

# IBM System/3 Communications Control Program Programmer's Reference Manual

## Program Numbers:

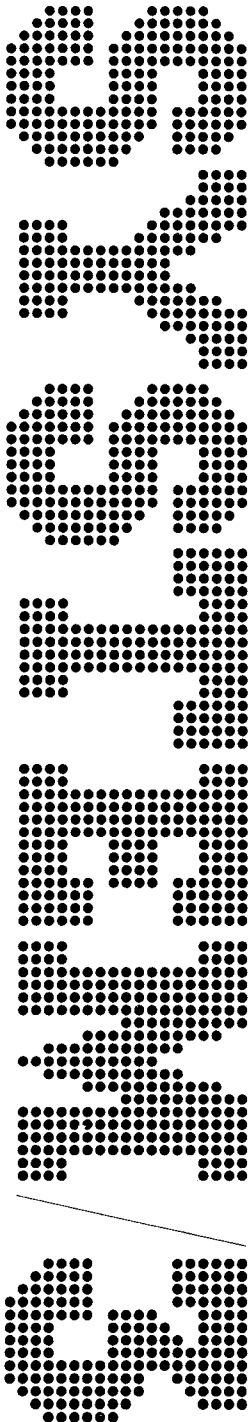
5702-SC1 (Model 10 Disk System)

5704-SC1 (Model 15)

5704-SC2 (Model 15)

5705-SC1 (Model 12)

Feature 6011/6012/6033/6070/6071



## **Sixth Edition (September 1979)**

This is a major revision of, and obsoletes, GC21-7579-5 and Technical Newsletters GN21-7970 and GN21-5628. Information has been added to support Program Number 5704-SC2 for the IBM System/3 Model 15D. Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change or addition.

This edition applies to the IBM System/3 Communications Control Program for:

- Version 15, modification 00, of Program Number 5702-SC1 for the IBM System/3 Model 10
- Version 7, modification 00, of Program Number 5704-SC1 for the IBM System/3 Model 15
- Version 4, modification 00, of Program Number 5704-SC2 for the IBM System/3 Model 15
- Version 5, modification 00, of Program Number 5705-SC1 for the IBM System/3 Model 12

This edition also applies to all subsequent versions and modifications unless otherwise indicated in new editions or technical newsletters.

Changes to the information herein are made periodically; changes will be reported in technical newsletters or in new editions of this publication.

Use this publication only for the purpose stated in the *Preface*.

Publications are not stocked at the address below. Requests for copies of IBM publications and for technical information about the system should be made to your IBM representative or to the IBM branch office serving your locality.

This publication could contain technical inaccuracies or typographical errors. Use the Reader's Comment form at the back of this publication to make comments about this publication. If the form has been removed, address your comments to IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901. IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

This publication describes how to write telecommunications application programs to run under control of the communications control program (CCP). CCP is a feature of disk system management that facilitates the implementation of telecommunications applications on the Model 10 Disk System, Model 12, and Model 15.

This manual is intended for programmers who use one or more of the following System/3 programming languages:

- RPG II
- Subset American National Standard (ANS) COBOL
- FORTRAN IV
- Basic Assembler

The introduction to this manual summarizes the purpose and operation of CCP. Subsequent chapters describe the standard application program interface to CCP, examples of typical application program logic, application programming in COBOL, FORTRAN IV, RPG II, and Basic Assembler, preparing source programs to run under CCP, program testing, and use of the optional 3270 display format facility of CCP.

### System/3 Model 8

The System/3 Model 8 is supported by System/3 Model 10 Disk System control programming and program products. The facilities described in this publication for the Model 10 are also applicable to the Model 8, although Model 8 is not referenced. However, the integrated communications adapter (ICA) and local display adapter are available on the Model 8. If you have the ICA or local display adapter, it is always designated on BSCA line 2. Therefore, you must specify line 2 whenever it is required, or enter the BSCA OCL statement (// BSCA LINE-2) at execution time.

It should be noted that not all devices and features which are available on the Model 10 are available on the Model 8. Therefore, Model 8 users should be familiar with the contents of *IBM System/3 Model 8 Introduction*, GC21-5114.

### Prerequisites

The CCP application programmer need not have extensive previous knowledge of telecommunications networks, data link control, and the characteristics of specific terminal devices. This manual assumes, however, that the programmer has a working knowledge of his programming language and is familiar with the configuration of the CCP system in his installation.

This manual has no specific prerequisite publications; however, many references are made to the following manuals that are required by the programmer:

- *IBM System/3 Models 8, 10, and 12 Communications Control Program System Reference Manual*, GC21-7588
- *IBM System/3 Model 15 Communications Control Program System Reference Manual*, GC21-7620
- *IBM System/3 Communications Control Program System Design Guide*, GC21-5165
- *IBM System/3 Model 15D Channel Connected Systems Program Reference and Logic Manual*, GC21-5199

Also, in order to fully utilize the display format facility of CCP, the programmer must have a basic understanding of the concepts and operation of the IBM 3270 Information Display System as given in *IBM 3270 Information Display System Component Description*, GA27-2749.

Other publications that are useful to the programmer are listed in *Appendix C: Bibliography*.

This page is intentionally left blank.

<b>CHAPTER 1: INTRODUCTION</b> . . . . .	<b>1-1</b>	Accept No-Wait Input (5704-SC2 Only) . . . . .	2-30
CCP Stages . . . . .	1-1	Function and Use of Accept No-Wait Input . . . . .	2-30
Generation Stage . . . . .	1-1	Stop Invite Input (or Get) . . . . .	2-31
Assignment Stage . . . . .	1-2	Function and Use of Stop Invite Input . . . . .	2-32
Operational Stage . . . . .	1-2	Get Terminal Attributes . . . . .	2-32
Terminals and Features Supported . . . . .	1-3	Special Information Returned in the Parameter List . . . . .	2-33
<b>CHAPTER 2: STANDARD APPLICATION PROGRAM</b>		Information Returned in the Record Area . . . . .	2-33
<b>INTERFACE TO CCP</b> . . . . .	<b>2-1</b>	Function and Use of Get Attributes . . . . .	2-36
Communications Service Subroutine . . . . .	2-2	Specifying the Terminal . . . . .	2-36
Parameter List . . . . .	2-2	Acquire Terminal . . . . .	2-36
Return Code (Positions 0-1) . . . . .	2-2	Function and Use of Acquire Terminal . . . . .	2-37
Operation Code (Positions 2-3) . . . . .	2-3	Generic Acquire Terminals (5704-SC2 Only) . . . . .	2-38
Third Field (Positions 4-5) . . . . .	2-3	Example . . . . .	2-38
Maximum Input Length (Positions 6-7) . . . . .	2-4	Port Command Request (5704-SC2 Only) . . . . .	2-39
Address of the Record Area (Positions 8-9) . . . . .	2-4	Task Chain Request (5704-SC2 Only) . . . . .	2-40
CCP Work Area (Positions 10-15) . . . . .	2-4	Function and Use of the Task Chain Request . . . . .	2-40
Record Area . . . . .	2-4	Release Terminal . . . . .	2-42
Program Name . . . . .	2-4	Function and Use of Release Terminal . . . . .	2-43
Symbolic Terminal Name . . . . .	2-4	Release and Task Chain (5704-SC2 Only) . . . . .	2-44
Multicomponent Terminal Considerations . . . . .	2-5	Function and Use of the Release and Task Chain . . . . .	2-44
Data Transfer and Translation . . . . .	2-6	Release and Task Chain Involving Sort Programs . . . . .	2-45
Terminal Attributes . . . . .	2-6	Shutdown Inquiry . . . . .	2-48
Input Data Transfer . . . . .	2-6	Function and Use of Shutdown Inquiry . . . . .	2-49
Input Data Translation . . . . .	2-7	Wait Operation (Model 15 Only) . . . . .	2-49
Output Data Transfer . . . . .	2-8	Function and Use of Wait Operation . . . . .	2-49
Output Data Translation . . . . .	2-8		
Transmitting 3735 FDPs on an ASCII Line . . . . .	2-9	<b>CHAPTER 3: COMMUNICATIONS PROGRAMMING</b>	
Record Separators (Variable Length and Spanned		<b>TOPICS</b> . . . . .	<b>3-1</b>
Records) . . . . .	2-10	Terminal Classes . . . . .	3-1
Device Control Characters . . . . .	2-11	Command Terminals . . . . .	3-1
MLTA Typewriter Terminals . . . . .	2-12	Data Terminals . . . . .	3-1
BSCA Terminals . . . . .	2-13	Program use of Terminals . . . . .	3-2
Line Control Characters . . . . .	2-13	Requesting Terminal . . . . .	3-2
Communicating with MLTA Terminals . . . . .	2-13	Program-Selected Terminal . . . . .	3-2
Communicating with BSCA Terminals . . . . .	2-13	Program Types . . . . .	3-3
Blocking . . . . .	2-13	Single Requesting Terminal (SRT) Program . . . . .	3-3
End of Transmission (EOT) . . . . .	2-14	Multiple Requesting Terminal (MRT) Program . . . . .	3-3
BSCA Input Operations . . . . .	2-14	Special Program Attributes . . . . .	3-4
BSCA Output Operations . . . . .	2-16	Never-Ending Program . . . . .	3-4
3284/3286 Printer Consideration . . . . .	2-17	Serially Reusable Program (Models 10 and 12) . . . . .	3-5
Operations . . . . .	2-18	Dedicated Program (Models 10 and 12) . . . . .	3-5
Program Errors . . . . .	2-18	Program Request Under Format . . . . .	3-5
3270 Display Format Facility Operations . . . . .	2-18	Sort Programs . . . . .	3-7
Get . . . . .	2-19	Task Chaining and Release and Task Chain With Sort	
Function and Use of Get . . . . .	2-19	(5704-SC2 Only) . . . . .	3-8
Specifying the Terminal . . . . .	2-19	Examples of Application Program Logic . . . . .	3-8
Put . . . . .	2-21	Single Requesting Terminal . . . . .	3-9
Function and Use of Put . . . . .	2-22	Single Requesting Terminal and Program-Selected	
Put-Then-Get . . . . .	2-23	Terminals . . . . .	3-11
Function and Use of Put-Then-Get . . . . .	2-24	Multiple Requesting Terminals . . . . .	3-13
Put-No-Wait . . . . .	2-25	Multiple Requesting Terminals and Program-Selected	
Function and Use of Put-No-Wait . . . . .	2-26	Terminals . . . . .	3-15
Invite Input . . . . .	2-27	Symbolic Files . . . . .	3-16
Function and Use of Invite Input . . . . .	2-27	Considerations and Restrictions in Using Symbolic	
Accept Input . . . . .	2-28	Files . . . . .	3-17
Function and Use of Accept Input . . . . .	2-28	Switched Lines . . . . .	3-17
		BSCA Switched Line . . . . .	3-17
		MLTA Switched Line . . . . .	3-20
		Switched Line Disconnect Considerations . . . . .	3-20

<b>CHAPTER 4: COBOL</b>	<b>4-1</b>
COBOL Use of the Standard Interface	4-1
Defining the Record Area and Parameter List	4-1
Record Area	4-1
Parameter List	4-3
Setting the Contents of the Parameter List and Record Area	4-4
Setting Fields in the Parameter List	4-4
Setting the Record Area	4-7
Calling the Communications Service Subroutine	4-9
Examining Returned Information	4-11
Return Code	4-11
Examining a Returned Name	4-13
Referencing Saved Information	4-13
Effective Input Data Length	4-13
Count of Outstanding Invite Inputs	4-13
Input Data	4-13
Using the System Operator Console	4-15
COBOL Programming Considerations	4-15
3270 Display Format Facility	4-15
Programming Examples	4-17
Example 1	4-17
Example 2	4-24
<b>CHAPTER 5: FORTRAN IV</b>	<b>5-1</b>
FORTRAN Use of the Standard Interface	5-1
Defining the Record Area and Parameter List	5-1
Record Area	5-1
Parameter List	5-2
Setting the Contents of the Parameter List and Record Area	5-3
Setting Fields in the Parameter List	5-3
Setting the Record Area	5-6
Calling the Communications Service Subroutine	5-8
Examining Returned Information	5-9
Return Code	5-9
Examining a Returned Name	5-9
Referencing Saved Information	5-11
Effective Input Data Length	5-11
Count of Outstanding Invite Inputs	5-13
Using the System Operator Console	5-13
FORTRAN Programming Considerations	5-13
3270 Display Format Facility	5-13
Programming Examples	5-15
Example 1	5-15
Example 2	5-21
<b>CHAPTER 6: RPG II</b>	<b>6-1</b>
RPG II Use of the Standard CCP Interface	6-1
Communications Interface Using RPG II Special Files	6-2
Parameter Array for SPECIAL	6-2
Record Area for SPECIAL	6-2
CCP Communications Service Subroutines for SPECIAL	6-3
SPECIAL Input Record Area	6-4
SPECIAL Output Record Area	6-4
Indicators Reserved for CCP Use	6-5
Defining Special Files for use with CCP	6-5
Main File Description for SPECIAL	6-8
Continuation Specification for SPECIAL	6-8
Defining the Parameter Array	6-9
Extension Specifications	6-9

CCP Operation Codes	6-10
Force End of File (SPECIAL Only)	6-12
Put With Invite Input (SPECIAL Only)	6-12
Put-No-Wait With Invite Input (SPECIAL Only)	6-12
Performing CCP Operations with SPECIAL	6-12
Performing CCP Operations Using Primary, Secondary, or Demand Input	6-13
Performing CCP Operations Using Heading, Detail, Total or Exception Output	6-16
Heading, Detail or Total Output	6-19
Put-Then-Get Operation	6-19
Non-I/O Operations	6-20
Operations Issued at Input Time	6-21
Operations Issued at Output Time	6-22
EXIT/RLABL Communications Interface	6-22
Parameter Array	6-23
Record Area	6-23
EXIT to SUBR91	6-23
Setting the Parameters for EXIT/RLABL Operations	6-24
Examining Returned Information	6-25
EXIT to SUBR90	6-26
Task Chaining (5704-SC2)	6-27
EXIT to SUBR87 and SUBR88	6-27
3270 Display Format Facility	6-30
RPG II Programming Considerations	6-30
Specific Restrictions	6-32
Programming Examples	6-32
Example 1	6-32
Example 2	6-38

## CHAPTER 7: BASIC ASSEMBLER PROGRAMMING

<b>FOR CCP</b>	<b>7-1</b>
Symbols Used in Defining Macro Instructions	7-1
Mnotes	7-1
Generate Equates for Common Values (\$NCOM)	7-1
Generate Equates for Parameter List Offsets (\$NPLO)	7-2
Generate Operation Code/Modifier Values (\$NOPV)	7-3
Generate Equates for Return Code Values (\$NRTV)	7-3
Generate Parameter List (\$NPL)	7-6
Example	7-7
Set Control Information for Communications Operation (\$NCIO)	7-8
Examples of Using \$NCIO	7-12
Programming Restrictions	7-14
Assembler Macro Support Mnotes	7-14
Programming a User Security Routine—Models 10 and 12	7-16
Sample Program—Model 10 or Model 12	7-19
Programming a User Security Routine—Model 15	7-22
Sample Program—Model 15	7-23

**CHAPTER 8: 3270 DISPLAY FORMAT FACILITY**

<b>(DFF)</b> . . . . .	<b>8-1</b>
General Information . . . . .	8-1
Overview . . . . .	8-1
Prerequisite Information . . . . .	8-2
DFF Routines . . . . .	8-2
Field Concepts . . . . .	8-4
Definition . . . . .	8-4
Field Classes . . . . .	8-4
Field Types . . . . .	8-6
Planning the Printer/Display Layout . . . . .	8-11
Attributes . . . . .	8-11
Output Class . . . . .	8-13
Input and Output/Input Classes . . . . .	8-13
SPD Class . . . . .	8-14
Autoskip and Cursor Positioning . . . . .	8-16
Defining Data . . . . .	8-17
Data in Numeric Input Edited Fields (5704-SC2 Only). . . . .	8-19
Number of Fields . . . . .	8-20
Record Concepts . . . . .	8-21
Display Output Record Format . . . . .	8-21
Printer Output Record Format . . . . .	8-21
Input Record Format . . . . .	8-21
Display Format Generator . . . . .	8-22
Printer/Display Layout Sheet . . . . .	8-22
Display Control Form . . . . .	8-22
Field Definition Form . . . . .	8-26
Additional Functions for the Field Definition Statement . . . . .	8-28
OCL Considerations for the Display Format Generator . . . . .	8-32
Display Format Generator Diagnostic Messages . . . . .	8-32
Printer Format Generator Routine (PFGR) . . . . .	8-33
Printer/Display Layout Sheet . . . . .	8-33
Printer Control Form . . . . .	8-33
Field Definition Form . . . . .	8-36
Printer Control on the Field Definition Statement . . . . .	8-38
PFGR Line/Partial-Line Duplication . . . . .	8-38
OCL Considerations for the Printer Format Generator . . . . .	8-40
Printer Format Generator Diagnostic Messages . . . . .	8-40
Display Format Control Routine (DFCR) . . . . .	8-41
3270 Display Operations . . . . .	8-43
Operation Considerations with DFF . . . . .	8-43
Put Message . . . . .	8-43
Put-No-Wait . . . . .	8-44
Put Override . . . . .	8-44
Selecting the WCC . . . . .	8-49
Copy . . . . .	8-50
Selecting the Copy Control Character . . . . .	8-52
Erase . . . . .	8-53
Return Codes . . . . .	8-53
Input Operations - Accept Input, Get, Stop Invite Input . . . . .	8-53
User Program Record Area . . . . .	8-54
Display Concepts . . . . .	8-56
New Screens . . . . .	8-56
Overlay Screens . . . . .	8-56
DFF, CCP Considerations . . . . .	8-58
Assignment Control Statements . . . . .	8-58
Storage Area . . . . .	8-58
Terminal Operator Actions . . . . .	8-59
Display Format Test Routine (\$CCPDT) . . . . .	8-61
Format Find Routine (5704-SC2 Only) . . . . .	8-64

Examples . . . . .	8-65
Example 1—DFF Formatting Example . . . . .	8-65
Example 2—RPG II MRT Program Using the Display Format Facility . . . . .	8-72
Example 3—SRT Inquiry Program . . . . .	8-95
Example 4—RPG II Order Entry Program (Using PRUF with the DFF) . . . . .	8-103
Example 5—COBOL Order Entry Program (Using PRUF with the DFF) . . . . .	8-122

**CHAPTER 9: PROGRAM PREPARATION . . . . . 9-1**

Compiling and Link-Editing the Program—Model 15 CCP . . . . .	9-1
Compiling the Program—Model 10 and Model 12 CCP . . . . .	9-2
Link-Editing the Program—Model 10 and Model 12 CCP . . . . .	9-3
Overlay Linkage Editor Control Statements—Model 10 and Model 12 CCP . . . . .	9-3
Link-Editing a Program to Run Under DSM—Models 10 and 12 . . . . .	9-4
Copying the Load Module . . . . .	9-6
Making Assignments . . . . .	9-6
Unit Record File Considerations—Model 10 and Model 12 CCP . . . . .	9-7
Unit Record File Considerations—Model 15 CCP . . . . .	9-7
Disk File Considerations . . . . .	9-8
Models 10 and 12 Considerations . . . . .	9-8
Model 15 Considerations . . . . .	9-10
Model 10 and Model 12 CCP File Sharing Considerations . . . . .	9-14
Model 15 CCP File Sharing Considerations . . . . .	9-14
Determining the Disk File Access Value . . . . .	9-19

**CHAPTER 10: PROGRAM TESTING . . . . . 10-1****CHAPTER 11. PORTLINE APPLICATION PROGRAM**

<b>INTERFACE TO CCP . . . . .</b>	<b>11-1</b>
Required Ports . . . . .	11-2
Operation Codes . . . . .	11-2
Portline Exception to Get Terminal Attributes . . . . .	11-3

**APPENDIX A: CPU TO CPU CONSIDERATIONS . . . . . A-1**

Attachment Configurations . . . . .	A-2
Programming Considerations . . . . .	A-2
Command Mode . . . . .	A-3
Data Mode . . . . .	A-5
Generation Considerations . . . . .	A-5
\$EMLA and \$EMLD Statements . . . . .	A-5
\$EBSC and \$EBSL Statements . . . . .	A-5
\$ECSL and \$ECSL Statements (5704-SC2 Only) . . . . .	A-6
Assignment Considerations . . . . .	A-6
Recommendations and Examples . . . . .	A-6
Example 1: Multipoint Command Mode . . . . .	A-7
Example 2: Point-to-Point Command Mode . . . . .	A-8
Example 3: Point-to-Point Switched Command Mode . . . . .	A-9
Example 4: Point-to-Point Data Mode . . . . .	A-11

**APPENDIX B: GLOSSARY . . . . . B-1****APPENDIX C: BIBLIOGRAPHY . . . . . C-1**

CCP . . . . .	C-1
General System/3 . . . . .	C-1
BSC and BSCA Terminals/Systems . . . . .	C-1
Programming Language Manuals . . . . .	C-2
General Telecommunications . . . . .	C-2

<b>APPENDIX D: OPERATION CODES . . . . .</b>	<b>D-1</b>
<b>APPENDIX E: RETURN CODES . . . . .</b>	<b>E-1</b>
Negative Return Codes . . . . .	E-1
Use of Data Truncated Return Code in 3270 DFF . . . . .	E-1
<b>INDEX . . . . .</b>	<b>X-1</b>



In order to gain an overall understanding of the requirements for writing application programs under CCP, read chapters 1, 2, and 3 before reading the chapter that applies to your programming language. These chapters contain:

Chapter 1: Summary of the purpose and operation of CCP.

Chapter 2: Description of the application program interface to CCP, independent of any particular programming language.

Chapter 3: General description of terminal classes, program types, and CCP application program logic.

After you have read the first three chapters, read thoroughly the chapter that applies to your programming language:

Chapter 4: COBOL

Chapter 5: FORTRAN IV

Chapter 6: RPG II

Chapter 7: Basic Assembler

If your program will use the 3270 Display Format Facility of CCP to communicate with components of the IBM 3270 Information Display System, read chapter 8 after you have an understanding of CCP application programming in your language.

Before attempting to write a CCP application program, be sure to read *Chapter 9: Program Preparation*, since that chapter contains important disk and unit record file considerations you must be aware of.

### Reference Aids

The appendixes provide convenient summaries of application program operation codes and return codes as well as a glossary of terms and a bibliography.

Use the index at the end of the manual to locate specific subjects.

### Terminology Notice

Unless otherwise indicated, references in this manual to BSCA (binary synchronous communications adapter) should be interpreted to mean BSCA and BSCC (binary synchronous communications controller). BSCC is supported on the Model 15D only, and as a multipoint nonswitched control station or in point-to-point nonswitched configurations.



The Communications Control Program (the CCP) is a system control program feature of the IBM System/3 Model 10 Disk System, IBM System/3 Model 12, and IBM System/3 Model 15 designed to facilitate the development and implementation of telecommunications applications. The CCP serves as the control program of a telecommunications subsystem, operating in conjunction with disk system management (referred to by the abbreviation DSM in this manual).

Under the CCP, an online network of terminals can call application programs as needed and access a common set of disk files. If sufficient main storage is available, the CCP permits several application programs to be executing concurrently under its control.

Communications application programs to be run under control of the CCP can be written in any of the high-level languages available with Models 10, 12, and 15 — RPG II, COBOL, and FORTRAN IV — and in Basic Assembler. Individual application programs can be written without detailed knowledge of the requirements for programming under a telecommunications system and, with few exceptions, as though they are to be run individually, with access to all system resources.

With the facilities provided by the CCP, the System/3 can be used either as a host system or as a subhost system:  
**Host System:** The System/3 is the central controller of a network of start-stop and/or binary synchronous terminals.  
**Subhost System:** The System/3, while directly controlling a group of terminals, is itself a tributary station to a large central processor, such as System/370.

**Note:** For an introduction to the CCP that includes more detailed descriptions of CCP services and relationships between the CCP and other System/3 programs, see the *CCP System Reference Manual*, GC21-7588 for Models 10 and 12, GC21-7620 for Model 15. If you are not acquainted with terms and abbreviations used in this manual, you can find definitions either in *Appendix B. Glossary* at the end of this manual, or in *IBM Data Processing Glossary*, GC20-1699.

## CCP STAGES

Establishing and operating the CCP in a particular environment is accomplished in three stages:

- Generation
- Assignment
- Operation

### Generation Stage

CCP generation is the process whereby your installation creates its individual version of the CCP. The purpose of generation is to establish the required capabilities of the CCP by creating a set of CCP object modules and sub-routines, unique to the requirements of your installation. The process of generation involves:

1. Describing the type of equipment to be used by the communications system and other permanent features of the CCP system.
2. Creating a set of control routines whose specific content may be unique to your installation.
3. Joining the routines by a link-editing process.
4. Copying appropriate additional supporting routines.
5. Initializing the control file that the assignment stage and the operational stage use (\$CCPFILE).

CCP Generation is described in *CCP System Reference Manual*, GC21-7588 for Models 10 and 12, GC21-7620 for Model 15.

## Assignment Stage

CCP assignment stage is a brief process by which one or more *sets* of specific environments in which the CCP can run are defined. 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 that 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.
- Terminal attributes.

The assignment run need be repeated only when some of the specific information given in a previous assignment run must be changed. For example, CCP assignment must be repeated when new programs and files are to be used under the CCP.

As a programmer, you must be familiar with the contents of the CCP assignment sets, since you must be aware of characteristics of files, terminals, and communication lines available to programs you write. You can determine the contents of assignment sets from the listing produced by the Assignment List program.

See *CCP System Reference Manual*, GC21-7588 for Models 8, 10, and 12, GC21-7620 for Model 15, for detailed information about CCP Assignment.

## Operational Stage

The operational stage begins with *operational startup*, when the CCP is loaded into main storage. During startup, CCP routines open disk files, adapters, and communication lines and complete various tables and control blocks. During *operation*, the CCP supervises the environment in which your application programs run and provides communications services to your programs. The operational stage is concluded by shutdown, which is initiated by the system operator. During *shutdown*, the CCP allows programs that are currently executing, or that are currently scheduled or chained, to finish processing, then it closes communication lines, adapters, and files.

See *CCP System Operator's Guide*, GC21-7581 for Models 8, 10, and 12, or GC21-7619 for Model 15 for a detailed description of CCP operation.

## TERMINALS AND FEATURES SUPPORTED

The following terminals may be used with the communications control program.

Through the multiple line terminal adapter:

- 1050 Data Communication System
    - Switched
    - Multipoint nonswitched
  - 2740 Communication Terminal Model 1
    - Basic
    - 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
    - Station control
    - Station control, checking
    - Station control, buffered receive
    - Station control, buffered receive, checking
  - 2741 Communication Terminal
    - Basic
    - Switched
  - 3767 Communication Terminal (when simulating a 2740 Model 1)
    - Checking
    - Dial with checking
    - Station control, checking(when simulating a 2740 Model 2)
    - Station control, checking
  - 3767 Communication Terminal (when simulating a 2741)
    - Basic
    - Switched
  - Communicating Magnetic Card SELECTRIC® Typewriter (appears identical to a 2741 switched)
    - Point-to-point switched
  - System/7 (appears identical to a 2740 Model 1)
    - Checking
    - Dial with checking
    - Station control with checking
  - 5100/5110 Portable Computer (when simulating a 2741)
    - Point-to-point switched
    - Point-to-point nonswitched
- With the binary synchronous communications adapter:
- 3270 Information Display System
    - Multipoint nonswitched
  - 3275 Information Display Station
    - Switched
    - Nonswitched
  - 3735 Programmable Terminal
    - Switched
    - Multipoint nonswitched
  - 3741 Data Station Model 2, Programmable Work Station Model 4
    - Point-to-point nonswitched or switched
    - Multipoint
  - 5110 Portable Computer
    - Point-to-point switched
    - Point-to-point nonswitched
    - Multipoint with CCP as control station
  - 5230 Data Collection System (appears identical to a 3741 Model 2 or 4)
    - Point-to-point switched
    - Point-to-point nonswitched
  - 5937 Display Station RPO
    - Appears to be a 240-byte 3277 Model 1
  - 3600 Finance Communication System (see *Note 2*)
    - Multipoint nonswitched
  - System/3
    - Point-to-point switched
    - Point-to-point nonswitched
    - Multipoint with CCP as control station
    - Multipoint with CCP as a tributary
  - System/7 Feature 2074 or RPO (see *Note 1*)
    - Point-to-point switched
    - Point-to-point nonswitched
    - Multipoint with CCP as control station
  - System/360, System/370
    - Point-to-point switched
    - Point-to-point nonswitched
    - Multipoint with CCP as tributary

- System/32
  - Point-to-point switched
  - Point-to-point nonswitched
  - Multipoint with CCP as control station
- System/34
  - Point-to-point switched
  - Point-to-point nonswitched
  - Multipoint with CCP as control station
- Series/1
  - Point-to-point switched
  - Point-to-point nonswitched

With the binary synchronous communications controller:

- 3270 Information Display System
  - Multipoint nonswitched
- 3735 Programmable Terminal
  - Multipoint nonswitched
- 3741 Data Station Model 2, Programmable Work Station Model 4
  - Multipoint nonswitched
  - Point-to-point nonswitched
- System/3
  - Multipoint with BSCC as control station
  - Point-to-point nonswitched
- System/7 Feature 2074 or RPQ (see *Note 1*)
  - Multipoint with BSCC as control station
  - Point-to-point nonswitched
- System/34
  - Point-to-point nonswitched
  - Multipoint nonswitched
- 3600 Finance Communications System (see *Note 2*)
  - Multipoint with CCP as control station
- 5230 Data Collection System (appears identical to a 3741 Model 2 or 4)
  - Point-to-point nonswitched

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.

*Notes:*

1. Under BSCA or BSCC, the System/7 is supported only as it is supported by the Multiline/Multipoint BSCA IOCS — see *IBM System/7 (RPQ) Binary Synchronous Module Programming Guide and Reference Manual, SC34-1510*.
2. Under BSCA or BSCC, each 3600 work station is supported as a CPU by CCP. RPQ number 540156 is required to attach the 3600 to System/3 Model 8, 10, and 12.

The *standard interface* (that is, the procedures and common data areas) used by application programs to request CCP to perform communications operations with remote terminals or the system operator's console is composed of the following basic elements:

- Communications Service Subroutine
- Parameter List
- Record Area
- A set of communications operations that can be issued to the CCP

The details of this interface differ slightly among the programming languages—RPG II, COBOL, FORTRAN IV, and Basic Assembler—but the functions performed by the basic elements remain essentially the same. Where the interface for a particular language differs from the standard interface, you are referred to the chapter covering that language.

In order to perform a communications operation, such as writing a message to a terminal, an application program must do the following:

1. Provide storage space within itself for a parameter list and record area and specify the format of these areas.
2. Prepare the record area for the operation.
3. Set the contents of the parameter list.
4. Invoke the communications service subroutine to perform the operation.
5. Check appropriate return codes to determine the result of the operation.

Since your program may be competing with other programs for system resources such as terminals, disk files, and unit record devices, CCP ensures that these resources are available to your program before your program is allowed to run. Each terminal required by your program is allocated exclusively to your program until your program releases it (see index entry *Release Terminal Operation*) or

until the execution of your program has ended. When either of these events has occurred, the terminal is free to be allocated to another program (or to enter commands, if it is a command terminal). Because CCP also allocates the use of unit record devices, you can code I/O operations using these devices as though your program has exclusive control of them. (Exception for Model 15 CCP: Your program can share use of the 1403 printer with another program running concurrently if PRINTER—SHR is specified in the PROGRAM assignment statement for your program [see *CCP System Reference Manual*]. You should consider in the design of your program that you do not have exclusive control of the printer.)

CCP may receive a request for a program that uses:

- A terminal that is presently allocated to another program.
- A disk file that is allocated to another program in such a manner that the access methods conflict. For example, a currently executing program adds consecutively to a file and the program being requested adds to the same file.
- A disk file is specified as NOSHR on the FILES parameter of the PROGRAM assignment statement or as SHARE-NO on the FILE OCL statement on 5704-SC2.

CCP rejects such a program request or queues it, depending on the queue status of the terminal (see /Q and /NOQ commands in *CCP Terminal Operator's Guide*, GC21-7580). When the previous program has terminated, terminals and disk files used by that program are available to subsequent programs.

*Note:* Model 10 and Model 12 CCPs can also queue (/Q) a request for a program that uses a unit record device that is temporarily unavailable. Model 15, however, normally rejects requests for programs that require a unit record device that is unavailable. The exception is if the requested program uses the printer and the printer is either permanently allocated to the CCP partition, or spool is intercepting the CCP partition, and the requested program uses no other unit record devices or terminals.

## COMMUNICATIONS SERVICE SUBROUTINE

Since RPG II, COBOL, and FORTRAN IV, do not include special statement types for general purpose terminal I/O operations and other communications services (see *Operations*), the CCP provides one or more communications service subroutines to application programs written in each language. (For Basic Assembler Programs, a macro instruction is provided — see index entry *\$NCIO macro*.) The communications service subroutine converts the application program's request into a standard request to the CCP communication facilities.

The communication service subroutine (RPG II programs may actually use more than one) must be link edited to each application program prior to using the program under the CCP. Thus, the communication service subroutine, although provided by the CCP, actually becomes a part of the application object program. See *Chapter 9: Program Preparation* for procedures for preparing an application program to run under the CCP.

In COBOL and FORTRAN IV, the application program initiates a communications operation by issuing a CALL statement to the communications service subroutine. In RPG II, the program can initiate an operation and invoke the communications service subroutine either through the SPECIAL or EXIT/RLABL facilities of the language.

### PARAMETER LIST

You must provide a parameter list within your program with each request for a communications operation. The parameter list specifies the details of the communications operation and provides locations within itself where the CCP returns information about the results of the operation. This chapter describes the parameter list as it is presented by the communications service subroutine to the CCP communications facilities. In RPG II, the parameter list as defined in the user program is somewhat different (see *Chapter 6: RPG II*).

The parameter list is 16 positions long, consisting of eight two-position fields, as shown in Figure 2-1.

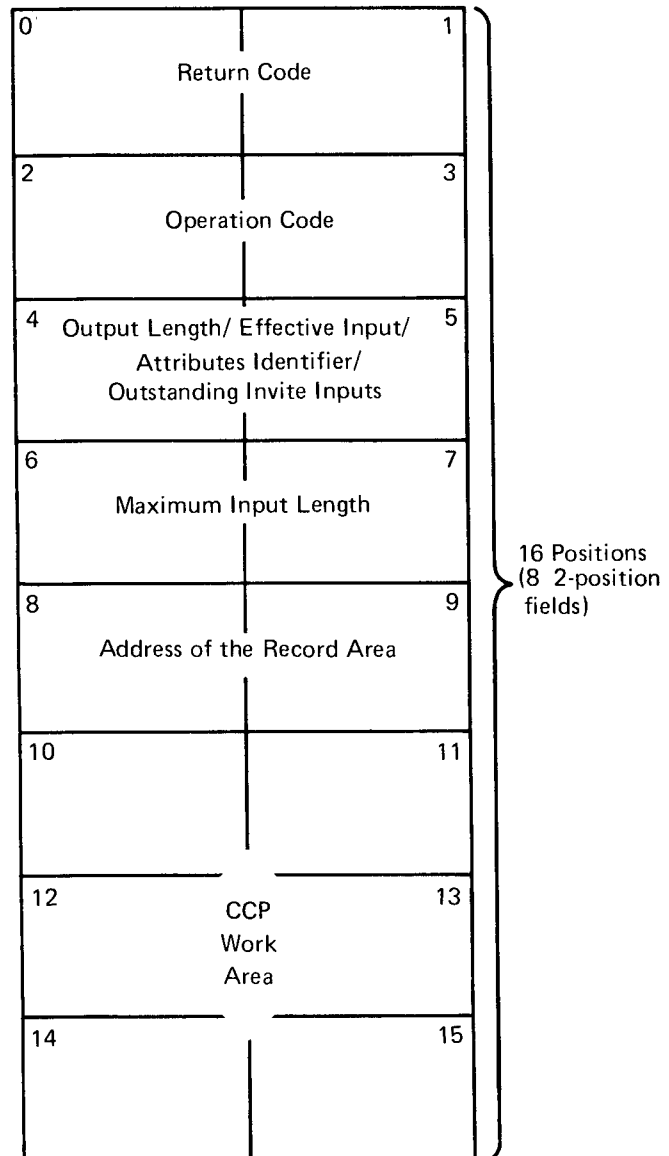
#### Return Code (Positions 0-1)

Although this field (see Figure 2-1) must be provided in the parameter list, the CCP ignores the contents at the beginning of the operation. At the completion of each

operation, before returning control to the application program, the CCP places a value in this field indicating the status of the operation:

- Operation completed normally (value of zero).
- Operation resulted in an I/O error (negative value).
- Operation resulted in an exception condition (positive value).

Specific return code values and meanings are given in *Appendix E: Return Codes*.



*Note:* In RPG II, the format of the parameter list is somewhat different (see *Chapter 6: RPG II*).

Figure 2-1. Parameter List



In order to determine the results of a communications operation, you must include coding in your program to test the return code. The degree of return code checking and the actions taken based on return code checking will vary in different applications, however, it is strongly recommended that return code checking at the level of normal completion (zero return code) or abnormal completion (non-zero return code) be done in all programs.

Examples of testing return codes are given in chapters 4 through 6. Recommended actions to be taken by your program for each return code are given in *Appendix E*. RPG II programmers should see *Chapter 6: RPG II* for additional information concerning handling of return codes in that language.

### Operation Code (Positions 2-3)

For each communications operation (except some RPG II operations), this field must contain a code that indicates the specific operation to be performed. The contents of this field are the same after completion of the operation as when the operation began. See *Operations*, later in this chapter, for descriptions of the valid operations and operation modifiers that can be issued to the CCP by an application program.

### Third Field (Positions 4-5)

This field in the parameter list can contain four different kinds of information:

1. *Output Length* – provided by your program for output operations (see *Output Operations*, following)
2. *Effective Input Length* – returned by the CCP (see *Input Operations*, following)
3. *Terminal Attributes Identifier* – provided by your program (see *Acquire Terminal Operation*, following)
4. *Count of Outstanding Invite Input Operations* – returned by the CCP (see *Input Operations and Release Terminal Operation*, following)

*Output Operations:* This field must contain the length of the data to be transmitted from your program, that is, the number of characters of data you wish to write from the record area in your program, not including the six positions for the symbolic terminal name and not including line control characters, which are added to your data by the CCP. (In RPG II, the output length is placed in the output record area; see Chapter 6.)

*Input Operations:* On each completed input operation, the CCP calculates and places into this field the actual length of the input data passed to the application program. This effective input length does not include the symbolic terminal name, line control characters, backspace characters, or data which the CCP cannot pass to the application program when the amount of data received exceeds the size of the record area (see *Maximum Input Length*, the next field in the parameter list). However, the effective input length does include record separator characters (see index entry *record separators*). The CCP ignores the contents of this field at the start of an input operation.

If *data mode escape* is allowed in your CCP system (see index entry) and a terminal enters the /RELEASE command after entering the data mode escape characters, your program will receive a 08 return code from any of the following input operations: Get, Accept Input, Put-Then-Get, and Stop Invite Input (see index entries). The 08 return code indicates that the terminal to which the input operation was issued is no longer available to your program. In this case, CCP places the current number of outstanding *Invite Inputs* for your program (see index entry) in positions 4-5 of the parameter list. This information is important in *multiple requesting terminal (MRT) programs* (see index entry).

*Acquire Terminal Operation:* If you issue an *Acquire Terminal* operation (see index entry) which sets the attributes of the terminal to be acquired, this field must identify the attribute set you want to assign to the terminal. The terminal attribute set is defined in the TERMATTR assignment statement – see *CCP System Reference Manual*.

*Release Terminal Operation:* If your program releases a terminal (see index entry *Release Terminal operation or Release and Task Chain operation (5704-SC2 only)*) and receives a zero return code from the operation, CCP places the current number of outstanding Invite Inputs for your program (see index entry) in the third field (positions 4-5) of the parameter list.

*Accept No-Wait Operation (5704-SC2 only):* In addition to the response given under *input operations*, the count of outstanding invites is returned in the third field of the parameter list for return code of 04 (shutdown was entered) and 16 (no outstanding invites completed).

*Shutdown Inquiry Operation (5704-SC2 only):* If you issue a shutdown inquiry operation, the count of outstanding invites is returned in the field, regardless of the return code.

### Maximum Input Length (Positions 6-7)

On each operation involving input data, you must enter a value into this field representing the maximum number of bytes of input data you expect to receive. This value does not include the six characters for the terminal name. This value must be greater than zero and no larger than the size of the record area provided by your program. The CCP does not alter this value during the operation.

### Address of the Record Area (Positions 8-9)

This field is set by the communications service subroutine (except in Basic Assembler, where this field is set by the \$NCIO macro) to contain the main storage address of the record area (see *Record Area*). This field addresses the first (leftmost) position of the name field in the record area, not the first position of data; therefore, the data actually begins at the address given, plus six. For operations not involving data transfer, this field may point to a record area containing only the name field.

This field is not present in the parameter list used by RPG II application programs.

### CCP Work Area (Positions 10-15)

These positions are used for a work area by the CCP. Your program must not use these positions.

## RECORD AREA

With each communications operation your program issues to the CCP (except Shutdown Inquiry), it must provide a *record area*. A record area is an area in the application program that consists of two parts. The standard record area for operations involving data transfer consists of a six-position *name field* followed by a *data area* (Figure 2-2). Exceptions to this standard format occur for RPG II (see *Chapter 6: RPG II*) and when the *3270 Display Format Facility* (see index entry) is used.

The name field contains either the name of the program (if a chained task operation), or the symbolic terminal name that is to be involved in the operation.

The parameter list field containing the record area address (Figure 2-1) always points to the leftmost position of the name field. Data transfer, however, always occurs into and out of the data area segment of the record area. Lengths specified in the parameter list for operations involving data transfer refer to the length of the data area portion of the record area, except in certain RPG II output operations.

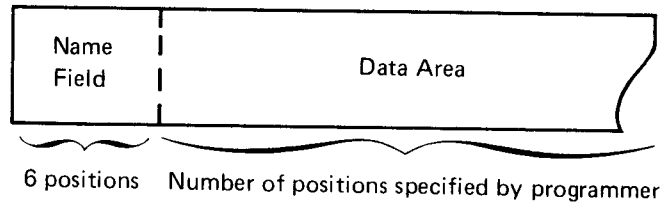


Figure 2-2. Standard Record Area

### Program Name

The program name is the name of the program to be called on a Chain Task Request operation (5704-SC2 only). For a task chain operation, your program must place the name of the program to be chained in the first six positions of the record area (left-justified and padded with blanks if less than six characters). If data is to accompany the chain request, the data follows the program name in the record area, and PGMDATA-YES must be specified on the PROGRAM assignment statement (see the *Model 15 CCP System Reference Manual*, GC21-7620) for the requested program.

### Symbolic Terminal Name

The terminal with which a communications operation is performed is identified by a symbolic terminal name in the first six positions of the record area (left justified). In most operations, the application program must place the name into the name field of the record area to specify the terminal with which to operate; in certain operations, however, the CCP places the name into the record area to inform the program with which terminal the operation took place. Each symbolic terminal name refers to a specific physical terminal device.

Three classes of terminal names are available for use in application programs:

1. **User-Defined Names:** These are the terminal names defined in TERMNAME statements during CCP assignment. The structure of these names must conform to the following rules:
  - The first character must be alphabetic (including #, \$, and @).
  - Each succeeding character can be either alphabetic or numeric.
  - One to five of the six possible positions in the name can be blank, but no blanks may be embedded between other characters. For example, the following names are valid: TERM

~~blank~~; the following are invalid: TERM2,  
TERM2.

- Each terminal name must be unique.
- The names CONSOL, ALL, and a name consisting of six blanks cannot be user-defined.

2. **CONSOL:** On the System/3 Model 10 and Model 12, the symbolic name CONSOL refers to the system operator's 5471 Printer/Keyboard. On the Model 15, the symbolic name CONSOL refers to the system operator's keyboard and 3277 Display Station, referred to as the CRT/Keyboard. Application programs can communicate with the system operator's console at any time; however, the console is never allocated to the program. Operations issued to the console by programs running under the CCP must be issued as communications operations; if issued in any other way, the results are unpredictable. The only operations that can be issued to the console are:

- Put
- Put-No-Wait (handled as a Put by the CCP)
- Put-Then-Get
- Get Attributes
- Accept Input (to accept only data that accompanies the program request)
- Accept No-Wait (to accept only data that accompanies the program request)

The CCP automatically releases the console from any program it requests as follows:

- If the console requested the program and the PROGRAM assignment statement (see *CCP System Reference Manual*) specifies PGMDATA-NO, the console is released when the program is loaded.
- If the console requested the program and the PROGRAM assignment statement specifies PGMDATA-YES, the console is released after an Accept Input operation results in the console program data being passed to the user program.

*Note to Model 10 and 12 users:* Programs that use *symbolic files* (see index entry) must allow data to be entered with the program request if they could be requested by the console (see *program request command* in *CCP System Operator's Guide*). These programs must also open all physical files to be

referenced by a symbolic file prior to issuing an Accept Input operation. (In RPG II, these files are automatically opened prior to the first input operation.)

3. **Blanks:** Programs that handle only one requesting terminal per execution (designated *single requesting terminal (SRT) programs*, see index entry), can issue communications operations with six EBCDIC blanks (hexadecimal 40) in the symbolic terminal name portion of the record area. The CCP interprets the blank name as a reference to the terminal that requested the program. Upon completion of such an operation, the CCP sets the first six positions of the record area to contain the name of the requesting terminal. The program cannot use blanks after it has released the requesting terminal (see index entry *Release Terminal* operation).

The use of symbolic names for terminals allows programs to be relatively independent of the specific terminals. However, the programmer must be aware of the type of terminal he is using since he must know the record length of the device; whether the terminal is capable of input only, output only, or both input and output; and other information (see index entry *Get Attributes*). The system operator can reassign a symbolic name of a terminal (perhaps the terminal is out of order or offline) to a different terminal during operation of the CCP to allow execution of programs using that terminal name.

Of those operations requiring a six-position symbolic terminal name area in the record area (only Shutdown Inquiry does not) only *Accept Input* does not require that the area contain a valid symbolic terminal name. The contents of the terminal name field for that operation are not used by the CCP.

Whenever you specify a symbolic terminal name other than CONSOL in an operation, you must ensure that the terminal is allocated to your program under that defined name. The only exceptions to this rule are the *Acquire Terminal* operation, (see index entry) which is a request to obtain a terminal, and *Get Terminal Attributes*, (see index entry) which can be requested for any defined terminal name in the system.

### Multicomponent Terminal Considerations

Multicomponent terminals are a special class of terminals that can have more than one input and/or output device attached. The 1050 Data Communications System is the only terminal currently supported by the CCP that is considered to be a multicomponent terminal. (Each component of the 3270 Information Display System is considered a separate terminal and has its own name.)

A 1050 system is treated by CCP as if it were one terminal regardless of the number of components attached. For example, the entire 1050 system is always allocated to a program; it is impossible for one component to be allocated to one program while another component of the same 1050 system is allocated to another program. Therefore, any program in control of a 1050 system has access to *all* components of that particular 1050 system.

As with every other terminal in the CCP system, the 1050 has a symbolic terminal name. However, this symbolic terminal name has a principal input and principal output component associated with it. When the symbolic terminal name is used in an operation, it refers to the principal components.

You can address other than the principal input and/or principal output component of a 1050 system. In addition to the symbolic terminal name, you can assign symbolic names to a component or pair of components. These are called symbolic sub-terminal names. To direct an operation to a specific component, use the symbolic sub-terminal name associated with that component.

The following special rules apply to use of multicomponent terminals:

- Only one *Invite Input operation* (see index entry) may be outstanding to the terminal at one time, regardless of the number of input components attached to the terminal.
- When an operation is issued in which CCP returns a symbolic terminal name, such as *Accept Input*, the name returned is always the master terminal name, never a symbolic sub-terminal name.
- The *Acquire Terminal* operation must specify a symbolic terminal name, not a symbolic sub-terminal name.
- The *Release Terminal* operation must specify a symbolic terminal name, not a symbolic sub-terminal name.

## DATA TRANSFER AND TRANSLATION

The CCP either moves data into your record area or out of your record area during a communications operation, according to the operation you specify in the *Operation Code* field of your parameter list. In order to know how data is transferred to or from a specific terminal, what the CCP does with the data, and what your program must do with the data, you must know what attributes are assigned to the terminal (for example, whether or not a 3270 is using the Display Format Facility).

## Terminal Attributes

TERMATTR assignment statements (see *CCP System Reference Manual*) define terminal attribute sets for terminals used under the CCP. Each attribute set is assigned an identification number. This number is then referenced in a BSCATERM or MLTATERM assignment statement to assign a particular set of attributes to a terminal. A terminal may have different attributes at different times and a single attribute set can be used by more than one terminal. See *Get Attributes* and *Acquire Terminal* for additional information about terminal attributes.

The terminal attribute sets specify the following information about terminals:

For BSCA and MLTA terminals:

- Whether or not the CCP will translate data sent to or received from the terminal.
- If data is to be translated, whether to force the data to uppercase EBCDIC.
- Whether the terminal is auto or manual answer (if on a switched line).

For BSCA terminals only:

- Record length
- Block length
- Input data mode (record, block, or message)
- Whether or not the EBCDIC transparency feature is used for output
- ITB (intermediate text blocks) used
- Variable length or spanned records used
- 3270 Display Format Facility used

For BSCA terminals on switched lines only:

- Whether or not the CCP will verify exchange identification sequences
- Whether the terminal is auto or manual call

## Input Data Transfer

Data received from a terminal as the result of an input operation (see *Operations*) is moved by CCP from the communication line buffer to your program's record area. Data

is received in the seventh and succeeding positions of your record area (the program or symbolic terminal name resides in positions 1-6 of the record area), except in the following instances:

- In RPG II, data may begin in a different position (see *Chapter 6: RPG II*).
- In 3270 Display Format Facility operations, the format of the record area varies with different operations (see *Chapter 8: 3270 Display Format Facility*).

With some exceptions, CCP removes teleprocessing line control characters from the terminal input data that it moves to your record area. These exceptions and a discussion concerning the length of the data passed to the record area follow:

1. For BSCA terminals, the ITB (intermediate text block) character is not removed from input data unless fixed length records are being processed in ITB record mode, with the correct record length. When using variable length records, the record separator character is returned in the record area as the last character of data. The effective input length returned in the third field of the parameter list includes the record separator character.
2. Programs that communicate with 3270 terminals without using the Display Format Facility will receive and must send the actual data and display control characters necessary for the 3270, such as Escape Command, Set Buffer Address, Start Field, buffer addresses, and others (see *Example 1* in chapters 4, 5, and 6 for specific examples in COBOL, FORTRAN, and RPG II).
3. For programs not using PRUF (program request under format) that are requested by 3270 terminals, the data appended to the program request is not processed by the Display Format Facility but is passed directly to the user program. See *Chapter 3: Communications Programming Topics*, for further description of PRUF. The data is provided in the program record area as a continuous string, but with no 3270 display control characters. The length of the data depends on the value specified in the SYSTEM assignment statement (see *Assignment Stage* in the *IBM System/3 CCP System Reference Manual* for your system). The maximum length of the data appended to the program request is the value of the PGMREQ parameter minus the length of the program name and one blank. Since 80 is the maximum value of PGMREQ, the maximum length of data that can be appended to the program request is 78 characters; any further data in the 3270 buffer at the time of the program request is not sent to the program. A positive input return code is posted if the data length exceeds the length specified. The return code can be tested and appropriate action taken.
4. For PRUF programs, more than 78 characters of program request data can be sent to the user program. The length of the data sent to the program can be up to the maximum length specified in the PRUFLNG parameter of that program's PROGRAM statement (see *Assignment Stage* in the *IBM System/3 Models 8, 10, and 12 Communications Control Program System Reference Manual*, GC21-7588, or the *IBM System/3 Model 15 Communications Control Program System Reference Manual*, GC21-7620). If the program being requested is a PRUF program, CCP will pass the entire 3270 text stream, control characters and data, to the user program at program request time. If PRUF\$Z was specified on the PROGRAM statement at assignment time, PRUF program request data is handled by the display format facility. See *Chapter 3: Communications Programming Topics* for a further description of PRUF.
5. For task chain requests with data, the maximum amount of data that can be transferred is limited only by the size of the teleprocessing buffer. If other tasks are active or the teleprocessing buffer is fragmented, the area for a chain task with a large amount of data may be temporarily unavailable. In this case, the task chain request should be reissued at a later time or reissued with a smaller amount of data. If the chained task is a sort program, the maximum amount of data that can be passed by the requesting program is 80 characters.
6. For CPU-or-CPU applications or MLTA applications for Program Number 5704-SC2, the maximum length of data that can be appended to the program request is 253.

#### Input Data Translation

The attribute set associated with a terminal specifies whether or not data received from that terminal is to be translated from the line transmission code (if other than EBCDIC) to EBCDIC. If translation is specified, the attribute set also indicates whether or not to force to upper case all alphabetic characters received.

*Note:* All input, including PRUF input, received from a terminal in command mode is forced to upper case.

### *EBCDIC Transmission Code Used or Translation Requested*

If the transmission code is EBCDIC, or if translation is requested, data is presented in EBCDIC in the record area. (If translation is requested, the data is converted to EBCDIC by the CCP.) No teleprocessing line control characters are included in the data except for the BSCA ITB character mentioned under *Input Data Transfer*. For MLTA, backspace characters sent from the terminal are not received in the data area; rather, the input data is received with all backspacing resolved. Also, if the last character of the input is a carriage return, the CCP removes it from the input data.

All other device control characters (such as 3270 control characters, tab key, carriage return in the middle of text) are treated as input data characters. Whether or not lower case alphabetic characters are translated to their corresponding upper case characters is determined by the attribute set currently associated with the terminal. If upper case translation is specified, all alphabetic data input appears in upper case EBCDIC in your program's record area.

If the length of the data received is greater than the maximum input length specified, the excess data is lost (truncated) and the effective input length equals the maximum input length. If the data length received is less than the maximum input length, the effective input length is set to equal the data length received, and the remainder of your record area is cleared to blanks up to the maximum input length.

### *Transmission Code Not EBCDIC and Translation Inhibited*

If the transmission code is not EBCDIC and the terminal attributes do not specify translation, the CCP places data into the record area as it is received, including backspace characters, but not including line control characters. The application program must be prepared to translate data to EBCDIC if the data is to be processed by the program. If more data is received than was specified as maximum input length in the parameter list, the excess data is lost and the CCP sets the effective input length equal to the maximum input length. If the data received is less than the maximum specified, the CCP sets the effective input length to the number of input characters received. The record area positions beyond the effective input length are set to blanks (X'40') (except for MLTA terminals under Model 10 and Model 12 CCP, when the content is unpredictable).

### **Output Data Transfer**

On output operations, the CCP moves data from your record area to the communication line buffer and transmits it to the terminal you specify. The data must begin in position 7 of the record area, following the symbolic terminal name, (except in some RPG II operations and when 3270 DFF is used). No teleprocessing line control characters are needed, since CCP adds the necessary line control characters before transmitting the data. However, you may include in your data any device control characters you desire (see *Device Control Characters*).

*Note:* For BSCA record mode output operations, if the output record length is less than the record length specified in the terminal attributes set, the number of characters specified as the output length (third field of the parameter list) is sent, followed by the number of blanks necessary to satisfy the record length specified in the terminal attributes set.

### **Output Data Translation**

The attribute set associated with a terminal specifies whether or not the data to be transmitted to that terminal is to be translated from EBCDIC to the line transmission code.

### *Translation*

If translation is specified in the terminal attributes, the CCP converts data from EBCDIC to the appropriate line transmission code. Any device control characters are treated as data; thus, if you include device control characters in your record area, they must be in EBCDIC form. If invalid characters are found during the translation of the data, data transfer does not occur and the CCP places a return code indicating translation error in the parameter list. If more data is sent in one output operation than the line buffer for the terminal can hold (in BSCA record mode operations, if the output length exceeds the record length specified in the terminal attributes set), then the excess data is lost (truncated). The return code indicates if there was either a translation error or a data truncation.

### *Translation Inhibited*

If translation is not specified, output data is taken from your record area and transmitted as is, except for the addition of line control characters. If more data is to be sent in one operation than the size of the line buffer can hold (in BSCA record mode operation, if the output length exceeds the record length specified in the terminal attributes set), then the excess data is lost (truncated). All the data that can fit into the line buffer (or record area, for BSCA record mode) is sent and a return code indicating the data has been truncated is placed in the parameter list.

### **Transmitting 3735 FDPs on an ASCII Line**

You must use a special procedure to transmit FDPs (form description programs) to a 3735 terminal under the following conditions:

- Transmitting on an ASCII line.
- CCP to translate input and/or output data.

This special procedure is necessary because the FDPs, themselves, must not be translated, but all other data, including the FDP header and trailer, must be translated.

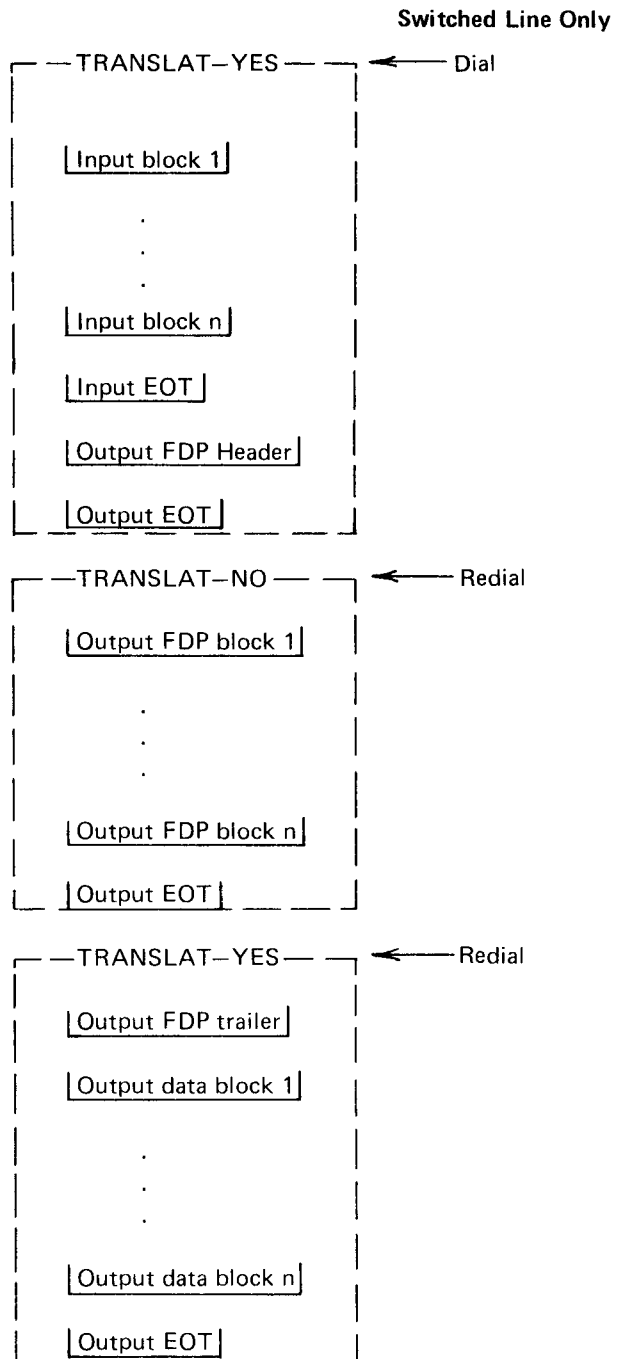
The procedure is as follows:

1. Define two attribute sets for the 3735 terminal at assignment time (TERMATTR statements), one specifying TRANSLAT=YES and the other specifying TRANSLAT=NO.
2. Initially, use the terminal attribute set that specifies TRANSLAT=YES for all input from the 3735 (until EOT is received).
3. Send the FDP header and an EOT using the same attribute set (TRANSLAT=YES).
4. Issue a Release Terminal (Keep-Line) operation followed by an Acquire Terminal (Set Terminal Attributes) operation, specifying the attribute set with TRANSLAT=NO.
5. Transmit all blocks of FDPs, followed by an EOT.
6. Issue a Release Terminal (Keep-Line) operation followed by an Acquire Terminal (Set Terminal Attributes) operation, specifying the attribute set with TRANSLAT=YES.
7. Transmit the FDP trailer in a block by itself.
8. Transmit all blocks of data, followed by an EOT.

On switched (dial) lines, sending an EOT to the 3735 causes the lines to be disconnected. Redialing is necessary to continue operations on the line. It is necessary to send an EOT before the attributes of the line can be changed.

If no data is to be read from the 3735, or no data is to be sent to the 3735 other than the FDPs, you can code the FDP header and/or trailer in ASCII, thereby eliminating the need for step 4 or 6 and the EOT in step 3 or 5 of the previous procedure. This also eliminates the need to redial after sending EOT.

The data stream to and from the 3735 appears as follows:



#### Record Separators (Variable Length and Spanned Records)

Record separator characters for variable length and spanned records can be processed by the CCP on record mode input operations and on any mode of output operations (record, block, message). The BSCA terminal (other than the 3270) transmitting variable length or spanned records must be



defined as supporting record separators at assignment time (see *TERMATTR* statement) and must be defined as record mode for input operations.

The CCP automatically provides record separator characters at the end of each record to indicate the end of the record. The normal character provided is X'1E'; however, an alternate character may be chosen during CCP generation (see *\$EBSC* statement in *CCP System Reference Manual*). The record separator character is considered a device control character, not a line control character.

*Note:* When sending blocks of field descriptor programs to a 3735 terminal for which RECSEP=YES is specified in the // *TERMATTR* assignment statement, you must specify a block length less than 476, because a record separator character is automatically added to the end of your data before it is sent.

### Variable Length Records

When using variable length records, no record (including its record separator character) can be longer than the block size defined for the terminal. The record separator must be considered a data position when determining block sizes and/or line buffer sizes.

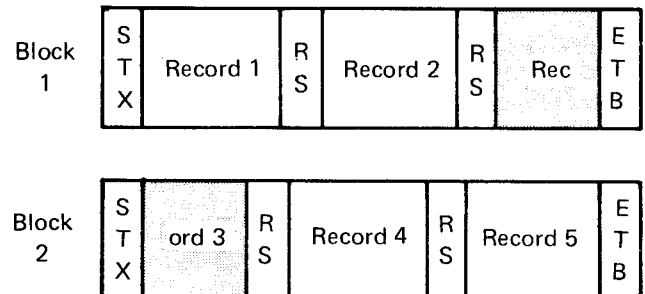
*Input:* When using variable length records, you must specify a maximum input length in your parameter list that is equal to or greater than the longest record you expect to receive. The record separator character is reflected as part of the effective input length in the parameter list.

For variable length, nonspanned input records, the last record separator character may be omitted. In this case, the ETB/ETX line control character suffices as a record separator, and is received in the user program record area in place of the normal record separator.

*Output:* The CCP automatically adds record separator characters after each record. Do not include the record separator in the output length field of your parameter list.

### Spanned Records

Spanned records can be used under the CCP only if record separators are used. A spanned record is not completely contained within a single block, but is continued in the next contiguous block, as shown in the following example of a data format (without ITB and without text transparency):



STX } BSCA control characters - see *Components Reference Manual*.  
 ETB }  
 RS - Record separator.

Record length, including the record separator, may not exceed block length.

### DEVICE CONTROL CHARACTERS

Device control characters are data characters that control certain aspects of terminal operation, such as carriage return for typewriter-like terminals and screen formatting for the 3270 terminal. Device control characters must be included in data that is transmitted to or from certain terminals. Certain device control characters can be automatically inserted into output data by the CCP:

- Carriage return and idle characters for control of MLTA typewriter terminals.
- 3270 screen format characters, when the Display Format Facility is used (see *Chapter 8: 3270 Display Format Facility*).

In all other cases, your program must provide the appropriate device control characters (such as tab characters and 3270 screen format control characters, when the 3270 Display Format Facility is not used). Therefore, before writing a program to communicate with any terminal, you must understand the device control required by the terminal and the physical characteristics and capabilities of the

terminal as described in the component description manual for the terminal (see *Appendix C: Bibliography*). See index entries for specific terminal types for additional information about the unique requirements of specific terminals.

## MLTA Typewriter Terminals

As part of the operation code in the parameter list, you can indicate whether you want the CCP to insert special device control characters into your data for the terminals with typewriter characteristics. The terminals considered to possess typewriter characteristics include the following:

- 2740, all models or equivalent (including System/7)
- 2741, all models or equivalent (including the Communicating Magnetic Card SELECTRIC® Typewriter)
- 1050 with typewriter component (1051/1053).

*Note:* If the 1050 multicomponent terminal number specifying all output components is specified, the 1050 is not treated as a typewriter device.

Unless you specify otherwise in your operation code, the CCP inserts a carriage return and idle characters at the beginning of an output record (New Line), if needed to assure the output starts on a new line, and at the end of an output record (End Line). By means of operation code modifiers, you can suppress New Line control characters (Not New Line), End Line control characters (Not End Line), or both sets of control characters (Not New Line and Not End Line).

You need not suppress New Line and End Line for non-typewriter terminals. The CCP ignores the indication in the operation codes and does not insert the typewriter control characters. Also, the CCP inserts New Line and End Line characters, unless suppressed, whether or not translation is specified.

### *New Line*

New Line causes a transmitted message to begin on a new line at the typewriter terminal. CCP does this by transmitting a carriage return and 15 idle characters before your data, if the typewriter is not already positioned at the beginning of a new line. The idle characters allow the typewriter time to reposition itself as a result of the carriage return. It is not always necessary to insert the typewriter control characters, since the typewriter may already be positioned at the beginning of a new line. CCP

attempts to keep track of the position of the typewriter and considers the typewriter to be positioned at a new line under the following conditions:

- The last operation was an input operation in which the last character received was carriage return.
- The last operation was an output operation which specified End Line.

If you specify New Line under either of these conditions, CCP does not insert the typewriter control characters. If your program is exchanging messages with a typewriter terminal, the terminal operator can decrease transmission time by keying a carriage return as the last character of his input to the program. Thus, when your program responds with a Put, CCP will not have to insert the additional control characters at the beginning of your output message.

*Note:* For a 2740 Model 2 terminal with the buffered receive feature, CCP sends the carriage return without idle characters, since this terminal allows for completion of the carriage return before continuing the printout.

### *End Line*

End Line causes the typewriter to be positioned at the beginning of a new line after receiving a message. The CCP does this by appending a carriage return and 15 idle characters to the end of your data.

### *Message Length Considerations*

You should not allow space for New Line and End Line control characters in your record area. When you provide an output message in your record area, the CCP must build the actual output data stream in the teleprocessing line buffer before transmission can occur. Any additional control characters added by the CCP must be in the line buffer along with your message. (The size of the line buffer is specified at assignment time, in the TERMATTR statement for BSCA and in the MLTALINE statement for MLTA, see *CCP System Reference Manual*.) Thus, if you want to transmit a 40-character message and you specify New Line and End Line, a 72-byte data stream is built in the line buffer by the CCP. If the data stream is larger than the line buffer, your message is truncated while all typewriter control characters remain appended to the message.

## BSCA Terminals

The CCP performs the following device control for BSCA terminals:

- The CCP inserts record separators for data transmission involving variable length or spanned records, if specified in the terminal attribute set associated with the terminals (see index entry *record separators*).
- If the *3270 Display Format Facility* (see index entry) is used with 3270 terminals, the CCP provides screen format control based on the descriptions of fields in the Display Format Specifications.

In communicating with other communications systems via the BSCA, you need not provide device control characters; however, the communications interface between the sending and receiving programs may require that you provide certain control data in your program that is understood by both programs, such as data delimiters and record identifiers. See Appendix A for additional considerations.

## LINE CONTROL CHARACTERS

Line control characters are the signals which control communication on either an MLTA or BSCA line. Line control characters are always removed from or added to data by the MLTA and BSCA communications IOCS facilities of the CCP. You need not provide space for line control characters in your record area and you need not manipulate line control characters in your program.

MLTA line control is described in the *MLTA RPQ Program Reference and Component Description Manual*, GC21-7560; BSCA line control is described in the *Components Reference Manual*.

## COMMUNICATING WITH MLTA TERMINALS

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

Programs communicate with MLTA terminals in a record-by-record manner; that is, each I/O operation in a program results in a record being sent or received. The program has effective control of the line only while a record is being sent or received. After the record has been sent or received, another program and/or terminal can use the line.

## COMMUNICATING WITH BSCA TERMINALS

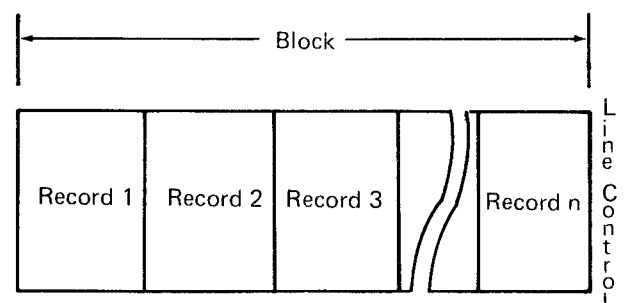
The term *BSCA terminals* refers to any of the terminals (including host and subhost systems) listed in *Chapter 1: Introduction* as supported by the binary synchronous communications adapter (BSCA) or binary synchronous communications controller (BSCC). BSCA terminals perform binary synchronous communications with the Model 10 Disk System, the Model 12, and the Model 15 through the CCP and the multiline/multipoint (MLMP) BSCA IOCS, which is included in the generated CCP if BSCA terminals are to be used. See *IBM System/3 Multiline/Multipoint Binary Synchronous Communications Reference Manual*, GC21-7573, for a complete description of the MLMP IOCS. Terminals on the BSCC lines perform binary synchronous communications with Model 15 (Program Number 5704-SC2) through CCP and the BSCC IOCS, which are included in the generated CCP if the BSCC lines are specified. Additional information regarding binary synchronous communications can be found in publications listed in *Appendix C: Bibliography*.

*Note:* BSCA conversational line control is not supported by the CCP.

### Blocking

When communicating with BSCA terminals, programs send or receive *blocks* of data. A block is the physical unit of data that is actually sent or received in each individual transmission on a BSCA line.

A block of data can be composed of one or more data records (Figure 2-3). Collecting records into blocks saves time when similar operations are performed on each record, since it is faster to send and receive more than one record at a time than to send and receive records individually.



1 block = n records

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

**Figure 2-3. Blocking in Binary Synchronous Communications**

## End of Transmission (EOT)

When communicating with a BSCA terminal, your program must perform Get operations until it receives an end-of-transmission (EOT) signal from the terminal (Figure 2-4) or until a transmission error occurs (resulting in a negative return code – See *Appendix E*). The EOT signal indicates the terminal has completed its current transmission. Likewise, your program must send an EOT signal when it has finished transmitting to a BSCA terminal (see *Put Message* under *BSCA Output Operations*), unless a transmission error has resulted in a negative return code.

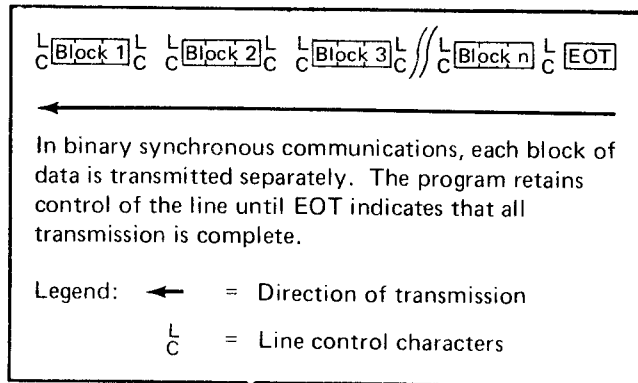
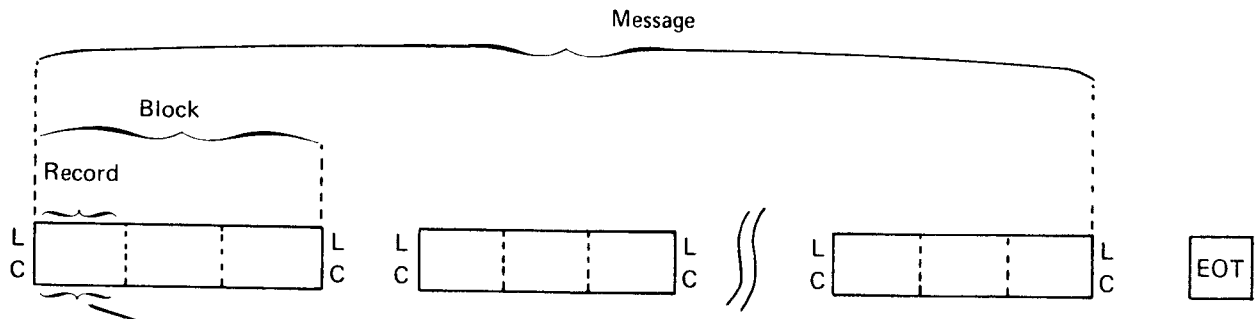


Figure 2-4. Data Transmission on BSCA Lines

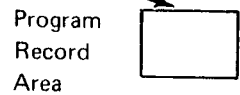
A BSCA line is dedicated to a program and a terminal once communication is initiated and is not freed for use by another program or terminal until EOT is transmitted (or a negative return code is received from an operation). Other terminals on a multipoint line may be allocated to other programs; however, a program can only be transmitting or receiving with one terminal at a time. A program that is receiving data from a BSCA terminal cannot transmit data to the terminal or communicate with any other terminal on that line until the terminal sends EOT (or a negative return code is received). Likewise, when a program is transmitting to a terminal on a BSCA line, that line cannot be used by any other program or terminal until either EOT is sent by the program or a negative return code is received by the program.

## BSCA Input Operations

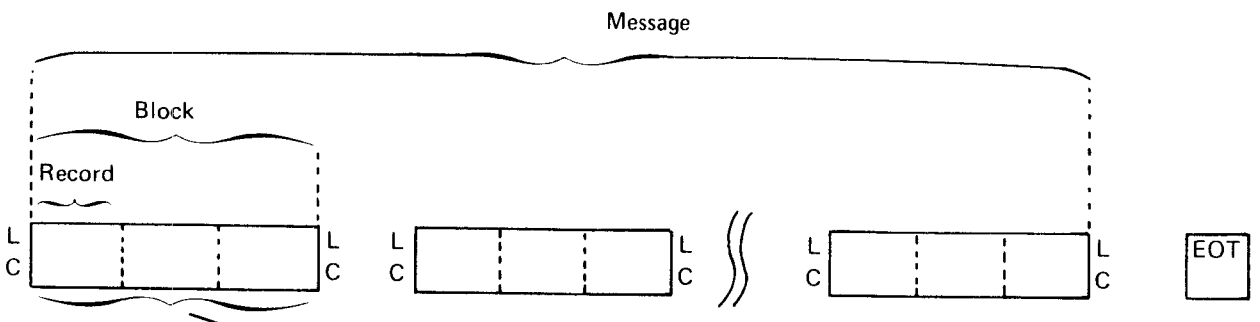
The CCP provides three levels (modes) of input operations for communication with BSCA terminals corresponding to three basic units of data: record mode, block mode, and message mode (Figure 2-5). The mode of input used by a program with a terminal is specified during the CCP assignment stage (see *TERMATTR* statement in *CCP System Reference Manual*). The actual input operations are used as described under *Operations* (see index entry).



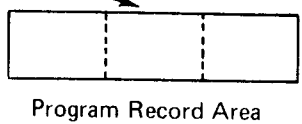
**RECORD  
MODE**



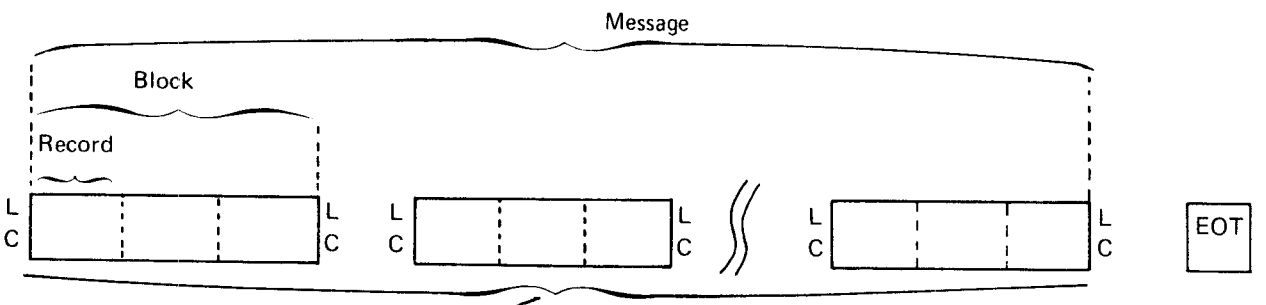
BSCA *record mode* input operations result in a single record being moved from the block in the input buffer to the program's record area (each block contains three records in this illustration). The length of the record area must be at least as great as the input record length plus the terminal name (and additional information, in certain RPG II operations).



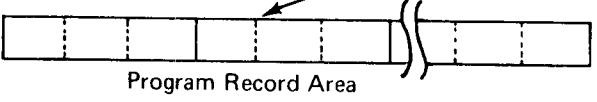
**BLOCK  
MODE**



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



**MESSAGE  
MODE**



BSCA *message mode* input operations result in an entire message (all blocks of input data preceding the EOT signal) being assembled in the input buffer and moved (without line control characters) to the program's record area. The record area must be as long as the longest message to be received plus the terminal name (and additional information, in certain RPG II and 3270 DFF operations), or the excess portion of the message is truncated.

Figure 2-5. BSCA Input Operation Modes

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

Message mode is always used with the 3270 Display Format Facility.

*Note:* Input modes do not affect output operations.

### BSCA Output Operations

The CCP provides three types of Put operations for use with BSCA terminals: *Put Record*, *Put Block*, and *Put Message*. Use of these operations in your program is not restricted by your program's mode of input operations (see *BSCA Input Operations*). See *Operations* later in this chapter, for complete descriptions of all Put operations.

Your program must always send an EOT when it has finished transmitting to a BSCA terminal, unless a transmission error occurs (resulting in a negative return code), when the CCP forces an EOT condition and terminates the operation. The CCP automatically sends the EOT after a Put Message operation and after the Put portion of a Put-Then-Get operation.

#### *Put Record*

The Put Record (or Put-No-Wait Record) operation causes a record to be sent to the terminal you specify in your program's record area. If block length equals record length, each Put Record operation results in a record being transmitted on the BSCA line. If each block contains several records (specified in the terminal attribute set), the block is transmitted when it does not have space for another record. Thus, your program may issue several Put Record operations before a block of data is actually transmitted. (The CCP will automatically issue a Put Block operation when a block is complete — see *Put Block*.) In order to send EOT following Put Record operations, your program must issue a Put Message operation (see *Put Message*).

In fixed-length record processing, CCP either pads a record with blanks or truncates a record if the record length does not equal the record length specified in the terminal attribute set (TERMATTR assignment statement). For example, if the attribute set defines the record length as 50, and you issue a Put Record with an output length of 40, CCP actually sends 50 characters; the last 10 characters are blank characters. Similarly, if you issue a Put Record with an output length of 60, a record of 50 characters is sent; the last 10 characters are truncated.

#### *Put Block*

The Put Block operation causes the current block in the output buffer to be transmitted, whether or not the block contains all the records it can hold. The next record Put by your program starts a new block. A Put Block operation may either be:

- Accompanied by the final data to be placed in the block before it is sent, or
- Issued with a record length of zero, which simply causes the block to be sent (if there is no data to be sent, the operation is ignored by the CCP).

When processing fixed-length records (see *Put Record*), if the Put Block operation is used to force transmission of a short block, a data length of zero is suggested. If data is to accompany the operation, it should be exactly one record length, as defined by the terminal attributes set, because the normal record truncating or padding is not performed by the Put Block operation.

#### *Put Message*

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

- Accompanied by the final data to be sent before EOT
- Issued with a message length of zero, which simply sends the EOT signal to the terminal.
- Program Request Under Format (PRUF), which indicates that the Put Message operation is transmitting a program request format out to the 3270 terminal.

When processing fixed-length records (see *Put Record*), if the Put Message operation is used to indicate the end of data, a data length of zero is suggested. If data is to accompany the operation, it should be exactly one record length, as defined by the terminal attributes set, because the normal record truncating or padding is not performed by the Put Message operation.

### *Put-Then-Get and Put-No-Wait Operations*

These operations have the same basic function as described under *Operations*. Put-Then-Get causes data (record, block, or message) to be transmitted to a specific terminal, followed by EOT and a Get operation for the terminal. The result of the Get portion of the operation depends on the mode of input specified at assignment time (TERMATTR statement); the result may be the equivalent of a Get Record, Get Block, or Get Message to the terminal. Put-No-Wait can be issued at the record, block or message level. A Put-No-Wait Record or Put-No-Wait Block are identical to Put Record or Put Block. On a Put-No-Wait Message, your program neither waits for nor receives a return code.

### **3284/3286 Printer Consideration**

When you are issuing operations to a 3284 or 3286 printer (components of the IBM 3270 Information Display System), the following situation should be considered: the user program issues an operation that starts the printer. Before the print operation is complete, the user program issues another operation to the printer, resulting in a "device busy" condition, for which CCP returns a -14 return code. If the busy printer option (BSYPRT-YES) is specified during generation, then code to handle a printer busy condition will be included in CCP. CCP will retain control until the printer has completed the operation; it will then return control to the application program (Model 12 and Model 15D only).

The user program should be written to recognize a -14 return code from the 3284 or 3286 printer and to take some appropriate action if busy printer was not specified during generation. Some possible courses of action are:

- Retry the print operation a number of times under control of a counter in the program. If the operation is not accepted after a number of retries, go on to other processing or inform the system operator. See index entry *Return Codes, Negative (DFF)* for special considerations when using the 3270 Display Format Facility.

This kind of action makes heavy use of the communication line and adversely affects the performance of other terminals using the line.

- Perform other operations in the program, such as disk I/O or console I/O, to allow some time for the device busy condition to clear. For example, sending a message to the system operator requesting a response is a way of delaying a retry of the printer operation without using system resources needed by other programs. The system operator can be asked, for example, to respond when the printer is free or after a specified period of time.
- (Model 15 only) After receiving a device busy return code, issue a CCP wait operation to cause your task to wait for a specified amount of time before retrying your put operation. If you are using assembler, the SSIT macro could also be used to set the interval timer to cause the task to wait.
- (Model 15 only) Use the RPG II operation code TIME to obtain the time of day. Repeat the TIME operation code until the desired time period has elapsed from the initial TIME operation; then reissue the Put operation to the printer. This method requires use of the DSM transient area and may, therefore, adversely affect system performance through heavy use of the transient area and disk access mechanism.
- Specify BSYPRT-YES at generation time. The busy printer support will ensure that any previous printer operation will be completed prior to scheduling the current operation. The -14 return code will not be returned by CCP for the "device busy" condition, and other user or system tasks will continue to execute. User programs that do not use DFF should ensure that the WCC "start print" bit is set on in the last block of a printer message only.

*Note:* Waiting for the "device busy" condition to clear by looping in your program (not issuing CCP operations) prohibits other user programs from executing during the loop, and is therefore not recommended.

## OPERATIONS

This section describes the valid teleprocessing operations that can be issued by application programs running under the CCP. Each description of an operation contains the following information:

*Purpose:* A brief description of the purpose of the operation.

*Operation Code(s):* Decimal value, hexadecimal value, and RPG II form of each variation of the operation code. A summary chart of operation code values is provided in Appendix D.

*Additional Requirements:* Information your program must provide in addition to the operation code and record area.

*Information Returned:* Information the CCP provides to your program as a result of the operation, including all positive return code values. Descriptions of all return codes and a summary chart of return codes by operation type are provided in Appendix E.

*Function and Use:* A detailed description of the results of the operation and rules, considerations, and recommendations for using the operation.

## Program Errors

The CCP checks every operation issued by an application program for validity before it performs the operation. Certain conditions are considered to be program logic errors, which result in termination of the application program. The CCP informs the system operator of the termination by printing a message that contains a *program termination code* identifying the error condition, the name of the program, and other information. The contents of this message and the meanings of the program termination codes are given in the *IBM System/3 Communications Control Program Messages Manual*, GC21-5170.

### 3270 Display Format Facility Operations

Requests for 3270 DFF operations are issued in the same manner as other requests for terminal operations; that is, each request is issued through a communications service subroutine and is accompanied by a parameter list and a record area. For certain 3270 DFF operations, however, you must supply additional information in the record area, besides the terminal name. For example, when the display format is written to a 3270, the name of the format is given in the record area following the terminal name.

Three operations are unique to 3270 DFF: Copy, Erase, and Put Override. These operations are described in *Chapter 8: 3270 Display Format Facility*. Considerations for using other CCP operations with 3270 DFF are summarized in that chapter and are also included in the descriptions of CCP operations in this chapter.



## GET

The purpose of the Get operation is to read a unit of data (record, block, or message) from a specific terminal into the record area.

### Operation Codes:

Hex	Dec	RPG II	Meaning
0001	1	bbbbA	Normal Get operation
0011	17	bbAA	Get operation with reverse interrupt (RVI) (See <i>Function and Use of Get</i> for an explanation of RVI.)

### Additional Requirements

- Set value of the Maximum Input Length field in the parameter list.
- Provide a symbolic terminal name (or blanks) in the record area.

### Information Returned

- Effective Length of Input Data, in parameter list.
- Input data, in record area.
- Count of outstanding Invite Inputs in the third field of the parameter list, if the 08 return code is received.
- Return Codes:
  - 0 Successful
  - 1 Data truncated
  - 2 EOT
  - 3 Data truncated and EOT
  - 5 Data pending (BSCA)
  - 7 3270 CLEAR (No AID is returned in the record area)
  - 8 Terminal no longer available (/RELEASE command was successfully entered by the terminal operator).
  - 9 Terminal offline
  - n Negative return codes (I/O errors - see explanations in Appendix E).

## Function and Use of Get

The Get operation reads a unit of data (record, block, or message) from a specific terminal and places the data in the record area. After issuing a Get operation, an application program waits for the CCP to complete the operation. The program resumes execution either after the CCP has moved the unit of data received from the specified terminal to the record area or after the CCP has terminated the operation because data transfer cannot succeed.

If the length of the input data actually received is greater than the maximum input length allowed, the data is truncated. If the data received is less than that specified, the CCP places blanks in the remainder of the record area.

The attributes of the data, which determine how the input data is handled, and the unit of data (record, block or message) are specified by the terminal attribute set currently associated with the terminal (see index entry *terminal attributes*). For an MLTA terminal, the unit of data is always record.

For BSCA terminals that acknowledge the receipt of reverse interrupts (RVI), the Get operation can be used to send an RVI to a terminal while receiving data from that terminal. *RVI* (see index entry) is generally used as a signal from a receiving device to a device that is transmitting to interrupt its transmission as soon as possible, usually because the receiving device wishes to transmit to the sending device.

### Get Operation with 3270 DFF

When you are using the 3270 DFF, you must issue a Put Message or Copy operation to format the display before you issue a Get operation to the 3270. See *Field Concepts* and *Record Concepts* in Chapter 8 for special requirements in handling input data. Also see index entry *Get operation, 3270 DFF*.

### Specifying the Terminal

You may, for a Get operation, specify either a defined symbolic terminal name or blanks in the record area. A defined terminal name must be either the name under which the referenced terminal was allocated to the program, or, if this is a *multicomponent terminal* (see index entry), a sub-terminal name subordinate to that name. A symbolic terminal name which is not assigned to a terminal cannot be used with this operation.

This operation must not be issued to the CONSOL (5471 Printer/Keyboard on Models 10 and 12; CRT/Keyboard on the Model 15).

A program can use a blank symbolic terminal name for this operation only if the program is a *single requesting terminal (SRT) program* (see index entry). A blank name references the terminal that requested the currently executing copy of the program. The CCP returns the name of the requesting terminal in the record area before returning control to the program. In the case of an SRT program, once the requesting terminal has been released (either by using its symbolic name or a blank name), the use of a blank terminal name in the record area is no longer valid.

#### *Considerations*

- The Get operation must not be issued as the initial data-transfer operation to a requesting terminal which entered data as part of the program request. The only valid operation which may be issued to such a terminal at that time is an Accept Input (see *Accept Input* operation).
- The Get operation must not be issued as the initial data-transfer operation in a program that was loaded by a chain task request. The only operation that can be issued in this situation is an Accept Input operation.
- This operation can be issued only to a terminal capable of transmitting data.
- A maximum input length greater than zero must be specified in the parameter list for this operation.
- The Get operation must not be issued to a terminal which has an Invite Input outstanding to it. Should it be necessary to read data from such a terminal, perform a Stop Invite Input operation. If the Stop Invite Input is successful, the Invite Input is cancelled and a Get may then be issued to the terminal. If the Stop Invite Input fails, then the operation is treated as a Get from the specified terminal.
- When communicating in record or block mode to a BSCA terminal that is on the same multipoint line with other BSCA terminals, it is recommended that, once input is received from the terminal, Get operations should be issued to that terminal until EOT is received (or the operation terminates with a negative return code). This procedure will free the line for use by other terminals as quickly as possible. An alternate procedure is to issue an Invite Input to the terminal, followed by Accept Input operations until the transmission is complete (EOT or a negative return code is received).
- In message mode, the EOT return code is never returned.

**PUT**

The purpose of the Put operation is to write a unit of data (record, block, message) to a specific terminal. Carriage returns are performed for MLTA typewriter terminals before and after writing the data (*New Line* and *End Line*, respectively), unless a modified form of the operation code is used to suppress carriage returns. New Line and End Line are ignored for BSCA operations.

The Put operation may specify that the unit of data is to be written as the last unit in the current block (Put Block); that is, an ETB (end of text block) signal is to be issued following the data, and the next unit of data is to begin a new block. The Put operation may also specify that the unit of data is the last unit to be put in the current transmission (Put Message); that is, the EOT (end of transmission) signal is to be issued following the data.

Put Block and Put Message are intended for use with BSCA terminals; however, these operations are valid for MLTA terminals and have the same effect as a Put operation without ETB or EOT (Put Record).

Put Message can be followed by additional input or output operations to the same terminal.

*Operation Codes*

Hex	Dec	RPG II	Meaning: Perform Put operation as follows:		
			New Line	End Line	Unit of Data
0002	2	␣␣␣B	yes	yes	Record
0102	258	␣A␣B	yes	no	
0202	514	␣B␣B	no	yes	
0302	770	␣C␣B	no	no	
0022	34	␣␣BB	yes	yes	Block
0062	98	␣␣FB	yes	yes	
0122	290	␣ABB	yes	no	
0222	546	␣BBB	no	yes	
0322	802	␣CBB	no	no	
0032	50	␣␣CB	yes	yes	Message
0072	114	␣␣GB	yes	yes	
0132	306	␣ACB	yes	no	
0232	562	␣BCB	no	yes	
0332	818	␣CCB	no	no	
0832	2098	␣HCB	yes	yes	
0872	2162	␣HGB	yes	yes	

### Additional Requirements

- Set value of output length field in the parameter list (see exception under RPG II for SPECIAL files).
- Provide a symbolic terminal name (or blanks) in the record area.
- See index entry *Put Overrides* for special requirements of that operation.

### Information Returned

- Return Codes:
  - 0 Successful (no exception conditions)
  - 1 Data truncated
  - 5 Data pending (BSCA)
  - 6 Terminal interrupt (MLTA) or RVI (BSCA)
  - 9 Terminal offline
  - n Negative return codes (I/O errors and device status conditions – see explanations in Appendix E).

### Function and Use of Put

The Put operation writes a unit of data to a terminal (record, block, or message). On Put operations to MLTA terminals, the application program waits for completion of the transmission to the terminal. For BSCA devices, the program resumes execution upon acceptance of the operation by the CCP, except for Put Message. On Put Message, control is not returned to the program until either the data is transmitted successfully and the EOT sent out, or until an error condition occurs.

### Put Operation with 3270 DFF

With 3270 DFF, you must use a Put Message to write the initial display format to the 3270 terminal. To override data at the terminal, use the Put Overrides operation. See index entry *DFF operations* for additional information and requirements.

### Specifying the Terminal

You can specify either a defined symbolic terminal name or blanks in the record area for a Put operation. A defined terminal name must be either the name under which the terminal was allocated to the program, or, if this is a multicomponent terminal, a sub-terminal name subordinate to that name. A symbolic terminal name which is not assigned to a terminal can not be used with this operation.

A Put operation can be issued to the CONSOL. The maximum length of output is:

Models 10 and 12 – 80  
Model 15 – 107

You can use a blank symbolic terminal name for this operation only if your program is an SRT Program (see index entry). A blank name references the terminal that requested the currently executing copy of the program. The CCP returns the name of the requesting terminal in the record area before returning control to the program. In the case of an SRT program, once the requesting terminal has been released (either by using its symbolic name or a blank name), the use of a blank terminal name in the record area is no longer valid.

### Considerations

- The Put operation must not be issued as the initial data-transfer operation to a requesting terminal which was specified to receive data as part of the program request. The only valid operation which may be issued to such a terminal at that time is an Accept Input (see *Accept Input* operation).
- The Put operation can be issued only to a terminal capable of receiving data.
- An output length greater than zero must be specified for this operation if transmitting to an MLTA terminal. However, a zero output length may be specified on Put Block operations and Put Message operations to BSCA terminals (except on the first such operation) to force sending of the current block or message.
- This operation must not be issued to a terminal which has an Invite Input outstanding to it.
- Put operations to the console, regardless of the form of Put operation code used, cause the data (message) to be sent to the console.
- PRUF Put operations issued to the console are invalid.
- When using block mode or record mode input with BSCA terminals, the following situation can occur and must be programmed for:
  1. The program issues an Accept Input; two terminals have outstanding Invite Inputs.
  2. The first terminal (T1) provides input data.
  3. The program issues input operations to T1 until EOT is received, then issues a Put to T1.

4. Prior to the Put to T1, the second terminal (T2) provides input data. Since T2 now has control of the BSCA line, the Put operation issued to T1 results in a 05 return code (data pending on the BSCA line).

- When transmitting fixed length data to a BSCA terminal using Put Record, any Put Block or Put Message operation should not have data specified unless the data is exactly one record as specified in the terminal attribute set. Truncation or padding of record data is not performed for Put Block or Put Message operations (see index entry *Put Record*).

## PUT-THEN-GET

The Put-Then-Get operation transmits a unit of data to a specific terminal and then reads data from the same terminal. Optionally, carriage returns are performed for MLTA typewriter terminals before and/or after writing the data (*New Line* and *End Line*, respectively). Put-Then-Get is more efficient than separate Put and Get operations.

Put-Then-Get is the *only* operation that can be used to read data from CONSOL except when an Accept Input is used to receive data entered with the program request.

### Operation Codes

Hex	Dec	RPG II	Meaning
0003	3	␣␣␣C	Put (Record) -Then-Get operation including New Line and End Line on Put.
0033	51	␣␣CC	Put (Message) -Then-Get
0103	259	␣A␣C	Put (Record) -Then-Get including New Line, but suppressing End Line.
0203	515	␣B␣C	Put (Record) -Then-Get including End Line, but suppressing New Line.
0303	771	␣C␣C	Put (Record) -Then-Get with neither New Line nor End Line.

### Notes:

1. New Line and End Line are ignored for BSCA terminals and the console.
2. This operation cannot be used with DFF terminals.

### Additional Requirements

- Set value of Output Length field in parameter list (see exception under RPG II for SPECIAL files).
- Set value of Maximum Input Length field in parameter list.
- Provide a symbolic terminal name (or blanks) in the record area.

### Information Returned

- Input data in the record area.
- Effective Input Length value in parameter list.
- Count of outstanding Invite Inputs in the third field of the parameter list, if the 08 return code is received.
- Return Codes:
  - 0 Successful (no exception conditions)
  - 1 Data Truncated - this return code indicates that input data was truncated
  - 2 EOT - applies to the Get, not returned to the program if Get was message mode (input mode is determined by *terminal attributes* – see index entry)
  - 3 Data Truncated and EOT - applies to the Get (MLTA terminals only)
  - 5 Data Pending - BSCA terminals only
  - 6 Terminal interrupt (MLTA) or RVI (BSCA)
  - 7 3270 CLEAR - applies to the Get. No AID is returned in the record area.
  - 8 Terminal no longer available - applies to the Get only (/RELEASE command was successfully entered by the terminal operator)
  - 9 Terminal offline - applies to the Put, since Get is not performed
  - n Negative return codes (I/O errors and device status conditions – see explanations in Appendix E) – see *Error Return Codes* under *Function and Use of Put-Then-Get* for additional information

## Function and Use of Put-Then-Get

This operation is a combination of a Put operation and a Get operation. First, the Put operation is issued to a specific terminal. Upon completion of the Put, a Get operation is issued to the same terminal. The application program resumes execution upon completion of the Get, when the input data resides in the record area. The same record area is used for both the Put and the Get (except with RPG II SPECIAL — see index entry *Put-Then-Get, RPG II SPECIAL*).

For BSCA terminals, the operation works as follows:

- For Put(Record)-Then-Get, the output data is padded or truncated according to normal record processing (see index entry *Put Record*). After the record is sent, EOT is sent, followed by the Get operation.
- For Put(Message)-Then-Get, either the output data is the only data sent (if this is message output only), or the output data is the last data sent (if the previous operation was a Put Record or Put Block operation). No record padding or truncating is performed by the Put(Message)-Then-Get operation (see index entry *Put Message*). After the output data is sent, EOT is sent, followed by the Get operation.

The mode of input (record, block, message) specified by the terminal attribute set which is currently associated with this terminal (see index entry *terminal attributes*) determines the unit of data received. For an MLTA terminal, the unit of data is always record.

### Specifying the Terminal

You can specify either a defined symbolic terminal name or blanks in the record area for Put-Then-Get. A defined terminal name must be either the name under which the terminal was allocated to the program or, if this is a multicomponent terminal, a sub-terminal name subordinate to that name. A symbolic terminal name which is not assigned to a terminal cannot be used with this operation.

The use of a blank symbolic terminal name is valid for this operation only if the program is an *SRT program* (see index entry). A blank name references the terminal that requested the currently executing copy of the program. The CCP returns the name of the requesting terminal in the record area before returning control to the program. In the case of an SRT program, once the requesting terminal has been released (either by using its symbolic name or a blank name), use of a blank terminal name is no longer valid.

Put-Then-Get is the only operation that can be used to get data from the system operator's console (CONSOL), except when an Accept Input is used to receive data entered with the program request. The maximum input and output lengths for this operation are:

Models 10 and 12 — 80

Model 15 — The maximum input that may be received is based on the maximum output length.

If the output length is 1 through 35, the maximum input received is truncated to 80. If the output length is 36 through 71, the input is truncated to 40. The output length may not exceed 71.

### Error Return Codes

When I/O errors occur on an input operation, the CCP sets the effective input length in the parameter list to zero and clears the record area to blanks. When errors occur during the execution of the Put-Then-Get operation, the operation is terminated immediately. Thus, if the error occurs on the Put portion of the operation, the CCP does not perform the Get, but returns control to your program. In order to determine whether an I/O error (negative return code) occurred on the Put or the Get, you can examine the effective input length in the parameter list. If the value of this field is the same as the output length value specified for the operation, then the I/O error occurred on the Put. However, if the value has been set to zero, the I/O error occurred on the Get.

### Considerations

- This operation must not be issued as the initial data-transfer operation to a requesting terminal which entered data as part of the program request. The only valid operation which may be issued to such a terminal at that time is an Accept Input (see *Accept Input Operation*).
- This operation can be issued only to a terminal capable of both transmitting and receiving data.
- A maximum input length greater than zero must be specified for this operation.
- An output length greater than zero must be specified for this operation if transmitting to an MLTA terminal or if this is the initial block being transmitted to a BSCA terminal.
- This operation must not be issued to a terminal which has an Invite Input outstanding to it.
- This operation must not be used with the 3270 DFF.

## PUT-NO-WAIT

The Put-No-Wait operation allows overlap of the output operation with continued program execution. Put-No-Wait writes a unit of data (record, block, or message) to a specific terminal. For MLTA operations and for BSCA message operations, your program resumes execution immediately upon acceptance of the operation by the CCP. Optionally, a carriage return is performed for MLTA typewriter terminals before and/or after writing the data (*New Line* and *End Line*, respectively). New Line and End Line are ignored for BSCA operations.

### Operation Codes

Hex	Dec	RPG II	Meaning: Perform Put-No-Wait as follows:		
			New Line	End Line	Unit of Data
0006	6	␣␣␣F	yes	yes	Record
0106	262	bA␣F	yes	no	
0206	518	␣B␣F	no	yes	
0306	774	␣C␣F	no	no	
0026	38	␣␣BF	yes	yes	Block
0126	294	␣ABF	yes	no	
0226	550	␣BBF	no	yes	
0326	806	␣CBF	no	no	
0036	54	␣␣CF	yes	yes	Message
0076	118	␣␣GF	yes	yes	
0136	310	␣ACF	yes	no	
0236	566	␣BCF	no	yes	
0336	822	␣CCF	no	no	Put overrides message used only with DFF
0836	2102	␣HCF	yes	yes	
0876	2166	␣HGF	yes	yes	

### *Additional Requirements*

- Set value of Output Length field in the parameter list (see exception in RPG II for SPECIAL files).
- Provide a symbolic terminal name (or blanks) in the record area.

### *Information Returned*

- Return codes (see explanations in Appendix E):
  - 0 Operation accepted by CCP
  - 5 Data pending (BSCA)

### **Function and Use of Put-No-Wait**

This operation causes the data in the Put record area to be Put to a specific terminal. The requesting program resumes execution upon acceptance of the operation by the CCP.

The Put-No-Wait operation may specify:

- A record is to be written in the current block (see index entry *blocking*).
- An EOB (end of block) signal is to be issued following the record and the next record is to begin a new block (Put Block).
- A record is the last to be Put in the current transmission; that is, the EOT (end of transmission) signal is to be issued following the data (Put Message).

Put-No-Wait with EOB, Put-No-Wait with EOT, and PRUF Put-No-Wait with EOT are intended for use with BSCA terminals; however, these operations are valid for MLTA terminals and have the same effect as a Put-No-Wait operation without EOB or EOT (Put Record), or without PRUF.

On Put-No-Wait (record) and Put-No-Wait (block) operations to a BSCA terminal, the requesting program does not resume execution until the specified terminal has gained control of the BSCA line.

### *Put-No-Wait Operation with 3270 DFF*

If Put-No-Wait is used with 3270 DFF to put a format on the display, the CCP changes the operation to a Put Message and your program does not regain control until the operation is complete.

### *Specifying the Terminal*

You can specify either a defined symbolic terminal name or blanks in the record area with a Put-No-Wait. A defined terminal name must be either the name under which the referenced terminal was allocated to the program, or, if this is a multicomponent terminal, a sub-terminal name subordinate to that name. A symbolic terminal name which is not assigned to a terminal cannot be used with this operation. Put-No-Wait can be issued to the console, but it is treated as a Put (with wait).

You can use a blank symbolic terminal name for this operation only if your program is an SRT program. A blank name references the terminal that requested the currently executing copy of the program. The CCP returns the name of the requesting terminal in the record area before returning control to the program. In the case of an SRT program, once the requesting terminal has been released (either by using symbolic name or a blank name), the use of a blank terminal name is no longer valid.

### *Considerations*

- The Put-No-Wait operation must not be issued as the initial data-transfer operation to a requesting terminal which entered data as part of the program request. The only valid operation which may be issued to such a terminal at that time is an Accept Input (see *Accept Input* operation).
- This operation can be issued only to a terminal capable of receiving data.
- An output length greater than zero must be specified for this operation if transmitting to an MLTA terminal or if this is the initial block being transmitted to a BSCA terminal.
- This operation must not be issued to a terminal which has an Invite Input outstanding to it.
- Since control is returned to the user program before completion of the data transfer, no indication as to the success or failure or the data transmission is returned to the user program.



## INVITE INPUT

The purpose of the Invite Input operation is to make a specific terminal eligible to send input data. The invited input is not made available to your program except as the result of a subsequent Accept Input operation (see next operation).

### Operation Code

Hex	Dec	RPG II	Meaning
0005	5	BBBE	Invite Input

### Additional Requirements

- Set value of Maximum Input Length in parameter list.
- Provide a symbolic terminal name (or blanks) in the record area.

### Information Returned

Return Codes:

- 0 Successful
- 9 Terminal offline

### Function and Use of Invite Input

This operation causes CCP to make a specific terminal eligible to transmit data to the application program. The program resumes execution after acceptance of the Invite Input operation by CCP.

More than one Invite Input may be outstanding at one time, but not to the same terminal. As each terminal completes transmission, the data is queued as input to the user program. An Accept Input operation (see next operation), causes the first completed input to be made available to the application program.

The attributes of the data and the unit of data (record, block, or message) for this operation are those specified in the terminal attribute set currently associated with the terminal. For an MLTA terminal, the unit of data is always record.

### Invite Input Operation with 3270 DFF

This operation must not be issued to a terminal under 3270 DFF until a Put Message or Copy operation has placed a format on the display.

### Specifying the Terminal

You can specify either a defined symbolic terminal name or blanks in the record area for the Invite Input operation. A defined terminal name must be either the name under which the terminal was allocated to the program, or, if this is a multicomponent terminal, a subterminal name subordinate to that name. A symbolic terminal name which is not assigned to a terminal cannot be used with this operation. This operation must not be issued to CONSOL.

You can use a blank symbolic terminal name for this operation only if your program is an SRT program. A blank name references the terminal which requested the currently executing copy of the program. The CCP returns the name of the requesting terminal in the record area before returning control to the program. In the case of an SRT program, once the requesting terminal has been released (either by using a symbolic terminal name or a blank name), the use of a blank terminal name is no longer valid.

### Considerations

- An Invite Input operation must not be issued as the initial data-transfer operation to a requesting terminal which entered data as part of the program request. The only valid operation which may be issued to such a terminal at that time is an Accept Input (see *Accept Input* operation).
- This operation can be issued only to a terminal capable of transmitting data.
- A maximum input length greater than zero must be specified for this operation.
- The maximum input length must not be greater than the size of the entire processing buffer put or get area (minus 4) defined in the current assignment set, or re-defined during Startup (four bytes are required for control information used by the main storage allocation routines).
- This operation must not be issued to a terminal which has an Invite Input outstanding to it.

## ACCEPT INPUT

The purpose of the Accept Input operation is to gain access to the earliest completed record from the terminals to which Invite Input operations were previously issued, and to obtain the data received with a program request or a chain task request. Invite Inputs to the other terminals remain in effect.

### Operation Code

Hex	Dec	RPG II	Meaning
0004	4	DDDD	Accept Input

### Additional Requirements

- Set value of Maximum Input Length field in the parameter list.

### Information Returned

- Input data in the record area.
- Effective length of Input Data, in the parameter list.
- Symbolic name of terminal from which data was received, in the record area.
- Count of outstanding Invite Inputs in the third field of the parameter list, if the 08 return code is received.
- Return Codes:
  - 0 Successful
  - 1 Data Truncated
  - 2 EOT or non-PRUF request to a PRUF program
  - 3 Data truncated and EOT
  - 4 Shutdown requested (no data returned)
  - 7 3270 CLEAR (No AID is returned in the record area)
  - 8 Terminal no longer available (/RELEASE command was successfully entered by the terminal operator)
  - 9 Terminal offline

- 14 An accept operation was successful. Data was received from a task chain operation. There is no terminal.
  - 15 Truncated data was received from a task chain operation. There is no terminal.
- n Negative return codes (I/O errors—see explanations in Appendix E).

### Function and Use of Accept Input

This operation makes data available to your program from one of three sources:

1. A terminal to which an Invite Input operation was previously issued by the program (*program* invite operation).
2. A terminal that presents data along with the program request. In this case, the Invite Input is done by CCP (*system* invite operation).
3. A CCP program that presents data along with a chain task request.

A successful Accept Input operation satisfies the previous Invite Input to the terminal from which data was received so that there is no longer an Invite Input outstanding to that terminal. However, Invite Inputs to other terminals remain in effect and can be satisfied by subsequent Accept Inputs.

Your program resumes execution after data from a terminal has been made available in the record area.

On an Accept Input operation, CCP ignores the data in the name field of the record area. However, CCP places the name of the terminal from which the data was received in this area before returning control to your program. On a chain task request operation, CCP places the name of the requesting program in the name field.

### Accept Input Operation with 3270 DFF

For systems without program request under format or when using the 3270 DFF, you must issue a Put Message or Copy operation to format the display before you use Invite Input and Accept Input operations with the 3270. See *Field Concepts* and *Record Concepts* in Chapter 8 for special requirements in handling input data. See also index entry *Accept Input Operation, 3270 DFF*.

DFF non-PRUF programs that use the AID byte position to set record identifying indicators on Accept Input operations must take into consideration that an Accept Input that satisfies a *system* invite operation does not return the AID byte to your record area. The first byte of the program request input data is in the AID byte field (position 15 if using SUBR92) and could duplicate a valid AID character, causing the corresponding record identifying indicator to be set on.

#### Accept Input Operation for Program Request Data

When input data is appended to the program request for non-PRUF programs, that data is not processed by the DFF, but is passed directly to the user program. In this case, the Accept Input must not be preceded by a Put Message or Copy operation to format the screen.

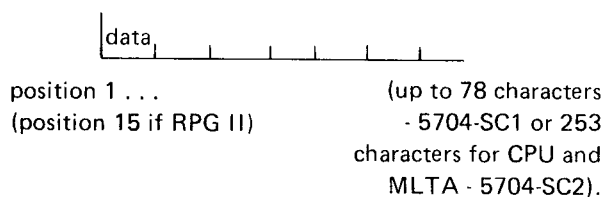
The input data is entered into the dynamic TP buffer area in a continuous string. If the data is from a 3270 terminal, the first 8 bytes are the control unit and device addresses, the AID byte, and the 3270 control characters. The length allocated to the dynamic TP buffer is determined by the PGMREQ value (maximum 80 bytes) of the SYSTEM assignment statement plus 8 bytes. When the input data is passed to the program record area, the 8 bytes and the program name information are also stripped from the input data, leaving a maximum of 78 bytes of actual program data. Any additional data in the 3270 buffer at the time of the program request is not sent to the program.

When input data is appended to the request for PRUF programs (identified by the PRUFLNG keyword given on the assignment PROGRAM statement), that data may or may not be processed by DFF. The maximum length of this data is specified by the PRUFLNG keyword. This data will be processed by DFF if the PRUF\$Z keyword was included in the assignment PROGRAM statement. The data stream returned to the program for PRUF non-DFF program requests differs from that for non-PRUF programs.

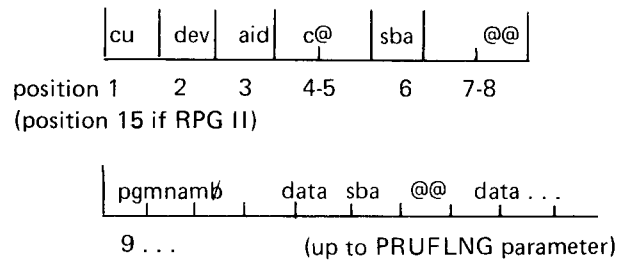
When input data is appended to the program request for CPU-to-CPU programs or MLTA programs, the maximum length is 253 characters (5704-SC2).

Program area examples of program request data:

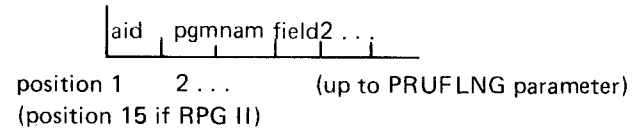
- Non-PRUF:



- PRUF non-DFF:



- PRUF DFF:



See Chapter 3 for an explanation of the 3270 control characters, aid, sba, etc.

If the PRUFLNG keyword was given in the PROGRAM statement for the requested program, but the last successful user Put was not a PRUF Put, the CCP will return a 02 return code to the PRUF program being requested. This indicates that non-PRUF data accompanies the program request.

Input data appended to the program request is passed to the user program in upper case even though it may have been entered in lower case at the 3270 terminal. Program Number 5704-SC2 allows data to be passed to the user program exactly as it is entered if the PF1 key is pressed instead of the ENTER key.

#### Accept Input Operation for Chain Task Requests with Data

When data is appended to a chain task request for a DFF program, the data is not processed by DFF but is sent directly to the input area in the program. This data does not contain any 3270 control characters.

When a program has been loaded via a chain task request with data, the program must issue an Accept Input operation before attempting any other CCP operation.

### Considerations

- You should specify a maximum input length in the parameter list (and a corresponding record area) large enough to accommodate the largest amount of data that can be received from an outstanding Invite Input, because, if several Invite Inputs are outstanding, you do not know which Invite Input will be the first to satisfy the Accept Input.
- Use caution when issuing Accept Input or Get operations after receiving a Shutdown return code (04). If the terminal operators do not key in data to satisfy the operations, the program remains in main storage until cancelled by the system operator. Stop Invite Input operations are recommended to cancel outstanding Invite Inputs and still permit processing of any data that may be received.
- An Accept Input can be issued only under the following conditions:
  1. There are one or more Invite Inputs outstanding for the program (including any implied Invite Inputs due to program requests with accompanying data).
  2. There are no outstanding Invite Inputs and
    - a. The program is defined as a never-ending program, and
    - b. The defined maximum number of requesting terminals that the program can handle concurrently is greater than the current number of requesting terminals the program is handling.
- An input length greater than zero must be specified for this operation.
- Accept Input is valid to the console only when issued for data with the program request. Any subsequent Invite Input to the console results in program termination.

### Shutdown Requested by Operator

A return code indicating that shutdown has been requested by the System Operator may be returned to this operation. In this case, the parameter list remains unchanged except for the return code field and the count of outstanding invites; no input data is received. If you still wish to have the operation performed, you must reissue the Accept Input. The shutdown-requested return code can only occur when there are no completed Invite Inputs to satisfy the

Accept Input. A shutdown-requested return code will be issued to an operation other than Shutdown Inquiry only once during the execution of the program.

Every program that uses Accept Input should check for the shutdown-requested return code.

*Note:* Only the Accept Input need be reissued if you still want to have the operation performed, because the Invite Input is still in effect.

### ACCEPT NO-WAIT INPUT (5704-SC2 ONLY)

The purpose of this operation is to allow the user to regain control even though no Invite Input operations have completed.

#### Operation Code

Hex	Dec	RPG II	Meaning
0044	68	DDDD	Accept No-Wait Input

#### Additional Requirements

- Set value of Maximum Input Length field in the parameter list.

#### Information Returned

The information returned is the same as that returned for Accept Input, except the following, which is unique for Accept No-Wait Input.

- Return Code:
  - 16 No invite input, chain task, or program requests have been completed.

### Function and Use of Accept No-Wait Input

Accept No-Wait Input performs the same functions that Accept Input performs, except that Accept No-Wait input checks for any completed invite input program request or task chain operations. If there are none, control is returned to the user with a 16 return code and the number of outstanding invites indicated in the third field of the parameter list.

If there are completed invites, program requests, task chains, CCP performs an accept input operation.

This operation can be issued even though no Invite Inputs have been issued. However, the defined maximum number of requesting terminals that the program can handle concurrently must be greater than the number of terminals currently being handled.

### Considerations

The considerations for this operation are the same as the considerations for Accept Input.

There are several return codes associated with accept input and accept no-wait input that imply no data. These codes are returned to the user so that the application program can be informed about these conditions. Some example return codes are:

- 04 Shutdown entered (only once).
- 07 A terminal operator has pressed the clear key on a 3277 terminal.
- 08 A terminal was released by the terminal operator.
- 09 A terminal was taken offline.

Any of these codes can be returned after an accept no-wait operation if there are no completed invites, program requests, or task chain operations. If none of the conditions exist, control is returned to the user with a return code of 16. In contrast, the accept input operation will wait until it has something to give the application program before returning control.

MRT application programs, which receive data via a program request from the console or task chain operation after processing data, do not know whether more requests are outstanding. Such an application program could do an accept no-wait operation if there was a request for the application. The application program would then have data and would process it. If there are no requests, control is returned with a return code of 16 and the count of the outstanding invites.

If the count of outstanding invites in the third field of the parameter list is not zero, the operator has these options:

1. Do an accept input and wait for data from the invited terminal.
2. Do a timer wait, then repeat the accept no-wait operation. If, after shutdown, none of the terminals are being used, a stop write operation may be issued to allow the program to go to end of job.
3. Drive a printer or some unit record device while waiting for a terminal.

*Note:* Looping on the accept no-wait operation may prohibit other user programs from executing; therefore, looping is not recommended.

### STOP INVITE INPUT (OR GET)

The purpose of this operation is to stop an Invite Input previously issued to a specific terminal and, if the Invite Input cannot be stopped, to accept (Get) the input. Stop Invite Input is used when some event has occurred in the program such that the program no longer wants input from the terminal.

#### Operation Code

Hex	Dec	RPG II	Meaning
0401	1025	bDøA	Stop Invite Input

#### Additional Requirements

- Provide a symbolic terminal name (or blanks) in the record area.
- Set value of the Maximum Input Length field in parameter list.

#### Information Returned

- Input data (if Invite Input is not stopped)
- Effective Input Length value in parameter list (if Invite Input is not stopped)
- Count of outstanding Invite Inputs in the third field of the parameter list, if the 08 return code is received.

- Return Codes:

- 0 Get successful (no exception conditions)
- 1 Data truncated
- 2 Get successful with EOT
- 3 Data truncated with EOT
- 7 3270 CLEAR (No AID is returned in the record area)
- 8 Terminal no longer available (/RELEASE command was successfully entered by the terminal operator)
- 9 Terminal offline
- 10 Stop Invite Input successful
- n Negative return codes (I/O errors and device status conditions – see explanations in Appendix E)

### Function and Use of Stop Invite Input

Stop Invite Input causes the CCP to attempt to cancel an Invite Input that has been previously issued to a specific terminal. If the Invite Input is stopped successfully, the terminal no longer has an Invite Input outstanding. However, if the Invite Input cannot be stopped, your program must be ready to handle any data received from the terminal as though your program issued a Get to the terminal. Thus, when requesting a Stop Invite Input, your program must present all the information needed for a Get operation. Your program resumes execution either when the Invite Input has been cancelled or when the Get has been completed.

This operation can only be issued to a terminal which has an Invite Input outstanding to it.

If the operation becomes a Get, the attributes of the data and the unit of data (record, block, message) for this operation are those specified in the terminal attribute set currently associated with this terminal (see index entry *terminal attributes*). For an MLTA terminal, the unit of data is always record.

### Specifying the Terminal

You can specify either a defined symbolic terminal name or blanks in the record area for this operation. A defined terminal name must be either the name under which the terminal was allocated to the program, or, if a multicomponent terminal, a sub-terminal name subordinate to that name. A symbolic terminal name which is not assigned to a terminal cannot be used with this operation. This operation must not be issued to the system operator console (CONSOL).

You may use a blank symbolic terminal name with this operation if your program is an SRT program. The blank name references the terminal which requested the currently-executing copy of the program. The CCP returns the name of the requesting terminal into the record area before returning control to your program. In the case of an SRT program, once the requesting terminal has been released (either by using a symbolic terminal name or a blank terminal name), the use of a blank name is no longer valid.

### Considerations for Using Stop Invite Input

- This operation must not be issued as the initial data-transfer operation to a requesting terminal which entered data as part of the program request. The only valid operation which may be issued to such a terminal at that time is an Accept Input.
- This operation can be issued only to a terminal capable of transmitting data.
- A maximum input length greater than zero must be specified for this operation.

### GET TERMINAL ATTRIBUTES

The purpose of this operation is to determine the attributes of a specified terminal.

#### Operation Code :

Hex	Dec	RPG II	Meaning
0008	8	ØØØH	Get Terminal Attributes

**Additional Requirements**

- Provide a symbolic terminal name (or blanks) in the record area.
- Set value of the maximum input length field in the parameter list.

**Information Returned**

- Effective input length value in the parameter list.
- Special information in positions 10-15 of the parameter list (see *Special Information Returned in the Parameter List* following).
- Terminal attributes, in the record area (see *Special Information Returned in the Parameter List* following).
- Return codes (see explanations in Appendix E):
  - 0 Success
  - 1 Data truncated

**Special Information Returned in the Parameter List**

Get Attributes returns the following information in the last three fields of the parameter list. This is the only situation in which you may want to reference information in these fields. These fields are not returned by the RPG II Get Attributes operation.

**Positions 10-11: (Work Area A)**

Contains the address of the Terminal Unit Block (TUB) for the terminal specified. The Terminal Unit Block is an internal CCP control block used to maintain control information for each terminal defined in the system with a MLTATERM or BSCATERM statement during the Assignment run.

**Positions 12-13: (Work Area B)**

**BSCA Terminal:** Contains the block length specified in the terminal attributes set associated with this terminal.

**MLTA Terminal:** Contains the record length specified in the terminal attributes set for the terminal.

**Positions 14-15: (Work Area C)**

**BSCA Terminal:** Contains the record length specified in the terminal attributes set associated with this terminal.

**MLTA Terminal:** Contains the record length specified in the terminal attributes set for the terminal.

**Information Returned in the Record Area**

As a result of a successful Get Attributes operation, the following information is returned in the first 21 positions of the data portion of the record area (all information is in EBCDIC).

**Position 1 - Allocation Status:** Position 1 contains a single character that indicates the following information about the terminal specified:

EBCDIC Character	Meaning
1	Allocated to this program. All the attribute data is provided.
2	Allocated to another program. All the attribute data is provided.
3	Not allocated to a program. All the attribute data is provided.
J K L	The specified name was a sub-terminal name. The master symbolic terminal name has been returned in record area. All attribute data applies to the master terminal. J = Master is allocated to this program. K = Master is allocated to another program. L = Master is not allocated to a program.
X	The specified symbolic terminal name is not assigned to a terminal. No other attribute data is provided.
Z	Terminal name not defined in the system.

**Position 2 - Terminal Class:** Position 2 contains a single character representing the class of terminal as follows:

EBCDIC Character	Terminal Class
0	CONSOL 5471 Printer/Keyboard (Models 10 and 12), CRT/Keyboard (Model 15)
1	MLTA terminal other than 1050
2	1050
3	3277M1 (480-character), 3284M1, or 3286M1
4	3277M2 (1920-character), 3284M2, or 3286M2
5	3275M1 (480-character), with or without 3284M3
6	3275M2 (1920-character), with or without 3284M3
7	3735
8	Another computer system (CPU) on a BSCA line.
9	3741 Data Station, 5231 Controller Model 2
A	Acquirable port <sup>1</sup>
B	Acquirable port <sup>1</sup>

**Position 3 - Line Number:** Position 3 contains the MLTA line or BSCA adapter number to which the terminal is attached. Determine whether MLTA or BSCA by checking position 2.

EBCDIC Character	Meaning
1	MLTA line 1/BSCA adapter 1
2	MLTA line 2/BSCA adapter 2
3	MLTA line 3
4	MLTA line 4
5	MLTA line 5
6	MLTA line 6
7	MLTA line 7
8	MLTA line 8

**Position 4 - Online:** Position 4 contains a single character that indicates whether the terminal is logically online or offline. This logical status of the terminal can be controlled by the system operator by use of the /VARY command (see *CCP System Operator's Guide*).

The value 'Y' indicates the terminal is online; the value 'N' indicates that the terminal is offline.

**Position 5 - Line Type:** Position 5 contains a character that indicates the type of line on which the terminal resides:

EBCDIC Character	Meaning
P	Point-to-Point
C	Control Station
M	Multi-point Tributary (BSCA only)
S	Switched
W	Control Station - Switched (MLTA only)

<sup>1</sup> For SIOC, see *IBM System/3 Model 15D Channel Connected Systems Program Reference and Logic Manual*, GC21-5199. For portline, see index entry *portline exception to get terminal attributes* under heading *portline application program interface to CCP*.



*Positions 6 - 21 - Terminal Attribute Set:* These 16 characters represent the 16 bit settings of the terminal attribute set currently associated with the terminal. (See index entry *terminal attributes*; also see the TERMATTR assignment statement in *CCP System Reference Manual*.)

Record Area Position	EBCDIC Character-Meaning	Corresponding Bit in Attribute Set
6	0 - Translate data 1 - Do not translate data	0
7	0 - Upper case translate 1 - Lower case translate	1
8	0 - Answer switched line 1 - Call switched line	2
9	0 - Manual switched line 1 - Auto switched	3
10	Reserved	4
11	Reserved	5
12	Reserved	6
13	0 - DFF not used for this terminal 1 - DFF used for this terminal	7
14	0 - Data format not record mode 1 - Data format is record mode	8
15	0 - Data format not block mode 1 - Data is block mode	9
16	0 - Data format not message mode 1 - Data format is message mode	10
17	0 - No ITB support 1 - Support ITB characters	11
18	0 - Non-transparency mode data <sup>1</sup> 1 - Transparency mode data	12
19	0 - Verify switched line exchange ID 1 - No exchange ID verification	13
20	0 - No spanned record support 1 - Support spanned records	14
21	0 - No variable length record support 1 - Support variable length records (record separators)	15

<sup>1</sup> Applies to data sent to the terminal (CCP output operations only).

## Functions and Use of Get Attributes

This operation is used to determine the attributes of any terminal defined by a BSCATERM or MLTATERM assignment statement (see CCP System Reference). It is not necessary that a terminal be allocated to a particular program for that program to issue a Get Attributes request to that terminal. The attributes of the specific terminal requested are available in the parameter list and the record area when the program resumes execution.

You might use a Get Attributes operation, for example, to determine which component of a 3270 is being used or whether or not the attribute set defined specifies DFF.

As a result of the Get Attribute operation, 21 characters of information are returned in the record area. If the maximum input length in the parameter list is less than 21, then only the number of characters indicated are returned. The effective input length field in the parameter list indicates the number of characters of information returned; the return code indicates data truncation occurs.

### Specifying the Terminal

Issuing this operation with a terminal name that is not defined causes CCP to return a Z in position 1 of the record area.

You may use a blank symbolic terminal name with this operation if your program is an SRT program. The blank name references the terminal which requested the currently-executing copy of the program. The CCP returns the name of the requesting terminal into the record area before returning control to the program.

In the case of an SRT program, once the requesting terminal has been released (either by using a symbolic terminal name or blanks), the use of a blank terminal name is no longer valid.

If a sub-terminal symbolic name is given with this operation, then the attributes of the master terminal are returned along with the name of the master terminal to which the sub-terminal name is subordinate.

### Considerations

- If an RPG II program uses position 15 of the input record area as the AID byte to set record identifying indicators, data from a previous operation can be lost or overwritten when a Get Attributes operation is performed.
- The PF 1, PF 2, and PF 3 keys return the characters 1, 2, and 3 respectively as the AID character in position 15 of the record area. A Get Attributes operation can also return the characters 1, 2, and 3 as valid allocation status information in the same position of the record area.

### ACQUIRE TERMINAL

This operation enables a program to request a specific terminal to be allocated to itself and to change the attributes of the terminal.

#### Operation Code :

Hex	Dec	RPG II	Meaning
0009	9	bbBI	Acquire Terminal—use the attribute set currently associated with the terminal (defined in TERMATTR statement)
0019	25	bbAI	Acquire Terminal—change the attributes to the set identified in third field of the parameter list
0029	41	bbBI	Acquire a command-mode, non-PRUF terminal (5704-SC2 only)
0039	57	bbCI	Acquire a command-mode non-PRUF terminal and change the attributes to the set identified in third field of parameter list (5704-SC2 only).

#### Additional Requirements

- Provide a symbolic terminal name in the record area.
- Provide an attributes identifier in the parameter list, if attributes are to be set by the operation.

## Information Returned

### Return Codes:

- 0 Successful
- 11 Acquire failed

## Function and Use of Acquire Terminal

Application programs can issue teleprocessing operations only to those terminals that are allocated to them. One way to allocate terminals is to specify the terminals in the PROGRAM assignment statement (see *CCP System Reference Manual*). However, there may be times when a program wishes to have a terminal dynamically allocated to it. This can be done with the Acquire Terminal operation.

The Acquire Terminal operation causes CCP to attempt to allocate a specific terminal to a program. If the terminal is online, not already allocated to a program, and not in command mode (signed on), it is eligible for allocation. If operating under control of 5704-SC2, an Acquire Terminal operation can acquire a command-mode, non-PRUF terminal. A non-PRUF terminal is a 3277 or 3275 that is not formatted with a PRUF display.

A PRUF terminal remains a PRUF terminal until any non-PRUF Put is issued to that terminal by the user or until the clear key is pressed. If PRUFOF-YES is specified at assignment time, the PRUF condition will be terminated after an Accept Input operation is sent to that terminal. This means a non-PRUF Put will not be required to put the terminal in non-PRUF mode.

Along with this operation you may identify a specific set of operational attributes to apply to that terminal while allocated to your program. If an Acquire Terminal operation is to set terminal attributes, then the identifier value of the terminal attribute set (as given in the TERMATTR assignment statement) must be given in the third field of the parameter list (in decimal for RPG II; in hexadecimal for other languages). CCP checks the attribute set specified against the terminal type for validity.

The Acquire Terminal operation is valid if the terminal is already allocated to your program only if the Acquire Terminal with Set Attributes modifier is issued to change the attribute set. (On a BSCA line, the last transmission must have completed a message; that is, EOT must have been sent or received.)

Your program resumes execution upon the completion of the allocation or upon recognizing that the terminal is not available for allocation.

## Specifying the Terminal

For this operation, you must specify a defined symbolic terminal name in the record area. The following names may *not* be specified for the operation:

- The name of a terminal already allocated to the program (except the requesting terminal, immediately after the first successful Accept Input).
- A symbolic sub-terminal name.
- A symbolic terminal name that is not assigned to a terminal.
- CONSOL
- Blank terminal name.

## Considerations

When both DFF and non-DFF programs can be requested from the same 3270 terminal, at least two TERMATTR statements (DFF3270-YES and DFF3270-NO) must be defined in the assignment set for these programs. The first terminal attributes set defined in the ATTRID parameter of the BSCATERM statement is the default attribute set for the terminal. A program that uses the requesting terminal in a manner other than that defined by the default attribute set must perform an Acquire Terminal operation with the appropriate attributes identifier to modify the terminal attributes. For further information about acquiring switched BSCA or MLTA devices, see *Program-Selected Terminals and Switched Lines* in Chapter 3.

### Generic Acquire Terminals (5704-SC2 Only)

The purpose of this operation code is to find an available command-mode, non-PRUF terminal or port by using a generic key given in the record. The terminal or port is found by matching the generic key to the terminal or port name that begins with the same characters.

#### Operation Code

Hex	Dec	RPG II	Meaning
0069	0105	bbFI	Generic Acquire
0079	0121	bbGI	Generic Acquire and Set Attributes

#### Additional Requirements

- Provide enough of the terminal name or port name needed to select a terminal within a defined group.
- Provide an attribute identifier in the parameter list, if acquiring a terminal from a group of terminals (for example, that begin with *PRNT*), and if attributes are to be set by the operation.
- Attribute may not be set for the port terminal.
- After shutdown is entered and accepted port terminals may not be acquired.

#### Return Codes

- 00 Successful. Name of terminal or port acquired is given in the record as the terminal name.
- 04 Acquire failed. Port terminals may not be acquired after shutdown is entered and accepted.
- 11 Acquire failed. None of the terminals or ports in the group selected are available to be acquired at this time.

### Example

In the first six bytes of the record, the user program places the first characters (see note) of the name of the terminals or port available to be acquired. Then the user program issues the GENERIC ACQUIRE operation code. The first terminal or port that is available, that may be acquired, and whose name begins with the characters contained in the record is then acquired. The name of the terminal or port which is acquired is returned to the user program in the first six bytes of the record.

If the user program was looking for an available port it would use the name given on the PORTPRFX parameter of the PORTLINE statement.

```
// PORTLINE MODE-P , . . . , PORTPRFX-PORT
```

```
Generates: PORT01
           PORT02
           PORT03
           . . .
           . . .
```

And the assignment set also has a terminal defined as follows:

```
// TERMNAME NAME-TERM01
// TERMNAME NAME-TERM02 } Terminals
// TERMNAME NAME- . . .
// TERMNAME NAME-PRNT01
// TERMNAME NAME-PRNT02 } Printers
// TERMNAME NAME-PRNT03
```

When using a TERMNAME of PORTbb, any available and acquirable port would be acquired (PORT01, PORT02, PORT03, . . .).

*Note:* The number of characters used depends on how many are needed to select the group of terminals or ports wanted.

In the above example only PObbbb was needed to acquire any port. If the user program had several printers, it could name the printers PRNT01, PRNT02, and acquire any available printer that was not being using by another program by specifying PRNTbb.

## Port Command Request (5704-SC2 Only)

The purpose of this operation code is for issuing commands and program requests to the command processor via an acquirable port.

### Operation Code

Hex	Dec	RPG II	Meaning
003A	0058	bbCK	Port Command Requests

### Additional Requirements

- All commands must be issued with the PCR operation code.
- Commands allowed are: /MSG, /FILES, and program requests.
- Commands may be issued only to an acquirable port.
- Set the value of the output length field in the parameter list (see exception under RPG II for SPECIAL files). The output length does not include the 6 character port name.
- A symbolic port terminal name must be provided in the record.
- The port prefix may be used as the port name if a port was required on the PROGRAM statement at assignment using the port prefix.

### Return Codes

- 0 Successful (no exception conditions)
- 1 Data truncated
- 5 Data pending (program must do a get)
- n Negative return codes (I/O errors and device status conditions—see explanations in Appendix E).

### Function and Use of Port Command Request

The PCR (port command request) op code operates functionally the same as the CCP PUT, MSG operation code. The command issued by the PCR operation code is given to the command processor of the other system. If a paired port in the other system was owned by a program, when the PCR is issued, then the paired port is released from the program and returned to the command processor. The program that would have received the command is instead given a return code of 08 (port released) on the program's next read operation or an 05 (data pending do a read) on the program's next put operation.

### Specifying the Terminal

You can specify either a defined symbolic port terminal name or the PORTPRFX name if it is a required port. A symbolic terminal name cannot be used with this operation. A PCR operation cannot be issued to the CONSOLE.

### Considerations

- The PCR operation must not be issued as the initial data-transfer operation to a requesting terminal which entered data as part of the program request. The only valid command that may be issued to such a terminal at that time is an Accept Input command.
- A blank terminal name may not be used since the requestor of a SRT (single requesting terminal) program by a port terminal is a nonacquirable port.
- An output length greater than zero must be specified for this operation.
- This operation must not be issued to a terminal that has an outstanding invite input operation.

## TASK CHAIN REQUEST (5704-SC2 ONLY)

This operation allows a program to request a second program that is to execute as an unrelated task. Data can optionally be passed with the request.

### Operation Code

Hex	Dec	RPG II	Meaning
002A	0042	00BK	Set up a program request from this program.

### Additional Requirements

- Provide the name of the requested program in the name field of the record area.
- Specify either the output length or 0 (if no data with the chain request) in the parameter list (see *Performing CCP Operations with Special* in Chapter 6).

### Information Returned

#### Return Codes:

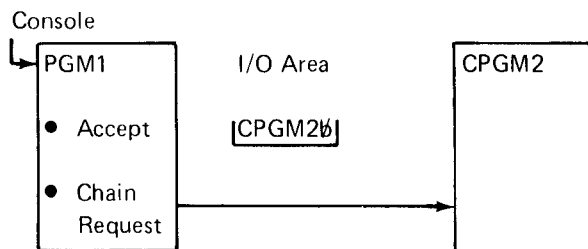
- |    |   |
|----|---|
| 0  | Successful operation (Chain Task Request accepted)                      |
| 12 | Request rejected (maximum number of chain task requests already queued) |
| 13 | Request rejected (insufficient TP buffer available)                     |

## Function and Use of the Task Chain Request

When successfully executed by a CCP program, the Task Chain Request operation causes a second program to be loaded and executed. If the resources (files, terminals, or storage) required by the second program are not available, the second program is automatically queued. If data is presented with the Task Chain Request and PGMDATA-YES was specified on the PROGRAM assignment statement, the data is passed to the second program when that program issues its first Accept Input operation.

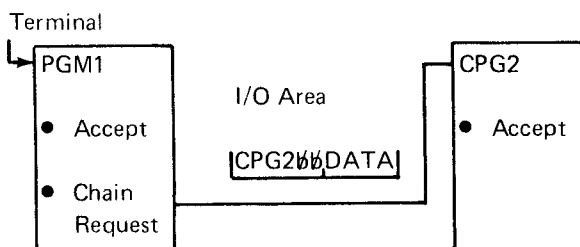
Following are some examples of Task Chain Request operations (for additional information, see *IBM System/3 Communications Control Program System Design Guide*, GC21-5165):

*Task Chaining from a Program Loaded from the Console:* PGM1 is loaded from the system operator's console by entering the program request: PGM10CPGM2. PGM1 issues an Accept Input operation and receives the data CPGM2, and sets up the chain request.



*Note:* The output length is zero in this example because no data is passed between programs.

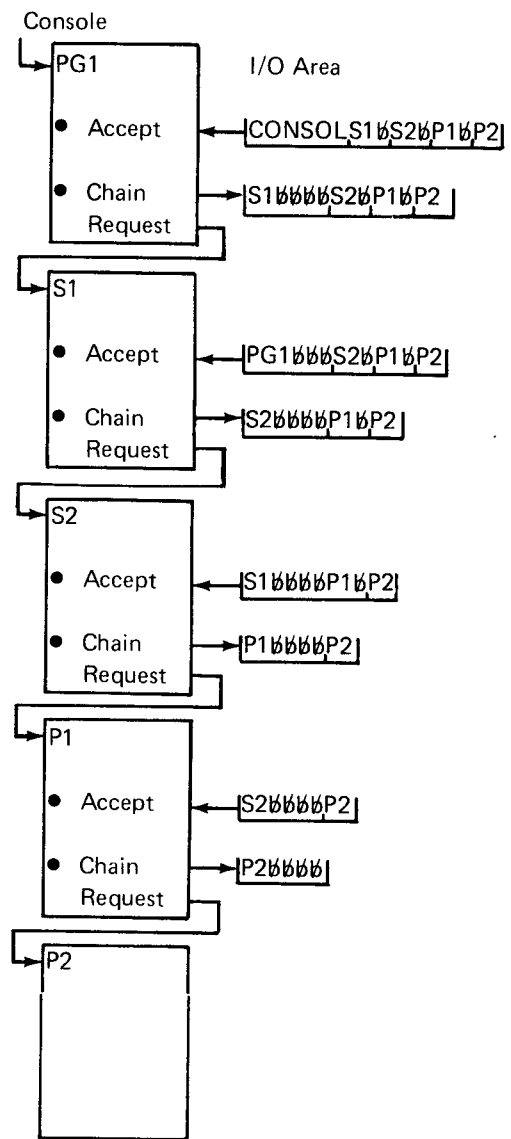
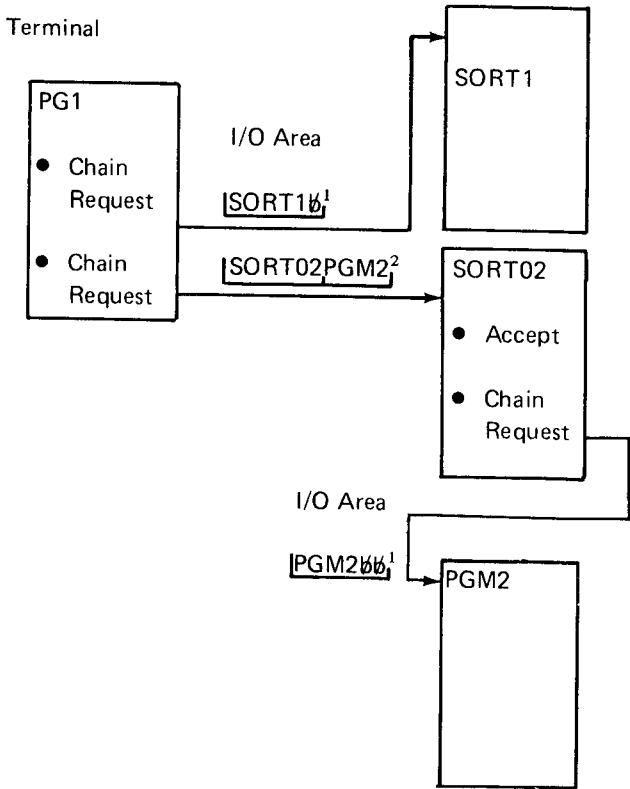
*Task Chaining with Data From A Program Loaded From a Terminal:* PGM1 is loaded from a terminal by entering the program request: PGM10CPG200DATA. When PGM1 issues an Accept Input operation, the program receives CPG200DATA as data, sets up and issues the chain request for CPG2, and passes the characters DATA to the requested program. When CPG2 is loaded, it issues an Accept Input operation to receive the characters DATA. If both accept operations are successful, PGM1 receives a 00 return code, CPG2 receives a +14 return code.



*Note:* In this example, the output length is set to four.

*Multiple Task Chaining Involving Sort Programs:* PG1 is requested from a terminal in this example. No program data accompanies this program request. When PG1 receives control, it sets up and issues the chain request for SORT1. SORT1 is then loaded and begins executing. PG1 then sets up and issues the chain request for SORT02. When SORT02 is loaded, it issues an accept operation to receive the data passed from PG1, performs the sort function, and sets up and issues a chain request for PGM2. PGM2 is then loaded and executed.

*Note:* It is possible for PG1, SORT1, SORT02, and PGM2 to be executing at the same time, provided the programs do not have conflicting resource requirements. It is not necessary that one program end before another can begin executing.



<sup>1</sup>Output length is zero.

<sup>2</sup>Output length is four.

**Note:** CCP/Disk Sort issues a chain request if chaining was specified during the generation of the sort program.

**Multiple Chain Requests Involving both Sort and Nonsort Programs:** PG1 is loaded from the console by entering the program request: PG1 S1 S2 P1 P2. PG1 issues an accept operation to receive the program data, and sets up the chain request. Sort programs S1 and S2, and nonsort programs operate similarly to the preceding examples.

**Note:** PG1 and P1 must be set up by the programmer to build the chain requests from the input data and to move the program name into the name field. S1 and S2, the sort programs, are set up to use up to 6 characters of input or until the first blank (delimiter) is detected, as the name of the program for a chain request. The remaining characters are passed as data.

*Considerations*

- Symbolic file names that are valid for the requesting program are not valid for the requested program because the /FILE commands and SYMFILE statements do not resolve file name references.
- The MAXCHAIN parameter of the SYSTEM assignment statement must contain a value greater than zero for programs that use the Task Chain Request operation. If a Task Chain Request operation is made and MAXCHAIN=0 was specified in the assignment set, the program is terminated with a 3F termination code. The MAXCHAIN parameter has a default value of zero.
- Once a Task Chain Request is accepted by CCP and a return code returned to the requesting program, any further diagnostics and messages for that request are issued to the system operator's console.
- CCP allocation does not normally queue requests for any unit record devices. If a Task Chain Request operation has been successfully executed, but the chained task requires a unit record device that is not available, the chained task is rejected and a message is issued to the system operator's console. An exception to this is the Model 15 CCP; it will queue an unavailable printer if the printer is spooled or if the printer is permanently allocated to CCP and not in use by an NEP or MRT.
- If a program that issued a Task Chain Request operation abnormally terminates, the program requested by the task request is allowed to execute.
- If a CCP user program does not terminate immediately after a Task Chain Request operation and instead, invokes a program that requires resources made unavailable by the invoking program (such as a NOSHR disk file), the invoked program is queued until the invoking task releases the resources or terminates.
- When a series of task chained programs are active when SHUTDOWN is entered, the chained programs continue executing to normal end of job.

- A potential lock-up situation exists when using Task Chain Requests. Each Task Chain Request requires TPBUFFER to hold the requested program name and any data to be passed to the requested program. Also required is one of the task chaining control blocks as specified on the MAXCHAIN parameter of the SYSTEM assignment statement. Either of these resources can become exhausted. When this happens, another Task Chain Request will be rejected with a return code of 12 or 13. A program receiving such a return code should not loop on the Task Chain Request. If a program loops on a Task Chain Request, lower priority CCP user tasks that are being task chained are prevented from doing an ACCEPT input which frees the above mentioned resources. A preferred action to take when receiving a 12 or 13 return code is to issue a WAIT operation and then retry the Task Chain Request or issue an ACCEPT if it is possible that the program has Task Chain Request data queued.

**RELEASE TERMINAL**

The Release Terminal operation enables an application program to give up control of a specific terminal in order to make the terminal available to the rest of the CCP system.

*Operation Codes*

Hex	Dec	RPG II	Meaning
000A	10	␣␣␣K	Release terminal and the Communication line.
001A	26	␣␣AK	Release terminal - keep line allocated to the program (switched lines only).

*Additional Requirements*

- Provide a symbolic terminal name (or blanks) in the record area.

*Information Returned*

- Count of Invite Inputs outstanding for this program, in the third field of the parameter list.



## Function and Use of Release Terminal

When issued to a terminal on a nonswitched communication line, a Release Terminal operation causes CCP to release the specified terminal from the program, making it available to other programs (if it was not the requesting terminal) or freeing it to enter commands (if it was the requesting terminal). When issued to a terminal on a switched line, Release Terminal can be issued so that it releases the terminal and either:

1. Keeps the line allocated to the program, or
2. Releases the line, making it available to other programs.

Keeping the line breaks the connection to the terminal, but allows the program to acquire any terminal on the line, since no other application program can gain access to the line.

After completion of the Release Terminal operation the third field of the parameter list contains a count of the outstanding Invite Input operations for the program, including any Invite Input operations issued by the CCP because of data accompanying a program request.

A typical use for the Release Terminal operation is to release a requesting terminal from a *multiple requesting terminal (MRT) program* (see index entry) after the program completes its work for the terminal. When an MRT program is handling its maximum number of requesting terminals, it must release a terminal before it can accept a request from a new terminal.

Your program should issue this operation when it no longer needs a specific terminal, so that the terminal can be available to other programs. Possibly the terminal has been yielding negative return codes because of a malfunction and the program can make no productive use of the terminal. The terminal cannot be placed in offline status by the system operator while it is allocated to a program. Thus, only after an application program releases the terminal can it be placed offline.

## Specifying the Terminal

For this operation you may specify either a defined symbolic terminal name or blanks in the record area. A defined terminal name *must* be the name under which the referenced terminal was allocated to the program.

The use of a blank symbolic terminal name is valid for this operation only if the program is an SRT program.

The blank name references the terminal which requested the currently executing copy of the program.

None of the following may be specified as the symbolic terminal name:

- An undefined name (a name not assigned to a terminal)
- The defined name of a terminal not allocated to this program
- A defined name which references a terminal allocated to this program, but which is not the name under which that terminal was allocated to the program
- A sub-terminal name
- CONSOL—the console is automatically released by the CCP at the beginning of execution of a program requested from it.

## Considerations

- This operation must not be issued as the initial data-transfer operation to a requesting terminal which entered data as part of the program request. The only valid operation which may be issued to such a terminal at that time is an Accept Input.
- This operation must not be issued to a terminal which has an Invite Input outstanding to it.
- In order to issue Release Terminal on a BSCA line, the line must be at EOT.
- (*Model 10 and Model 12 CCPs*) SRT programs using symbolic files must open the symbolic file before releasing the requesting terminal. In RPG II and FORTRAN, all files are opened before any I/O operations can be performed by the program; however, COBOL and Basic Assembler programs must explicitly open all symbolic files prior to issuing a Release Terminal operation to the requesting terminal.

## RELEASE AND TASK CHAIN (5704-SC2 Only)

The Release and Task Chain operation enables an application program to give up control of a specific terminal and, at the same time, issue a program request to the released terminal. The released terminal is then allocated to the requested program. Any symbolic file references associated with the released terminal may be used by the requested program.

### Operation Code

Hex	Dec	RPGII	Meaning
004A	74	b̄bDK	Release terminal and the communication line and issue the program request given in the record area.

### Additional Requirements

- Provide a symbolic terminal name (or blanks) in the record area.
- Place the program name to be requested in the record area after the terminal name.
- If data is to accompany the request, place a blank after the program name that is to be followed by the data. The format is the same as that which the terminal operator would type if the request were to come from the terminal itself.
- Specify the total output length (program name plus the blank and data if data is supplied). Do not include the terminal name. An output length of zero is not allowed. The maximum length is limited by the size of the TP buffer (add 14 to the length if using SUBR92).

### Information Returned

#### Return Codes:

- 0 Successful operation (terminal was released and the program request has been accepted.)
- 13 Rejected (terminal was not released because sufficient space was not available in the TP buffer for the program request.)

If the operation is accepted (return code = 0), the count of outstanding invite inputs is returned in the third field of the parameter list.

*Note:* This is the same field that contained the PUT length.

## Function and Use of the Release and Task Chain

When successfully executed by a CCP program, CCP will release the terminal from the program and cause a second program to be loaded on and executed by the terminal. If the resources (files, terminals, or storage) required by the second program are not available, the second program is automatically queued. The terminal does not have to be in the /QUEUE mode. If data is presented along with the program name and PGMDATA=YES was specified on the PROGRAM statement of the second program, the data is passed to the second program when that program issues its first Accept Input operation. The terminal released from the first program is now allocated to the second program. The terminal being released is placed in a non-PRUF mode.

The Release and Task Chain operation may be used when a terminal or symbolic file reference is needed by a subsequent program in a chaining sequence. See *Release and Task Chain Involving Sort Programs* later in this chapter.

### Specifying the Terminal

For this operation, you may specify either a defined symbolic terminal name or blanks in the record area. A defined terminal name *must* be the name under which the referenced terminal was allocated to the program.

The use of a blank symbolic terminal name is valid for this operation only if the program is an SRT program.

The blank name references the terminal that requested the currently executing copy of the program.

The referenced terminal must be a command capable terminal.

None of the following may be specified as the symbolic terminal name:

- An undefined name (a name not assigned to a terminal)
- The defined name of a terminal that is not allocated to this program
- A defined name that references a terminal that is allocated to this program, but the name is not the name under which that terminal was allocated to the program
- A terminal that is not command capable
- A sub-terminal name
- CONSOLE—The console is automatically released by CCP at the beginning of the execution of a program requested from the console

### Considerations

- This operation must not be issued as the initial data-transfer operation to a requesting terminal that entered data as part of the program request. The only valid operation that may be issued to such a terminal at that time is an Accept Input operation.
- This operation must not be issued to a terminal that has an outstanding Invite Input operation.
- If the terminal is on a BSCA or BSCC line, the line must be at EOT.
- Once the Release and Task Chain operation is accepted by CCP and a return code is returned to the requesting program, any further diagnostics and messages for that request are issued to the released terminal.
- CCP allocation does not normally queue requests for any unit record devices. If a Release and Task Chain operation has been successfully executed, but the requested program requires a unit record device that is not available, the requested program is rejected and a message is issued to the terminal released by the operation. An exception to this is the Model 15 CCP; it will queue an unavailable printer if the printer is spooled or if it is permanently allocated to CCP and not in use by an NEP or MRT.
- If a program that issues a successful Release and Task Chain operation later abnormally terminates, the program requested by the operation is allowed to execute.
- If a CCP user program does not terminate immediately after a Release and Task Chain operation and, instead, invokes a program that requires resources (such as a NOSHR disk file) made unavailable by the invoking program, the invoking program is queued until the invoking task either releases the resources or terminates.
- The Release and Task Chain operation issues a program request for the terminal operator, and the program request is processed as though the terminal operator had entered it. If the terminal is not signed on or if the symbolic file references are undefined, the program request will be rejected, and a message will be sent to the terminal.

- When a series of programs are requested by the Release and Task Chain operation while SHUTDOWN is entered, the requested programs continue executing to normal end of job.
- When SHUTDOWN is entered and a command mode terminal is owned by CCP and not by an application, the terminal is automatically signed off. If an application then acquires the terminal and issues a Release and Task Chain operation, the program requested by the operation is rejected because the terminal has been signed off.

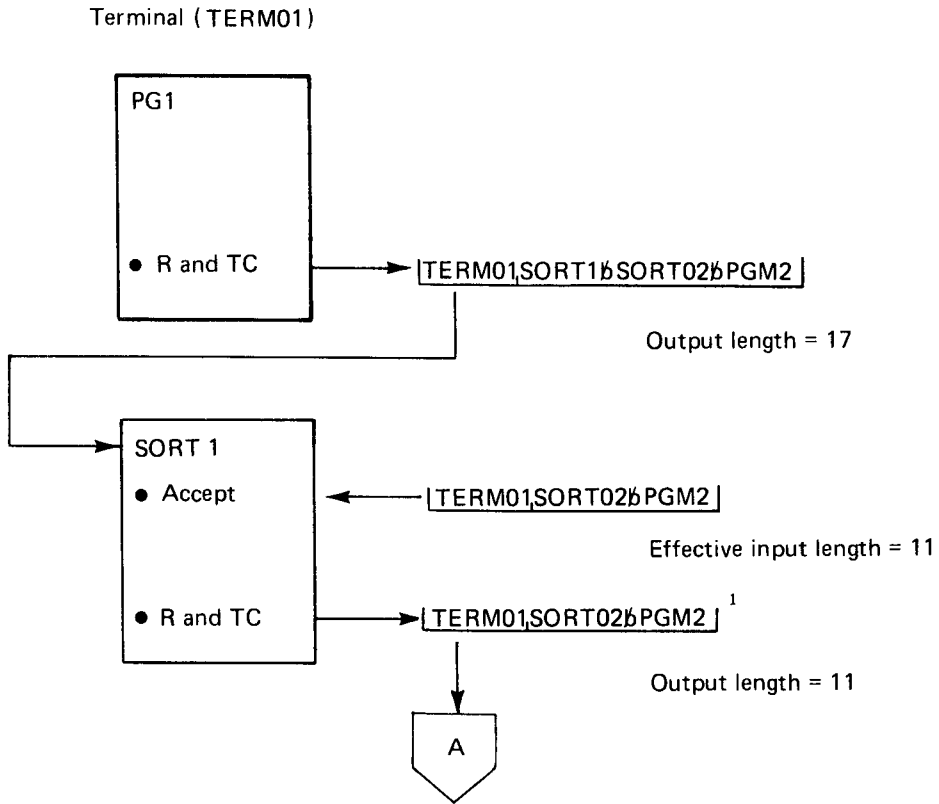
### Notes:

1. Symbolic file names that are defined for the terminal being released may be used by the program being requested if the requested program is an SRT. The /FILE command and SYMFILE statements will resolve the file name references.
2. This operation does *not* require Chain Task Request queues (MAXCHAIN) to be specified on the SYSTEM assignment statement.

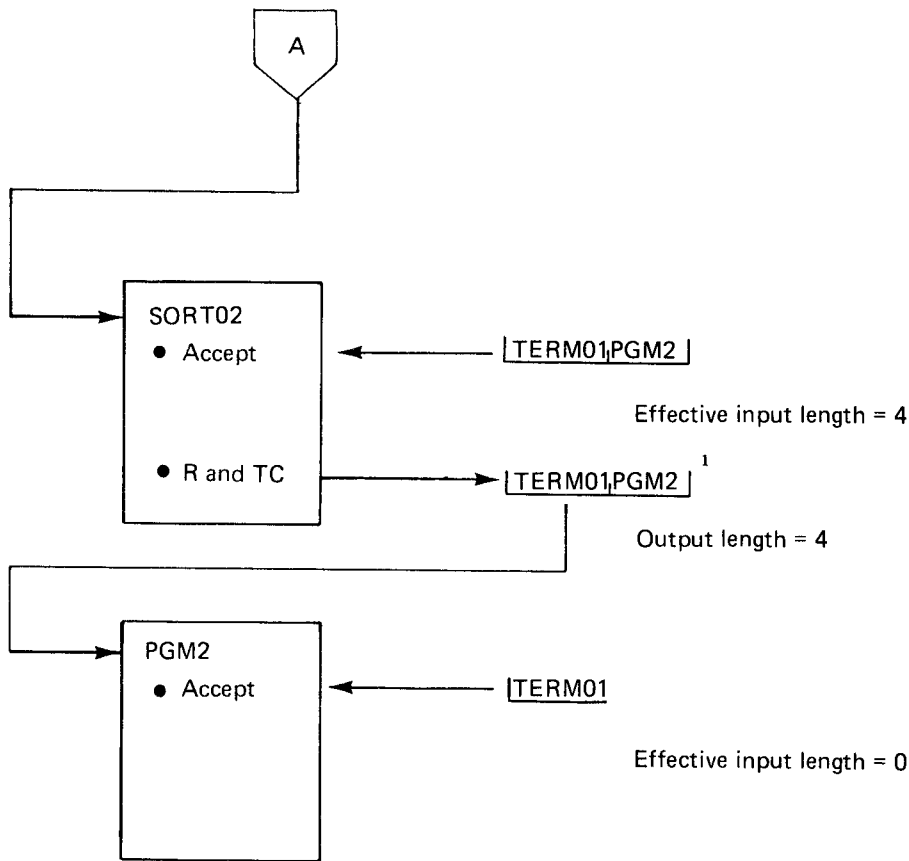
### Release and Task Chain Involving Sort Programs

The Release and Task Chain operation may be used to invoke a CCP/disk sort program. This allows symbolic files defined to the terminal to be used by the CCP/disk sort program. The CCP/disk sort program can continue to pass the symbolic file reference and terminal to subsequent sort or application programs. An option is allowed in the sort header card to allow the CCP/disk sort program to issue a Release and Task Chain operation upon the completion of the sort.

In the following example, the terminal operator requests program PG1. PG1, in turn, requests SORT1 and instructs SORT1 to request SORT02. SORT02 then requests PGM2.



<sup>1</sup> CCP/SORT issues the Release and Task Chain Operation if this operation is specified during the generation of the sort program.



PGM2 then displays the results of the two sorts to the terminal TERM01. TERM01 is allocated to PGM2 as its requester.

<sup>1</sup> CCP/DISK SORT issues the Release and Task Chain Operation if this operation is specified during the generation of the sort program.

**Note:** If the sort program is requested by a Task Chain operation or is requested by the console operator, the sort program cannot issue the Release and Task Chain operation because no terminal exists for the Release and Task Chain operation. In the above case, if the sort program has been passed data with the request, the sort program will perform a Task Chain operation instead.

## SHUTDOWN INQUIRY

The Shutdown Inquiry operation is used to determine whether the system operator has requested CCP system shutdown.

### Operation Code

Hex	Dec	RPG II	Meaning
0000 0100	0 256	0000 0A00	Shutdown Inquiry Shutdown Delay Inquiry (5704-SC2)

### Additional Requirements

None.

### Information Returned

- Count of outstanding Invite Inputs in third field of parameter list (5704-SC2 only)
- Return Codes:
  - 0 Shutdown not requested
  - 4 Shutdown requested
  - 17 Shutdown delay has been entered and the delay time has not expired (only for the Shutdown Delay Inquiry Operation) 5704-SC2 only

## Function and Use of Shutdown Inquiry

The system operator can request shutdown of the CCP at any time. When he does, programs currently running should terminate their own execution in a controlled manner, rather than be cancelled by the system operator without warning. The Shutdown Inquiry operation determines whether or not shutdown has been requested and sets an appropriate return code in the parameter list.

If you want your program to perform a controlled termination when shutdown is requested, you may include a Shutdown Inquiry (or an Accept Input) and a test of the return code in your program. For example, perhaps you want your program to inform attached terminals that the CCP is about to shut down before your program terminates.

Shutdown Delay Inquiry (5704-SC2 only) allows the application program to inquire if shutdown delay has been entered by the system operator. A return code of 17 is returned if the system operator has requested shutdown delay and the time has not expired. Once the time has expired, a 04 is returned to the program. This allows a PRUF program to inform the terminal operator of the shutdown delay, or this provides an orderly termination of the PRUF program.

**WAIT OPERATION (Model 15 only)**

The Wait operation enables an application program to suspend operation for a specified time period (hours/minutes/seconds).

*Note:* To use the Wait operation, the timer option must be selected at system generation time.

*Operation Code*

Hex	Dec	RPG II	Meaning
0014	20	␣␣AD	Wait Operation

*Additional Requirements*

- Set output length equal to 10. (If using RPG SUBR92, set output length to 24.)
- Specify the time limit in the user's record area.

*Information Returned*

Return Code 0 – Successful

**Function and Use of Wait Operation**

This operation enables the application program to issue a wait request for the amount of time specified (in decimal) in the data area. The data area is specified in the following format:

Terminal name	␣	hhmmss	␣␣␣
1-6	7	8-13	14-16

where:

- 1-6 – Symbolic terminal name
- 7 – Blank
- 8-13 – hh = number of hours in decimal  
mm = number of minutes in decimal  
ss = number of seconds in decimal
- 14-16 – Blanks

*Wait Operation Format for RPG SUBR92*

op code	output length	terminal name	␣	hhmmss	␣␣␣
1-4	5-8	9-14	15	16-21	22-24

where:

- 1-4 – ␣␣AD
- 5-8 – 0024
- 9-14 – Symbolic terminal name
- 15 – Blank
- 16-21 – hh = number of hours in decimal  
mm = number of minutes in decimal  
ss = number of seconds in decimal
- 22-24 – Blanks

Your program resumes execution after the wait operation is completed.

-



This chapter describes several aspects of communications programming that are important for you to understand before you write application programs that use the CCP communications interface facilities. (The interface facilities are described in Chapter 2 and are further defined in later chapters for each programming language.) The following topics are discussed:

- Terminal classes
- Program use of terminals
- Program types
- Program attributes
- Communications program logic
- Symbolic files
- Switched lines

For overall system design and performance considerations involving these topics, refer to *IBM System/3 Communications Control Program System Design Guide*, GC21-5165.

### TERMINAL CLASSES

Under the CCP, terminals are divided into two classes, based on whether or not they are allowed to enter commands to the CCP:

- Command terminals
- Data terminals

Terminals are designated either command terminals or non-command (data) terminals during CCP assignment. (See BSCATERM and MLTATERM assignment statements in *CCP System Reference Manual*.)

### Command Terminals

A command terminal can request the CCP to perform special services; the most significant service request is a request to load and initiate a program. Command terminals must be capable of both input and output, since they must be able to transmit commands to the CCP and receive messages from the CCP. Once a command terminal has requested a program, it is capable of sending and receiving data under direction of the program. The terminal remains under control of the program until one of the following occurs:

- The terminal is released by the program.
- The program terminates.
- The terminal operator enters a /RELEASE command, releasing the terminal from the program.

At that time, the terminal is again allowed to enter commands to the CCP. A command terminal that is not currently signed on to enter commands may also be used by a program to send and receive data.

When a switched line with one or more command terminals is not under control of a program, the CCP awaits calls from command terminals on that line.

### Data Terminals

A data terminal cannot request CCP services, but can only transmit and receive data under control of programs that use the terminal. When a program releases the terminal, the terminal is not used until it is required by another program.

A switched line that is connected only to data terminals is allocated by the CCP to a single application program at a time. Connections are established (answers or calls) when the program performs I/O operations referencing the symbolic name of a terminal on the line (see index entry *switched lines*).

A data terminal might have only input capability, only output capability, or both input and output capability.

## PROGRAM USE OF TERMINALS

Your communications program differentiates between terminals with which it communicates based on whether the terminal requested the program.

### Requesting Terminal

A requesting terminal is a command terminal that entered into communication with your program by entering a request for your program while in *command mode* (see *CCP System Operator's Guide* for a description of command mode and program requesting). Once a requesting terminal is in communication with your program, it is in *data mode* and is directed by your program to transmit data, receive data, or both.

From a programming point of view, there are few differences between handling requesting terminals and program-selected terminals. However, the following considerations apply to requesting terminals, but not program-selected terminals:

- A requesting terminal can include data with a program request (if the program is written to handle data with the program request, see PROGRAM assignment statement in *CCP System Reference*), therefore, the program can issue an Accept Input operation as the first operation to the terminal (the CCP in effect issues the first Invite Input operation to the terminal in this case).
- A program that handles only a single requesting terminal (see Program Types) can use a blank terminal name as a reference to the requesting terminal.
- The program does not know which terminal will be the requesting terminal and, therefore, must determine which terminal is requesting by examining the terminal name returned with an initial operation.

### Program-Selected Terminal

A program-selected terminal is a terminal that has not requested your program, but is needed by your program to transmit data, receive data, or both.

A program-selected terminal can be attached to your program in two ways: it can be either specified at assignment time as required for your program or acquired by your program during its execution by means of an Acquire Terminal operation. A program-selected terminal can be one of the following:

- A data terminal in standby mode.
- A command terminal in initial mode.
- (5704-SC2 only) A command terminal in command mode and not formatted by a PRUF display.

There are hardware characteristics of certain of the MLTA terminals that need to be considered in designing and writing programs that use program-selected terminals:

A request for a program that uses a program-selected terminal is rejected by the CCP if the terminal is on a switched line that is already connected.

*Data Terminals:* No special consideration.

*Command Terminals:* Command terminals operating under 5704-SC1 must be in initial mode in order to be program selected. Command terminals operating under 5704-SC2 must be in either initial mode or in command mode and not formatted by a PRUF display in order to be program selected. However, while in initial mode, command terminals are invited for input by CCP. Thus, one of the steps in the allocation of a command terminal in initial mode is a Stop Invite Input request from CCP. If the Stop Invite Input fails (that is, the read to the terminal cannot be cancelled), the program selection of the terminal is considered to have failed. Thus, the capability of cancelling a read operation is crucial to the program selection of command terminals.

There are certain terminal types for which reads cannot be cancelled once they have been issued. Once a read has been issued to the terminal, no other operation can be started to the terminal until the input operation completes. The terminals whose hardware characteristics fall into this category are:

1. 2741
2. 2741 Dial

3. 2740 Dial with transmit control
4. 2740 Dial with transmit control and checking
5. 3767 simulating a 2741
6. 5100 simulating a 2741

Therefore, if these terminals are assigned as command terminals, it will be extremely difficult for them to be program selected. Application programs using these terminals should be written so as to handle them as requesting terminals. This may require writing a program as a multiple requesting terminal (MRT) program.

## PROGRAM TYPES

Two general types of communications programs can be written to run under the CCP:

- Single Requesting Terminal (SRT) Program
- Multiple Requesting Terminal (MRT) Program

Programs are designated as either MRT programs or non-MRT (SRT) programs by the PROGRAM statement during CCP assignment (see *CCP System Reference Manual*). If the MRTMAX keyword is used on the PROGRAM statement, the program is considered an MRT program. (MRTMAX specifies the maximum number of requesters the program can handle concurrently.) SRT and MRT programs have different characteristics and place significantly different requirements on the application programmer (see *Examples of Application Program Logic*, later in this chapter).

### Single Requesting Terminal (SRT) Program

An SRT program can service the needs of only one requesting terminal on each execution (that is, from the time the program is initiated by the CCP until it terminates). A typical example of an SRT program is an inquiry program that processes one or more messages from its requester and then terminates, using system resources only briefly. The program may access or update several different files in order to complete its processing. An SRT program might also transmit batched data to a host system, such as a System/370, where the host is the requester of the program.

An SRT program can be written to acquire (or require, by means of the TERMS parameter of the // PROGRAM assignment statement) one or more program-selected terminals while servicing the requesting terminal. For

example, perhaps the requester wants information from several terminal locations or wants to send information to other locations. An example of such a program might be an inquiry program that serves a credit office application. The requesting terminal asks for information about a customer from terminals in other offices by issuing a message to program-selected terminals in those offices specifying the customer identification. The attached offices reply with the latest credit information.

If sufficient resources are available, the CCP can load and initiate separate, duplicate copies of an SRT program, each copy servicing a different requesting terminal. If resources are not available, second and subsequent requests for the program may be placed on a queue by the CCP (see /Q command in *CCP Terminal Operator's Guide*). Multiple copies of an SRT never-ending program are not permitted under the CCP.

### Multiple Requesting Terminal (MRT) Program

An MRT program can service requests from one or more terminals each time it is executed. An MRT program may be written to handle multiple requesting terminals concurrently. The maximum number is specified by the MRTMAX keyword of the PROGRAM statement at assignment time. Requests for the program that are received while the program is already handling its maximum number of terminals are queued by the CCP (if the requester has indicated queuing) to be honored when the program has released a terminal; therefore, you need not check this in your program. Only a single copy of any one MRT program can be in main storage at a time.

New requesting terminals are attached to an MRT program by means of *Accept Input operations* (see index entry). A program is notified that a new terminal is attached by receiving a new terminal name from an Accept Input.

MRT programs must maintain status information regarding several requesting terminals in order to remember which terminals are attached and the status of each terminal relative to the program.

You must explicitly release a requesting terminal once your program has completed the processing required by the terminal (see index entry *Release Terminal operation*). If not released, the terminal remains allocated to the program until the program terminates. An MRT program should be written to terminate when it has no more requests to service, unless the program is defined as a *never-ending program* (see *Program Attributes*).

Many application programs, both order entry and inquiry, could be written in either single requester or multiple requester form. If a program is likely to be often requested from more than one terminal concurrently, it is more efficient to code the program as an MRT program, since the MRT version is not as likely to cause resource conflicts as are numerous requests of an SRT program. Also, an MRT program should take less main storage space than separate copies of an SRT program.

An MRT program can be defined as MRTMAX-1 on the PROGRAM assignment statement. In this case, only one request is processed at a time, as in an SRT program, but subsequent requests for the program can be queued to the same copy of the program if the requesting terminal has elected to queue requests (/Q command). An MRT program defined in this way does not have to keep track of multiple attached terminals; however, the program cannot process multiple requests concurrently and only one copy of the program can be in main storage at a time.

### SPECIAL PROGRAM ATTRIBUTES

The two general communications program types—SRT and MRT—may have additional special attributes:

- Never-ending
- Serially reusable (Models 10 and 12)
- Dedicated (Models 10 and 12)
- Program request under format
- Sort

The PROGRAM assignment statement specifies whether a program has any of these attributes.

### Never-Ending Program

If a program is to be requested frequently throughout the CCP run and if sufficient main storage is available, it can be defined as a never-ending program (NEP). An NEP, once it has been loaded and initiated by the CCP, does not terminate (except in an unusual situation) until the CCP is shut down. Under Model 10 and Model 12 CCP, once an NEP is loaded, the main storage it occupies is permanently unavailable for other programs, even if it terminates in an unusual situation. Under Model 15 CCP, the main storage occupied by an NEP is released for use by other programs if the NEP terminates in an unusual situation.

An NEP with no work to do (no outstanding Invite Inputs) issues an Accept Input operation and waits until it is requested again. See *Disk File Considerations* in *Chapter 9: Program Preparation* for a Model 15 CCP consideration in this situation.

When using NEPs, you should be aware of facts concerning system resource allocation:

- While a particular system resource is allocated to a NEP, requests for programs that also require the resource will be rejected, regardless of the queuing status (/Q or /NOQ) of the requesting terminal.
- When an NEP requires a system resource that is already allocated to another program, the request for the NEP will be rejected without regard to the queuing status of the requesting terminal. Under Model 15 CCP, a request for a NEP can be queued if CCP is handling the maximum number of tasks or if user program area is not immediately available.

An NEP should either check for the shutdown return code or issue the Shutdown Inquiry operation so that CCP shutdown can be successfully completed.

When writing an SRT program as a never-ending program, be aware that second and subsequent requests for the program are rejected by the CCP.

To ensure that the required main storage is available for the never-ending program, it is recommended that all never-ending programs be loaded into main storage prior to requesting any other programs. Main storage fragmentation could result if never-ending programs are not started as the initial programs in the system.

*Note to Model 10 and 12 Users:* It is especially important to load NEPs first if they use the 3270 Display Format Facility, because the Display Format Facility is loaded into the user program area for execution.

*Note to Model 15 Users:* CCP always loads NEPs at the opposite boundary of the user program area from non-NEPs to avoid fragmentation of this area, which could otherwise severely impair CCP performance.

### **Serially Reusable Program (Models 10 and 12)**

A serially reusable program terminates normally after executing and can be re-executed without requiring a fresh copy of the program to be loaded in main storage. A serially reusable program must restore data areas and modified instructions to their initial condition prior to reusing those data areas and instructions when the program is reinitiated. Only COBOL and Basic Assembler programs can be written to be serially reusable. Never-ending programs cannot be specified as serially reusable.

Use of serially reusable programs can increase CCP efficiency, especially if the programs are requested frequently, since the need for repeated loading of the program can be avoided in some cases. If other programs are being requested and loaded, however, timing of the requests may be such that, when a subsequent request for the serially reusable program is entered, the main storage space previously occupied by the serially reusable program is already occupied by another program. In this case, the serially reusable program must be reloaded.

### **Dedicated Program (Models 10 and 12)**

When a program defined as a *dedicated program* is running,

it must be the only program running in the CCP program level, even though multiprogramming is allowed by the CCP. It may not be initiated while other programs are running and other programs may not be initiated while it is running. This program attribute applies only in CCP systems that allow more than one program to be executing concurrently.

You might want to designate a program as dedicated if it requires exclusive use of disk files that are otherwise shareable, for example, a program that performs summary operations at a particular cut-off time, such as day-end or month-end, when concurrent operations on the files are not desired. A dedicated program might also be used in applications where fast response time is important and the program relies on exclusive access to all communication lines, disk files, and other system resources.

### **Program Request Under Format**

With a non-PRUF request, the maximum amount of data that can be passed to a user program, as a program request, is 78 characters. This is not an efficient use of the 3270 terminal buffer. An efficient method of using the 3270 terminal is to write a short SRT program which will put a display at a terminal, and then go to end of job. The terminal operator can then fill in the display with data, cause attention (PF key or ENTER key) and have the display at the terminal essentially request another program. The whole display will be used as program request data. This concept of requesting programs and passing up to a full screen of data to the requested program is called Program Request Under Format (PRUF).

The use of PRUF will provide the following capabilities:

- More than one field of data may be passed as program request data.
- More than 78 characters of data may be accepted as program request data.
- The AID character is passed as program request data to PRUF programs, but not to non-PRUF programs.
- The data passed to the user program with the program request may or may not be processed by DFF under format control if the program being requested is a PRUF program. However, DFF does not process non-PRUF program request data.
- Main storage can be used efficiently, as a program need not be in main storage during a lengthy terminal operator keying operation.

A program is defined as a PRUF program if the PRUFLNG parameter is specified in the assignment PROGRAM statement. The PRUFLNG parameter specifies the maximum length of program request data to be sent by the terminal. If the PRUF program is also a DFF program, the PRUF\$Z parameter is specified in the PROGRAM statement. This gives the name of the format which will be used by DFF to format the program request data. (See the *IBM System/3 Models 8, 10, and 12 Communications Control Program System Reference*, GC21-7588, or the *IBM System/3 Model 15 Communications Control Program System Reference Manual*, GC21-7620, for a complete description of these keywords.)

To inform CCP that the next program request from a 3270 terminal will be a PRUF program request, user program A (which may be a PRUF or a non-PRUF program) executes a PRUF-Put operation to the 3270 terminal as its last output operation prior to releasing that terminal or going to end-of-job. If the terminal receiving the PRUF-Put operation was a requestor of the program issuing the PRUF-Put, that program must have ENDMSG-NO specified on its PROGRAM statement.

Before returning the terminal to command mode status, CCP will reserve an area from the TP buffer, equal in length to the maximum PRUFLNG, as a temporary hold area for the program request data from that terminal. It should be noted that CCP will only reserve a TP buffer area equal in length to the PGMREQ, as specified in the SYSTEM statement, if the last user output operation to that terminal is not a PRUF-Put operation.

The first field received from the 3270 must be the program name of the PRUF program to be requested and must begin in row 1, column 2 or later on the screen. The program name must be either entered in the first field on the screen (that field defined as an input field) or sent to the screen (by a PRUF-Put operation) in the first field defined as an output/input field. The terminal operator then keys in data to all needed input fields on the screen.

When all the needed fields have been keyed in, press the ENTER key, a PF key, or insert a card into the card reader (this action is device dependent). Now the program request for program B enters the system. If program B is a non-DFF program, the following data will be passed as program request data to program B:

cu	dev	aid	c@	sba	@@	pgmnam	pgmdata . . .
1	2	3	4-5	6	7-8	9 . . .	

**Note:** If the program is written in RPG II, these fields start in column 15 of the input specifications,

where:

- cu = Control unit address of the 3270 terminal
- dev = Device address of the 3270 terminal
- aid = AID character
- c@ = Cursor address
- sba = Set buffer address (X'11')
- @@ = Address of start of pgmnam field
- pgmnam = Name of program B
- ␣ = EBCDIC character for a blank (X'40')
- pgmdata = Remainder of 3270 text stream or the number of characters specified by PRUFLNG parameter, whichever is smaller.

A PRUF program request will return 8 plus *PGMNAM length* plus 1 *additional* character of data more than a program request for a non-PRUF program. If a program B is a DFF PRUF program, DFF will attach the PRUF\$Z format to the program using that format for control and move data into program B's input record area at program request time. (See Chapter 8 for a description DFF handling of accept input data.)

The following considerations apply when running CCP assignment sets with PRUF programs:

- PRUF-Put operations to the system console are invalid.
- If the 3270 terminal has been formatted by a PRUF-Put operation, and the program being requested is a non-PRUF program, CCP will reject the program request.
- If the 3270 terminal has not been formatted by a PRUF-PUT operation, and the program being requested is a PRUF program, CCP will issue a 02 return code following the Accept Input operation. The program request data returned in this case will begin with the first character of data following the program name and a blank. This will not have been processed by DFF.
- If a Non-PRUF Put of a DFF Format is used to prompt for a PRUF Program, the program name and any data should be placed in one DFF field. If separate fields are used for the program name and the data, the attribute byte for the first data field terminates the data stream and no data is passed to the program. (The 02 return code is returned, and the effective input length value is zero.)
- If PRUF is active on the terminal, all system messages to that terminal will be output in positions 82 through 160. Therefore, these positions should be used with caution at program request time to PRUF programs.
- A terminal which had a PRUF-Put format sent to it becomes a PRUF terminal. This condition (PRUF-type terminal) can be terminated by the next non-PRUF Put operation sent to the terminal, or if the terminal CLEAR key is pressed. If PRUFOF-YES is specified at assignment time, the PRUF condition will be terminated after an Accept Input operation is sent to that terminal. This means a non-PRUF Put will not be required to put the terminal in non-PRUF mode.

## Sort Programs

Sort programs (using CCP/disk sort rather than disk sort) must be generated offline from CCP but can be executed as user programs under CCP. See the section entitled *Program Preparation* in this manual. Multiple sort programs can be active under CCP at one time providing each program has a unique name, and unique work and output files. A minimum of three files must be defined on the FILES parameter of the PROGRAM assignment statement for a sort program:

- An input file with CG access. Input files can be opened with CO, IO, and IOU access methods but the opening program must terminate successfully before the file can be shared. Up to eight input files can be defined.
- A work file with CA access and specified as nonshareable.
- An output file with CO access and specified as nonshareable. After a sort program has completed successfully, the output file can be opened using CA, CG, CO, CU, DG, or DU access methods.

*Note:* For Model 15, the sort input file cannot be specified as the sort output file. Although the use of SYMFILE statements and /FILE commands to resolve sort file name assignments is supported, do not attempt to overlay the input file with the output file by using the SYMFILE and /FILE facilities.

After a sort program has terminated successfully, the input, work, and output files are available to other sort and non-sort programs.

When a sort program is requested from a terminal, additional modules with the prefix \$DG are loaded by the sort program. These additional modules (initially supplied on the system pack) must be on the same pack from which the sort program was loaded. For example, if a sort program has been placed on the program pack (PACK-PROGRAM specified on the PROGRAM assignment statement), the \$DG modules must be copied to the program pack from the system pack. This can be accomplished using the following OCL:

```
// COPY FROM-ss, TO-pp, LIBRARY-O,RETAIN-P,
      NAME-$DG.ALL
```

where:

```
ss = disk unit where system pack is located
pp = disk unit where program pack is located
```

Because of space considerations, it may be more convenient to put the sort program on the system pack.

Once a sort program has started, the requesting terminal is released and is free to perform other operations. All sort messages indicating the status of the sort program are issued to the system operator's console after the requesting terminal is released.

**Note:** If another program is allowed to update the sort input file at the same time that a sort program is processing the file, improper results can occur. For this reason, either specify NOSHR for the sort input file in the assignment set, or allow input only programs to access the file while the sort program is executing. See *CCP System Reference manual* for your system and the *IBM System/3 Disk Sort Reference Manual*, SC21-7522, for additional information about sorts.

If shutdown is requested, an active sort program is allowed to complete execution to normal termination.

### **Task Chaining and Release and Task Chain With Sort (5704-SC2 Only)**

When a sort program is to issue a chain request or a Release and Task Chain, the name of the requested program and the data (if required) must be passed to the sort program with the program request. The sort program interprets the first six characters (or up to the first blank if the name is less than six characters) as the name of the program to be requested. The remaining data is passed as data along with the chain request.

If more than 80 characters of data are passed to a sort program, the sort program issues a CCP DATA TRUNCATED message to the console and issues the chain request or a Release and Task Chain, even though the data has been truncated.

If CCP is handling the maximum number of chain requests when a sort program issues a chain request, CCP issues a TASK CHAIN UNSUCCESSFUL message to the console. In this case, the sort program is allowed to complete execution, but the chain request is neither issued nor executed.

For an example of a sort program issuing a chain request, see the following index entry: *task chain request and release and task chain*.

## **EXAMPLES OF APPLICATION PROGRAM LOGIC**

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

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

The following examples illustrate the typical logic required to deal with:

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



### Single Requesting Terminal

Figure 3-1 shows the program logic that might be used in a program that deals with only a single requesting terminal on each execution. The numbered notes further explain aspects of the logic.

- ❶ If data is expected with the program request, no Invite Input is required, since the first Accept Input will return the requester's name and data in the record area. If data is expected with the program request, an Accept Input *must* be issued.
- ❷ A Put operation with a blank terminal name causes the name of the requesting terminal to be returned in the record area. Invite Input can then be issued to that terminal. If no data is expected with the program request, a Get operation with a blank terminal name can be used after the Put operation instead of the Invite Input and Accept Input operations.
- ❸ If data is expected with the program request, the first input operation is an Accept Input. Subsequent input operations can either be Accept Input or Get operations. If Accept Input is used, Invite Input must be issued prior to the Accept Input except for the first Accept Input operation when data is expected with the program request.

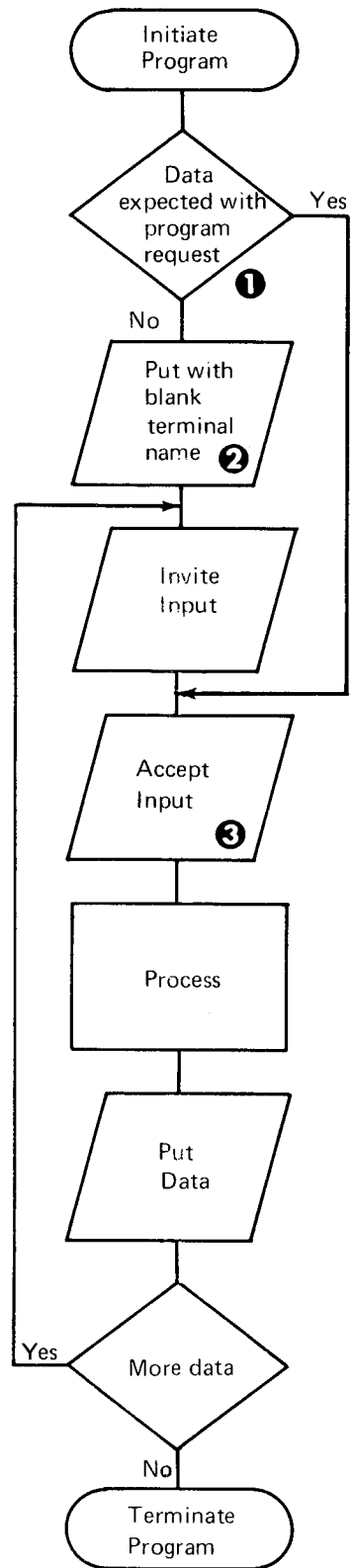


Figure 3-1. Program that Communicates with a Single Requesting Terminal

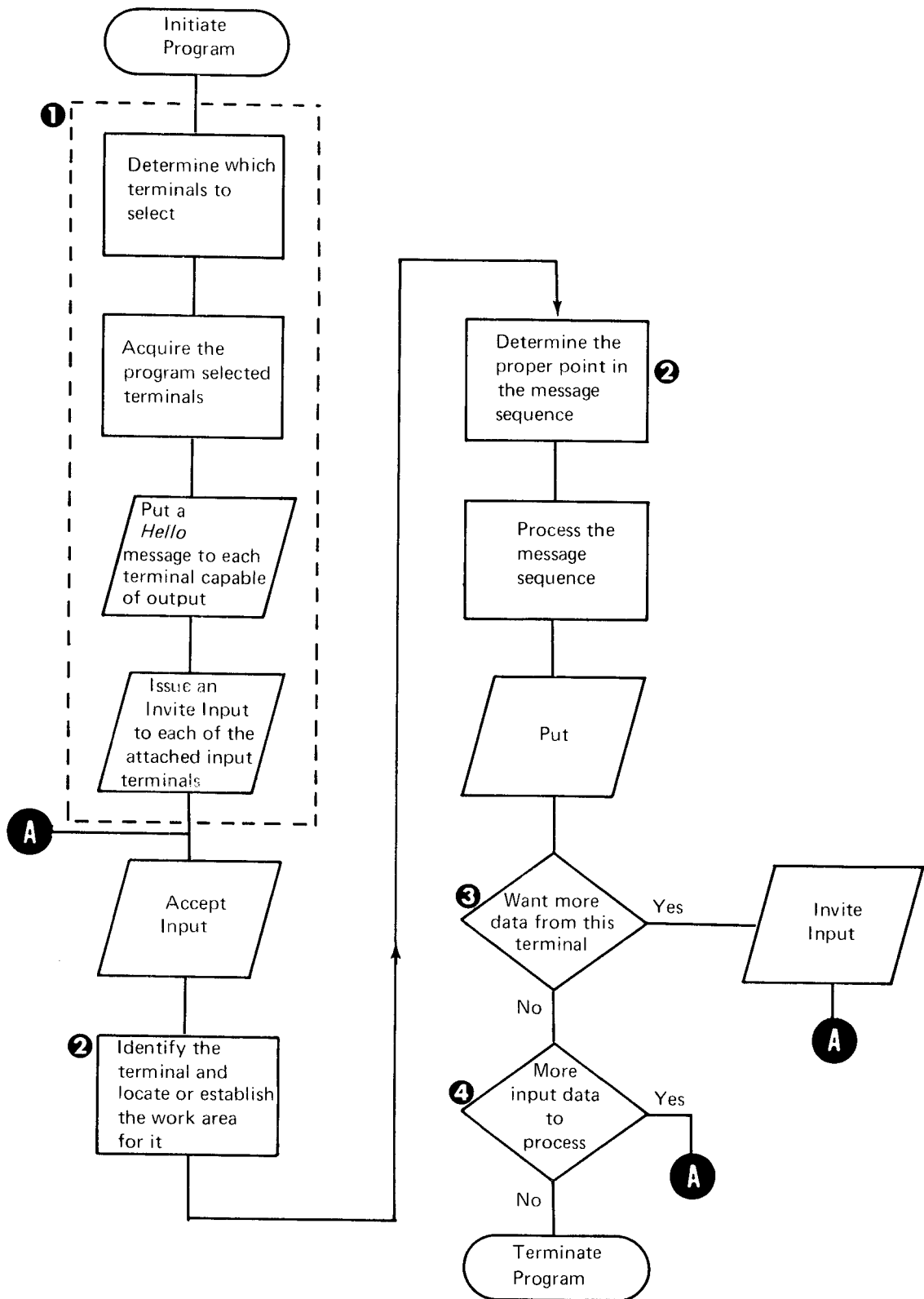


Figure 3-2. Program that Communicates with a Single Requesting Terminal and Program-Selected Terminals

## Single Requesting Terminal and Program-Selected Terminals

Figure 3-2 illustrates the logic that could be used by a communications program to deal with a single requesting terminal and one or more program-selected terminals. The numbered notes further explain characteristic aspects of this type of logic.

- ❶ These steps (enclosed by the broken line) are performed only the first time through the program. A program can determine which terminals to select in various ways:
  - The terminals required by the program are specified at assignment time and the terminals have been allocated to the program before it gains control.
  - The program knows which terminals it needs, but must acquire them itself.
  - The program does not know which terminals to select. The program might have to obtain this information from the system operator, a terminal, or from the data he is processing.

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

- ❷ When a program is capable of handling more than one terminal, it may need to set aside a separate work area for each. The program would use the work area to retain enough information to remember what it has previously received from each terminal. When, for example, input data consists of more than one part, a separate routine often processes each different part. A complete input message might consist of a customer name or number, an order number, item numbers, quantities, prices, and other information, entered as separate lines of input data and, in fact, as separate transmissions from the terminal. The program must be able to determine which portion of the data it is processing, where to store that data in the work area, and which routine processes that portion of the data.
- ❸ When a program-selected terminal has completed a message sequence, the program must determine whether it wants additional input from the terminal. For example, if the program has received an end-of-input signal or if the system operator has issued the SHUTDOWN command (and the program recognizes the shutdown return code), the program should not issue an Invite Input to the terminal.
- ❹ If any other program-selected terminals have input messages to transmit (have outstanding Invite Input operations issued to them), the program finishes processing them. When all input from the program-selected terminals and the requesting terminal has been processed, the program terminates.

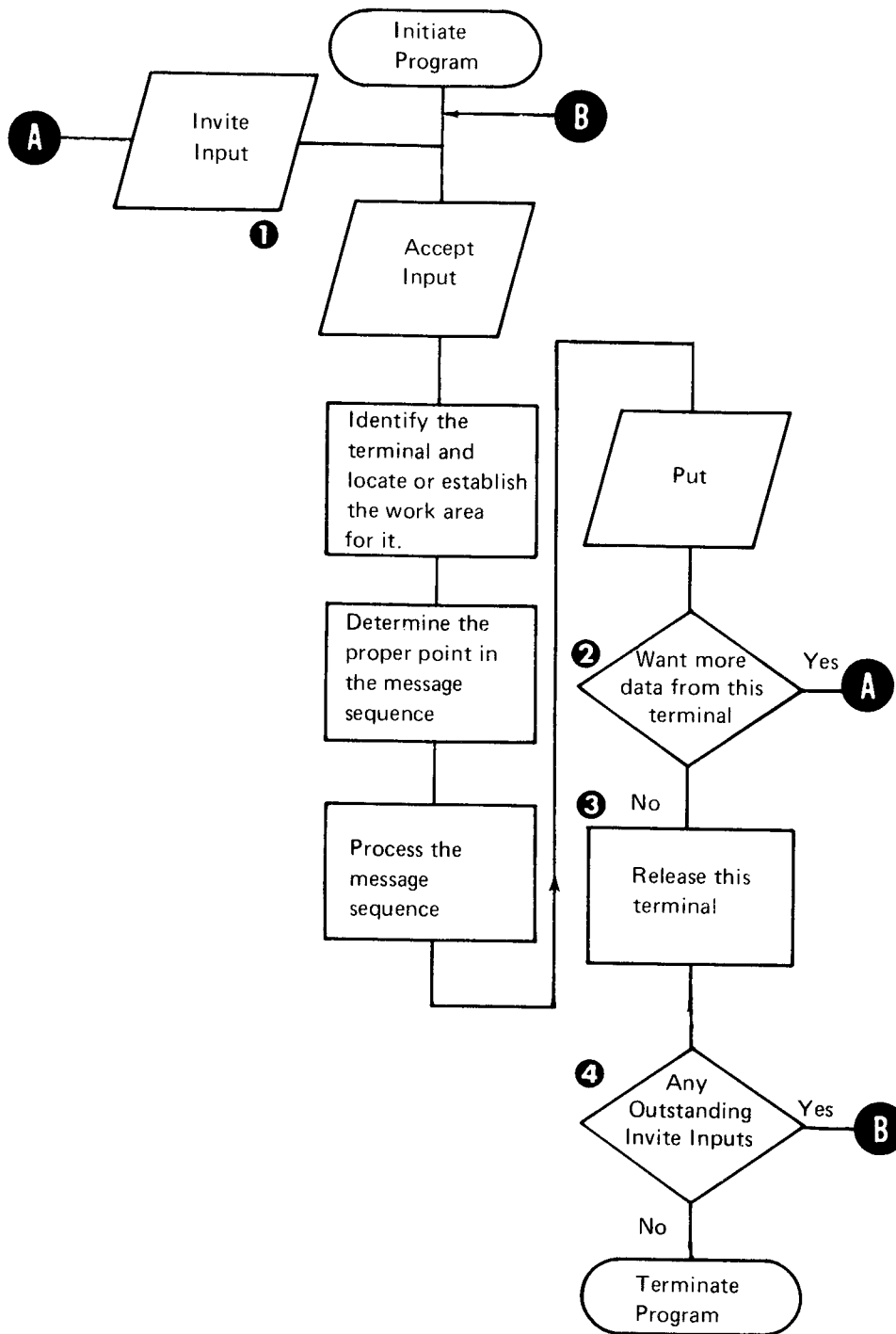


Figure 3-3. Program that Communicates with Multiple Requesting Terminals and No Program-Selected Terminals

## Multiple Requesting Terminals

Figure 3-3 shows the program logic that might be used in a program that deals with multiple requesting terminals. The numbered notes further describe key steps in the logic.

- ❶ The Invite Input is bypassed the first time through since it is not known which terminal requested the program until after the Accept Input operation.
- ❷ When the program has received the final portion of a message sequence from a particular terminal, it must determine whether it wants additional input from the terminal. If, for example, the terminal has indicated that this is the last message it will send or if the system operator has issued the SHUTDOWN command to shutdown the CCP (and the program recognizes the shutdown return code), the program should not issue an Invite Input to the terminal.
- ❸ If no Invite Input is to be issued to this terminal, the terminal is released from the program.
- ❹ If requests from other terminals are in process or awaiting processing, they must be completed before the program terminates. The number of remaining requests can be determined from the count of outstanding Invite Inputs, returned in the third field of the parameter list by the previous Release Terminal operation.

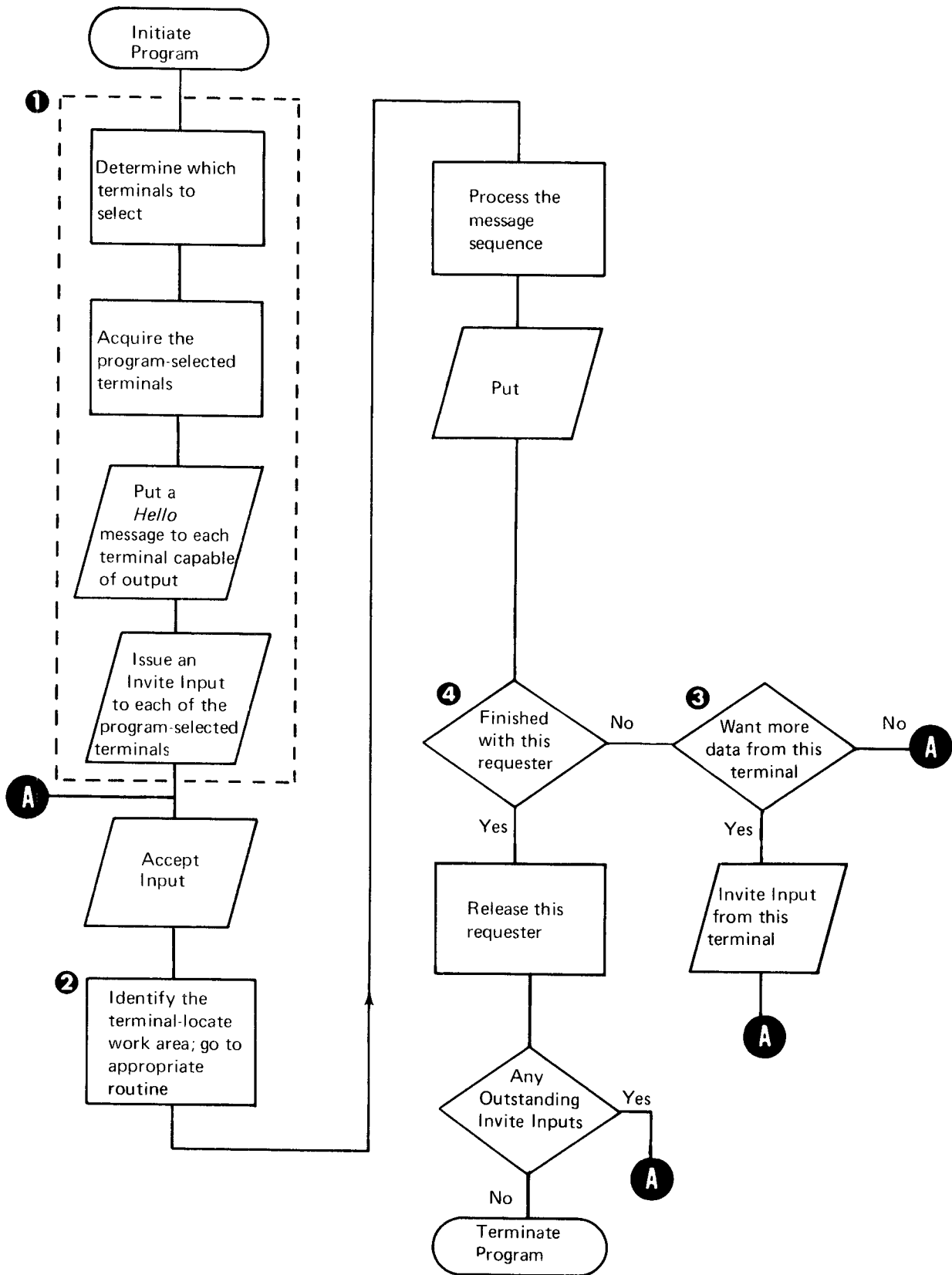


Figure 3-4. Program that Communicates with Multiple Requesting Terminals and Program-Selected Terminals

## Multiple Requesting Terminals and Program-Selected Terminals

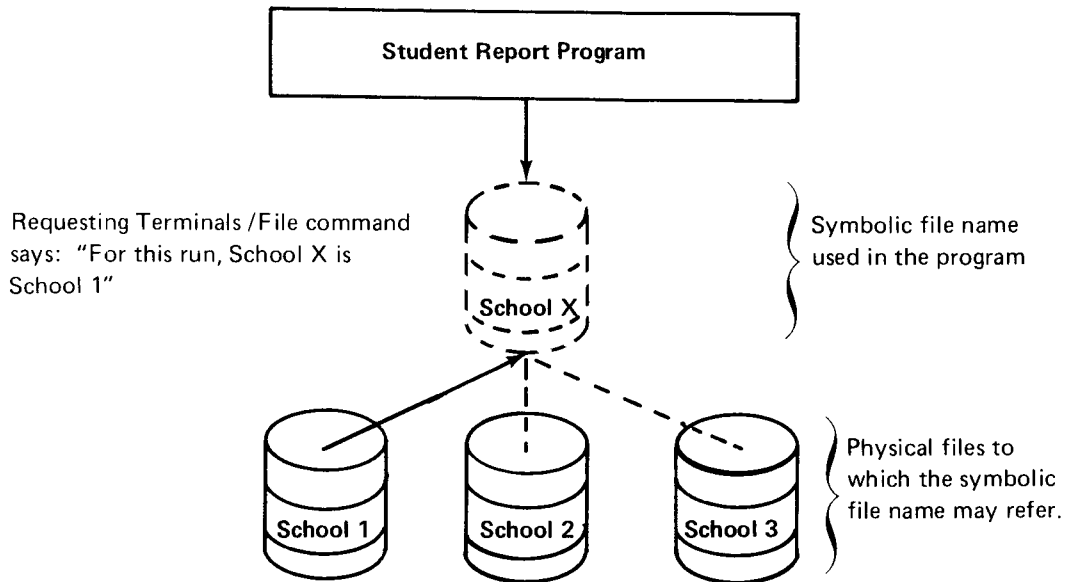
Figure 3-4 shows an example of the general logic of a communications program that accepts requests concurrently from several requesting terminals and, in satisfying the requests, contacts one or more program-selected terminals. As each requester is satisfied, the program releases it, enabling the requester to enter other commands to the CCP. The program-selected terminals in this example are not released from the program until the last requester has been served and the program is terminated. The numbered notes further explain key steps in the logic.

- ❶ The first-time processing required when program-selected terminals are used is described in Figure 3-2.
- ❷ When a program is capable of handling more than one terminal, it may need to set aside a separate work area for each. The program may have to retain enough information to remember what it has previously received from each terminal. When, for example, input data consists of more than one part, a separate routine often processes each different part. A complete input message might consist of a customer name or number, an order number, item numbers, quantities, prices, and other information, entered as separate lines of input data and, in fact, as separate transmissions from the terminal. The program must be able to determine which portion of the data it is processing, where to store that
- ❸ When a program-selected terminal has completed a message sequence, the program must determine whether it wants to invite additional input from the terminal. For example, if the program has received an end-of-input signal or if the system operator has issued the SHUTDOWN command, the program should not issue an Invite Input to the terminal.
- ❹ If requests from other terminals are in process or awaiting processing, they must be completed before the program terminates.

data in the work area, and which routine processes that portion of the data.

## SYMBOLIC FILES

Under the CCP, you can write a program using a file name (symbolic file) which might refer to a different physical disk file each time the program is executed. The physical (actual) files must be similar in format (that is, the same file organization, access method, record length, key length, and key position), although different in content. For example, you could write a student report program that generates a report for a different school on each execution, depending on which school's student file is associated with the symbolic file for a particular run (Figure 3-5).



The assignment statements for this example might look like this (see *CCP System Reference Manual* for explanations of the assignment statements):

```
// DISKFILE NAME-SCHOOL1, ORG-I, RECL-100, KEYPOS-94, KEYL-7
// DISKFILE NAME-SCHOOL2, ORG-I, RECL-100, KEYPOS-94, KEYL-7
// DISKFILE NAME-SCHOOL3, ORG-I, RECL-100, KEYPOS-94, KEYL-7
// SYMFILE NAME-SCHOOLX, DISKFILE-'SCHOOL1, SCHOOL2, SCHOOL3'
// PROGRAM NAME-STURPT, PGMDATA-NO, FILES-'SCHOOLX/IR'
```

Figure 3-5. A Symbolic File

The specific physical file to be used on a particular run of the program is determined by the operator of the requesting terminal by means of a /FILE command prior to the program request (see *CCP Terminal Operator's Guide* or *CCP System Operator's Guide* for a description of this command.) The

names of one or more valid physical files are associated with the symbolic file name by means of the DISKFILE and SYMFILE assignment statements (see *CCP System Reference Manual*). // FILE OCL statements for the physical files are entered by the system operator during CCP startup.



## Considerations and Restrictions in Using Symbolic Files

- An MRT program cannot use symbolic files.
- On the Model 10 Disk System or the Model 12, if your program releases the requesting terminal prior to the initial opening of any symbolic file, the CCP will cancel your program. You must be especially careful in programs that can be requested from the console, since the console is automatically released from your program by the CCP.

To use symbolic files in Model 10 and Model 12 CCP application programs that can be requested from the console, you must specify at assignment time (// PROGRAM statement) that data is allowed with the program request, even if no data will actually be entered. The CCP releases the console when the first Accept Input in the program has resulted in the name CONSOL and program data being passed to the user program. Therefore, the symbolic file must be opened before the Accept Input. In RPG II all files are opened before any I/O operations can be performed.

- On the Model 10 Disk System or a Model 12, a serially reusable program (COBOL or Basic Assembler) will access the same physical file when it is reused (executed without being reloaded) as it was used on its initial execution.

## SWITCHED LINES

A switched line is a communication line on which the connection between the system and the terminal is established by dialing a data set (telephone) number either automatically or manually. After the connection is completed, data can be transmitted. A terminal on a switched line is disconnected under control of an application program by means of the Release Terminal operation (see *Operations*, in Chapter 2). This operation can specify whether to keep the line allocated to the program or to "return" it to the CCP.

If command mode terminals are attached to a switched line, the CCP awaits calls from the terminals, rather than polling the terminals for commands.

Under the CCP, a data set (telephone) number can be established at assignment time (TERMNAME statement) for each terminal name assigned to a terminal that might be called by an application program running under the CCP. Also, the attribute set associated with a terminal on a switched line can assign the attributes auto/manual call and auto/manual answer.

## Auto Call

If a terminal is defined as *auto call* by the TERMATTR assignment statement, an I/O operation from a user program to the terminal on a switched line that is not connected causes the CCP to place a call to the terminal automatically. Auto call cannot be used with MLTA terminals. In order to use auto call on BSCA terminals, the Auto Call feature must be installed on the BSCA hardware.

## Manual Call

If a terminal is defined as *manual call* by the TERMATTR assignment statement, an I/O operation from a user program to the terminal on a switched line that is not connected causes the CCP to inform the system operator that he must dial a data set number on a certain line.

## Auto Answer

If a terminal is defined as auto answer, the CCP awaits a call from the terminal, and automatically answers the call (if the auto answer feature is activated on the data set). An I/O operation from a user program to a switched line that is not connected causes the CCP to inform the system operator that the program is awaiting a call on the switched line.

## Manual Answer

In manual answer, the system operator answers a call from a terminal. The system operator and terminal operator then place their data sets in data mode; the CCP waits for input from the terminal. An I/O operation from a user program to a switched line that is not connected causes the CCP to inform the system operator that the program is awaiting a call on the switched line.

## BSCA Switched Line

On a BSCA switched line, the CCP allocates the line to the user program when the first terminal on the line is allocated. In order to communicate with a terminal on the line, the program must either already have the line allocated or the line must be free for allocation (not currently allocated to another program).

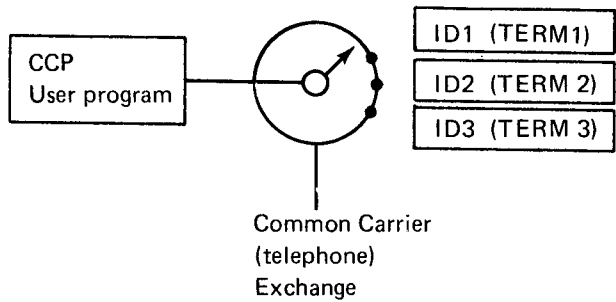
Invite Input operations can be outstanding to multiple terminals on the same line after a connection has been made. The CCP determines which symbolic name to return with an Accept Input operation from the exchange identification characters, which are associated with a specific terminal

name by the BSCATERM assignment statement (see *CCP System Reference Manual*).

**Note:** If a communications operation is issued to a terminal for which VERIFYID=NO is specified in the TERMATTR assignment statement and the operation is an answer operation, then any terminal that calls satisfies the operation.

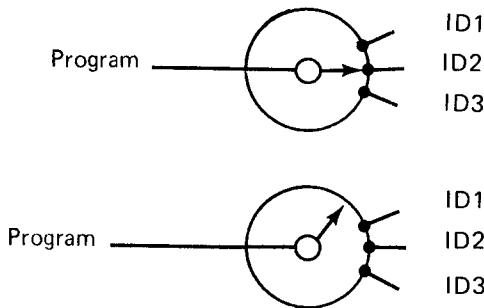
**BSCA Requesting Terminals**

The following examples illustrate the use of BSCA switched lines with requesting terminals. Assume you have the following switched point-to-point network:



ID1 }  
 ID2 } Exchange ID's  
 ID3 }  
 TERM1 }  
 TERM2 } Symbolic terminal names  
 TERM3 }

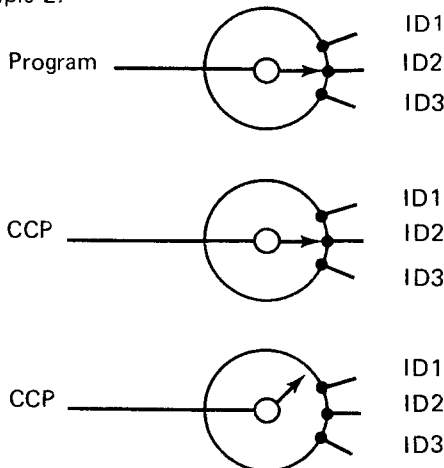
**Example 1:**



1. ID2 (TERM2) calls, makes connection, signs on, makes a program request, and communicates with program under the name TERM2.
2. Program issues Release Terminal operation, keeping the line. ID2 is signed off automatically.
3. Program now has control of the line, can call or accept calls from ID1, ID2, or ID3 in data mode.

*Note:* No terminals are allowed to call in and sign on while the line is in control of the application program.

**Example 2:**

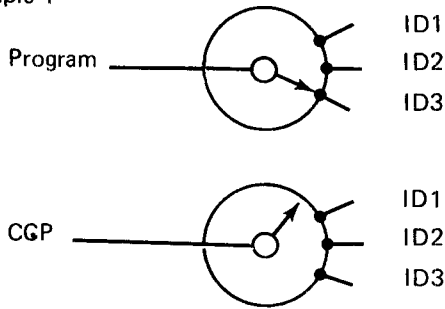


1. ID2 (TERM2) calls, makes connection, signs on, makes a program request, and communicates with program under the name TERM2.
2. Program reaches end of job without issuing a Release Terminal operation.
3. TERM2 is still connected and can enter other commands and program requests.
4. TERM2 signs off specifying DROP. The connection to TERM2 is broken.
5. ID1, ID2, and ID3 can call and sign on.

*Program-Selected Terminals*

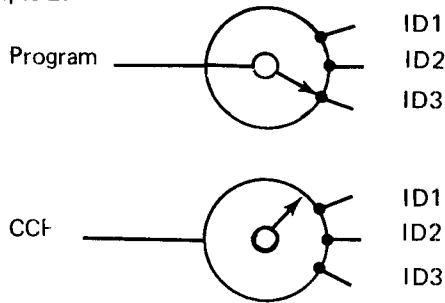
The following examples illustrate the use of BSCA switched lines with program-selected terminals. Assume, again, a switched point-to-point network.

Example 1



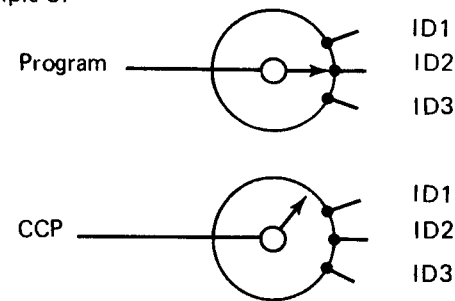
1. Program issues Invite Input operations to ID1, ID2, and ID3.
2. ID3 calls and communicates with program as TERM3 (Accept Input operation code).
3. Program issues Release Terminal to TERM3 with or without keeping the line and issues another Accept Input (the line is disconnected).
4. ID1 and ID2 can call and communicate with the program; ID3 cannot, since it no longer has an Invite Input.

Example 2:



1. Program issues Invite Input operations to ID1, ID2, and ID3.
2. ID3 calls and communicates with program as TERM3 (Accept Input operation code).
3. Program goes to end of job without releasing terminals (programs should issue Stop Invite Inputs to terminals with outstanding Invite Inputs before going to end of job).
4. All Invite Inputs are cancelled; terminals are available to other programs.

Example 3:



1. Program calls ID2 and communicates with ID2 as TERM2.
2. Program issues Release Terminal to TERM2 with or without keeping the line (the line is disconnected).
3. Program can call ID1, ID2, or ID3 and issue Invite Inputs. CCP either automatically calls the correct data set number (auto call) or provides the correct data set and line number to the system operator (manual call).

## MLTA Switched Line

On MLTA switched lines, there is only one terminal per line. Both the line and the terminal are allocated to the program by the CCP. The examples of using a switched line with command terminals and data terminals given under *BSCA Switched Line* apply also to MLTA switched lines, except that:

- All terminals that call on an MLTA switched line have the same symbolic terminal name; they cannot be uniquely distinguished from each other.
- MLTA terminals on switched lines do not have exchange identification characters associated with them.

## Switched Line Disconnect Considerations

The CCP disconnects a terminal on a switched line in any of the following circumstances:

- User program issues a Release Terminal operation specifying the keep-line modifier.
- User program issues a Release Terminal operation without the keep-line modifier to a *data terminal* (see index entry).

- User program issues a Release Terminal operation to a command terminal that is not the requesting terminal.
- The user program terminates while the line is being used with a program-selected terminal.
- The system operator issues a VARY OFFLINE command to a terminal connected to the switched line.
- A requesting terminal on the line signs off (/OFF) and the DROP option is in effect.

The only Release Terminal operation for which the terminal is not disconnected is a Release Terminal operation to the requesting terminal without the keep-line modifier. In this case, the requesting terminal is still in command mode and can continue to enter commands to the CCP.

When a command terminal is connected on a switched line, the CCP attempts to maintain the terminal connection as long as possible. After a program has terminated and the CCP has sent the "ended" or "released" message (which can optionally be suppressed by specifying ENDMSG-NO on the // PROGRAM assignment statement), the CCP attempts to receive from the terminal for an amount of time that is based on the error retry count specified by the NRETRY parameter on the // BSCALINE assignment statement.

To request CCP communication services, you must write your COBOL programs using the standard application program interface described in Chapter 2. This standard interface is composed of the following elements:

- Communications Service Subroutine
- Parameter List
- Record Area

*Note:* This chapter assumes that you are familiar with the COBOL language. For more information on writing and executing COBOL programs, see *IBM System/3 Subset American National Standard COBOL*, GC28-6452.

## COBOL USE OF THE STANDARD INTERFACE

To use the standard application program interface to CCP, the COBOL application program must:

1. Define the record area and the parameter list (see *Defining the Record Area and Parameter List*).
2. Set the contents of the parameter list and the record area (see *Setting the Contents of the Parameter List and Record Area*).
3. Call the communications service subroutine, identifying the program's parameter list and record area, to initiate the operation (see *Calling the Communications Service Subroutine*).
4. Examine information returned by CCP in the parameter list and record area and, for input operations, process the input data (see *Examining Returned Information*).

## DEFINING THE RECORD AREA AND PARAMETER LIST

Before your COBOL program can perform communications operations, you must define one or more record areas and parameter lists.

### Record Area

The number of record areas you must define depends upon the logic of your program. You need not always define separate record areas for input data and output data, or for operations with different terminals.

Each record area defined must be large enough to contain either the program name (if a chained task), or the terminal name, and the maximum length of data to be either received as input in the record area or to be transmitted as output from the record area. Define the record areas in the WORKING-STORAGE SECTION of the DATA DIVISION of your COBOL program.

The name field portion of the record area must be specified as an alphanumeric character field. In the following example, TERM-NAME is the name of a symbolic terminal name field that has been initialized to blanks:

```
05 TERM-NAME PIC X(6) VALUE SPACES.
```

Define the data portion of the record area as required by your formats. Unless you are using a BSCA line with the Text Transparency feature (see index entry *Terminal Attribute*), you should define all elementary data items as one of the following types:

- Alphanumeric
- Alphabetic (PIC A).
- Numeric DISPLAY (PIC 9 . . . or S9 . . . with USAGE as DISPLAY or omitted).
- Numeric zoned-decimal (PIC 9 or S9 with USAGE specified as COMP or COMPUTATIONAL).

Do not define numeric data fields of the record area with a USAGE specified as COMPUTATIONAL-3, COMP-3, COMPUTATIONAL-4, or COMP-4 unless data is being transferred over a BSCA line using Text Transparency.



## Parameter List

You must also define one or more parameter lists in the WORKING-STORAGE SECTION of your program's DATA DIVISION (see index entry *parameter list*). The first four fields of the parameter list should be defined as two-byte binary (PIC S9(4), USAGE specified as COMPUTATIONAL-4 or COMP-4) fields. Because the parameter list is defined in the WORKING-STORAGE SECTION of DATA DIVISION rather than the FILE SECTION, you can also specify initial values for these fields. The fields are, in the sequence they must be defined in the parameter list:

1. Return code field.
2. Operation code and modifier field.
3. Field used jointly for output data length, actual input data length, count of outstanding Invite Inputs, and attributes identifier.
4. Maximum input data length field.

These fields are the only fields you reference in your application program. The remaining four fields of the parameter list are not referenced directly by your COBOL program. However, they must be defined because space must be reserved for them. You can define them simply by specifying FILLER with a PICTURE of X(8). Your program should never initialize or set these fields.

*Example:* Figure 4-2 shows how to define a parameter list in a COBOL program. The operation field is initialized to 2 for a PUT operation. The output data length field is initialized to 48. This value might be the length of the first output message. The maximum input data length is initialized to 60. This value might be the total length of the data portion of a record area used with this parameter list.

## Return Code Values

The CCP ignores the contents of the return code field of the parameter list at the beginning of a communications operation. At the completion of each operation, the CCP places a binary value in this field indicating the status of the operation. This value indicates:

- The operation completed normally (value of zero) for nonchained operations; value of 14 for task chained operations)
- The operation resulted in an I/O error (negative value)
- The operation resulted in an exceptional condition (positive value)

The CCP places this value in the return code field of the parameter list before returning to the COBOL program. The COBOL program must check the return code value upon the completion of each operation. Specific return code values and their meanings are given in Appendix E. *Return Codes*.

## Operation Code Values

For each communications operation, you must set the operation code field of a parameter list to a value which indicates the specific operation being requested. You must set this value within your COBOL program. This field can be set by initializing the field in the definition of the parameter list or by moving an appropriate value into the operation code field during execution (see *Setting Fields in the Parameter List* later in this chapter).

The CCP does not change this field during the communications operation; the contents of the field are the same after completion of the operation as they were at the beginning of the operation. See Chapter 2: *Standard Application Program Interface to the CCP* for descriptions of the valid operation. Appendix D: *Operation Codes* summarizes the operation code values.

```

06
07 X
08 X   PARAMETER LIST -- INITIALIZED FOR A 'PUT' OPERATION
09 X
10 01  PARM-LIST-1.
11    05  PL1-RTC PIC S9(4) COMP-4.
12    05  PL1-OPC PIC S9(4) COMP-4 VALUE 2.
13    05  PL1-OUL PIC S9(4) COMP-4 VALUE 48.
14    05  PL1-EFL REDEFINES PL1-OUL PIC S9(4) COMP-4.
15    05  PL1-ATI REDEFINES PL1-OUL PIC S9(4) COMP-4.
16    05  PL1-INL PIC S9(4) COMP-4 VALUE 60.
17    05  FILLER PIC X(8).
18
19
20

```

Figure 4-2. Defining a Parameter List

#### SETTING THE CONTENTS OF THE PARAMETER LIST AND RECORD AREA

You must set the contents of the following areas before performing a communications operation in COBOL:

1. Parameter list fields, if different from the last operation.
2. Symbolic terminal name in the first six positions of the record area. (This can be omitted if a terminal name is not required for the operation or if the name is the same as in the previous operation.)
3. Output data in the data portion of the record area if the operation is an output operation.

#### Setting Fields in the Parameter List

You reference four parameter fields within your COBOL program:

- Return Code field.
- Operation Code field.
- Field used jointly for output length, effective input length, count of outstanding Invite Inputs, and attributes identifier.
- Maximum input length field.

You need set only the operation code and the Maximum Input Length field for input operations. For output operations, you must set the operation code and the Output Length field. For an Acquire Terminal operation, you must set the operation code and, if this Acquire Terminal also sets the terminal attributes, the Attributes Identifier field. You need never set the return code field; it is used only by the CCP to return information about the operation to your COBOL program.



**Operation Code**

Whenever a communications operation is issued, this field must contain a value indicating the operation to be performed. You can set this field when you define it in the WORKING-STORAGE SECTION of the DATA DIVISION by specifying a VALUE clause:

```
05 PL-OPC PIC S9(4) COMP-4 VALUE 4.
```

You can also set this field with a MOVE statement in the PROCEDURE DIVISION of your COBOL program. You can move either a numeric literal or a named numeric value into the operation code field of a parameter list you defined. In the following example, the numeric literal 4 (Accept Input operation) is moved into the operation code field PL-OPC:

SEQUENCE		L CON	A	B	COBOL STATEMENT															
(PAGE)	SERIAL																			
1	3	4	6	7	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	
		01																		
		02																		
		03																		
		04																		
		05																		
		06																		
		07																		
		08																		
		09																		
		10																		
		11																		
		12																		
		13																		
		14																		
		15																		
		16																		
		17																		
		18																		
		19																		
		20																		

The following example sets the Operation Code field by moving the named numeric field, ACPTIN, into it. ACPTIN is defined with the value 4.

SEQUENCE	COBOL STATEMENT
(PAGE)	SERIAL
1	3 4 6 7
01	
02	DATA DIVISION.
03	
04	WORKING-STORAGE SECTION.
05	77 ACPTIN PIC S9(4) COMP-4 VALUE 4.
06	
07	
08	01 PARM-LIST.
09	
10	05 PL-OPC PIC S9(4) COMP-4.
11	
12	
13	PROCEDURE DIVISION.
14	
15	
16	MOVE ACPTIN TO PL-OPC.
17	
18	
19	
20	

The CCP never modifies the value in the Operation Code field. You do not need to reset the field if the operation to be performed is the same as the last operation using this parameter list.

For more information on the valid operations, see the chapter *Standard Application Interface to the CCP*. Appendix D: *Operation Codes* summarizes the operations and operation code values.

*Output Length/Attributes Identifier/Count of Outstanding Invite Inputs/Effective Input Length*

The third field of the parameter list can contain one of four different values depending on the type of operation:

- Output Length
- Attributes Identifier
- Count of Outstanding Invite Inputs
- Effective Input Length

The first two values you must set; the third and fourth are returned values set by the CCP for certain operations.

You can set this field when you define it in the WORKING-STORAGE SECTION of the DATA DIVISION by means of a VALUE clause, or in the PROCEDURE DIVISION by specifying a MOVE statement, just as you set the operation code field. You can move either a numeric literal or a named numeric value into the field.

*Output Length:* For task chaining and output operations, you must place into this field the length of the data you wish to write from the record area in your program. This length does not include the six positions for the program name or the symbolic terminal name. The output operations you must set a data length for are:

- Put
- Put-No-Wait
- Put-Then-Get
- Chain Task Request
- Release and Task Chain Request

You must reset this value if the output data length differs from the last operation using this parameter list or if the field was modified by the CCP. This field is modified by CCP for the following operations:

- Get
- Put-Then-Get
- Accept Input
- Accept no-wait input (5704-SC2)
- Get Terminal Attributes
- Acquire Terminal
- Release Terminal
- Release and Task Chain

**Attributes Identifier:** If your operation code specifies an Acquire Terminal operation which sets the attributes of the terminal to be acquired, you must place into this field the identifier of the attributes set. This numeric value must correspond to the number you assigned to the desired set of attributes in an Assignment run.

**Effective Input Length:** You do not need to set this value. For each input operation, CCP places the actual length of the input data passed to your COBOL program in this field before it returns control to your program. This is the length of the data only, it does not include the length of the terminal name.

**Count of Outstanding Invite Inputs:** On Release Terminal and Release and Task Chain operations and on any input operation that results in an 08 return code (the terminal that entered the data mode escape and issued a /RELEASE command), this field is set by CCP to the number of Invite Input operations still outstanding. On accept no-wait input operations that result in a return code of 04 or 16, the count of outstanding invites is returned to your program (5704-SC2 only). If this is a multiple requesting terminal (MRT) program, this number includes not only the Invite Inputs you have issued that have not yet been satisfied by an Accept Input operation, but also the number of additional terminals that have requested your program but have not yet been served by your program.

#### **Maximum Input Data Length**

For each operation involving input data, you must enter a numeric value into the fourth field of the parameter list, indicating the maximum amount of input data you expect to receive. This value must be greater than zero and no

larger than the size of the data portion of the record area with which this parameter list is used. The value does not include the six positions at the beginning of the record area for the name field. The input operations for which you must place a value in this field are:

- Get
- Invite Input
- Accept Input
- Accept no-wait input (5704-SC2)
- Put-Then-Get
- Get Terminal Attributes
- Stop Invite Input (in case input cannot be stopped)

You can set the value of this field either when you define it in the WORKING-STORAGE SECTION of the DATA DIVISION or by means of a MOVE statement in the PROCEDURE DIVISION. CCP never modifies the value in this field. Therefore, you do not need to reset it unless the maximum input length for this operation is different from the value set in this field the last time this parameter list was used. However, if this parameter list is used with more than one record area, you may need to alter this value during execution of your COBOL program.

#### **Example of Setting Fields in the Parameter List**

Figure 4-3 shows how you can set the operation, output data length, and maximum input data length fields of a parameter list. Each operation code value is assigned a name. These names are then used on a MOVE statement that moves the named numeric values into the operation field of the parameter list. The output data length and maximum input data length fields are set by moving literals into them.

#### **Setting the Record Area**

The record area consists of a six-position name field and a data area. For an operation with a terminal, except for Accept Input and Shutdown Inquiry operations, you must place the symbolic name of the terminal to be involved with the operation in the name field. For a Task Chain Request, you must place the name of the requested program in the name field. You must also provide the data to be transmitted in the data portion of the record area when an output operation is to be performed.



# COBOL Coding Form

SYSTEM		PUNCHING INSTRUCTIONS				PAGE	OF
PROGRAM		GRAPHIC				CARD FORM #	
PROGRAMMER		DATE	PUNCH				

SEQUENCE	(PAGE)	SERIAL	A	B	COBOL STATEMENT	IDENTIFICATION
01					DATA DIVISION.	
02					}	
03					}	
04					}	
05					WORKING-STORAGE SECTION.	
06					77 ACPTIN PIC S9(4) COMP-4 VALUE 4.	
07					77 PUTNWT PIC S9(4) COMP-4 VALUE 54.	
08					77 PUTMWT PIC S9(4) COMP-4 VALUE 50.	
09					77 INVINP PIC S9(4) COMP-4 VALUE 5.	
10					77 PUTGET PIC S9(4) COMP-4 VALUE 3.	
11					77 RELTRM PIC S9(4) COMP-4 VALUE 10.	
12					77 STPINV PIC S9(4) COMP-4 VALUE 1025.	
13					}	
14					}	
15					01 PARM-LIST.	
16					05 PL-RTC PIC S9(4) COMP-4.	Return code field
17					05 PL-OPC PIC S9(4) COMP-4.	Operation code field
18					05 PL-OUT PIC S9(4) COMP-4.	Output data length field
19					05 PL-INL PIC S9(4) COMP-4.	Maximum input data length field
20					05 FILLER PIC X(8).	Required work area
					}	

Assign names to operation code values that can later be used in MOVE statements to set the operation field of the parameter list.

Return code field  
 Operation code field  
 Output data length field  
 Maximum input data length field  
 Required work area

SEQUENCE	(PAGE)	SERIAL	A	B	COBOL STATEMENT	IDENTIFICATION
01					PROCEDURE DIVISION	
02					}	
03					MOVE ACPTIN TO PL-OPC.	Move the named numeric value for the Accept Input operation into the operation field.
04					MOVE 8 TO PL-INL.	Move a numeric literal into the input length field.
05					}	
06					}	
07					MOVE 8 TO PL-OUT.	Move a numeric literal into the output length field.
08					}	
09					}	
10					MOVE PUTMWT TO PL-OPC.	Reset the operation to Put.
11					MOVE 34 TO PL-OUT.	Reset the output length to 34.
12					}	
13					}	
14					MOVE INVINP TO PL-OPC.	Reset the operation to Invite Input.
15					MOVE 8 TO PL-INL.	Reset the input length field to 8
16					}	
17					}	
18					}	
19					}	
20					STOP RUN.	

Figure 4-3. Setting Fields in the Parameter List

### Name Field

For an operation involving a terminal, the terminal name you place in a record area must have been assigned to your program. You may also identify the requesting terminal by using six blanks as the terminal name if your program is *not a multiple requesting terminal (MRT) program* (see index entry). See *Chapter 2: Standard Application Program Interface to the CCP* for more information on the valid terminal names.

For a Chain Task Request operation, you must provide the name of the program to be loaded in the name field.

You may set the name when you define the record area in the WORKING-STORAGE SECTION of the DATA DIVISION, or by means of a MOVE statement in the PROCEDURE DIVISION. You do not need to reset the terminal name if the terminal to be used is the same that was named the last time the record area was used, unless the name was modified by CCP. CCP modifies the name field of the record area in the following situations:

- Upon completion of an Accept Input operation, CCP sets the name field to the name of the program or terminal whose data is placed in the record area.
- Upon completion of any operation using the name field that was set to blanks, CCP sets the name field to the name of the requesting program or terminal.

### Output Data Area

If the operation to be performed is an output operation, you must provide the data to be transmitted in the data portion of the record area. You do not need to provide data in the record area for operations other than output operations because either the data area is not used or data is provided to your program by CCP. Data provided to your program by CCP overlays the information previously in the data portion of the data area. For example, the input data transmitted to your program by the Get part of the Put-Then-Get operation overlays the output data transmitted from your program by the Put part of the operation. See the *Chapter 2: Standard Application Program Interface to the CCP* for more information on transferring data.

**Note:** If the message to be sent is shorter than the total length of the data area, you need not clear the excess area to blanks.

### Example of Setting the Record Area

Figure 4-4 shows how you can define and set the record area when it is used for both input and output operations. Assume the CCP has set the terminal name and data area as the result of an Accept Input operation. The COBOL program then resets the data area for an output operation by moving the message "TRY AGAIN INV DATA" to the data portion of the record area. This message overlays the input data transmitted to the record area by the Accept Input operation. Later in the program, the terminal name is reset to a named alphanumeric value.

### CALLING THE COMMUNICATIONS SERVICE SUBROUTINE

Since COBOL does not include special statement types for terminal I/O operations and other communications services, the CCP provides a communications service subroutine, 'CCPCIO,' that converts the COBOL program's communications requests into a standard request to the CCP communication facilities. The functions performed by CCPCIO for the COBOL program are:

- Loads index register 2 with the address of the program's parameter list.
- Places the address of the record area into the program's parameter list.
- Invokes CCP.

The CCPCIO subroutine must be linkage edited with the COBOL application program. See *Chapter 9. Program Preparation*.

After you have set the required parameter list fields and the terminal name in the record area, and have prepared any output data, you are ready to request the CCP to perform the operation specified in the parameter list. You make this request by issuing a CALL statement specifying 'CCPCIO'. The names of your parameter list and record area must be passed as arguments to the subroutine.

The format of the CALL statement is as follows:

```
CALL 'CCPCIO' USING parameter-list-name, record-area-name.
```



### COBOL Coding Form

SYSTEM	PUNCHING INSTRUCTIONS				PAGE	OF
PROGRAM	GRAPHIC				CARD FORM #	*
PROGRAMMER	DATE	PUNCH				

SEQUENCE	(PAGE)	SERIAL	A	B	COBOL STATEMENT	IDENTIFICATION
01					DATA DIVISION.	
02						
03						
04						
05					WORKING-STORAGE SECTION.	
06						
07						
08					01 INPUT-OUTPUT-AREA.	
09					05 TERM-NAME-IO PIC X(6).	Defining the record area. Terminal Name
10					05 DATA-IN1.	
11					10 DATA-REC PIC X(8).	Data Area defined for input
12					10 FILLER PIC X(26).	
13					05 DATA-OUT REDEFINES DATA-IN1.	Data Area redefined for output
14					10 DATA-CHAR PIC X(34).	
15						
16						
17					PROCEDURE DIVISION.	
18						
19						
20						

SEQUENCE	(PAGE)	SERIAL	A	B	COBOL STATEMENT	IDENTIFICATION
01						
02						
03					MOVE 'TRY AGAIN INY DATA' TO DATA-OUT.	Move message to record area to be transmitted as output.
04						
05						
06					MOVE 'TERMA' TO TERM-NAME-IO.	Reset terminal name for next operation.
07						
08						
09						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						

Figure 4-4. Setting the Record Area

The CALL statement appears in the PROCEDURE DIVISION of your COBOL program.

In the following example, the name of the parameter list is PARM-LIST. The name of the record area is INPUT-OUTPUT-AREA.

```
CALL 'CCPCIO' USING PARM-LIST, INPUT-OUTPUT-AREA.
```

Control returns to your COBOL program at the statement immediately following the CALL statement. When the return occurs, the following actions have already taken place:

- For output operations, any output data has been accepted by CCP and, depending upon the output operation specified, has been received by the terminal. In any case, the record area is now free for you to use again.
- For input operations, any input data which was to be received in the record area is now in the record area, unless an error condition or one of several exception conditions occurred.
- For Accept Input operations, the name of the program or the symbolic terminal name of the terminal that provided the data in the record area has been set in the first six positions of the record area.
- For successful task chain operations, the requested program is placed on the program request input queue when control is returned to the requesting program.
- For operations that validly specified a blank terminal name, the symbolic terminal name of the requesting terminal has been set in the first six positions of the record area.
- For all operations, the return code field in your parameter list has been set indicating the result of the operation.
- For input operations, the actual input data length has been set in your parameter list.
- For Release Terminal operations or for input operations where the terminal has released itself from the program, the count of outstanding Invite Input operations has been set in your parameter list.

## EXAMINING RETURNED INFORMATION

After control has returned to your COBOL program from the communications service subroutine, you should examine returned information supplied by CCP, including one or more of the following:

- The return code
- The symbolic terminal name (if it was set by CCP) or the name of the program that issued the Task Chain
- Request operation
- The count of outstanding Invite Inputs, if a Release Terminal operation was performed, or if the return code from an input operation indicates the terminal was released
- The actual input data length, if an input operation was successfully performed
- The input data, if an input operation was performed

### Return Code

CCP always provides a return code after an operation. You should never assume that an operation is successful; you should always check the return code. In certain cases, you will find that no data transfer has occurred. See Appendix E for the meanings of specific return codes and see *Programming Examples*, later in this chapter, for examples of checking return codes.

You may wish to perform certain operations in your COBOL program depending upon the return code value set by CCP. The example in Figure 4-5 assumes that you want to branch to one of several procedure names depending upon the value of the return code. The program examines the return code value for the following conditions:

- The operation was successful and no exceptions occurred
- An EOT was received on a successful operation, or on an operation in which data was truncated.
- Some other exception condition occurred.
- An I/O error occurred.

Assume that all data names have been defined earlier in this program. Note the use of comments in the example.



### COBOL Coding Form

SYSTEM		PUNCHING INSTRUCTIONS				PAGE	OF
PROGRAM		GRAPHIC					
PROGRAMMER		DATE	PUNCH			CARD FORM #	*

SEQUENCE	COBOL STATEMENT	IDENTIFICATION
(PAGE) SERIAL	A	B
1 3 4 8	12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72	76 80
01		
02	ASSUME REQUIRED CONTROL FIELDS SET -- NOW REQUEST THE OPERATION	
03		
04	CALL 'CCPCIO' USING PARM-LIST-1, REC-AREA-A-1.	
05		
06	EXAMINE RETURN CODE FOR SUCCESSFUL OPERATION	
07		
08	IF P11-RTC ZERO, GO TO NORMAL-OPERATION.	
09		
10	EXAMINE THE RETURN CODE FOR I/O ERROR	
11		
12	IF P11-RTC NEGATIVE, GO TO I-O-ERROR-OPERATION.	
13		
14	DISTINGUISH BETWEEN EOT-RECEIVED AND OTHER EXCEPTIONS	
15		
16	GO TO OTHER-EXCEPTION, EOT-RECEIVED, EOT-RECEIVED	
17	DEPENDING ON P11-RTC	
18		
19	FOR VALUE GREATER THAN +3, CONTROL FALLS HERE	
20		
	OTHER-EXCEPTION.	

Figure 4-5. Examining Return Code Values



### COBOL Coding Form

SYSTEM		PUNCHING INSTRUCTIONS				PAGE	OF
PROGRAM		GRAPHIC					
PROGRAMMER		DATE	PUNCH			CARD FORM #	*

SEQUENCE	COBOL STATEMENT	IDENTIFICATION
(PAGE) SERIAL	A	B
1 3 4 8	12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72	76 80
01	DATA DIVISION.	
02	}	
03	}	
04	WORKING-STORAGE SECTION.	
05	}	
06	}	
07	01 REC-AREA-A-1.	
08	05 TERM-NAME-A PIC X(6).	
09		
10		
11	01 SAVED-INFORMATION.	
12	05 SAV-ENTRY OCCURS 5 TIMES INDEXED BY SAV-IX.	
13	10 SAV-TERM-NAME PIC X(6).	
14	}	
15	}	
16	PROCEDURE DIVISION.	
17	}	
18	}	
19	SET SAV-IX TO 1.	
20	}	
	MOVE TERM-NAME-A TO SAV-TERM-NAME (SAV-IX).	
	}	

Figure 4-6. Saving the Symbolic Terminal Name



## Examining a Returned Name

On certain operations, CCP returns the symbolic terminal name to your program's record area. You may need to examine this name.

For example, you may need to examine the name of the terminal that provided the input data. You can then compare the terminal name in the record area with a saved terminal name to associate new data with data previously received from this terminal. You can save a terminal name for later comparison by specifying the terminal name field of the record area in a MOVE statement. The field to which the terminal name is moved must be defined with a PICTURE of X(6).

The example in Figure 4-6 saves the terminal name CCP sets in the name field of the record area, TERM-NAME-A, by moving it to the field SAV-TERM-NAME. SAV-TERM-NAME is the name field in a table of saved values.

If a program can be requested from both a terminal and another program using the Task Chain Request operation, you may want to determine how the program was requested. This can be accomplished by checking for a 14 return code, indicating a Task Chain Request operation. This information is useful if a program communicates with the requestor since your program cannot communicate with a chain task requesting program.

## Referencing Saved Information

In some of your COBOL programs, you may need to save the information entered on the terminals and reference it later in your program. For example, if your program receives data from several different terminals, you may need to associate new data entered on a terminal with data previously entered on the same terminal. To do this, you must save the significant data received from every terminal you are using and identify that saved data with the name of the terminal from which it was received. You can then associate new data with the saved data by comparing the terminal name set by CCP in the record area with the saved terminal names.

One way you can save information received from each terminal is to define a table of group items. Specify the number of terminals from which information must be saved as the integer in the group item's OCCURS clause. For example, if information must be saved from five terminals, specify that the group item OCCURS 5 TIMES. Each group item consists of a set of elementary items, one of which is the terminal name. Upon completion of an Accept

Input operation, you can then search the table of saved information until you find the saved terminal name that corresponds to the name of the terminal which just transmitted data to your program. Once you have found the table entry you are searching for, you can address any field of the save information by indexing that field name with the index name.

Figure 4-7 shows how to set up a table for saved information and reference the saved information in your COBOL program. By searching the table for the saved terminal names that corresponds to the terminal name in the record area, you can associate the new data with the data that was saved.

## Effective Input Data Length

If the communications service subroutine requested an operation which transferred data to your program (Get, Accept Input, Get Attributes, Put-Then-Get, or Stop Invite Input), CCP also places the effective length of the input data into the parameter list. Because this is the length of the data that was actually received by your program, you may wish to use this length to control subscripted or indexed operations in your program. For example, you may need to scan the input data for a specific character or string of characters. To do this you must know the length of the input data you must scan.

## Count of Outstanding Invite Inputs

On a Release Terminal operation or on an input operation where the return code indicates that the terminal released itself from your program, the count of outstanding Invite Input operations is returned to your program. You may use this number to determine whether your program has any further terminals to serve or whether it can go to end of job.

## Input Data

If the operation requested by your program is an input operation that transfers data, CCP places the input data received by your program in the seventh and succeeding positions of your record area before it returns control to your COBOL program. Any excess positions, beyond the end of the actual data received, are filled with blanks by CCP, up to the maximum input length you specified for the operation. The data is then available for you to use in your program.

SYSTEM		PUNCHING INSTRUCTIONS				PAGE	OF
PROGRAM		GRAPHIC				CARD FORM #	
PROGRAMMER		DATE					

SEQUENCE	COBOL STATEMENT	IDENTIFICATION
01		
02	DATA DIVISION.	
03		
04		
05	WORKING-STORAGE SECTION.	
06		
07		
08	01 RECORD-AREA-A.	
09	05 TERM-NAME-A PIC X(6).	
10	05 REC-TYPE-A PIC X.	
11		
12		
13		
14	* ONE SET OF FIELDS FOR EACH OF 5 TERMINALS	
15		
16	01 SAVED-INFORMATION.	
17	05 SAV-ENTRY OCCURS 5 TIMES INDEXED BY SAV-IX.	
18	10 SAV-TERM-NAME PIC X(6).	
19	10 SAV-CUST-NO PIC 9(4).	
20	10 SAV-CUST-NAME PIC X(55).	
	10 SAV-LAST-REC-TYPE PIC X.	
	10 SAV-NUMBER-TRANSACS PIC 99 COMP.	
	10 SAV-TRANSAC-AMT-SUBTOTAL PIC S9(5)V99 COMP.	

Defines a table of saved values for data received from five different terminals. The group item, SAV-ENTRY, consists of 6 fields. These fields can be referenced later in your program.

SEQUENCE	COBOL STATEMENT	IDENTIFICATION
01		
02	PROCEDURE DIVISION.	
03		
04		
05	SET SAV-IX TO 1.	
06	SEARCH-LOOP.	
07	IF TERM-NAME-A = SAV-TERM-NAME (SAV-IX)	
08	GO TO FOUND-THE-ENTRY.	
09	IF SAV-IX LESS THAN 5	
10	SET SAV-IX UP BY 1	
11	GO TO SEARCH-LOOP	
12	ENTRY-NOT-FOUND.	
13		
14		
15	FOUND-THE-ENTRY.	
16		
17		
18	IF REC-TYPE-A = SAV-LAST-REC-TYPE (SAV-IX)	
19		
20		
	ADD 1 TO SAV-NUMBER-TRANSACS (SAV-IX)	

Search through the table of saved entries for the saved terminal name that is the same as the terminal name in the name field of the record area (TERM-NAME-A). When you find the corresponding terminal name proceed to FOUND-THE-ENTRY.

Once you have found the saved entry you were searching for, you can reference the saved fields by indexing the field name by SAV-IX:

- Compares input record type field with previous record from this terminal.
- Updates the number of transactions field in the saved entry by adding the new transaction from the input record.

Figure 4-7. Referencing Saved Information

## USING THE SYSTEM OPERATOR CONSOLE

If you wish to communicate with the system operator through either the 5471 Printer/Keyboard (Models 10 and 12) or CRT/Keyboard (Model 15), you must specify operations as though the device is a remote terminal. You cannot address the system operator's console by either the DISPLAY .... UPON console-name or the ACCEPT....FROM console-name statement. Instead of using these statements, you must specify either a Put or Put-Then-Get operation to a terminal named CONSOL. CONSOL is the only name that can be assigned to the system operator console.

Your program can communicate with the system operator's console at any time. To receive data from the console, you must issue a Put-Then-Get operation, which:

1. Transmits a message to the system operator; and
2. Accepts a reply from the system operator.

Control is not returned to your program until the system operator has transmitted input data to your program.

The operations that can be issued to the console are:

- Put
- Put-Then-Get
- Get Attributes

The console is available at all times to communicate with any program or to enter system operator commands. However, if the console requests a program, it cannot request another program until the first program is initiated by CCP. It is not necessary, nor is it valid, to issue an Acquire Terminal operation to the console in order to communicate with it.

*Example:* The example in Figure 4-8 uses the system operator console as the terminal for a Put-Then-Get operation to notify the system operator of an I/O error.

## COBOL PROGRAMMING CONSIDERATIONS

When writing your COBOL program, remember:

- (Model 10 and Model 12) You cannot use either the ACCEPT or the DISPLAY statements when addressing the CONSOL.
- You cannot use the Checkpoint/Restart facility of Disk Data Management. Therefore, your COBOL program cannot specify the RERUN statement.
- (Model 10 and Model 12 CCP) You must not issue a *STOP literal* statement. Programs running under the CCP are not permitted to issue halts.
- You should not use the APPLY CORE-INDEX clause in your COBOL program to create an in-storage index (master index) for randomly processed indexed files. The index is built as a result of the MSTRINDX keyword of the DISKFILE assignment statement (see *CCP System Reference*); therefore, if you use an APPLY CORE-INDEX clause, you will only add unnecessary storage size to your program.
- You cannot use the COBOL TRACE option under CCP
- The DISPLAY statement cannot be used for a printer.

## 3270 DISPLAY FORMAT FACILITY

You can use the 3270 Display Format Facility (DFF) of CCP to aid you in formatting and using the 3270 display. *Chapter 8: 3270 Display Format Facility* describes the programming requirements that are unique to using 3270 DFF, including the unique 3270 DFF operations, additional information that must be placed in the record area for certain operations, field types that are unique to the 3270, and other information.

See *Chapter 8: 3270 Display Format Facility* for an example of a COBOL program that uses the DFF to support a single requesting 3270 terminal.



### COBOL Coding Form

SYSTEM		PUNCHING INSTRUCTIONS				PAGE	OF
PROGRAM		GRAPHIC					
PROGRAMMER		DATE	PUNCH			CARD FORM #	*

SEQUENCE	A	B	COBOL STATEMENT	IDENTIFICATION
01				
02			DATA DIVISION.	
03			WORKING-STORAGE SECTION.	
04			}	
05			}	
06	X		PARAMETER LIST	
07	X			
08			01 PARM-LIST.	
09			05 PL-RTC PIC S9(4) COMP-4.	
10			05 PL-OPC PIC S9(4) COMP-4.	
11			05 PL-OUT PIC S9(4) COMP-4.	
12			05 PL-INL PIC S9(4) COMP-4.	
13			05 FILLER PIC X(8).	
14	X			
15	X		RECORD AREA	
16	X			
17			01 INPUT-OUTPUT-AREA.	
18			05 TERM-NAME-IO PIC X(6).	
19			05 MSG-DATA PIC X(22).	
20	X			
			PROCEDURE DIVISION.	
			}	
			(see next Page)	

Define the parameter list and the record area.

SEQUENCE	A	B	COBOL STATEMENT	IDENTIFICATION
01				
02	X		PREPARE ERROR MESSAGES HERE AND SET	
03	X		UP PARAMETER LIST FOR PUT THEN GET TO CONSOLE	
04	X			
05			PUT-GET.	
06			MOVE 3 TO PL-OPC.	● Set operation code for Put-then-Get.
07			MOVE 28 TO PL-OUT.	● Set output length field.
08			MOVE 2 TO PL-INL.	● Set maximum input data length field.
09	X			
10			INSERT TERMINAL NAME CONSOL AND	
11	X		ERROR MESSAGE TO RECORD AREA	
12	X			
13	X			
14			MOVE 'TP IO ERROR' TO MSG-DATA.	● Set data portion of record area.
15			MOVE 'CONSOL' TO TERM-NAME-IO.	● Set terminal name field of record area to CONSOL.
16	X			
17	X		DO PUT THEN GET OPERATION TO CONSOL	
18	X			
19			CALL 'COPCIO' USING PARM-LIST, INPUT-OUTPUT-AREA.	● Call the Communications Service Subroutine to perform the Put then Get operation.
20			}	

Figure 4-8. Using the Console

## PROGRAMMING EXAMPLES

Two programming examples are explained in this section:

*Example 1* – A COBOL program that supports a single requesting 3270 without using the Display Format Facility.

*Example 2* – A COBOL program that supports multiple requesting terminals.

See Chapter 8 for an example of a COBOL program that uses the 3270 Display Format Facility to support a single requesting 3270 terminal.

### Example 1

Figures 4-9, 4-10, and 4-11 show the flowcharts, messages, and listing for a sample single requesting terminal (SRT) COBOL program. This program transmits two messages to a 3270 Model 1 Display System (480 character screen). The first message from the program requests the terminal operator to enter a room number. The program uses the room number as the relative record number to access a disk file whose records contain guest and rate information about the room. This information is then formatted and displayed as the second message transmitted to the 3270 terminal. Figure 4-9 also shows how these messages appear on the 3270 terminal.

Because this program is a *single requesting terminal* (SRT) program (see index entry) without any program-selected terminals, it can receive data from and transmit data to only one 3270 terminal. However, multiple copies of this program could be in main storage at the same time, each communicating with a different 3270 Display System. (If multiple copies are in core at the same time, the disk file must be specified as sharable during the Assignment stage – see index entry *disk file sharing*.)

### Formatting the Messages for the 3270 Display

Because this sample program does not use the Display Format Facility, this sample program must set all formatting control characters for the 3270 display screen into the data portion of the record area and transmit them as part of the messages to be displayed. Figure 4-10 shows the messages and the 3270 control characters as they are transmitted to the 3270 terminal. You can find the meanings of each of the 3270 screen format characters shown in Figure 4 of the *General Information Binary Synchronous Communications System Reference Library Manual*, GA27-3004.

The printable control characters are set by defining them as part of the message in the VALUE clauses of the record area definition. Blanks are left in the VALUE clauses where the unprintable format characters will be set by MOVE statements later in the program.

The unprintable format characters (hexadecimal values that have no corresponding printable character in 96-column card code) are set by first coding the hexadecimal format characters as decimal values and initializing fields to these values (PSEUDO and PSEUDO2). The fields assigned these decimal values are then redefined so that the COBOL program can access these values, which are stored in hexadecimal internally, as the format characters. These redefined fields (INSERT-CURSOR, START-FIELD, SET-BUFFER-ADDR, and ESCAPE) are then moved into the appropriate position in the message. The notes to the right of the listing in Figure 4-11 explain the statements used by this program to format the 3270 display screen. You will also find the comments in the listing helpful.

### Notes Concerning this Sample Program

- Message Mode was defined during the Assignment Stage for the 3270 terminal used by this program. (See TERMATTR statement in *CCP System Reference Manual*.) This eliminates the need to do repetitive input operations until EOT is received.
- To run this program using a terminal other than the 3270, you must remove all coding dependent on the 3270. This includes all screen formatting specifications and 3270 screen control characters within the data.
- This program will not accept data with the program request.
- Two different lengths are used for the output length field of the parameter list because the two messages transmitted by this sample program have different length.
- This program specifies a PUT operation and a GET operation using six blanks as the terminal name. The CCP places the name of the 3270 terminal being used in the terminal name field of the record area after the first PUT operation is performed.

- To keep this sample program simple, return code checking is kept to a minimum. You may want to do more return code checking in your application programs. For example, when you issue Accept Input you should check for the Shutdown Requested return code (04). Also, if data mode escape is allowed in the CCP system, programs should check for return code 08 (terminal has released itself from the program). It is recommended that each installation design its own return code checking and console communication routines so that a standard is established that is satisfactory to the installation and can be used by all application programs.
- This program does not check the length of the input data because the terminal operator is requested to enter a three-digit room number. If less than three digits are entered, the program branches to the EXIT-DONE procedure and the program is canceled. However, you may want to check the input data length in your application programs.
- Since there are only two different screen formats used by this program, they are both contained within the program. For more complete applications, you might store the screen formats on disk and read them when they are needed by your program.
- You could also use the Get Attributes operation in this program. If you do not know whether the 3270 Model 1 or the 3270 Model 2 will request the program, you can issue a Get Attributes operation to find out which type of terminal requested the program.
- If this program were coded and specified as a multiple requesting terminal (MRT) program with a MRTMAX=1 keyword on the PROGRAM assignment statement (see *CCP System Reference Manual*), multiple copies of the program would not be allowed in main storage at the same time. As the program is written, multiple copies could be in main storage at the same time and the disk file must be specified as sharable (FILES keyword of PROGRAM assignment statement).

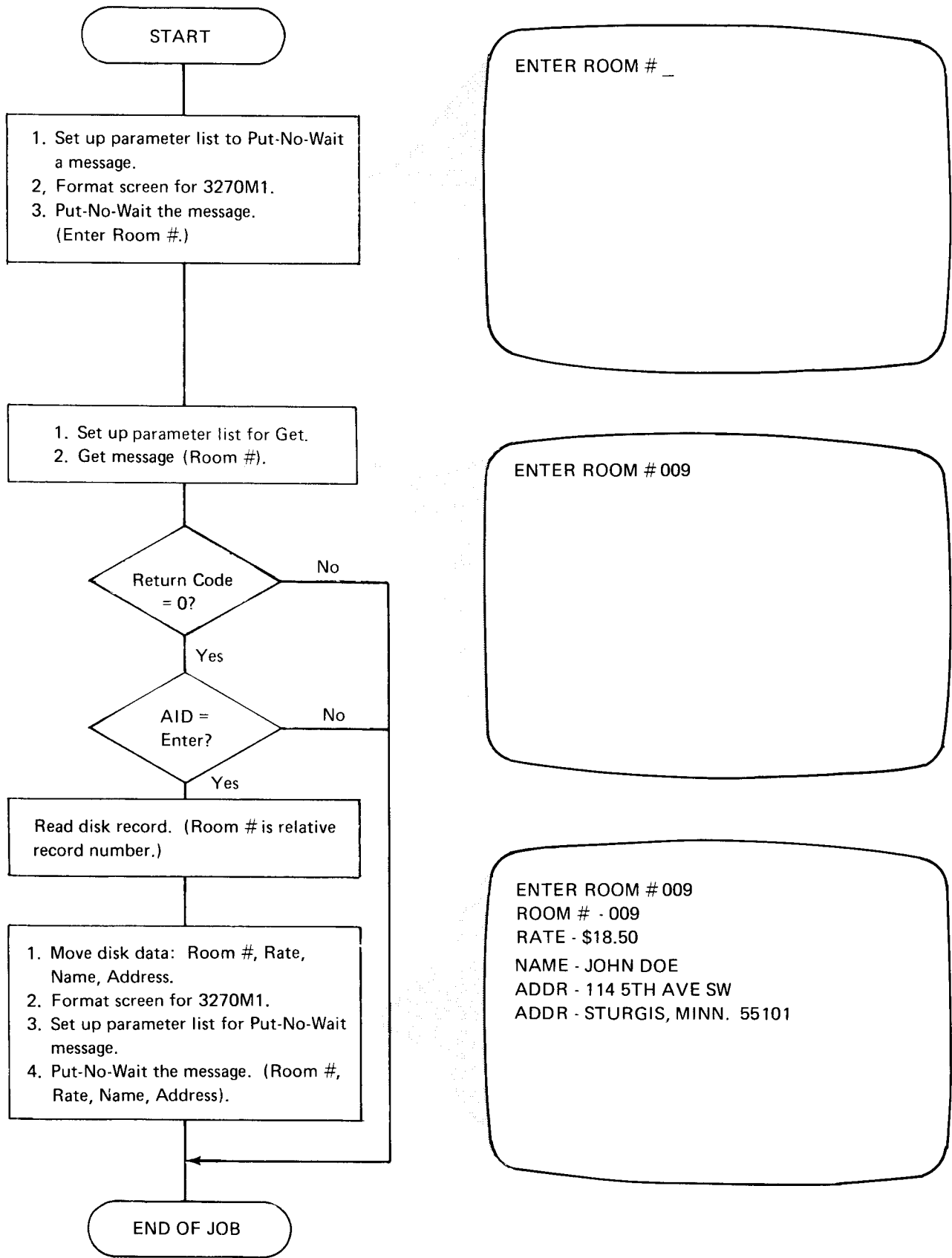
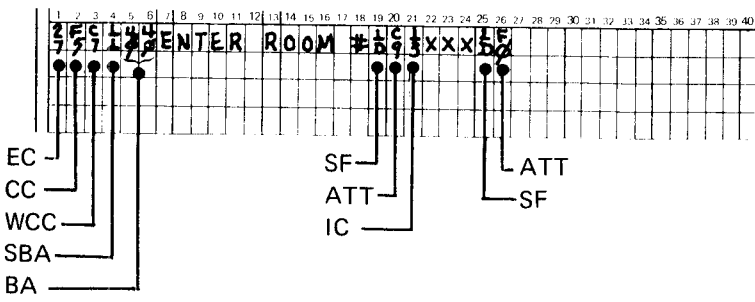
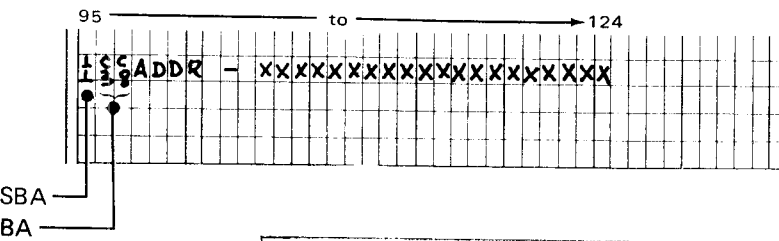
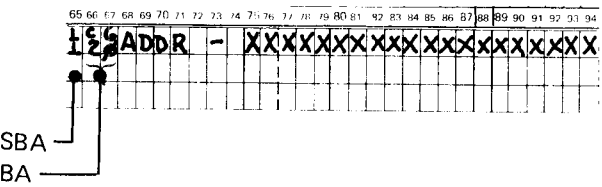
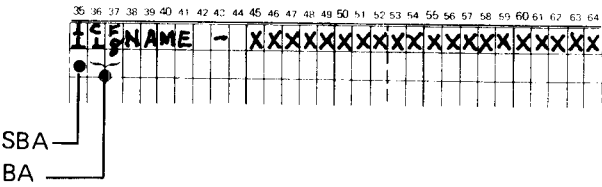
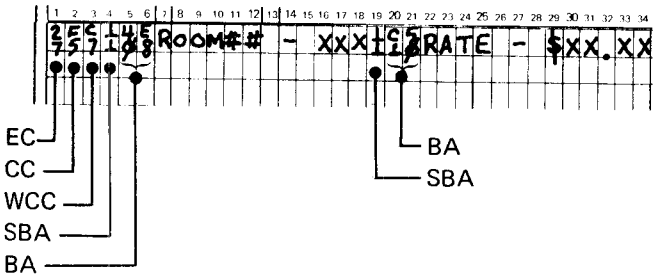


Figure 4-9. Program Logic of Example 1 (COBOL SRT Program)

First Message



Second Message



SF	— Start Field	CC	— Command Code
ATT	— Attribute Character	WCC	— Write Control Character
IC	— Insert Cursor	SBA	— Set Buffer Address
x	— Data Character	BA	— Buffer Address of first character position in the field
EC	— Escape Character		

Figure 4-10. Message Formats for Example 1 (COBOL SRT Program)



IBM SYSTEM/3 AMERICAN NATIONAL STANDARD COBOL

STNO -A...B... COBOL SOURCE STATEMENTS .....IDENTFCN SEQ/NO S

```

PROCESS MAP,LIST,GODECK
1 IDENTIFICATION DIVISION.
2 PROGRAM-ID. SRCOBL.
3 REMARKS. THIS IS A SAMPLE SINGLE REQUESTING TERMINAL PROGRAM
  DESIGNED TO RUN UNDER CCP. A 3 DIGIT ROOM NUMBER
  WHOSE VALUE IS BETWEEN 1 AND 10 IS ENTERED FROM A 3270
  TERMINAL. THE ROOM NUMBER IS RECEIVED BY THIS PROGRAM,
  AND IS USED TO ACCESS A FILE WHOSE RECORDS CONTAIN
  GUEST AND RATE INFORMATION ABOUT THE ROOM. THE PROGRAM
  RECEIVES THIS INFORMATION FROM THE DISK AND FORMATS IT
  AND THEN SENDS IT BACK TO THE 3270 TO BE DISPLAYED.
4 ENVIRONMENT DIVISION.
5 CONFIGURATION SECTION.
6 SOURCE-COMPUTER. IBM-S3.
7 OBJECT-COMPUTER. IBM-S3.
8 INPUT-OUTPUT SECTION.
9 FILE-CONTROL.
10 SELECT GUEST-FILE ASSIGN TO DA-5444-R-GSTFILE
  ACCESS MODE IS RANDOM
  ACTUAL KEY IS GUEST-KEY.
11 DATA DIVISION.
12 FILE SECTION.
*****
* THIS IS THE RECORD THAT CONTAINS THE GUEST AND RATE *
* INFORMATION FROM THE DISK FILE. *
*****
13 FD GUEST-FILE LABEL RECORDS ARE STANDARD
  DATA RECORD IS GUEST-REC.
14 01 GUEST-REC.
15 02 RPG-DATA PIC X.
16 02 ROOM-NMBR PIC X(3).
17 02 ROOM-RATE PIC 99V99.
18 02 GUEST-NAME PIC X(20).
19 02 ADDR-HOME PIC X(20).
20 02 ADDR-WORK PIC X(20).
21 02 FILLER PIC X(2).
22 WORKING-STORAGE SECTION.
*****
* INDEPENDENT FIELDS AND CONSTANTS AND KEYS *
*****
23 77 GUEST-KEY PIC S9(7) COMP.
*****
* THESE ARE SPECIAL HEX-DECIMAL CHARACTERS USED FOR FORMATTING *
* THE 3270 SCREEN *
*****
24 01 PSEUDO PIC 9999 COMP-4 VALUE 4893. ①
25 01 IC-SF REDEFINES PSEUDO.
26 05 INSERT-CURSOR PIC X. ②
27 05 START-FIELD PIC X.
28 01 PSEUDO2 PIC 9999 COMP-4 VALUE 4391. ③
29 01 SBA-ESC REDEFINES PSEUDO2.
30 05 SET-BUFFER-ADDR PIC X. ④
31 05 ESCAPE PIC X.

```

- ① Initialize PSEUDO using decimal values corresponding to the hexadecimal values for Insert Cursor and Start Field. These values will be internally represented in binary:  
 Insert Cursor = X'13'  
 Start Field = X'1D'  
 X'131D' = decimal 4893 (see *Note*)
- ② Redefine PSEUDO to make the resulting two hexadecimal values available to be manipulated individually in the program.
- ③ Initialize PSEUDO2 using decimal values corresponding to the hexadecimal values for Set Buffer Address and Escape Character. These values will be internally represented in binary:  
 Set Buffer Address = X'11'  
 Escape Character = X'27'  
 X'1127' = decimal 4391 (see *Note*)
- ④ Redefine PSEUDO2 to make the resulting two hexadecimal values available to be manipulated individually in the program.

*Note:* The hexadecimal value to be converted to decimal must never exceed X'270F', or the resulting decimal value will exceed four digits and will require a three-byte field. If this occurs, rearrange the order of the hexadecimal fields to see if it results in a lower decimal value. If it does not, use a three-byte field and place a X'00' filler in the first byte.

Figure 4-11 (Part 1 of 3). Example 1 – COBOL SRT Program

```

/*****
* INPUT-OUTPUT PARAMETER LIST
*****/
32 01 PARM-LIST.
33 05 PL-RTC PIC S9(4) COMP-4.
34 05 PL-DPC PIC S9(4) COMP-4 VALUE 54.
35 05 PL-OUTL PIC S9(4) COMP-4 VALUE 26.
36 05 PL-INL PIC S9(4) COMP-4 VALUE 11.
37 05 FILLER PIC X(8).
/*****
* THIS IS THE INPUT/OUTPUT AREA
*****/
38 01 INPUT-OUTPUT-AREA.
39 05 I-O-TERM PIC X(6) VALUE SPACES.
40 05 I-O-AREA.
41 10 MSG1 PIC X(21) VALUE ' 5G ENTER ROOM # I'.
42 10 ROOM-NUM PIC X(3) VALUE SPACES.
43 10 CHRS1 PIC X(5) VALUE ' 0- $'.
44 10 CHRS2 PIC 99.99.
45 10 NAME PIC X(10) VALUE ' A8NAME - '.
46 10 NAM-CHR PIC X(20) VALUE SPACES.
47 10 ADDR1 PIC X(10) VALUE ' B-ADDR - '.
48 10 AD1-CHR PIC X(20) VALUE SPACES.
49 10 ADDR2 PIC X(10) VALUE ' CHADDR - '.
50 10 AD2-CHR PIC X(20) VALUE SPACES.
51 05 I-O-CHARS REDEFINES I-O-AREA.
52 10 I-O-CHAR PIC X OCCURS 124 INDEXED BY INDX.}
53 05 I-O-AREA2 REDEFINES I-O-AREA.
54 10 ROOM PIC X(15).
55 10 ROOM-NM PIC X(3).
56 10 RATE PIC X(8).
57 10 FILLER PIC X(98).
58 05 INPUT-AREA REDEFINES I-O-AREA.
59 10 DEVICE PIC X.
60 10 CNTRL-U PIC X.
61 10 AID PIC X.
62 10 CRS-ADD PIC X(2).
63 10 SBA PIC X.
64 10 SBA-ADD PIC X(2).
65 10 RM-NUM PIC X(3).
66 10 FILLER PIC X(113).
/*****
* NOW BEGIN EXECUTION BY OPENING THE DIRECT ACCESS FILE
*****/
67 PROCEDURE DIVISION.
68 OPEN-THE-FILE.
69 OPEN INPUT GUEST-FILE.
/*****
* INSERT THE HEXADEcimal CONTROL CHARACTERS INTO DATA STREAM
*****/
70 FIRST-CHARS.
71 MOVE ESCAPE TO I-O-CHAR(1).
72 MOVE SET-BUFFER-ADDR TO I-O-CHAR(4).
73 NEXT-CHARS.
74 MOVE START-FIELD TO I-O-CHAR(19).
75 MOVE INSERT-CURSOR TO I-O-CHAR(21).
/*****
* THIS FIELD IS DEFINED TO PREVENT DATA FROM BEING ENTERED
* BEYOND THIS POSITION ON THE SCREEN.
*****/
76 MOVE START-FIELD TO I-O-CHAR(25).
/*****
* DO PUT MESSAGE NO WAIT OPERATION TO 3270 TERMINAL
* REQUESTING THE ROOM NUMBER BE ENTERED
*****/

```

Return Code Field  
Operation Code Field  
Output Length Field  
Maximum Input Length Field  
Required CCP Work Area

Terminal Name Field

Data area for Messages: Initialize the contents of message fields to be displayed and of any printable 3270 formatting characters. Leave blanks for any unprintable 3270 control characters (characters that cannot be represented by a character in the COBOL character set). The blank fields are set by MOVE statements later in the program. The first half of this definition is used for the first message; the second half is used only for the second message. The first part of the second message will be added later by overlaying the first message.

Redefine the data area with an index so each position in the area can be referenced separately.

Redefine the data area to set up the first part of the second message.

Redefine data area for Get operation.

Move the hexadecimal values for the remaining 3270 formatting control characters to appropriate positions in the data area. These characters are unprintable.

Figure 4-11 (Part 2 of 3). Example 1 – COBOL SRT Program

```

77     CALL 'CCPCIO' USING PARM-LIST, INPUT-OUTPUT-AREA.
*****
*     DU GET OPERATION FROM 3270 TERMINAL AND OBTAIN ROOM NUMBER *
*****
78     MOVE 1 TO PL-OPC.
79     CALL 'CCPCIO' USING PARM-LIST, INPUT-OUTPUT-AREA.
*****
*     IF THE RETURN CODE IS NOT ZERO GO TO END OF JOB *
*****
80     IF PL-RTC NOT = 0 GO TO EXIT-DONE.
*****
*     CHECK TO SEE IF THE ENTER KEY WAS PRESSED, IF IT WAS NOT GO *
*     TO END OF JOB. *
*****
82     IF I-O-CHAR(3) NOT = QUOTE GO TO EXIT-DONE.
*****
*     VALIDITY CHECK THE ROOM NUMBER IF ROOM NUMBER BAD GO TO END *
*     OF JOB *
*****
84     MOVE RM-NUM TO GUEST-KEY.
85     IF GUEST-KEY LESS THAN 1 GO TO EXIT-DONE.
87     IF GUEST-KEY GREATER THAN 10 GO TO EXIT-DONE.
89     MOVE RM-NUM TO ROOM-NM.
*****
*     READ RECORD FROM DIRECT ACCESS FILE. THE ROOM NUMBER *
*     REPRESENTS THE RELATIVE POSITION OF THE RECORD IN THE FILE*
*****
90     READ GUEST-FILE INVALID KEY GO TO EXIT-DONE.
*****
/*****
*     MOVE THE ROOM NUMBER, RATE PER DAY, THE NAME AND ADDRESS OF *
*     THE GUEST INTO THE OUTPUT AREA *
*****
92     MOVE ' 1G YROOM # - ' TO ROOM.
93     MOVE ' AGRATE ' TO RATE.
94     MOVE GUEST-NAME TO NAM-CHR.
95     MOVE ROOM-RATE TO CHRS2.
96     MOVE ADDR-HOME TO AD1-CHR.
97     MOVE ADDR-WORK TO AD2-CHR.
*****
*     INSERT THE HEXADEcimal CONTROL CHARACTERS INTO DATA STREAM *
*****
98     PERFORM FIRST-CHARS.
99     MOVE SET-BUFFER-ADDR TO I-O-CHAR(19).
100    MOVE SET-BUFFER-ADDR TO I-O-CHAR(35).
101    MOVE SET-BUFFER-ADDR TO I-O-CHAR(65).
102    MOVE SET-BUFFER-ADDR TO I-O-CHAR(95).
*****
*     SET UP PARAMETER LIST FOR A PUT MESSAGE NO WAIT *
*****
103    MOVE 54 TO PL-OPC.
104    MOVE 124 TO PL-OUTL.
*****
*     DO PUT MESSAGE NO WAIT OPERATION TO THE 3270 TERMINAL *
*****
105    CALL 'CCPCIO' USING PARM-LIST, INPUT-OUTPUT-AREA.
106    EXIT-DONE.
107    CLOSE GUEST-FILE.
108    STOP RUN.

```

Move the message, data, and printable 3270 control characters for the first part of the second message into the data area of the record area, overlaying the first message.

Move the hexadecimal values for the 3270 formatting control characters that are not already set in the data area into the appropriate positions of the data area. These are the unprintable control characters.

Figure 4-11 (Part 3 of 3). Example 1 – COBOL SRT Program

## Example 2

Figures 4-12, 4-13, and 4-14 show the flowchart, input/output messages, and listing for a sample COBOL multiple requesting terminal (MRT) program designed to run under the CCP. This program handles up to four MLTA requesting terminals. The terminal operator enters a seven-digit number preceded by a +, -, or N. The CCP transmits this signed number to the COBOL program. The COBOL program:

- Adds the number to the value in the accumulator field associated with the terminal that transmitted the data if the first position is +
- Subtracts the number from the accumulator if the first position is -
- Releases the terminal if the first position is N

If a value was either added or subtracted, the new value accumulated for the terminal is inserted into the message *CURRENT VAL = sxxxxxxxxx ENTER DATA* and the message is sent to the terminal.

This sample program also checks for several error conditions and transmits the appropriate error message to the terminal.

This sample program is not designed to show the most effective way of performing operations. Instead, it shows a variety of ways to do things. It uses a variety of operation codes that show how data can be associated with a terminal by defining a save area for the terminal names and accumulated data. It frequently checks return codes; but you can do even more return code checking if you wish. Data entered by the terminal operator must be fixed length. To allow variable length input fields, you could include a subroutine in your program to check the effective input length returned in the parameter list and align the data correctly. This program communicates with the console in addition to the requesting terminals.

The notes to the right of the listing in Figure 4-14 and the comments in the listing explain each section of the sample program.

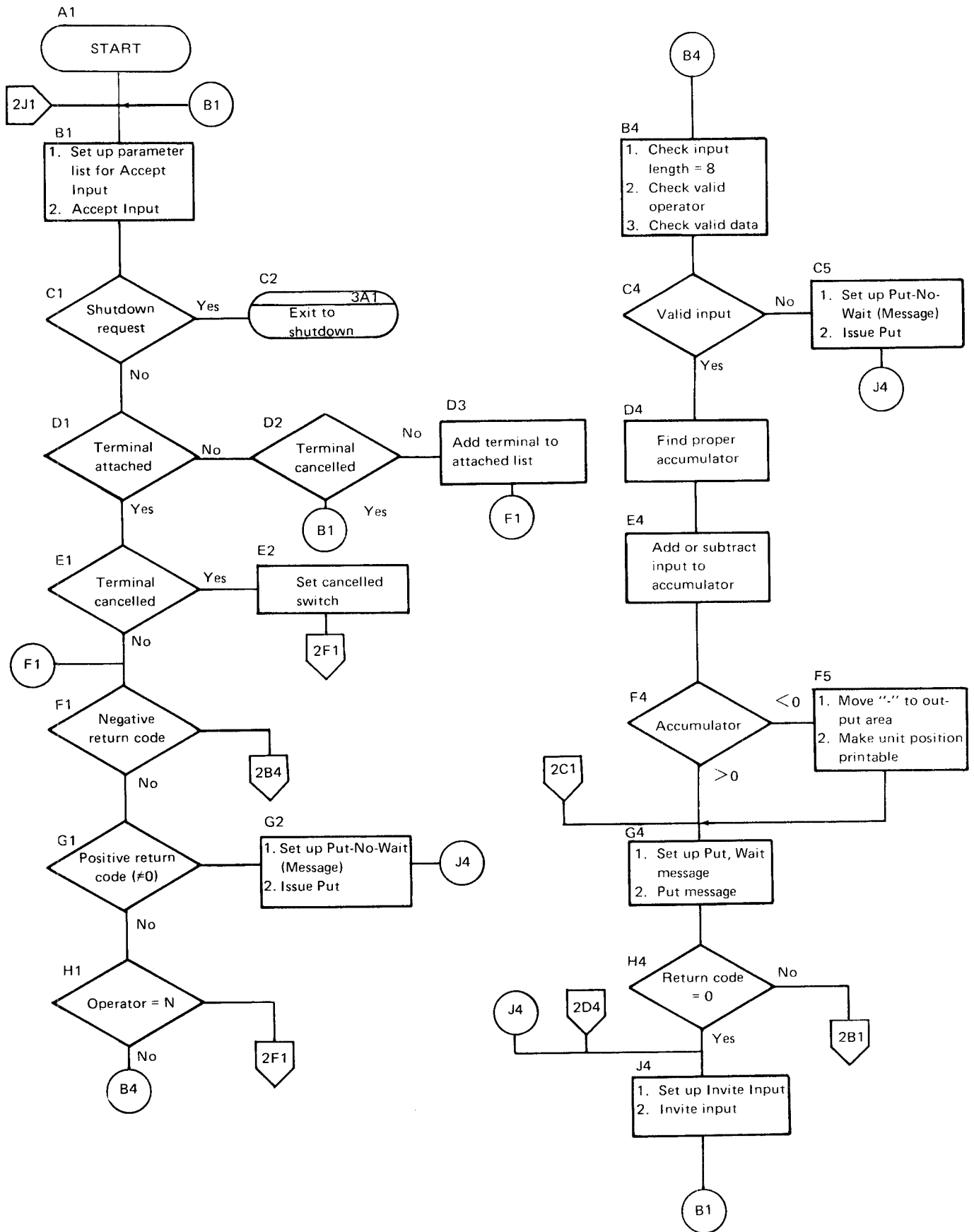


Figure 4-12 (Part 1 of 3). Program Logic of Example 2 (COBOL MRT Program)

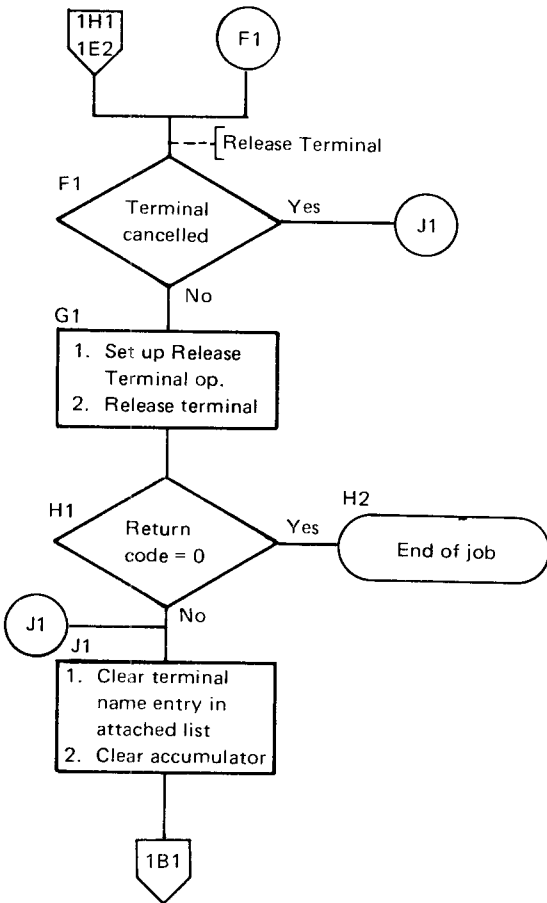
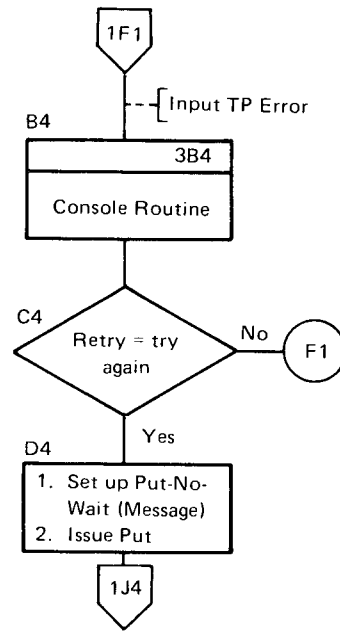
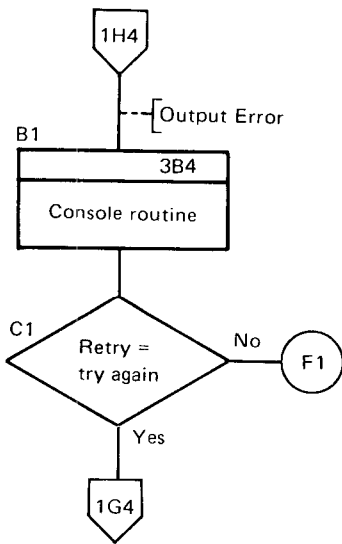
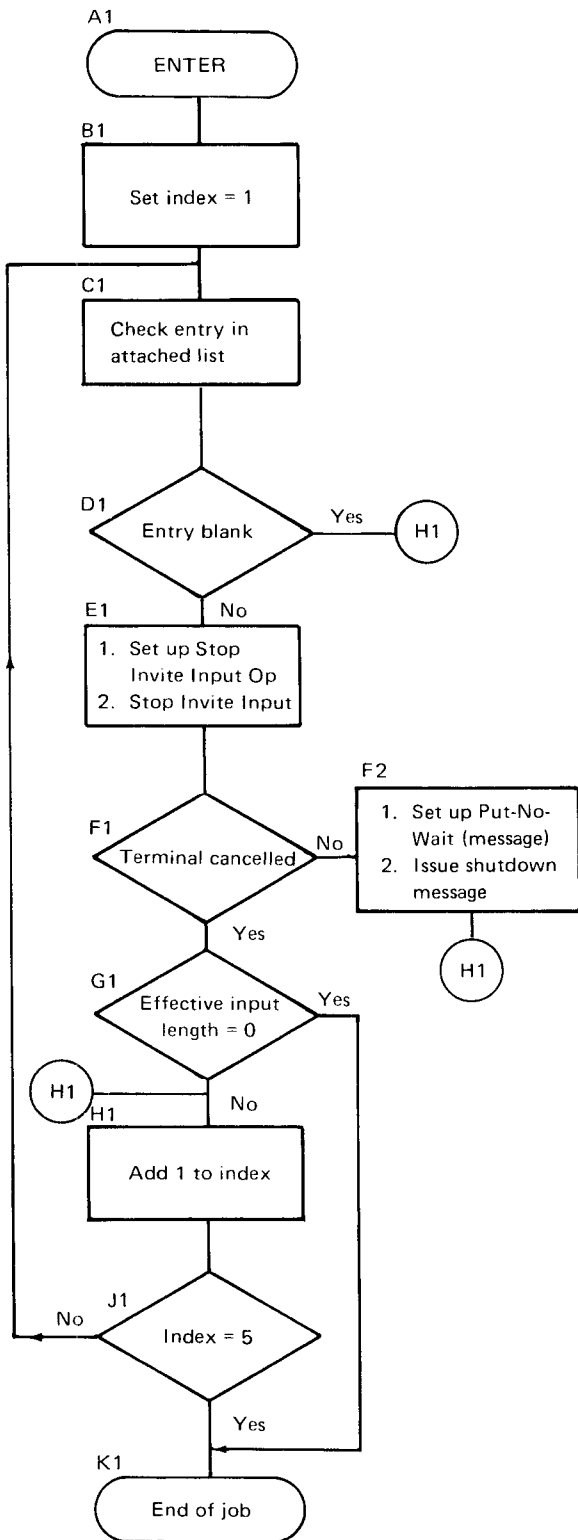


Figure 4-12 (Part 2 of 3). Program Logic of Example 2 (COBOL MRT Program)

### Shutdown Routine



### Console Routine

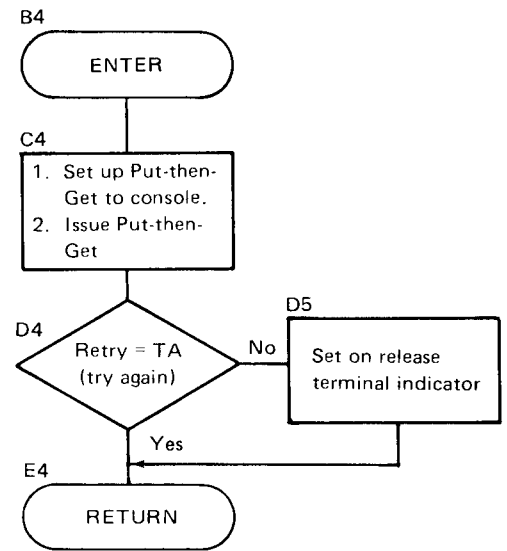


Figure 4-12 (Part 3 of 3). Program Logic of Example 2 (COBOL MRT Program)

### Input Data Entered by Terminal Operator

1	2	3	4	5	6	7	8	9	10	11
S	X	X	X	X	X	X	X	X		

A fixed length numeric field where S is a +, -, or N and X is a numeric digit. All eight positions must be entered, except when N is entered in the first position.

### Data Entered by System Operator on 5471 Printer/Keyboard (Models 10 and 12) or CRT/Keyboard (Model 15)

1	2	3	4	5	6	7	8	9	10	11
T	A									
C	C									

In response to the messages INPUT TP ERROR TNAME-cccccc and OUTPUT TP ERROR TNAME-cccccc to the console, the system operator replies TA if he wants to try again. Any other reply (cc) causes the terminal to be released.

### Output to the Console

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
I	N	P	U	T		T	P		E	R	R	O	R		T	N	A	M	E	-	C	C	C	C	C	C				
O	U	T	P	U	T		T	P		E	R	R	O	R		T	N	A	M	E	-	C	C	C	C	C				

These messages are transmitted to the console (cccccc = terminal name).

### Output to Terminal

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	
C	U	R	R	E	N	T		V	A	L	=	S	X	X	X	X	X	X	X	X		E	N	T	E	R		D	A	T	A					
T	R	Y		A	G	A	I	N		I	N	V		D	A	T	A																			
T	R	Y		A	G	A	I	N		T	P		E	R	R	O	R																			
C	C	P		S	H	T	D	W	N		L	A	S	T		R	E	C	-		T	P		E	R	R	O	R								
C	C	P		S	H	T	D	W	N		L	A	S	T		R	E	C	-		B	A	D		D	A	T	A								
C	C	P		S	H	T	D	W	N		L	A	S	T		R	E	C	-		S	X	X	X	X	X										
C	C	P		S	H	T	D	W	N		L	A	S	T		R	E	C	-		N	O		D	A	T	A									

Transmitted with value in accumulator associated with the terminal.  
 Issued if data is invalid.  
 Issued if system operator replies TA (negative return code on Accept Input).  
 Issued for negative return code on Stop Invite Input.  
 Issued for positive return code other than 10 on Stop Invite Input.  
 Issued for return code of 0 on Stop Invite Input.  
 Issued for return code of 10 on Stop Invite Input

Figure 4-13. Input and Output Message Formats for Example 2 (COBOL MRT Program)



STNO -A...B... COBOL SOURCE STATEMENTS .....IDENTFCN SEQ/NO S

```

PROCESS MAP,LIST
1 IDENTIFICATION DIVISION.
2 PROGRAM-ID. MRCOB1.
3 REMARKS. THIS IS A SAMPLE MULTIPLE REQUESTING TERMINAL PROGRAM
  DESIGNED TO RUN UNDER CCP. A NUMBER CONSISTING
  OF UP TO 7 NUMERIC CHARACTERS AND A + OR - OPERATOR IS
  TRANSMITTED TO THIS PROGRAM BY ANY ONE OF UP TO 4
  TERMINALS. THE + OR - OPERATION IS PERFORMED AND THE
  RESULTS PLACED IN THE ACCUMULATOR ASSOCIATED WITH THE
  TERMINAL THAT REQUESTED THE OPERATION. THE VALUE IN THE
  ACCUMULATOR IS THEN SENT BACK TO THE TERMINAL FOR
  DISPLAY. IF AN N IS ENTERED AS THE OPERATOR THE
  TERMINAL WILL BE RELEASED. IF THIS IS THE ONLY TERMINAL
  LINKED WITH THE PROGRAM, THE PROGRAM WILL END
  EXECUTION.
4 ENVIRONMENT DIVISION.
5 CONFIGURATION SECTION.
6 SOURCE-COMPUTER. IBM-S3.
7 OBJECT-COMPUTER. IBM-S3.
8 DATA DIVISION.
9 WORKING-STORAGE SECTION.
*****
* INDEPENDENT FIELDS AND CONSTANTS *
*****
10 77 SWITCH PIC 9 COMP-4.
*****
* OPERATION CODES *
*****
11 77 ACPPIN PIC S9(4) COMP-4 VALUE 4.
12 77 PUTNWT PIC S9(4) COMP-4 VALUE 54.
13 77 PUTMWT PIC S9(4) COMP-4 VALUE 50.
14 77 INVINP PIC S9(4) COMP-4 VALUE 5.
15 77 PUTGET PIC S9(4) COMP-4 VALUE 3.
16 77 RELTRM PIC S9(4) COMP-4 VALUE 10.
17 77 STPINV PIC S9(4) COMP-4 VALUE 1025.
*****
* TERMINAL DATA STORAGE ARRAY *
*****
18 01 TERMINAL-STORAGE-ARRAY.
19 05 TERMINAL-ENTRY OCCURS 4 INDEXED BY TERM-X.
20 10 TERM-NAME PIC X(6).
21 10 ACCUMULATOR PIC S9(11) COMP.
/*****
* COMMUNICATIONS AREA *
* CCP-COBOL INTERPACE PARAMETER LIST *
*****
22 01 PARM-LIST.
23 05 PL-RTC PIC S9(4) COMP-4.
24 05 PL-OPC PIC S9(4) COMP-4.
25 05 PL-OUT PIC S9(4) COMP-4.
26 05 PL-EPL REDEFINES PL-OUT PIC S9(4) COMP-4.
27 05 PL-INL PIC S9(4) COMP-4.
28 05 FILLER PIC X(8).
*****
* THIS IS THE INPUT OUTPUT AREA *
*****
29 01 INPUT-OUTPUT-AREA.
30 05 TERM-NAME-IO PIC X(6).
31 05 DATA-IN.
32 10 OPERATOR PIC X.
33 10 DIGITS PIC 9(7) COMP.
34 10 FILLER PIC X(26).
35 05 DATA-IN1 REDEFINES DATA-IN.
36 10 DATA-REC PIC X(8).
37 10 FILLER PIC X(26).
38 05 DATA-OUT REDEFINES DATA-IN.
39 10 DATA-CHAR PIC X(34).
40 05 ACCUM-OUT REDEFINES DATA-IN.
41 10 FILLER PIC X(12).
42 10 ACCUM-VALUE PIC +++++++9.
43 10 FILLER PIC X(11).
44 05 MSG-DATA REDEFINES DATA-IN.
45 10 MSG-DATA1 PIC X(22).
46 10 MSG-DATA2 PIC X(8).
47 10 FILLER PIC X(4).
48 05 MS-DATA REDEFINES DATA-IN.
49 10 MS-DATA1 PIC X(6).
50 10 MS-DATA2 PIC X(17).
51 10 MS-DATA3 PIC X(6).
52 10 FILLER PIC X(5).
53 05 TRY-AGAIN REDEFINES DATA-IN.
54 10 TA PIC X(2).
55 10 FILLER PIC X(32).

```

Define a symbolic name for each operation used in this program. These names are used in the procedure division instead of the numeric operation code values.

Set up a save area for the four terminals used by this program and their accumulators.

Return Code Field  
 Operation Code Field  
 Output Length Field  
 Input Length Field  
 Required Work Area

Terminal Name Field

The data portion of the record area is first defined for an eight-position field whose first position is for the operator (+, -, or N). It is then redefined for output and various messages.

Figure 4-14 (Part 1 of 5). Example 2 – COBOL MRT Program

```

/*****
* INITIALLY SET UP THE TERMINAL ARRAY IN ORDER THAT THIS
* PROGRAM BE RE-ENTRANT
*****/
56 PROCEDURE DIVISION.
57 INIT.
58 SET TERM-X TO 1.
59 LOOP.
60 MOVE ZEROES TO ACCUMULATOR(TERM-X).
61 MOVE SPACES TO TERM-NAME(TERM-X).
62 SET TERM-X UP BY 1.
63 IF TERM-X LESS THAN 5 GO TO LOOP.
*****
* SET UP PARAMETER LIST FOR ACCEPT INPUT OPERATION
*****
65 ACCEPT-INPUT.
66 MOVE 0 TO SWITCH.
67 MOVE ACPTIN TO PL-OPC.
68 MOVE 8 TO PL-INL.
*****
* DO ACCEPT INPUT OPERATION
*****
69 CALL 'CCPCIO' USING PARM-LIST, INPUT-OUTPUT-AREA.
*****
* CHECK TO SEE IF SHUTDOWN HAS BEEN REQUESTED
*****
70 IF PL-RTC = 4 GO TO SHUTDOWN.
*****
* DETERMINE IF TERMINAL HAS ALREADY BEEN ATTACHED, IF IT HAS
* GO CHECK THE RETURN CODE
*****
72 SET TERM-X TO 1.
73 TERM-SEARCH.
74 IF TERM-NAME-IO = TERM-NAME(TERM-X) GO TO TERM-FOUND.
75 SET TERM-X UP BY 1.
76 IF TERM-X LESS THAN 5 GO TO TERM-SEARCH.
*****
* CHECK TO SEE IF THE TERMINAL HAS BEEN CANCELED, IF IT HAS
* RETURN TO ACCEPT INPUT IF NO INVITE INPUTS ARE OUTSTAND-
* ING. IF INVITES OUTSTANDING GO TO ACCEPT INPUT.
*****
79 IF PL-RTC NOT = 8 GO TO ADD-TERM.
81 IF PL-EPL = 0 GO TO DONE-EXIT.
83 GO TO ACCEPT-INPUT.
84 ADD-TERM.
*****
* ADD TERMINAL NAME TO ATTACHED LIST IF NOT ALREADY PRESENT
* LOCATE A BLANK 6 CHARACTER TERMINAL NAME SPACE IN THE
* TERMINAL DATA STORAGE ARRAY
*****
85 SET TERM-X TO 1.
86 BLANK-SEARCH.
87 IF TERM-NAME(TERM-X) NOT = SPACES
88 SET TERM-X UP BY 1
89 GO TO BLANK-SEARCH.
*****
* NOTE: NO MORE THAN 4 TERMINALS WILL BE ALLOWED TO
* COMMUNICATE WITH THIS PROGRAM IF ASSIGNMENT SPECIFIES
* 4 TERMINALS
*****
90 MOVE TERM-NAME-IO TO TERM-NAME(TERM-X).
91 GO TO VALIDITY-CHK.
/*****
* CHECK TO SEE IF TERMINAL HAS BEEN CANCELLED. IF IT HAS AND
* THERE ARE NO INVITES OUTSTANDING GO TO EXIT. IF THERE ARE
* INVITES OUTSTANDING GO REMOVE FROM ACTIVE TERMINAL ARRAY.
*****
92 TERM-FOUND.
93 IF PL-RTC = 8 GO TO CANCEL-CHK.
*****
* CHECK FOR INPUT ERROR INDICATIONS, ISSUE ERROR MESSAGE IF
* RETURN CODE NOT = 0, OR IF LENGTH NOT WITHIN RANGE
* CHECK FOR VALID OPERATOR, IF OPERATOR EQUAL TO N GO RELEASE
* TERMINAL
*****
95 VALIDITY-CHK.
96 IF PL-RTC LESS THAN 0 GO TO PUT-GET.
98 IF PL-RTC GREATER THAN 0 GO TO INVALID-DATA.
100 IF OPERATOR = 'N' GO TO CANCEL-CHK.
102 IF PL-OUT NOT = 8 THEN GO TO INVALID-DATA.
104 IF OPERATOR = '+' GO TO ADD-ACCUM.
106 IF OPERATOR = '-' GO TO SUB-ACCUM.
*****
* ASSUME BAD OPERATOR, ISSUE INVALID DATA MESSAGE
*****
108 INVALID-DATA.
109 MOVE 'TRY AGAIN INV DATA' TO DATA-OUT.
110 MOVE 18 TO PL-OUT.
111 GO TO PUT-NO-WAIT.

```

Initialize the accumulators to zeros and the terminal name save areas to blanks.

Set the value for the accept input operation in the operation code field of the parameter list.

Set the input field length to 8, the length of the expected input.

Determine if the terminal name for the terminal that transmitted the input data is in the terminal name save area. If it is, the data is added to the value in the accumulator associated with that terminal. If it is not in the save area and the terminal is not cancelled, the terminal name is added to the save area.

If the terminal name is not already in the terminal name save area, it is moved to the first blank terminal name field in the save area.

If the return code is not equal to 0, indicating a successful operation, or if the input length field is not equal to 8, an error message is transmitted to the terminal.

If the first position of the input field is +, the data is added to the accumulator associated with the terminal that transmitted the data. If the first position is -, the data is subtracted from the terminal. If the first position is N, the terminal is checked to see if it is cancelled.

If the first position of the input data is not +, -, or N, a message is transmitted to the terminal.

Figure 4-14 (Part 2 of 5). Example 2 -- COBOL MRT Program

```

/*****
*   NOW ADD THE VALUE RECEIVED AS INPUT TO THE VALUE IN THE
*   ACCUMULATOR
*****
112 ADD-ACCUM.
113   ADD DIGITS TO ACCUMULATOR(TERM-X) .
114   GO TO DISPLAY-ACCUM.
*****
*   SUBTRACT THE VALUE RECEIVED AS INPUT FROM THE VALUE IN THE
*   ACCUMULATOR
*****
115 SUB-ACCUM.
116   SUBTRACT DIGITS FROM ACCUMULATOR(TERM-X).
*****
*   SET UP TO DISPLAY THE RESULTS RESIDING IN THE ACCUMULATOR
*   TO THE TERMINAL THAT REQUESTED THE OPERATION
*****
117 DISPLAY-ACCUM.
118   MOVE 'CURRENT VAL=+          ENTER DATA' TO DATA-OUT.
119   MOVE ACCUMULATOR(TERM-X) TO ACCUM-VALUE.
*****
*   SET UP PARM-LIST FOR PUT MESSAGE WAIT
*****
120   MOVE PUTMWT TO PL-OPC.
121   MOVE 34 TO PL-OUT.
*****
*   DO PUT MESSAGE WAIT OPERATION
*****
122   CALL 'CCPCIO' USING PARM-LIST, INPUT-OUTPUT-AREA.
*****
*   CHECK RETURN CODE TO SEE IF OPERATION WAS SUCCESSFUL, IF
*   RETURN CODE NOT EQUAL TO 0 GO ISSUE ERROR MESSAGE
*****
123   IF PL-RTC NOT = 0 MOVE 6 TO SWITCH
125   GO TO PUT-GET.
*****
*   SET UP PARAMETER LIST FOR INVITE INPUT
*****
126 INVITE-INPUT.
127   MOVE INVIMP TO PL-OPC.
128   MOVE 8 TO PL-INL.
*****
*   DO INVITE INPUT OPERATION
*****
129   CALL 'CCPCIO' USING PARM-LIST, INPUT-OUTPUT-AREA.
130   GO TO ACCEPT-INPUT.

```

Insert accumulated value associated with the terminal in the output message and display on the terminal.

Figure 4-14 (Part 3 of 5). Example 2 – COBOL MRT Program

```

/*****
* HANDLE SHUTDOWN REQUEST BY ISSUING STOP INVITES TO ALL
* OUTSTANDING INVITE INPUTS PREVIOUSLY ISSUED
*****
131 SHUTDOWN.
132 SET TERM-X TO 1.
133 STOP-SEARCH.
134 IF TERM-NAME(TERM-X) = SPACES GO TO INCR-INDEX.
136 MOVE TERM-NAME(TERM-X) TO TERM-NAME-IO.
*****
* SET UP PARAMETER LIST FOR STOP INVITE INPUT
*****
137 MOVE STPINV TO PL-OPC.
138 MOVE 8 TO PL-INL.
*****
* DO STOP INVITE INPUT OPERATION
*****
139 CALL 'CCPCIO' USING PARM-LIST, INPUT-OUTPUT-AREA.
*****
* IF TERMINAL NOT CANCELLED, THEN ISSUE SHUTDOWN MESSAGE
* IF CANCELLED THEN IF NO INVITES OUTSTANDING GO TO EXIT,
* OTHERWISE GO SET UP FOR NEXT STOP INVITE OPERATION.
*****
140 IF PL-RTC NOT = 8 GO TO SET-UP.
142 IF PL-EPL = 0 GO TO DONE-EXIT.
144 GO TO INCR-INDEX.
145 SET-UP.
*****
* SET UP PARAMETER LIST FOR PUT NO WAIT
*****
146 MOVE PUTNWT TO PL-OPC.
147 MOVE 30 TO PL-OUT.
*****
* INSERT PROPER SHUTDOWN MESSAGE TO TERMINAL REFERENCED IN
* TERMINAL DATA ARRAY. THE MESSAGE IS SET ACCORDING TO
* THE RETURN CODE
*****
148 IF PL-RTC LESS THAN 0 MOVE 'TP ERROR' TO MSG-DATA2
150 GO TO DISPLAY-OUT.
151 IF PL-RTC = 10 MOVE ' NO DATA' TO MSG-DATA2
153 GO TO DISPLAY-OUT.
154 IF PL-RTC = 0 MOVE DATA-REC TO MSG-DATA2
156 GO TO DISPLAY-OUT.
157 MOVE 'BAD DATA' TO MSG-DATA2.
158 DISPLAY-OUT.
159 MOVE 'CCP SHUTDOWN LAST REC - ' TO MSG-DATA1.
*****
* DO PUT NO WAIT OPERATION
*****
160 CALL 'CCPCIO' USING PARM-LIST, INPUT-OUTPUT-AREA.
161 INCR-INDEX.
162 SET TERM-X UP BY 1.
163 IF TERM-X = 5 GO TO DONE-EXIT.
165 GO TO STOP-SEARCH.

```

Find every terminal name in the save area and issue Stop Invite Input to it. If the terminal has not been cancelled, a shutdown message is issued to it.

*Note:* When the last terminal attached to an MRT program is processed, issue a Release Terminal operation to that terminal in order to check the count of outstanding Invite Inputs. If the count is greater than zero, the program can issue an Accept Input operation. For example, suppose an MRT program is servicing the maximum number of requestors and one or more additional requests are queued to the program. If the program receives a shutdown-requested return code (O4) and goes to end of job without checking the count of outstanding Invite Inputs, the program terminates with a 2C termination code (going to end of job with outstanding Invite Inputs), and each of the queued terminals receives an S06 message (program cancelled – shutdown).

Figure 4-14 (Part 4 of 5). Example 2 – COBOL MRT Program

```

/*****
*   PREPARE INPUT OR OUTPUT ERROR MESSAGES HERE AND SET UP
*   PARAMETER LIST FOR PUT THEN GET TO CONSOL
*****/
166  PUT-GET.
167      MOVE PUTGET TO PL-OPC.
168      MOVE 29 TO PL-OUT.
169      MOVE 2 TO PL-INL.
*****/
*   INSERT TERMINAL NAME = CONSOL, TERMINAL NAME WHERE ERROR
*   OCCURRED, AND INPUT OR OUTPUT ERROR MESSAGE
*****/
170      MOVE ' INPUT TP ERROR TNAME =' TO MS-DATA.
171      IF SWITCH = 6 MOVE 'OUTPUT' TO MS-DATA1.
173      MOVE TERM-NAME-IO TO MS-DATA3.
174      MOVE 'CONSOL' TO TERM-NAME-IO.
*****/
*   DO PUT GET OPERATION TO CONSOL
*****/
175      CALL 'CCPCIO' USING PARM-LIST, INPUT-OUTPUT-AREA.
*****/
*   MOVE TERMINAL NAME BACK TO TERMINAL NAME AREA IN TERMINAL
*   DATA STORAGE ARRAY. CHECK FOR REPLY REQUESTING TO TRY
*   AGAIN-- TA, IF TA NOT PRESENT THEN GO DISCONNECT
*****/
176      MOVE TERM-NAME(TERM-X) TO TERM-NAME-IO.
177      IF TA NOT = 'TA' GO TO CANCEL-CHK.
179      IF SWITCH = 6 GO TO DISPLAY-ACCUM.
*****/
*   IF OUTPUT ERROR MESSAGE THEN GO TRY TO OUTPUT AGAIN
*   IF INPUT ERROR MESSAGE THEN GO TRY TO INPUT AGAIN
*****/
181      MOVE 'TRY AGAIN TP ERROR' TO DATA-OUT.
182      MOVE 18 TO PL-OUT.
/*****
*   SET UP PUT NO WAIT PARAMETER LIST
*****/
183  PUT-NO-WAIT.
184      MOVE PUTNWT TO PL-OPC.
*****/
*   DO PUT NO WAIT OPERATION
*****/
185      CALL 'CCPCIO' USING PARM-LIST, INPUT-OUTPUT-AREA.
186      GO TO INVITE-INPUT.
*****/
*   CHECK TO SEE IF THIS TERMINAL HAS BEEN CANCELLED. IF IT HAS
*   AND THERE ARE NO INVITES OUTSTANDING GO TO EXIT,
*   OTHERWISE GO CLEAR FROM ACTIVE LIST. IF IT HAS NOT BEEN
*   CANCELLED DO A RELEASE TERMINAL OPERATION.
*****/
187  CANCEL-CHK.
188      IF PL-RTC NOT = 8 GO TO RELEASE-TERM.
190      IF PL-EPL = 0 GO TO DONE-EXIT.
192      GO TO CLEAR-ENTRY.
*****/
*   SET UP PARAMETER LIST FOR RELEASE TERMINAL OPERATION
*****/
193  RELEASE-TERM.
194      MOVE RELTRM TO PL-OPC.
*****/
*   DO RELEASE TERMINAL OPERATION
*****/
195      CALL 'CCPCIO' USING PARM-LIST, INPUT-OUTPUT-AREA.
196      IF PL-EPL = 0 GO TO DONE-EXIT.
*****/
*   INITIALIZE THE TERMINAL DATA STORAGE ARRAY ENTRY FOR THE
*   RELEASED TERMINAL TO BLANKS AND ZERO THE ACCUMULATOR AND
*   RETURN TO ACCEPT INPUT
*****/
198  CLEAR-ENTRY.
199      MOVE SPACES TO TERM-NAME(TERM-X).
200      MOVE ZEROES TO ACCUMULATOR(TERM-X).
201      GO TO ACCEPT-INPUT.
202  DONE-EXIT.
203  STOP RUN.

```

If an input error occurred, the message 'INPUT TP ERROR TNAME xxxxxx' is issued to the console (xxxxxx = terminal on which the error occurred). If switch equals 6, a similar output error message is built and issued.

If the system operator keys in TA, the terminal name of the terminal on which the error occurred is placed in the terminal name field of the record area and the operation is retried. If the operator keys in any other characters, the terminal name for which the error occurred is placed in the record area and, if the terminal has not been cancelled, a release terminal operation is issued to it.

When a terminal is released, reinitialize the accumulator to zeros and the terminal name save area to blanks.

Figure 4-14 (Part 5 of 5). Example 2 – COBOL MRT Program



To request CCP communication services, you must write your FORTRAN programs using the standard application program interface, described in Chapter 2.

This standard interface is composed of the following elements:

- Communications Service Subroutine
- Parameter List
- Record Area

*Note:* This chapter assumes that you are familiar with the FORTRAN language. For more information on writing and executing FORTRAN programs, see the publication *IBM System/3 FORTRAN IV Reference Manual*, SC28-6874.

#### FORTRAN USE OF THE STANDARD INTERFACE

To use the standard application program interface to the CCP, your FORTRAN application program must:

1. Define the record area and the parameter list (see *Defining the Record Area and Parameter List*).
2. Set the contents of the parameter list and the record area (see *Setting the Contents of the Parameter List and Record Area*).
3. Call the communications service subroutine, identifying the program's parameter list and record area, to initiate the operation (see *Calling the Communications Service Subroutine*).
4. Examine information returned by the CCP in the parameter list and record area and, for input operations, process the input data (see *Examining Returned Information*).

#### DEFINING THE RECORD AREA AND PARAMETER LIST

Before your FORTRAN program can perform communications operations, you must define one or more record areas and parameter lists.

#### Record Area

The number of record areas you must define depends upon the logic of your program. You need not always define separate record areas for input data and output data, or for operations with different terminals.

Each record area defined must be large enough to contain the name field and the maximum length of data to be received as input in the record area or to be transmitted as output from the record area. Define each record area you require as an array using an explicit specification statement. Define the array as type `INTEGER*2`. You may specify an initial value for the elements of your record area array by using the `DATA` statement.

Define the data portion of the record area as required by your record formats. You should define all data items as literal or unpacked data unless data is to be transferred over a BSCA line using Text Transparency (see index entry *terminal attributes*), when you can define data fields of the record area as binary, packed, or hexadecimal.

Many FORTRAN application programs require that the same record areas be used for records with different formats. By defining each record area array needed by the program and using `EQUIVALENCE` statements, you can redefine the record area array in a different format. The `EQUIVALENCE` statement specifies that the redefined record area format shares the same storage locations as the original record area array definition.

*Example:* Figure 5-1 shows how to define a record area whose record may be in either of two formats. The `EQUIVALENCE` statement assigns the array `LTERM` to the same storage locations used by the first six elements of `MAREA`, `ARRAY1` to the same locations used by the next 30 elements of `MAREA`, and `ARRAY2` to the same locations used by the last 20 elements of `MAREA`. The `DATA` statement initializes the six elements of the terminal name array, `LTERM`, to blanks.







## FORTRAN Coding Form

PROGRAM		PUNCHING INSTRUCTIONS	GRAPHIC PUNCH																																																																												
PROGRAMMER		DATE																																																																													
LINE	STATEMENT NUMBER	CONT	FORTRAN STATEMENT																																																																												
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	E																		
C		X																																																																													
C		X	PARAMETER LIST ARRAY INITIALIZED FOR A PUT OPERATION																																																																												
C		X	INTEGEX2 LSTPRM(8)																																																																												
			EQUIVALENCE (LRTC,LSTPRM(1)),(LOPC,LSTPRM(2)),																																																																												
	1		(LOUTL,LSTPRM(3)),(LINL,(LSTPRM(4)))																																																																												
			DATA LOPC,LOUTL,LINL/2,48,60/																																																																												

Figure 5-2. Defining a Parameter List Array

The CCP places this value in the return code field of the parameter list before returning to your FORTRAN program. The FORTRAN program must check the return code value upon the completion of each operation. Specific return code values and their meanings are given in *Appendix F: Return Codes*.

*Operation Code Values*

For each communications operation, you must set the operation code field of a parameter list to a value which indicates the specific operation being requested. You must set this value within your FORTRAN program. This field can be set by initializing the field in the definition of the parameter list array or by moving constants into the operation code array element during execution (see *Setting Fields in the Parameter List* later in this chapter).

The CCP does not change this field during the communications operation; the contents of the field are the same after completion of the operation as they were at the beginning of the operation. See *Chapter 2: Standard Application Interface to the CCP* for descriptions of the valid operations, *Appendix E: Operations Codes* summarizes the operations and operation code values.

**SETTING THE CONTENTS OF THE PARAMETER LIST AND RECORD AREA**

You must set the contents of the following areas before performing a communications operation in FORTRAN:

1. Parameter list array, if different from the last operation.
2. The program name or the symbolic terminal name in the first six elements of the record area array. (This step can be omitted if a terminal name is not required for the operation, or if the name is the same as in the previous operation.)
3. Output data in the data portion of the record area array if the operation is an output operation.

**Setting Fields in the Parameter List**

You reference four parameter fields within your FORTRAN program.

- Return Code field.
- Operation Code field.
- Field used jointly for output length, effective input length, count of outstanding Invite Inputs, and attributes identifier.
- Maximum input/output length field.

You need set only the operation code field and the maximum input/output length field for input operations. If you are doing an output or an Acquire Terminal operation, you must also set the field used as the output length or attributes identifier. You need never set the return code field; it is used only by the CCP to return information about the operation to your FORTRAN program.

**Operation Code**

Whenever a communications operation is issued, this field must contain a value indicating the operation to be performed. You can set this field when you define the parameter list array by specifying a DATA statement:

```
DATA OPC/2/
```

You can also set this field by specifying an assignment statement. You can assign it either a numeric value or a numeric variable. In the following example, the operation code element of the parameter list, LOPC, is assigned the value 2.

The image shows an IBM program card with the following fields:

- PROGRAM
- PROGRAMMER
- Statement 6: LOPC = 2

The following example sets the operation code array element, LOPC, by moving the numeric variable, PUTOP, into it. PUTOP is defined with the value 2.

The image shows an IBM program card with the following fields:

- PROGRAM
- PROGRAMMER
- Statement 6: LOPC = PUTOP

The CCP never modifies the value in the operation code field. You do not need to reset the field if the operation to be performed is the same as the last operation using this parameter list.

For more information on the valid operations, see the chapter *Standard Application Interface to the CCP*. Appendix D: *Operation Codes* summarizes the operations and operation code values.

**Output Length/Attributes Identifier/Count of Outstanding Invite Inputs/Effective Input Length**

The third field of the parameter list can contain one of four different values depending on the type of operation:

- Output Length
- Attributes Identifier
- Count of Outstanding Invite Inputs
- Effective Input Length

The first two values you must set; the third and fourth are returned values set by the CCP for certain operations.

You can set this field when you define the parameter list array by means of a DATA statement, or by means of an assignment statement, just as you set the operation code field. You can assign it either a numeric value or a numeric variable.

**Output Length:** For output operations, you must place into this field the length of the data you wish to write from the record area in your program. This length does not include the six elements at the beginning of the record area array for the name field. This length must be less than or equal to the output length specified for the fourth field of the parameter list. The output operations you must set a data length for are:

- Put
- Put-No-Wait
- Put-Then-Get
- Chain Task Request
- Release and Task Chain

You must reset this value if the output data length differs from the last operation using this parameter list or if the field was modified by the CCP. This field is modified by the CCP for the following operations:

- Get
- Put-Then-Get
- Accept Input
- Accept No-Wait Input (5704-SC2)
- Get Terminal Attributes
- Acquire Terminal
- Release Terminal
- Release and Task Chain

**Attributes Identifier:** If your operation code specifies an Acquire Terminal operation which sets the attributes of the terminal to be acquired, you must place into this field a value that identifies the attributes you want to assign to the terminal. This numeric value must correspond to the number you assigned to the desired set of attributes in an Assignment run.

**Effective Input Length:** You do not need to set this value. For each input operation, the CCP places the actual length of the data passed to your FORTRAN program in this field before it returns control to your program.

**Count of Outstanding Invite Inputs:** On a Release Terminal operation and on any input operation that results in a 08 return code (terminal entered data mode escape and issued

a /RELEASE command), this field is set by the CCP to the number of Invite Input operations still outstanding. If this is a multiple requesting terminal (MRT) program, this number includes not only the Invite Inputs you have issued that have not yet been satisfied by an Accept Input operation, but also the number of additional terminals that have requested your program but are not yet being served by your program.

#### *Maximum Input Length/Output Data Length*

For each operation involving input data, you must enter a numeric value into the fourth field of the parameter list indicating the maximum amount of input data you expect to receive. For each operation involving output data, you must enter a numeric value indicating the maximum amount of output data you expect to transmit (in this respect, the FORTRAN communications interface differs from the standard interface defined in Chapter 2). This output length must be greater than or equal to the output length specified in the third field of the parameter list, but *no greater than the size of the data portion of the record area with which this parameter list is used*, or unpredictable results can occur. The value does not include the six elements at the beginning of the record area array for the terminal name. The input operations for which you must place a value in this field are:

- Get
- Invite Input
- Accept Input
- Accept No-Wait Input (5704-SC2)
- Put-Then-Get
- Get Terminal Attributes
- Stop Invite Input (in case input cannot be stopped)

The output operations for which you must place a value in this field are:

- Put
- Put-Then-Get
- Put-No-Wait
- Chain Task
- Release and Task Chain

You can set the value of this field when you define the parameter list area by specifying a DATA statement or by specifying it in an assignment statement. CCP never modifies the value in this field. Therefore, you do not need to reset it unless the maximum input/output length for this operation is different from the value set in this field the last time this parameter list was used. However, if this parameter list is used with more than one record area, you may need to alter this value during execution of your FORTRAN program.

#### *Example of Setting Fields in the Parameter List*

Figure 5-3 shows how you can set the operation, output data length, and maximum input/output data length fields of a parameter list. The maximum input/output data length element is set by initializing it to 125 in a DATA statement. It does not need to be reset unless you wish to receive (Get) or transmit (Put) data longer than 125. The operation code element and the output length element are set by assigning them numeric values.

#### **Setting the Record Area**

The record area consists of a six-position name field and a data area. For an operation with a terminal, except for Accept Input and Shutdown Inquiry operations, you must place the symbolic name of the terminal to be involved with the operation. For Task Chain Request, you must place the name of the requested program in the name field. You must also provide the data to be transmitted in the data elements of the record area array when an output operation is to be performed.

#### *Name Field*

For operations involving a terminal the name you place in a record area array must have been assigned to your program. You may also identify the requesting terminal by using six blank elements as the terminal name if your program is *not* a *multiple requesting terminal (MRT)* program (see index entry). See *Chapter 2: Standard Application Program Interface to CCP* for more information on the valid terminal names.

For a Task Chain Request operation, you must provide the name of the program to be loaded in this field.

You may set the name field when you define the record area array by specifying a DATA statement or by specifying it in an assignment statement. You need not reset the terminal name array elements if the terminal to be used is the same that was named the last time the record area array was used, unless the name was modified by CCP. CCP modifies the terminal name field of the record area in the following situations:

- Upon completion of an Accept Input operation, CCP sets the name field to the name of the program or terminal whose data is placed in the record area array.
- Upon completion of any operation using the field name array element set to blanks, CCP sets the name element to the name of the requesting terminal.

#### *Output Data Area*

If the operation to be performed is an output operation, you must provide the data to be transmitted in the data portion of the record area. You do not need to provide data in the record area for operations other than output operations because either the data area is not used or data is provided to your program by CCP in this area. Data provided to your program by CCP overlays the information previously in the data portion of the data area. For example, the input data transmitted to your program by the Get part of the Put-Then-Get operation overlays the output data transmitted from your program by the Put part of the operation. See the *Chapter 2: Standard Application Program Interface to CCP* for more information on transferring data.

*Note:* If the message to be sent is shorter than the total length of the data area, you need not clear the excess area to blanks.

#### *Example of Setting the Record Area*

Figure 5-4 shows how you can define and set the record area when it is used for both input and output operations. Assume CCP has set the terminal name and data area as the result of an Accept Input operation. The FORTRAN program then resets the data area for an output operation by moving the message "TRY AGAIN INV DATA" to the data portion of the record area array. This message overlays the input data transmitted to the record area array by the Accept Input operation. Later in the program, the terminal name is reset.

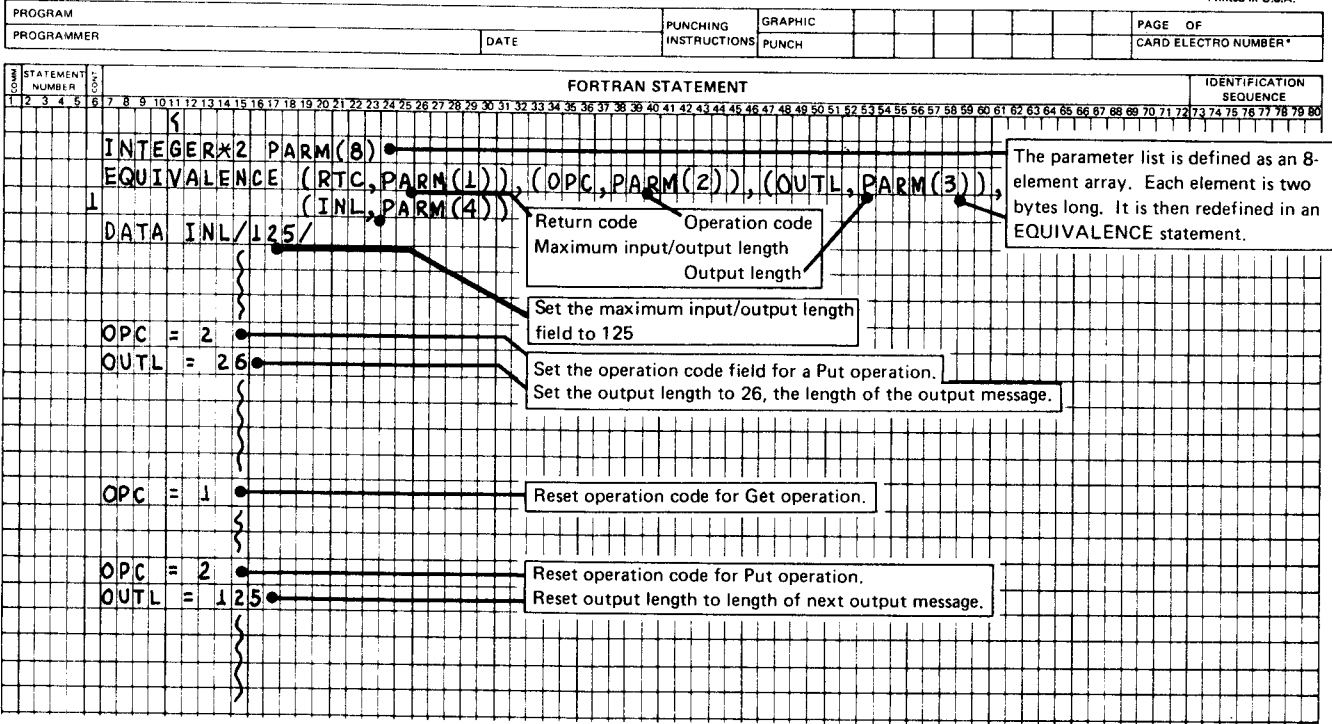


Figure 5-3. Setting Elements in the Parameter List Array

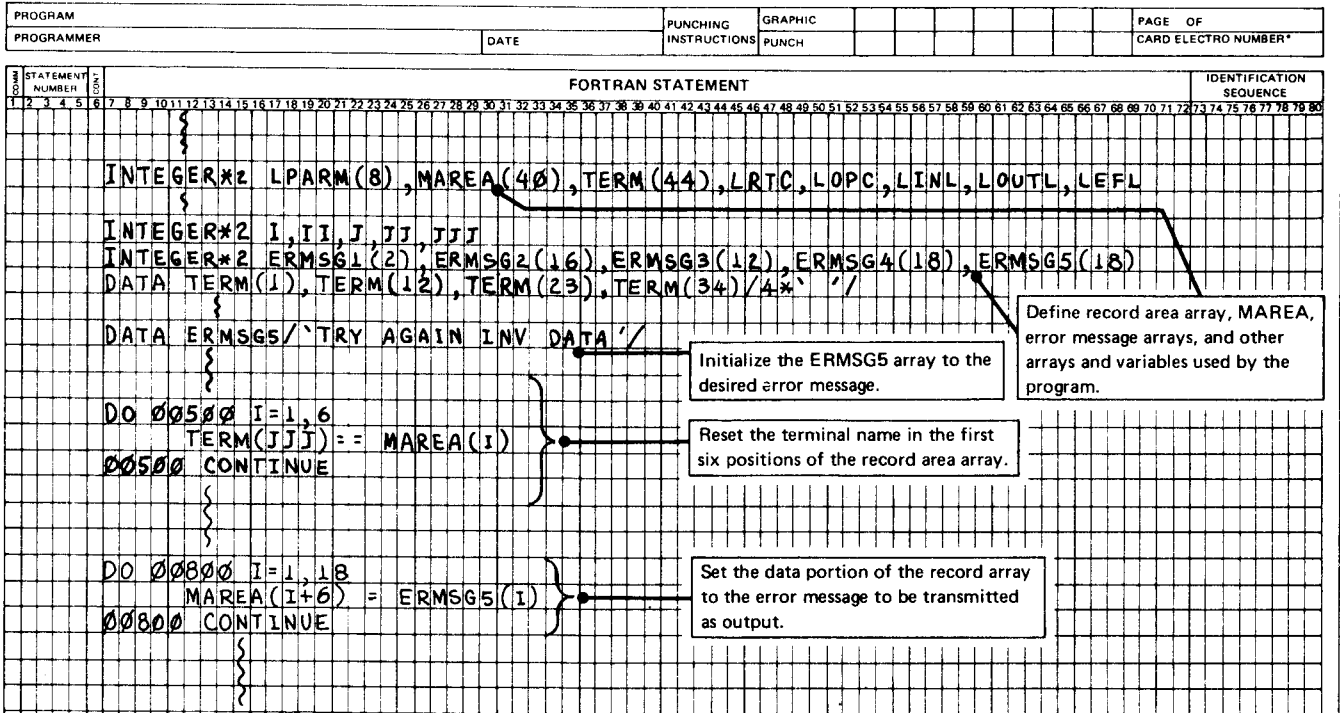


Figure 5-4. Setting the Record Area

## CALLING THE COMMUNICATIONS SERVICE SUBROUTINE

Since FORTRAN does not include special statement types for terminal I/O operations and other communications services, the CCP provides a communications service subroutine, CCPFIO, that converts your FORTRAN program's communications requests into a standard request to the CCP communication facilities. The functions performed by CCPFIO for your FORTRAN program are:

- Loads index register 2 with the address of your program's parameter list.
- Places the address of the record area into your program's parameter list.
- Packs the data from A1 format to A2 before it is passed to CCP.
- Branches to the CCP.
- Unpacks the data from A2 format back to A1 before it is passed to the FORTRAN program.

The CCPFIO subroutine must be linkage edited with your FORTRAN application program. See *Chapter 9. Program Preparation*.

After you have set the required parameter list array element and the terminal name in the first six elements of the record area array, and have prepared any output data, you are ready to request the CCP to perform the operation specified in the parameter list array. You make this request by issuing a CALL statement, specifying CCPFIO. The names of your parameter list array and record area array must be passed as arguments to the subroutine.

The format of the CALL statement is as follows:

```
CALL CCPFIO (parameter-list-array-name, record-area-array-name)
```

In the following example, the name of the parameter list array is PARM-LIST; the name of the record area array is INPUT-OUTPUT-AREA:

```
CALL CCPFIO (PARM-LIST, INPUT-OUTPUT-AREA)
```

Control returns to your FORTRAN program at the statement immediately following the CALL statement. When the return occurs, the following actions have already taken place:

- For output operations, any output data has been accepted by the CCP and, depending upon the output operation specified, has been received by the terminal. In any case, the record area array is now free for you to use again.
- For input operations, any input data which was to be received in the record area array is now in the record area array.
- For Accept Input operations, the symbolic terminal name of the terminal which provided the data in the record area array has been set in the first six elements of the record area array.
- For all operations, the return code field in your parameter list array has been set indicating the result of the operation.
- For input operations, the actual input data length has been set in your parameter list array.
- For Release Terminal operations or for input operations where the terminal has released itself from the program, the count of outstanding Invite Input operations has been set in your parameter list.
- For successful Task Chain Request operations, the requested program is placed on the program request input queue when control is returned to the requesting program.

## EXAMINING RETURNED INFORMATION

After control has returned to your FORTRAN program from the Communications Service Subroutine, you should examine returned information supplied by the CCP, including one or more of the following:

- The return code
- The symbolic terminal name (if it was set by the CCP) or the name of the program that issued the Task Chain Request operation.
- The count of outstanding Invite Inputs, if a Release Terminal operation was performed or if the return code value from an input operation indicates the terminal released itself.
- The actual input data length, if an input operation was successfully performed.
- The input data, if an input operation was performed

### Return Code

The CCP always provides a return code after an operation. You should never assume that an operation is successful; you should always check the return code. In certain cases, you will find that no data transfer has occurred. See Appendix E for the meanings of specific return codes and see *Programming Examples*, later in this chapter, for examples of checking return codes.

You may wish to perform certain operations in your FORTRAN program depending upon the return code value set by the CCP. The example in Figure 5-5 assumes that you want to branch to one of several locations depending upon the value of the return code. The program examines the return code value for the following conditions:

- The operation was successful and no exceptions occurred.
- An EOT was received on a successful operation.
- Some other exception condition occurred.
- An I/O error occurred.

Assume that all array names have been defined earlier in this program. Note the use of comments in the example.

### Examining a Returned Name

On certain operations, the CCP returns the symbolic terminal name to your program's record area array. You may need to examine this name.

For example, you may need to examine the name of the requesting terminal or the terminal that provided the input data to associate new data with data previously received by comparing the terminal name in the record area array with a saved terminal name. You can do this by specifying a DO loop that sets the elements of a six-element save area array equal to the terminal named elements of the record area array. The save area array must be defined as an area of type INTEGER\*2.

If a program can be requested from both a terminal and another program using the Task Chain Request operation, you may want to determine how the program was requested. This can be accomplished by checking for a 14 return code, indicating a Task Chain Request operation. This information is useful if a program communicates with the requestor since your program cannot communicate with a task chain requesting program.

The example in Figure 5-6 saves the terminal name the CCP sets in the terminal name elements of the record area array, MAREA, by specifying a DO loop. The terminal name elements are saved in the array LFEFER.



### FORTRAN Coding Form

GX28-7327  
Printed in U.S.A.

PROGRAM	DATE	PUNCHING INSTRUCTIONS	GRAPHIC PUNCH	PAGE OF
PROGRAMMER				CARD ELECTRO NUMBER*

STATEMENT NUMBER	CONT.	FORTRAN STATEMENT	IDENTIFICATION SEQUENCE
1			
2			
3			
4			
5			
6			
7	*		
8	*		
9	*		
10	*		
11	*		
12	*		
13	*		
14	*		
15	*		
16	*		
17	*		
18	*		
19	*		
20	*		
21	*		
22	*		
23	*		
24	*		
25	*		
26	*		
27	*		
28	*		
29	*		
30	*		
31	*		
32	*		
33	*		
34	*		
35	*		
36	*		
37	*		
38	*		
39	*		
40	*		
41	*		
42	*		
43	*		
44	*		
45	*		
46	*		
47	*		
48	*		
49	*		
50	*		
51	*		
52	*		
53	*		
54	*		
55	*		
56	*		
57	*		
58	*		
59	*		
60	*		
61	*		
62	*		
63	*		
64	*		
65	*		
66	*		
67	*		
68	*		
69	*		
70	*		
71	*		
72	*		
73	*		
74	*		
75	*		
76	*		
77	*		
78	*		
79	*		
80	*		
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			
27			
28			
29			
30			
31			
32			
33			
34			
35			
36			
37			
38			
39			
40			
41			
42			
43			
44			
45			
46			
47			
48			
49			
50			
51			
52			
53			
54			
55			
56			
57			
58			
59			
60			
61			
62			
63			
64			
65			
66			
67			
68			
69			
70			
71			
72			
73			
74			
75			
76			
77			
78			
79			
80			



### FORTRAN Coding Form

GX28-7327  
Printed in U.S.A.

PROGRAM	DATE	PUNCHING INSTRUCTIONS	GRAPHIC PUNCH	PAGE OF
PROGRAMMER				CARD ELECTRO NUMBER*

STATEMENT NUMBER	CONT.	FORTRAN STATEMENT	IDENTIFICATION SEQUENCE
1			
2			
3			
4			
5			
6			
7	*		
8	*		
9	*		
10	*		
11	*		
12	*		
13	*		
14	*		
15	*		
16	*		
17	*		
18	*		
19	*		
20	*		
21	*		
22	*		
23	*		
24	*		
25	*		
26	*		
27	*		
28	*		
29	*		
30	*		
31	*		
32	*		
33	*		
34	*		
35	*		
36	*		
37	*		
38	*		
39	*		
40	*		
41	*		
42	*		
43	*		
44	*		
45	*		
46	*		
47	*		
48	*		
49	*		
50	*		
51	*		
52	*		
53	*		
54	*		
55	*		
56	*		
57	*		
58	*		
59	*		
60	*		
61	*		
62	*		
63	*		
64	*		
65	*		
66	*		
67	*		
68	*		
69	*		
70	*		
71	*		
72	*		
73	*		
74	*		
75	*		
76	*		
77	*		
78	*		
79	*		
80	*		
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			
27			
28			
29			
30			
31			
32			
33			
34			
35			
36			
37			
38			
39			
40			
41			
42			
43			
44			
45			
46			
47			
48			
49			
50			
51			
52			
53			
54			
55			
56			
57			
58			
59			
60			
61			
62			
63			
64			
65			
66			
67			
68			
69			
70			
71			
72			
73			
74			
75			
76			
77			
78			
79			
80			

Figure 5-5. Examining Return Code Values







### FORTRAN Coding Form

GX28-7327  
Printed in U.S.A.

PROGRAM		PUNCHING INSTRUCTIONS	GRAPHIC PUNCH		PAGE OF
PROGRAMMER	DATE				CARD ELECTRO NUMBER*

STATEMENT NUMBER	C	FORTRAN STATEMENT	IDENTIFICATION SEQUENCE
1	*		
2	*	TERMINAL INFORMATION REFER-BACK ARRAY -- LREFER	
3	*	THE LAST ELEMENT OF THE TERMINAL NAME IS USED AS	
4	*	TERMINAL NAME CODE	
5	*	THE ROW LREFER(1,1) THROUGH LREFER(1,5) CONTAINS	
6	*	THE TERMINAL NAME CODE	
7	*	THE ROW LREFER(2,1) THROUGH LREFER(2,5) CONTAINS	
8	*	STORAGE FOR A VARIABLE CALLED A	
9	*	THE ROW LREFER(3,1) THROUGH LREFER(3,5) CONTAINS	
10	*	STORAGE FOR A VARIABLE CALLED B	
11	*	THE ROW LREFER(4,1) THROUGH LREFER(4,5) CONTAINS	
12	*	STORAGE FOR A VARIABLE CALLED C	
13		INTEGER*2 LREFER(4,5)	
14		}	
15		DO 20 I=1,5	
16		IF (LREFER(1,I) .EQ. MAREA(6)) GO TO 5	
17	20	CONTINUE	
18	*		
19	*	SET UP ERROR RECOVERY OR ADD NEW TERMINAL	
20	*	NAME TO THE ARRAY	
21	*		
22		}	

Assume that the parameter list and record area arrays have been defined and that the last element in each terminal name is unique. The first subscript of the same area array is the row, the second element is the column. Thus row 1 of the array contains the saved terminal names.

Define the save area array.

Compare each element in the first row of the save array area to the last element in the terminal name. When the terminal name is found, process the data.



### FORTRAN Coding Form

GX28-7327  
Printed in U.S.A.

PROGRAM		PUNCHING INSTRUCTIONS	GRAPHIC PUNCH		PAGE OF
PROGRAMMER	DATE				CARD ELECTRO NUMBER*

STATEMENT NUMBER	C	FORTRAN STATEMENT	IDENTIFICATION SEQUENCE
1	*		
2	*	PROCESS INFORMATION FOR TERMINAL LOCATED BY	
3	*	COLUMN I OF ARRAY	
4	*		
5		LREFER(2,I) = A+B	
6			
7			
8		LREFER(2,I) = LREFER(2,I) + MAREA(8)	
9		LREFER(3,I) = LREFER(3,I) + MAREA(9)	
10		LREFER(4,I) = MAREA(10)	

When the terminal name is found, set the corresponding array element equal to A + B.

If MAREA(8) = A, MAREA(9) = B, and MAREA(10) = C, you can update the corresponding elements by addition, subtraction, or replacement.

Figure 5-7. Referencing Saved Information

## Count of Outstanding Invite Inputs

On a Release Terminal operation or on an input operation when the return code indicates that the terminal released itself from your program, the count of outstanding Invite Input operations is returned to your program. On accept no-wait input operations with a return code of 04 or 16, the count of outstanding invites is returned to your program (5704-SC2 only). On shutdown inquiry operations, the count of outstanding invites is returned (5704-SC2 only). You may use this number to determine whether your program has any further terminals to serve or whether it can go to end of job.

### Input Data

If the operation requested by your program is an input operation that transfers data, the CCP places the input data received by your program in the seventh and succeeding elements of your record area array before it returns control to your FORTRAN program. The data is then available for you to use in your program.

## USING THE SYSTEM OPERATOR CONSOLE

If you wish to communicate with the system operator through either the 5471 Printer/Keyboard (Models 10 and 12) or the CRT/Keyboard (Model 15), you must specify operations as though the device is a remote terminal. You cannot address the system operator console by the TYPER subroutine or accept information from the system operator console via the KEYBD subroutine. You also cannot address the console or a terminal by using the READ or WRITE statements. Instead, you must specify a Put or Put-Then-Get operation to CONSOL. CONSOL is the only name that can be used for the system operator console.

Your program can communicate with the system operator console at any time. To receive data from the console, you must issue a Put-Then-Get operation, which:

1. Transmits a message to the system operator; and
2. Accepts a reply from the system operator.

Control is not returned to your program until the system operator has transmitted input data to your program.

Operations that can be issued to the console are:

- Put
- Put-Then-Get
- Get Attributes

The console is available at all times to communicate with any program or to enter system operator commands. However, if the console requests a program, it cannot request another program until the first program is initiated by the CCP. It is not necessary, nor is it valid to issue an Acquire Terminal operation to the console in order to communicate with it.  
an input operation

*Example:* The example in Figure 5-8 uses the system operator console as the terminal for a Put-Then-Get operation. Assume that the parameter list array (LPARM), the record area array (MAREA), the console name array (CONSOL), and all other symbolic names have been previously defined.

## FORTRAN PROGRAMMING CONSIDERATIONS

When writing your FORTRAN program, remember:

- You cannot use either the READ or WRITE statements when addressing either the console or teleprocessing terminals.
- You cannot use the TYPER and KEYBD subroutines to address the console.
- Use of the GLOBAL and INVOKE statements will lead to undefined results due to the storage managing characteristic of CCP.
- (Model 10 and Model 12 CCP) You must not use a PAUSE statement. Programs running under the CCP are not permitted to issue halts.

## 3270 DISPLAY FORMAT FACILITY

You can use the 3270 Display Format Facility (DFF) of the CCP to aid you in formatting and using the 3270 display. *Chapter 8. 3270 Display Format Facility* describes the programming requirements that are unique to using 3270 DFF, including the unique 3270 DFF operations, additional information that must be placed in the record area for certain operations, field types that are unique to the 3270, and other information.

See *Chapter 8: 3270 Display Format Facility* for an example of a FORTRAN IV program that uses the DFF to support a single requesting 3270 terminal.



## PROGRAMMING EXAMPLES

Two programming examples are explained in this section:

- *Example 1* – A FORTRAN program that supports a single requesting 3270 terminal without using the Display Format Facility.
- *Example 2* – A FORTRAN program that supports multiple requesting terminals.

See *Chapter 8: 3270 Display Format Facility* for an example of a FORTRAN IV program that uses the DFF to support a single requesting 3270 terminal.

### Example 1

Figures 5-9, 5-10, and 5-11 show the flowcharts, messages, and listing for a sample hotel reservations inquiry program written in FORTRAN. This program transmits two messages to a 3270 Model 1 Display System (480 character screen). The first message from the program requests the terminal operator to enter a room number. The program uses the room number as the relative record number to access a disk file whose records contain guest and rate information about the room. This information is then formatted and displayed as the second message transmitted to the 3270 terminal. Figure 5-9 also shows how these messages appear on the 3270 terminal.

This program is a single requesting terminal (SRT) program (see index entry) with no program-selected terminals; it can receive data from and transmit data to only one 3270 terminal. However, multiple copies of this program could be in main storage at the same time (if your configuration of the CCP allows this), each communicating with a different 3270 Display System. (If multiple copies are in core at the same time, the disk file must be specified as sharable during the Assignment stage—see index entry *disk file sharing*.)

### *Formatting the Messages for the 3270 Display*

Because this sample program does not use the Display Format Facility, this sample program must set all formatting control characters for the 3270 display screen into the data portion of the record area array and transmit them as part of the messages to be displayed (see Figure 5-10). You can find the meanings of each of the 3270 screen format characters shown in Figure 4 of the *General Information Binary Synchronous Communications System Reference Library Manual*, GA27-3004.

The format characters are set by specifying the arrays LOC1, LOC2, LOC3, LOC4, LOC5, LOC6, LOC7, and LOC8 and initializing these arrays to the required 3270 format characters (Figure 5-11). The character Z is used to indicate hexadecimal characters since some of the format characters can only be specified in hexadecimal. The format characters are inserted in the data portion of the record area array by specifying an EQUIVALENCE statement that sets the format character array elements equal to the corresponding element in the data portion of the record area array.

The notes to the right of the listing in Figure 5-11 explain the statements used by this program to format the 3270 display screen. You will also find the comments in the listing helpful.

### Notes Concerning this Sample Program

- Message mode was defined during the Assignment Stage for the 3270 terminal used by this program. (See TERMATTR statement in *CCP System Reference Manual*.) This eliminates the need to do repetitive input operations until EOT is received.
- To run this program using a terminal other than the 3270, you must remove all coding dependent on the 3270. This includes all screen formatting specifications and 3270 screen control characters within the data.
- This program will not accept data with the program request.
- Two different lengths are used for the output length field of the parameter list because the two messages transmitted by this sample program have different lengths.
- This program specifies a Put operation and a Get operation using six blanks as the terminal name. CCP places the name of the 3270 terminal being used in the terminal name field of the record area after the first Put operation is performed.

- To keep this sample program simple, return code checking is kept to a minimum. You may want to do more return code checking in your application programs. For example, when you issue Accept Input you should check for the Shutdown Requested return code (04). Also, if data mode escape is allowed in the CCP system, programs should check for return code 08 (terminal has released itself from the program). It is recommended that each installation design its own return code checking and console communication routines so that a standard is established that is satisfactory to the installation and can be used by all application programs.
- This program does not check the length of the input data because the terminal operator is requested to enter a three-digit room number. If less than three digits are entered, the program is canceled. However, you may want to check the input data length in your application programs.
- Since these are the only two different screen formats used by this program, they are both contained within the program. For more complete applications, you might store the screen formats on disk and read them when they are needed by your program.
- You could also use the Get Attributes operation in this program. If you do not know whether the 3270 Model 1 or the 3270 Model 2 will request the program, you can issue a Get Attributes operation to find out which type of terminal requested the program.
- If this program were coded and specified as a multiple requesting terminal (MRT) program with a MRTMAX-1 keyword on the PROGRAM assignment statement (see *CCP System Reference*), multiple copies of the program would not be allowed in main storage at the same time. As the program is written, multiple copies could be in main storage at the same time and the disk file must be specified as sharable (FILES keyword of PROGRAM assignment statement).

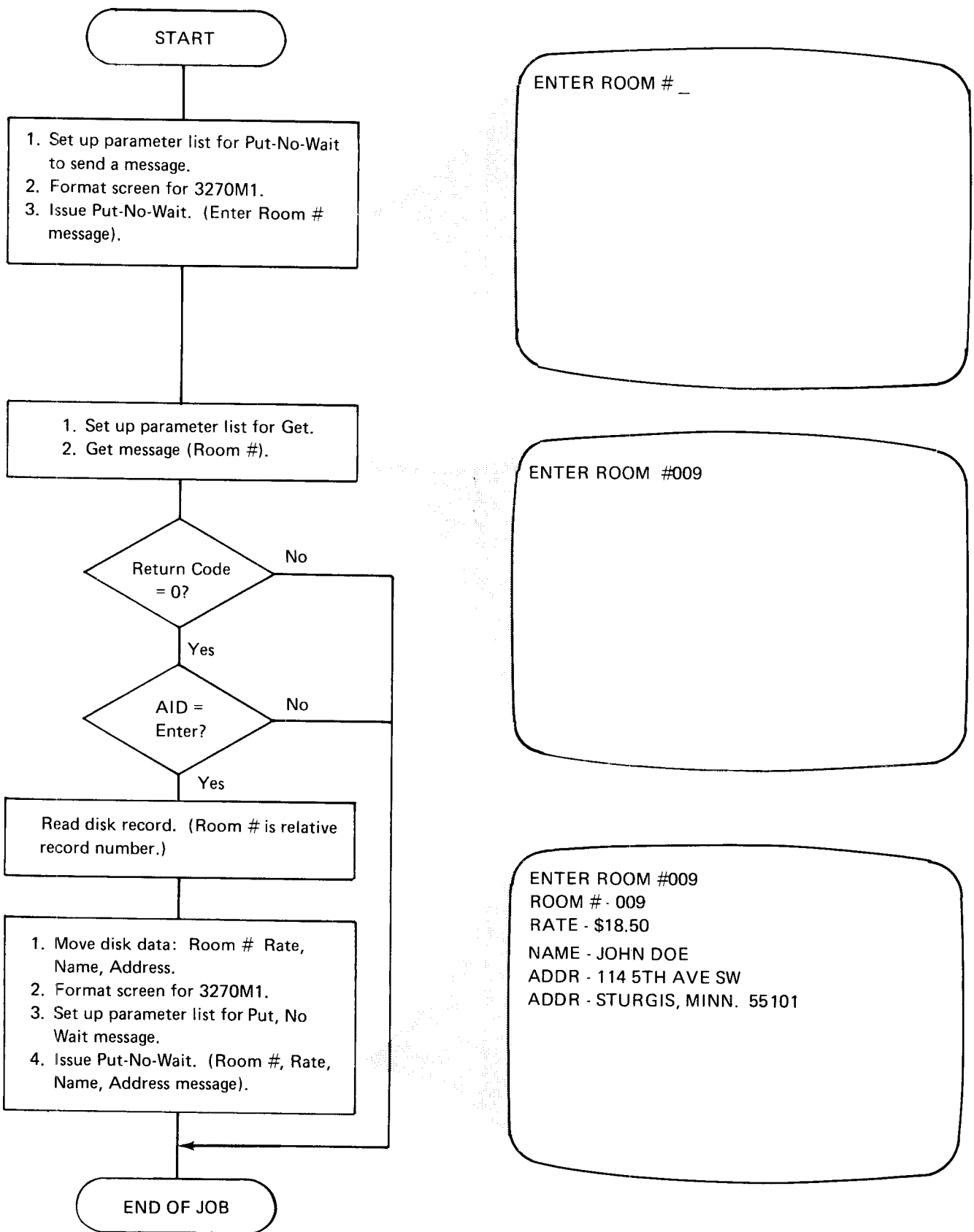
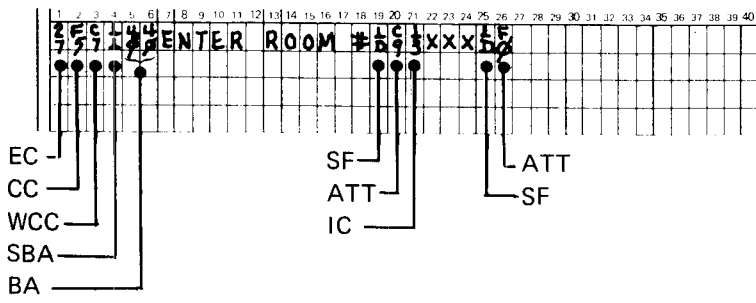
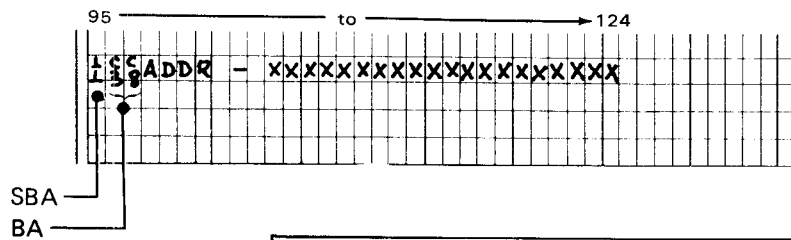
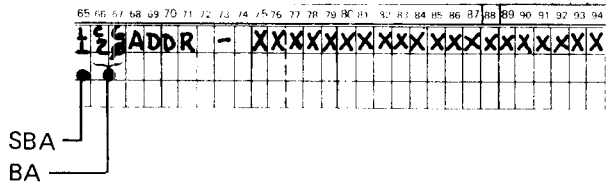
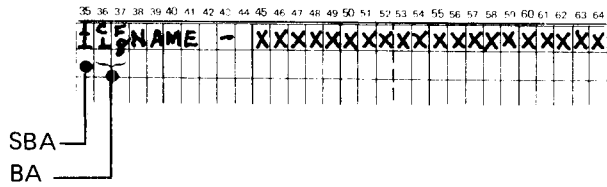
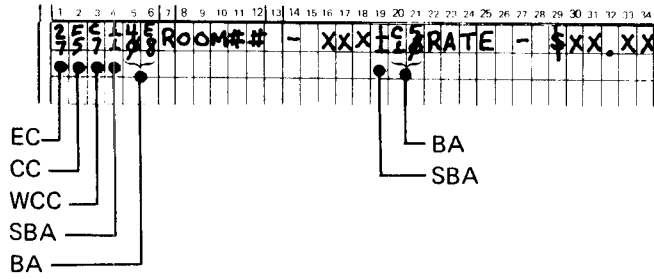


Figure 5-9. Communications Flow of Example 1 (FORTRAN SRT Program)

First Message



Second Message



SF	– Start Field	CC	– Command Code
ATT	– Attribute Character	WCC	– Write Control Character
IC	– Insert Cursor	SBA	– Set Buffer Address
x	– Data Character	BA	– Buffer Address of first character position in the field
EC	– Escape Character		

Figure 5-10. Message Formats for Example 1 (FORTRAN SRT Program)



```

FORTRAN IV
// DAD44 UNITNO=8
*PROCESS MAP,GUJECK
1      PROGRAM SRF01
2      C
3      C      DEFINE FILE 8(10,70,L,LAST)
4      C
5      C*****
6      C      COMMUNICATIUN AREA
7      C*****
8      C
9      C      INTEGER * 2 PARM(8),RTC,OPC,OUTL,INL
10     C
11     C      INTEGER * 2 IOAREA(130),TNAME(6),DATA1(124)
12     C
13     C      INTEGER * 2 IOUT1(124),LUC1(6),ROOMX(9),LOC2(3),RATEX(7),
14     C      *
15     C      *      LUC3(3),NAMEX(7),LOC4(3),ADDRX1(7),
16     C      *      LUC5(3),ADDRX2(7)
17     C
18     C      INTEGER * 2 IOUT2(26),LOC6(6),MSG(12),
19     C      *
20     C      *      LOC7(3),IBLANK(3),LOC8(2)
21     C
22     C      INTEGER * 2 ID,ROOM(3),RATE(4),NAME(20),ADDR1(20),ADDR2(20)
23     C      INTEGER * 2 IOJLR,IPOINT,ENT
24     C
25     C      EQUIVALENCE (RTC,PARM(1)),(OPC,PARM(2)),(OUTL,PARM(3)),
26     C      *      (INL,PARM(4))
27     C
28     C      EQUIVALENCE (TNAME(1),IOAREA(1)),(DATA1(1),IOAREA(7))
29     C
30     C      EQUIVALENCE (LUC1(1),IOUT1(1)),(ROOMX(1),IOUT1(7)),
31     C      *
32     C      *      (LOC2(1),IOUT1(19)),(RATEX(1),IOUT1(22)),
33     C      *      (IOJLR,IOUT1(29)),(IPOINT,IOUT1(32))
34     C      EQUIVALENCE (LUC3(1),IOUT1(35)),(NAMEX(1),IOUT1(38)),
35     C      *
36     C      *      (LOC4(1),IOUT1(65)),(ADDRX1(1),IOUT1(68)),
37     C      *      (LOC5(1),IOUT1(95)),(ADDRX2(1),IOUT1(98))
38     C
39     C      EQUIVALENCE (LOC6(1),IOUT2(1)),(MSG(1),IOUT2(7)),
40     C      *
41     C      *      (LOC7(1),IOUT2(19)),(IBLANK(1),IOUT2(22)),
42     C      *      (LOC8(1),IOUT2(25))
43     C
44     C      DATA LUC1 /Z2740,ZF140,ZC740,Z1140,Z4040,ZE840/
45     C      DATA ROOMX /'R O O M # - '/
46     C
47     C      DATA LOC2 /Z1140,ZC140,Z5040/
48     C      DATA RATEX /'R A T E - '/
49     C
50     C      DATA LOC3 /Z1140,ZC140,ZF840/
51     C      DATA NAMEX /'N A M E - '/
52     C
53     C      DATA LOC4 /Z1140,ZC240,Z6040/
54     C      DATA ADDRX1 /'A D D R - '/
55     C
56     C      DATA LOC5 /Z1140,ZC340,ZC840/
57     C      DATA ADDRX2 /'A D D R - '/
58     C
59     C      DATA LOC6 /Z2740,ZF540,ZC740,Z1140,Z4040,Z4040/
60     C      DATA MSG /'E N T E R R O O M # '/

```

Parameter list array

Record area array

Output area array

Message area array

Disk fields  
Other variables used by the program

Redefine parameter list array

Redefine record area array

Redefine output area array

Redefine message area array

Initialize 3270 format characters and variables

Figure 5-11 (Part 1 of 2). Example 1 - FORTRAN SRT Program

```

C
26 DATA LUC7 /Z1040,ZC940,Z1340/
27 DATA IBLANK /3*' '/
C
28 DATA LUC8 /Z1040,ZF040/
C
29 DATA TNAME /6*' '/
C
30 DATA IJULR/'S'/,IPJINT/'.'/,ENT/' '/
C
C*****
C A PUT SENDING THE MESSAGE 'ENTER ROOM #' IS SENT TO THE 3270 *
C*****
C
31 OPC = 54
32 OUTL = 26
33 INL = 26
34 CALL MOVE (IOUT2,1,26,DATA1,1)
35 CALL CUPFIJ (PARM,IOAREA)
C
C*****
C DO A GET FOR THE ROOM NUMBER AND CHECK THE RETURN CODE FOR *
C ZERO. IF IT IS NOT ZERO GO TO E0J. *
C*****
C
36 OPC = 1
37 OUTL = 11
38 INL = 11
39 CALL CUPFIJ (PARM,IOAREA)
40 IF (RTC.NE.0) GO TO 20
C
C*****
C CHECK IF ENTER AID KEY WAS ON AT GET, IF IT WAS NOT GO TO E0J *
C CONVERT THE ROOM NUMBER FROM A1 FORMAT TO INTEGER FORMAT *
C THEN CHECK TO INSURE IT IS IN THE RANGE OF FROM 001 TO 010. *
C*****
C
41 IF (DATA1(3).NE.ENT) GO TO 20
42 XROOM = GET (DATA1,9,11,1.0)
43 INROOM = XROOM
44 IF (INROOM.LT.1) GO TO 20
45 IF (INROOM.GT.10) GO TO 20
C
C*****
C READ A RECORO FROM A DIRECT-ACCESS DISK FILE. THE ROOM NUMBER *
C REPRESENTS THE RELATIVE POSITION OF THE RECORD IN THE FILE *
C*****
C
46 READ (8'INROOM,10000) ID,ROOM,RATE,NAME,ADDR1,ADDR2
C
C*****
C MOVE THE ROOM NUMBER, RATE PER DAY, THE NAME AND ADDRESS OF *
C THE GUEST INTO THE GUEST ARRAY, THEN DO A PUT OF IT TO THE *
C 3270 *
C*****
C
47 CALL MOVE (ROOM,1,3,IOUT1,16)
48 CALL MOVE (RATE,1,2,IOUT1,30)
49 CALL MOVE (RATE,3,4,IOUT1,33)
50 CALL MOVE (NAME,1,20,IOUT1,45)
51 CALL MOVE (ADDR1,1,20,IOUT1,75)
52 CALL MOVE (ADDR2,1,20,IOUT1,105)
53 CALL MOVE (IOUT1,1,124,DATA1,1)
54 IRETRY = 0
55 OPC = 54
56 OUTL = 124
57 INL = 124
58 10 CALL CUPFIJ (PARM,IOAREA)
C
C*****
C AFTER THE PUT CHECK THE RETURN CODE FOR ZERO, IF IT IS NOT *
C ZERO RETRY THE PUT ONE TIME. *
C*****
C
59 IF (RTC.EQ.0) GO TO 20
60 IF (IRETRY.EQ.1) GO TO 20
61 IRETRY = IRETRY + 1
62 GO TO 10
63 20 STOP
64 10000 FURMAT (A1,3A1,4A1,20A1,20A1,20A1)
65 END

```

Figure 5-11 (Part 2 of 2). Example 1 – FORTRAN SRT Program

## Example 2

Figures 5-12, 5-13, and 5-14 show the flowchart, input/output messages, and listing for a sample FORTRAN multiple requesting terminal (MRT) program designed to run under the CCP. This program supports up to four MLTA requesting terminals. The terminal operator enters a seven-digit number preceded by a +, -, or N. The CCP transmits this signed number to the FORTRAN program.

The FORTRAN program:

- Adds the number to the value in the accumulator associated with the terminal that transmitted the data if the first position is +
- Subtracts the number from the accumulator if the first position is -
- Releases the terminal if the first position is N.

If a value was either added or subtracted, the new value accumulated for the terminal is inserted into the message *CURRENT VAL = sxxxxxxxxx ENTER DATA* and the message is sent to the terminal.

This sample program also checks for several error conditions and transmits the appropriate error message to the terminal requesting the operation.

This sample program is not designed to show the most effective way of performing operations. Instead, it shows a variety of ways to do things. It uses a variety of operation codes that show how data can be associated with a terminal by defining a save area array for the terminal names and accumulated data. It frequently checks return codes, but you can do even more return code checking if you wish. Data entered by the terminal operator must be fixed length. To allow variable length input fields you could include a subroutine in your program to check the effective input length returned in the parameter list and align the data correctly. This program communicates with the console in addition to the requesting terminals.

The notes to the right of the listing in Figure 5-14 and the comments in the listing explain each section of the sample program.

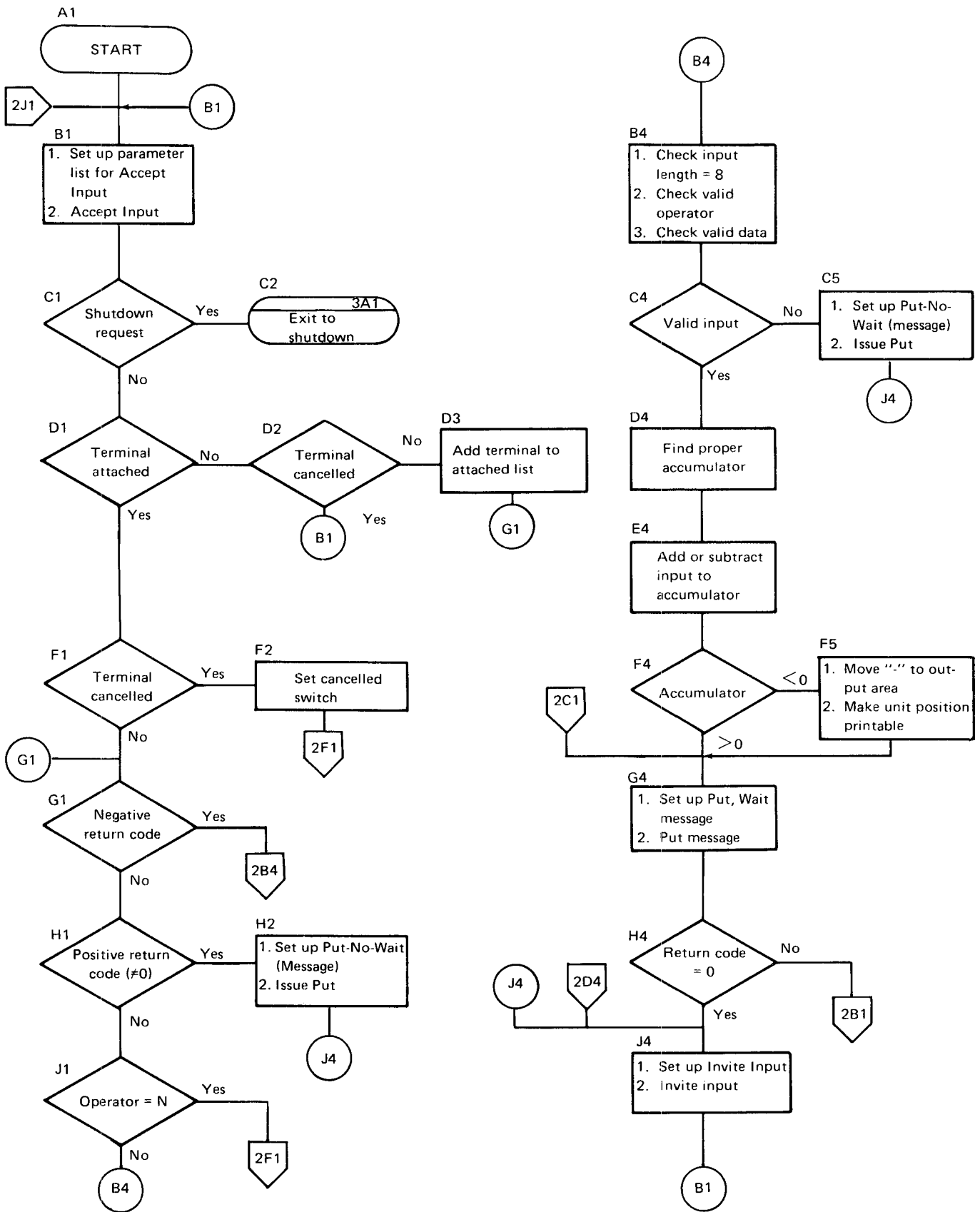


Figure 5-12 (Part 1 of 3). Program Logic of Example 2 (FORTRAN MRT Program)

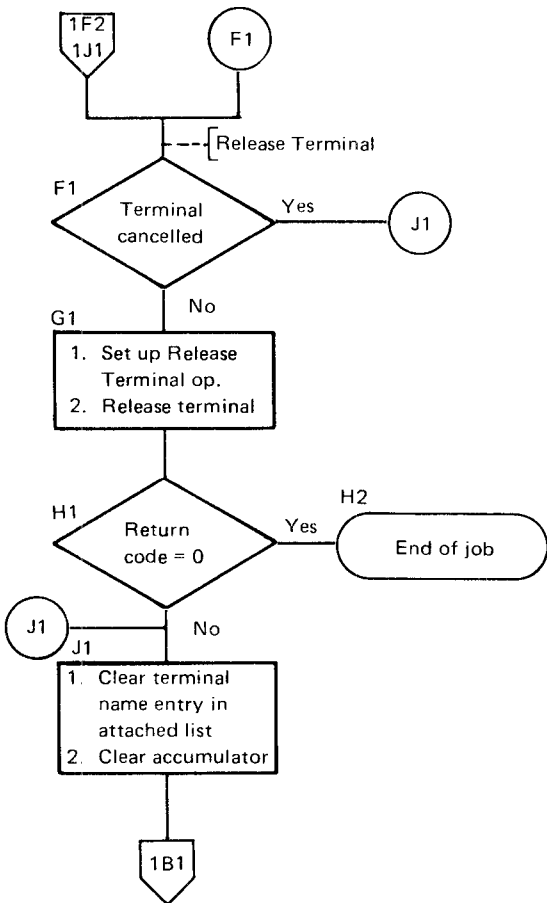
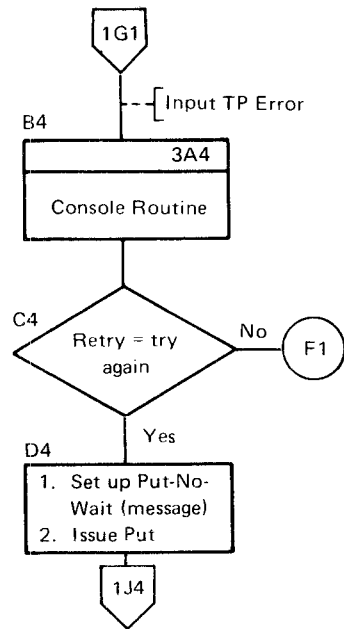
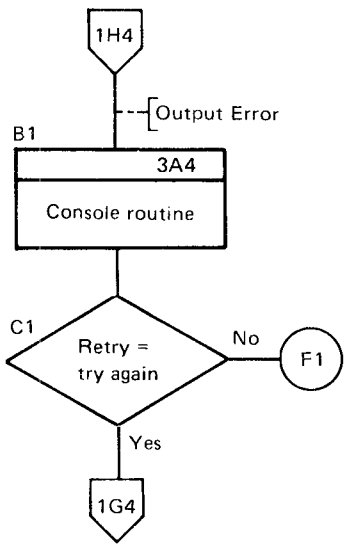
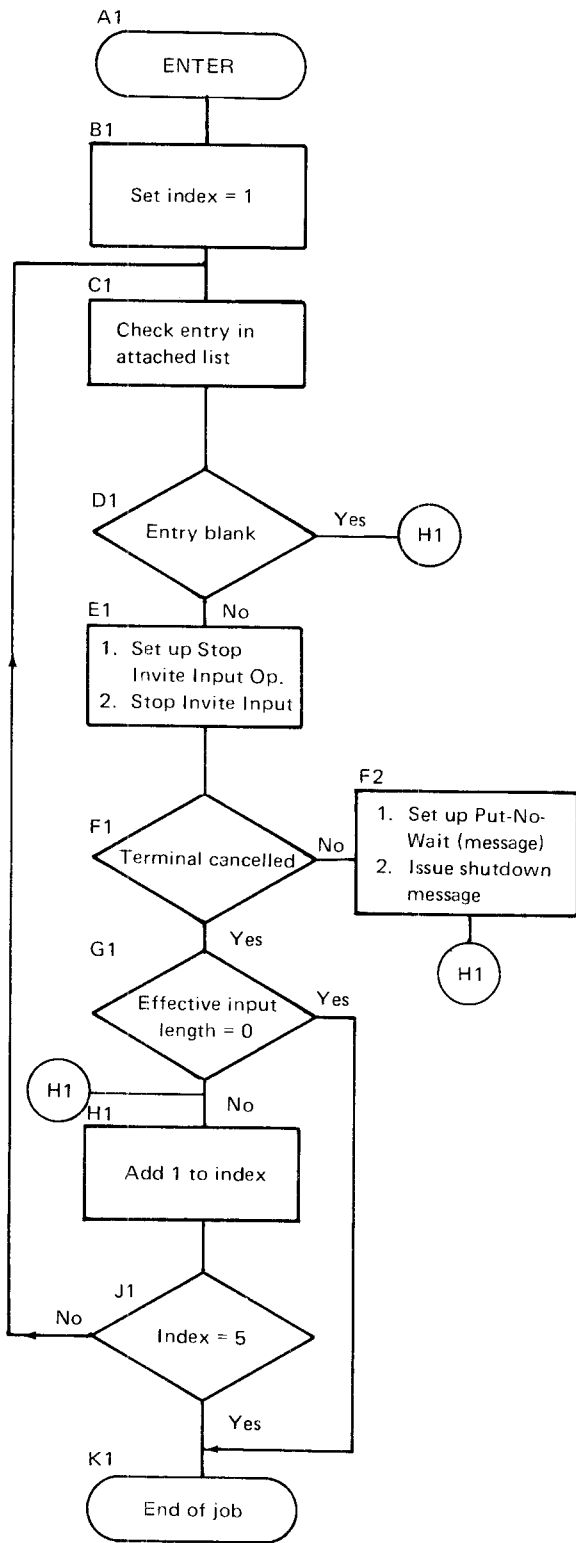


Figure 5-12 (Part 2 of 3). Program Logic of Example 2 (FORTRAN MRT Program)

### Shutdown Routine



### Console Routine

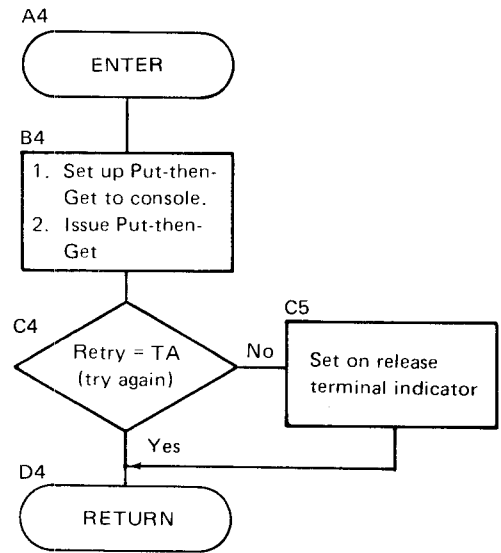


Figure 5-12 (Part 3 of 3). Program Logic of Example 2 (FORTRAN MRT Program)

**Input Data Entered by Terminal Operator**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	
S	X	X	X	X	X	X	X	X																

A fixed length numeric field where S is a +, -, or N and X is a numeric digit. All eight positions must be entered, except when N is entered in the first position.

**Data Entered by System Operator on 5471 Printer/Keyboard (Models 10 and 12) or CRT/Keyboard (Model 15)**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	
T	A																							
C	C																							

In response to the messages INPUT TP ERROR TNAME-cccccc and OUTPUT TP ERROR TNAME-cccccc to the console, the system operator replies TA if he wants to Try Again. Any other reply (cc) causes the terminal to be released.

**Output to the Console**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39			
I	N	P	U	T		T	P		E	R	R	O	R		T	N	A	M	E	-	C	C	C	C	C	C															
O	U	T																																							

These messages are transmitted to the console (cccccc = terminal name).

**Output to Terminal**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39			
C	U	R	R	E	N	T		Y	A	L	=	S	X	X	X	X	X	X	X	X		E	N	T	E	R															
T	R																																								
T	R																																								
C	C	P		S	H	T	D	W	N		L	A	S	T		R	E	C	-		T	P		E	R	R	O	R													
C	C	P		S	H	T	D	W	N		L	A	S	T		R	E	C	-		B	A	D		D	A	T	A													
C	C	P		S	H	T	D	W	N		L	A	S	T		R	E	C	-		S	X	X	X	X	X	X														
C	C	P		S	H	T	D	W	N		L	A	S	T		R	E	C	-		N	O		D	A	T	A														

- Transmitted with value in accumulator associated with the terminal.
- Issued if data is invalid
- Issued if system operator replies TA (Negative return code on Accept Input)
- Issued for negative return code on Stop Invite Input
- Issued for positive return code other than 10 on Stop Invite Input
- Issued for return code of 0 on Stop Invite Input
- Issued for return code of 10 on Stop Invite Input

Figure 5-13. Input and Output Message Formats for Example 2 (FORTRAN MRT Program)

FORTRAN IV

\*PROCESS MAP

```

1 PROGRAM MRPOR1
2 INTEGER*2 LPARM (8), MAREA (40), TERM (44), LRTC, LOPC, LINL, LOUPL, LEPL
3 INTEGER*2 ACCEPT, PUTNWT, PUTMWT, INVINP, PUTGET, RELTRM, STPINV
4 INTEGER*2 CRNTVL (34), CONSOL (6), LBLNK, LPLUS, LMINUS, LNNN, N
5 INTEGER*2 I, II, J, JJ, JJJ, MSAV (2)
6 INTEGER*2 ERMSG1 (2), ERMSG2 (16), ERMSG3 (12), ERMSG4 (18), ERMSG5 (18)
7 INTEGER*2 ERMSG6 (22), ERMSG7 (24)
8 INTEGER*4 LZERO
9 EQUIVALENCE (LRTC, LPARM (1)), (LOPC, LPARM (2)), (LOUPL, LPARM (3)),
10 (LINL, LPARM (4)), (LEPL, LPARM (5))
11 DATA ACCEPT, PUTNWT, PUTMWT, INVINP, PUTGET, RELTRM/4,54,50,5,3,10/
12 DATA STPINV/1025/
13 DATA LPLUS, LMINUS, LNNN, LBLNK/'+', '-', 'N', ' ' /
14 DATA LZERO/' 00' /
15 DATA CRNTVL/'C U R R E N T   V A L = +
16 E R   D A T A ' /
17 DATA CONSOL/'C O N S O L ' /
18 DATA TERM (1), TERM (12), TERM (23), TERM (34)/4*' ' /
19 DATA LINL/34/
20 DATA ERMSG1/'T A ' /
21 DATA ERMSG2/' T P   E R R O R   T N A M E = ' /
22 DATA ERMSG3/'I N P U T   O U T P U T ' /
23 DATA ERMSG4/'T R Y   A G A I N   T P   E R R O R ' /
24 DATA ERMSG5/'T R Y   A G A I N   I N V   D A T A ' /
25 DATA ERMSG6/'C P   S H T D W N   L A S T   R E C   -   ' /
26 DATA ERMSG7/'T P   E R R O R   B A D   D A T A   N O   D A T A ' /
27 C *****
28 C II=INDEX FOR TERMINAL ARRAY, JJ=SWITCH, JJ=1 INPUT, JJ=7 OUTPUT.
29 C JJJ=ADDITIONAL INDEX FOR VARIOUS SHORT TERM PURPOSES.
30 C
31 C SET UP PARAMETER LIST FOR ACCEPT INPUT
32 C *****
33 00100 JJ=1
34 LOPC=ACCEPT
35 C *****
36 C 00 ACCEPT INPUT OPERATION
37 C *****
38 CALL CCFPIO (LPARM, MAREA)
39 C *****
40 C CHECK TO SEE IF A SHUTDOWN HAS BEEN REQUESTED
41 C *****
42 IF (LRTC-4) 00190,02500,00190
43 C *****
44 C DETERMINE IF TERMINAL IS ALREADY ATTACHED, NCOMP IS SET TO 0 IF
45 C FIRST 6 ELEMENTS IN MESSAGE AREA MATCH A 6 ELEMENT TERMINAL
46 C NAME IN THE TERMINAL DATA STORAGE ARRAY
47 C *****
48 00190 DO 00200 II=1,34,11
49 IF (NCOMP(MAREA,1,6,TERM,II)) 00200,00600,00200
50 00200 CONTINUE
51 C *****
52 C CHECK TO SEE IF TERMINAL HAS BEEN CANCELLED. IF IT HAS RETURN
53 C TO ACCEPT INPUT IF INVITES ARE OUTSTANDING, IF NO INVITES ARE
54 C OUTSTANDING GO TO EXIT.
55 C *****
56 IF (LRTC-8) 00250,00220,00250
57 00220 IF (LEPL) 00100,09000,00100
58 C *****
59 C LOCATE A BLANK 6 ELEMENT TERMINAL NAME SPACE IN THE TERMINAL DATA
60 C STORAGE ARRAY
61 C NOTE: NO MORE THAN 4 TERMINALS WILL BE ALLOWED TO COMMUNICATE
62 C WITH THIS PROGRAM IF ASSIGNMENT SPECIFIES 4 TERMINALS
63 C *****
64 00250 DO 00300 II=1,34,11
65 IF (TERM(II)-EQ.LBLNK) GO TO 00400
66 00300 CONTINUE
67 C *****
68 C INSFR THE NEW TERMINAL NAME INTO THE SPACE JUST LOCATED AND ZERO
69 C THE ACCUMULATOR ASSOCIATED WITH THIS LOCATION
70 C *****
71 00400 CALL MOVE (MAREA,1,6,TERM,II)
72 CALL FILL (TERM,II+6,II+10,LZERO)
73 GO TO 00610
74 C *****
75 C CHECK TO SEE IF TERMINAL HAS BEEN CANCELLED. IF IT HAS GO MAKE
76 C TERMINAL NAME AREA AVAILABLE FOR NEW ENTRY
77 C *****

```

Define parameter list array, record area array, save area array, console name array, output message array, error messages, and other variables used by program.

Set parameter list fields equal to elements in parameter list arrays.

Initialize operation code fields, output messages, save area array, error messages, and other variables used in program.

Set operation code field equal to numeric variable for Accept Input.

Determine if the terminal name for the terminal that transmitted the input data is in the save area array. If it is, the data is added to the accumulator associated with that terminal. If it is not in the save area array and the terminal is not cancelled, the terminal is added to the save area array.

Figure 5-14 (Part 1 of 4). Example 2 – FORTRAN MRT Program



```

40 00600 IF (LRTC-8) 00610,04900,00610
C*****
C CHECK FOR INPUT ERROR INDICATIONS, ISSUE ERROR MESSAGE IF RETURN *
C CODE NOT EQUAL TO 0, OR IF LENGTH NOT 8 *
C CHECK FOR VALID OPERATOR, IF OPERATOR EQUAL TO N GO RELEASE TERMINAL *
C*****
41 00610 IP (LRTC) 02800,00650,00700
42 00650 IF (MAREA(7).EQ.LNNN) GO TO 04800
43 IF (LEPL-8) 00700,00660,00700
44 00660 IF (MAREA(7).EQ.LPLUS) GO TO 00900
45 IF (MAREA(7).EQ.LMINUS) GO TO 00900
C*****
C ASSUME BAD OPERATOR, ISSUE INVALID DATA MESSAGE *
C*****
46 00700 CALL MOVE(ERMMSG5,1,18,MAREA,7)
47 LOUTL=18
48 GO TO 04300
C*****
C CONVERT THE VALUE RECEIVED AS INPUT FROM CHARACTER (A1) TO ZONED *
C DECIMAL (D1) FORMAT *
C IF N NOT EQUAL TO 0 SET UP INVALID DATA MESSAGE *
C*****
49 00900 N=0
50 CALL A1DEC(MAREA,8,14,N)
51 IF (N) 00700,01000,00700
C*****
C NOW ADD THE VALUE RECEIVED AS INPUT TO THE VALUE IN THE *
C ACCUMULATOR ASSIGNED WITH THE TERMINAL NAME *
C*****
52 01000 JJJ=II+6
53 N=0
54 IF (MAREA(7).EQ.LMINUS) GO TO 01500
C*****
C ADD VALUE RECEIVED TO VALUE IN ACCUMULATOR *
C*****
55 CALL ADD(MAREA,11,14,TERM,JJJ,JJJ+4,N)
56 GO TO 01600
57 01500 CALL SUB(MAREA,11,14,TERM,JJJ,JJJ+4,N)
C*****
C SET UP TO DISPLAY RESULTS IN ACCUMULATOR, TO TERMINAL *
C NOTE: IF OVERFLOW OCCURRED MAX VALUE WILL BE DISPLAYED, ALL 9'S *
C*****
58 01600 CALL MOVE(CRNTVL,1,34,MAREA,7)
59 CALL MOVE(TERM,JJJ,JJJ+4,MAREA,25)
C*****
C SET INDICATOR N ACCORDING TO THE SIGN OF THE ACCUMULATOR VALUE THEN *
C SET LAST DIGIT OF ACCUMULATOR VALUE POSITIVE *
C*****
60 CALL NSIGN(MAREA,29,1,N)
61 IF (N) 01800,01900,01900
62 01800 MAREA(19)=LMINUS
C*****
C CONVERT THE ZONED RESULTS TO CHARACTERS IN A1 FORMAT FOR OUTPUT *
C*****
63 01900 N=0
64 CALL DECA1(MAREA,20,29,N)
C*****
C SET UP PARAM LIST FOR PUT MESSAGE WAIT *
C*****
65 LOPC=PUTMWT
66 LOUTL=34
C*****
C DO PUT MESSAGE WAIT OPERATION *
C*****
67 CALL CCPPIO(LPARN,MAREA)
C*****
C CHECK RETURN CODE TO SEE IF OPERATION WAS SUCCESSFUL *
C IF RETURN CODE NOT EQUAL TO ZERO GO ISSUE ERROR MESSAGE *
C*****
68 IF (LRTC) 02100,02200,02100
69 02100 JJ=7
70 GO TO 02800
C*****
C SET UP PARAMETER LIST FOR INVITE INPUT *
C*****
71 02200 LOPC=INVINP
C*****
C DO INVITE INPUT OPERATION *
C*****
72 CALL CCPPIO(LPARN,MAREA)
73 GO TO 00100
C*****
C HANDLE SHUTDOWN REQUEST BY ISSUING STOP INVITES TO ALL OUTSTANDING *
C INVITE INPUTS PREVIOUSLY ISSUED *
C*****
74 02500 DO 02600 II=1,34,11
75 IF (TERM(II).EQ.LBLNK) GO TO 02600
C*****
C SET UP PARAMETER LIST FOR STOP INVITE INPUT *
C*****

```

If the first position of the input field is +, the data is added to the accumulator element associated with the terminal that transmitted the data. If the first position is -, the data is subtracted from the terminal. If the first position is N, the terminal is checked to see if it is cancelled.

If the first position of the input data is not +, -, or N, a message is issued to the terminal.

Insert accumulated value associated with the terminal into the output message and display it on the terminal.

*Note:* When the last terminal attached to an MRT program is processed, issue a Release Terminal operation to that terminal in order to check the count of outstanding Invite Inputs. If the count is greater than zero, the program can issue an Accept Input operation. For example, suppose an MRT program is servicing the maximum number of requestors and one or more additional requests are queued to the program. If the program receives a shutdown-requested return code (04) and goes to end of job without checking the count of outstanding Invite Inputs, the program terminates with a 2C termination code (going to end of job with outstanding Invite Inputs), and each of the queued terminals receives an S06 message (program cancelled – shutdown).

Figure 5-14 (Part 2 of 4). Example 2 – FORTRAN MRT Program

```

76         LOPC=STPINV
77         CALL MOVE (TERM,II,II+5,MAREA,1)
C*****
C DO STOP INVITE INPUT OPERATION *
78         CALL CCPPIO (LPARM,MAREA)
C*****
C IF TERMINAL NOT CANCELLED, THEN ISSUE SHUTDOWN MESSAGE. IF TERMINAL*
C CANCELLED THEN GO TO EXIT IF NO OUTSTANDING INVITES, ELSE GO *
C SEARCH FOR NEXT ACTIVE TERMINAL. *
C*****
79         IF (LRTC=8) 02510,02508,02510
80 02508 IF (LEPL) 02600,09000,02600
C*****
C SET UP PARAMETER LIST FOR PUT NO WAIT *
C*****
81 02510 LOPC=PUTNWT
82 LOUTL=30
C*****
C INSERT PROPER SHUTDOWN MESSAGE TO TERMINAL REFERENCED IN TERMINAL *
C DATA ARRAY. THE MESSAGE IS SET ACCORDING TO THE RETURN CODE *
C*****
83         IF (LRTC) 02520,02535,02525
84 02520 JJJ=1
85 GO TO 02528
86 02525 IF (LRTC-10) 02526,02527,02526
87 02526 JJJ=9
88 GO TO 02528
89 02527 JJJ=17
90 02528 CALL MOVE (ERMSG7,JJJ,JJJ+7,MAREA,29)
91 GO TO 02558
92 02535 CALL MOVE (MAREA,7,14,MAREA,29)
93 02558 CALL MOVE (ERMSG6,1,22,MAREA,7)
C*****
C DO PUT NO WAIT OPERATION *
C*****
94         CALL CCPPIO (LPARM,MAREA)
95 02600 CONTINUE
96 GO TO 09000
C*****
C PREPARE INPUT OR OUTPUT ERROR MESSAGES HERE *
C SET UP PARAMETER LIST FOR PUT THEN GET TO CONSOL *
C*****
97 02800 LOPC=PUTGET
98 LOUTL=28
C*****
C INSERT TERMINAL NAME=CONSOL, TERMINAL NAME WHERE ERROR OCCURRED, *
C AND INPUT OR OUTPUT ERROR MESSAGE *
C*****
99         CALL MOVE (CONSOL,1,6,MAREA,1)
100        CALL MOVE (ERMSG3,JJ,JJ+5,MAREA,7)
101        CALL MOVE (TERM,II,II+5,MAREA,29)
102        CALL MOVE (ERMSG2,1,16,MAREA,13)
C*****
C DO PUT GET OPERATION TO CONSOLE *
C*****

```

If an input error occurred, the message 'INPUT TP ERROR TNAME xxxxxx' is issued to the console (xxxxxx = terminal on which the error occurred). If an output error occurred, a similar output error message is built and issued.

Figure 5-14 (Part 3 of 4). Example 2 – FORTRAN MRT Program

```

103      CALL CCPPIO (LPARM,MAREA)
C*****
C  MOVE TERMINAL NAME BACK TO TERMINAL NAME AREA IN TERMINAL DATA *
C  ARRAY *
C  CHECK FOR REPLY REQUESTING TO TRY AGAIN--TA *
C  IF TA NOT PRESENT THEN GO DISCONNECT *
C*****
104      MSAV(1)=MAREA(7)
105      MSAV(2)=MAREA(8)
106      CALL MOVE(TERM,II,II+5,MAREA,1)
107      IF (NCOMP(MSAV,1,2,ERRMSG1,1)) 04800,03800,04800
C*****
C  IF OUTPUT ERROR MESSAGE THEN GO TRY TO OUTPUT AGAIN *
C*****
108      03800 IF (JJ-7) 03810,01600,03810
C*****
C  IF INPUT ERROR MESSAGE THEN GO TRY TO INPUT AGAIN *
C*****
109      03810 CALL MOVE(ERRMSG4,1,18,MAREA,7)
110      LOUPL=18
C*****
C  SET UP PUT NO WAIT PARAMETER LIST *
C*****
111      04300 LOPC=PUTNWT
C*****
C  DO PUT NO WAIT OPERATION *
C*****
112      CALL CCPPIO (LPARM,MAREA)
113      GO TO 02200
C*****
C  CHECK TO SEE IF THIS TERMINAL HAS BEEN CANCELLED. IF IT HAS GO *
C  REMOVE IT FROM THE ACTIVE LIST. IF NOT CANCELLED DO A RELEASE *
C  TERMINAL OPERATION. *
C*****
114      04800 IF (LRTC-8) 04850,04900,04850
C*****
C  SET UP PARAMETER LIST FOR RELEASE TERMINAL OPERATION *
C*****
115      04850 LOPC=RELTRM
C*****
C  DO RELEASE TERMINAL OPERATION *
C*****
116      CALL CCPPIO (LPARM,MAREA)
C*****
C  SET THE FIRST ELEMENT OF THE TERMINAL NAME AREA OF THE ARRAY TO A *
C  BLANK, SIGNIFYING THAT THIS ENTRY AND ITS ACCUMULATOR ARE *
C  AVAILABLE. RETURN TO ACCEPT INPUT IF MORE INVITES ARE OUTSTAND- *
C  ING, OTHERWISE EXIT. *
C*****
117      04900 TERM(II)=LBLNK
118      IF (LEPL) 00100,09000,00100
119      09000 STOP
120      END

```

If the system operator keys in TA, the terminal name of the terminal on which the error occurred is placed in the terminal name field of the record area and the operation is retried. If he keys in any other characters, the terminal name for which the error occurred is placed in the record area and, if the terminal has not been cancelled, a Release Terminal operation is issued to it.

Figure 5-14 (Part 4 of 4). Example 2 – FORTRAN MRT Program



Programs written in RPG II can communicate with terminal devices via the CCP by means of either or both of the following two RPG II facilities:

- **SPECIAL files** - One or more SPECIAL files can be defined for communications devices; different file types can be assigned, depending on how the device is used.
- **EXIT/RLABL operations** - Terminal operations can be requested using EXIT and RLABL operations with subroutines provided by the CCP.

You can issue all of the communications operations described under *Operations* in Chapter 2 through either the SPECIAL or the EXIT/RLABL interface. In general, however, use of SPECIAL files is recommended for operations involving data transfer, such as Get, Put, Accept Input, Put-Then-Get, and Stop Invite Input (which may result in a Get). EXIT/RLABL is best suited for use in non-I/O operations, such as Acquire Terminal, Release Terminal, and Shutdown Inquiry. By following this general rule, you can eliminate the need for RPG II logic to move fields from a record area when no data has been received.

Programming for non-TP devices (console, card devices, printer, and disk) under the CCP is the same as in RPG II programs that are not written to run under the CCP, with a few exceptions:

- The system operator console is treated as a terminal device under the CCP. Therefore, you must communicate with the console through either SPECIAL or EXIT/RLABL; you cannot use the name CONSOLE (Model 10 Disk System and Model 12) or CRT77 (Model 15) on File Description Specifications.
- You should not specify a core index for disk files (columns 60-65 of the File Description Specification). The CCP builds a core index based on the MSTRINDX keyword of the DISKFILE assignment statement (see *CCP System Reference Manual*). If you also specify a core index in your program, you will cause unnecessary code to be generated for your program.

- You should be aware of how other programs running under the CCP are using disk files that you use in your program, especially if records are being added to the files. See index entry *disk file considerations* for detailed information.
- You cannot use magnetic tape files in programs written to run under the CCP.

**Note:** You are assumed to be familiar with the RPG II object program cycle and the RPG II language, including SPECIAL files, arrays, and EXIT/RLABL, READ, and EXCPT operations. If you are not familiar with these RPG II facilities and operations, you should consult the following publications:

- *IBM System/3 RPG II Reference Manual*, SC21-7504
- *IBM System/3 RPG II Additional Topics Programmer's Guide*, GC21-7567
- *IBM System/3 RPG II Disk File Processing Programmer's Guide*, GC21-7566

### RPG II Use of the Standard CCP Interface

The interface used by RPG II application programs for CCP communications operations differs somewhat from the standard interface described in Chapter 2. However, you should read Chapter 2 before reading this chapter, since most of the information in that chapter also applies to RPG II. The differences in the RPG II interface are due mainly to the fixed nature of RPG II object program cycle, a characteristic that COBOL, FORTRAN, and Basic Assembler do not possess. The basic elements of the interface, although somewhat different in content and use, remain the same:

- Parameter list
- Record area
- Communications service subroutine

## COMMUNICATIONS INTERFACE USING RPG II SPECIAL FILES

One method of performing operations with terminals under the CCP is to define one or more SPECIAL files for terminals in the File Description Specifications. SPECIAL files can be defined in a variety of ways, depending on the needs of the particular application. For example, a program that performs both input and output operations with a single terminal might define one SPECIAL file as a primary input file, to be used for input operations with the terminal, and a second SPECIAL file as a detail output file, for output operations to the terminal. Thus, two SPECIAL files would be defined for a single terminal.

Suppose, on the other hand, the program communicates with several terminals. The same two SPECIAL files defined for communication with a single terminal could be used for communication with several terminals - one for all terminal input and one for all terminal output.

It is also possible to define a single SPECIAL file for both input and output operations with one or more terminals. For example, a SPECIAL file could be defined as a Combined Demand Input and Exception Output file to be processed by READ and EXCPT operations in Calculations.

These are just a few examples of the variety of ways of defining SPECIAL files for use in terminal operations via the CCP. Any of the following file types can be defined as a SPECIAL file to be used for I/O operations with one or more terminals:

- Primary Input file
- Secondary Input file
- Demand Input file (used with the READ operation)
- Exception, Detail, or Total Output file
- Combined Demand Input and Exception Output file (used with READ and EXCPT)
- Combined Primary or Secondary Input file and Exception Output file
- Combined Primary or Secondary Input and Detail Output file

## Parameter Array for SPECIAL

In the RPG II communications interface to the CCP, using SPECIAL, each parameter list must be contained in an RPG II array consisting of five elements — six positions each. The contents of the parameter array for SPECIAL are shown in Figure 6-1. Notice that the contents of the parameter array are similar to the contents of the standard parameter list defined in Chapter 2, except that no CCP work area is provided and, for SPECIAL input operations, the symbolic terminal name with which the operation is to be performed is placed in the parameter list.

Unlike the standard interface defined in Chapter 2, certain parameters for the SPECIAL interface are placed in the record area, instead of parameter array. See *Performing CCP Operations with SPECIAL*, later in this chapter, for details of using the parameter array and record area in various operations.

The parameters for an operation can be placed in the parameter array during RPG II compilation, at pre-execution time, or during execution of the program, depending on how the array is defined in Extension Specifications (see *Defining the Parameter Array*, later in this chapter).

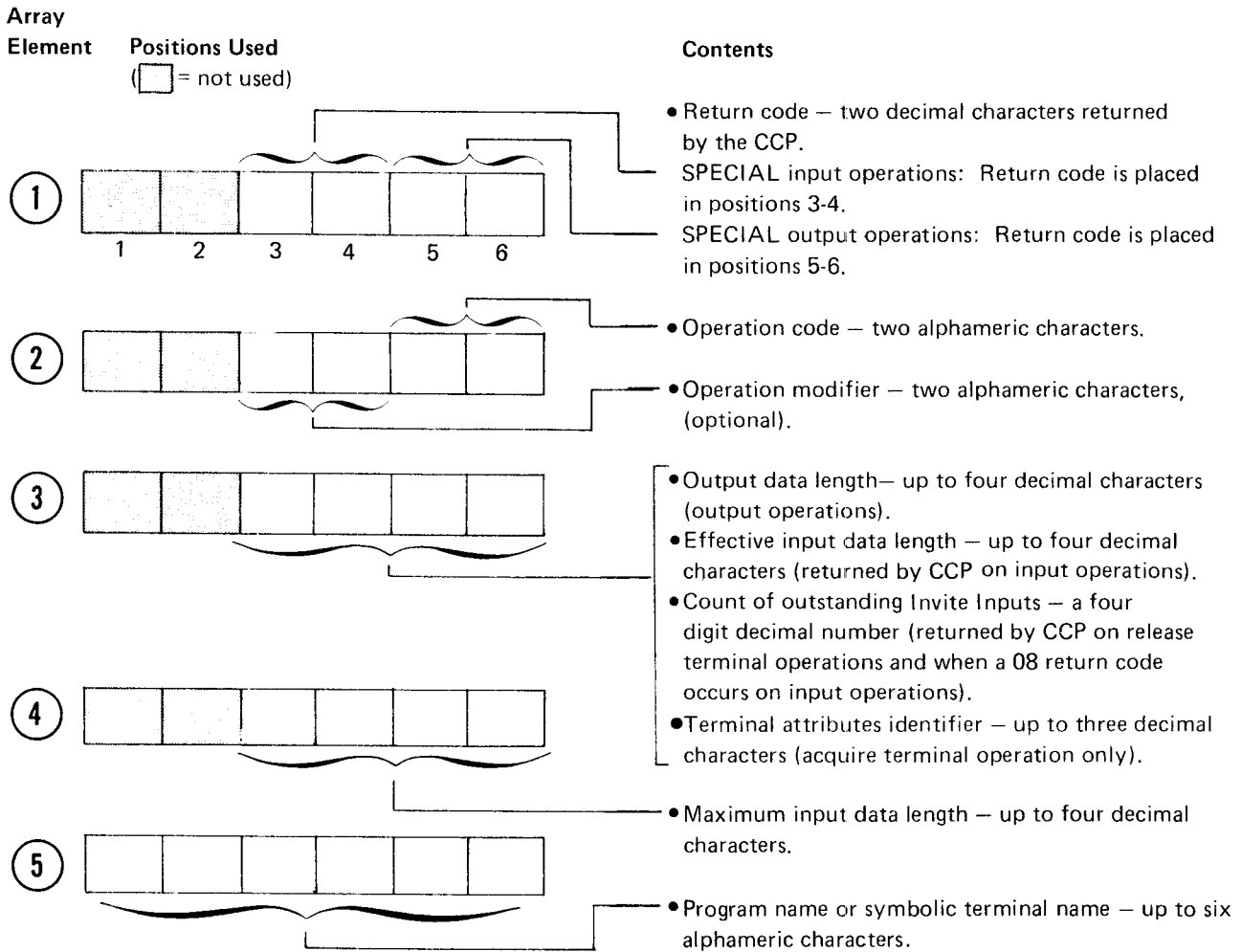
## Record Area for SPECIAL

When using RPG II SPECIAL, input and output record areas for CCP operations are defined by coding Input Specifications and/or Output-Format Specifications. As shown in Figure 6-2, the formats of the input and output record areas for SPECIAL are different from the standard record area format defined in Chapter 2. An additional eight positions, besides the name field, must be set aside in both input and output record areas; therefore, data always begins in position 15. Figures 6-3 and 6-4 show example RPG II specifications for input and output record areas.

### *Input Record Area*

On input operations, the CCP places the following information in the record area, as well as in the parameter array:

- Return code for the input operation and the last output operation
- Effective input length from the input operation
- The program name or the symbolic terminal name



Note: See *Summary – Performing CCP Operations Using RPG II SPECIAL Files*, later in this chapter, for a summary of the use of the parameter array.

Figure 6-1. RPG II Parameter Array for SPECIAL Files.

Because this information is available in the record area, you can check the contents of these areas without moving array elements into fields. You can also check the input return code and effective input length using field indicators. See *Performing CCP Operations with SPECIAL*, later in this chapter, for more information about examining the return code, effective input length, and terminal name.

#### Output Record Area

On heading, detail, total, or exception (H, D, T, or E) output operations, you must place the following information in the output record area:

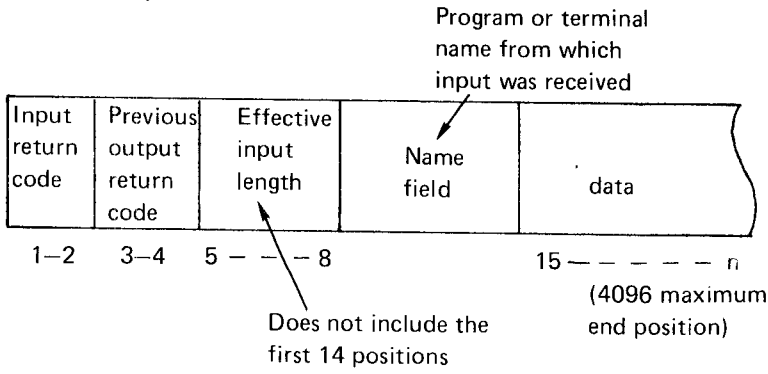
- Operation code (positions 1-4)
- Output length (positions 5-8)
- Symbolic terminal name, CONSOL, program name, or blanks (positions 9-14)

H, D, T, and E output specifications can be used to issued either output or non-I/O operations (such as Release Terminal and Invite Input -- for Invite Input, the maximum input length must be provided in the parameter array). See *Performing CCP Operations with SPECIAL*, later in this chapter, for additional information about setting parameters for various operations.

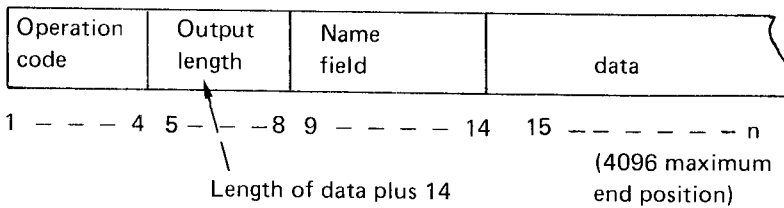
#### CCP Communications Service Subroutines for SPECIAL

Two SPECIAL file subroutines, SUBR92 and SUBR93, are provided by the CCP for terminal operations. SUBR92 is used for processing requests for terminal and task chaining operations. SUBR93 is used only with a "false" SPECIAL file (see *Defining SPECIAL Files*.) in order to bypass the record selection part of the RPG II object program cycle so that input operations can be performed in calculations using the RPG II READ operation code. The first entry to SUBR93 returns to your program without data; all subsequent entries to SUBR93 cause the end of the file condition to be set on for the false SPECIAL file.

### SPECIAL Input Record Area



### SPECIAL Output Record Area



Note: See Summary – Performing CCP Operations Using RPG II SPECIAL Files, later in this chapter, for a summary of the use of record areas.

Figure 6-2. RPG II Communications Record Area Formats for SPECIAL

IBM International Business Machine Corporation			RPG INPUT SPECIFICATIONS																												GX219094 Printed in U.S.A.	
Program						Punching Instruction			Graphic						Card Electro Number						Page 1 of 2		Program Identification						75 76 77 78 79 80			
Programmer						Date			Punch																							
Line	Form Type	Filename	Sequence Number (L,N)	Operation (O)	Record Identifying Indicator	Record Identification Codes									Field Location		Field Name	Decimal Positions	Control Level (L,L,Lg)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators										
						1	2	3	Position	Not (N) C/Z/D Character	Position	Not (N) C/Z/D Character	Position	Not (N) C/Z/D Character	From	To						Plus	Minus	Zero or Blank								
01	I	TERMIN	AA	02													1	3	INRC													
02	I																4	5	OUTRC													
03	I																8	9	EFFLEN													
04	I																14		TRMNAM													
05	I																															
06	I																															
07	I																															
08	I																															
09	I																															
10	I																															

Figure 6-3. Input Record Area for SPECIAL





**RPG CALCULATION SPECIFICATIONS**

Program \_\_\_\_\_ Date \_\_\_\_\_ Punching Instruction \_\_\_\_\_ Graphic Punch \_\_\_\_\_ Card Electro Number \_\_\_\_\_  
 Programmer \_\_\_\_\_ Page 1 2 of \_\_\_\_\_ Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (L, O, L, S, R, SR, AN, OR)	Indicators						Factor 1	Operation	Factor 2	Result Field		Resulting Indicators			Comments		
			And	And	Not	Not	Not	Not				Name	Length	Decimal Positions	Half Adjust (H)	High		Low	Equal
01	C																		
02	C	OR	91																
03	C		92																
04	C		LR																
05	C																		
06	C																		
07	C																		
08	C																		
09	C																		
10	C																		
11	C																		
12	C																		

This instruction is required to define indicators 91 and 92 to the RPG II compiler.

**RPG CALCULATION SPECIFICATIONS**

Program \_\_\_\_\_ Date \_\_\_\_\_ Punching Instruction \_\_\_\_\_ Graphic Punch \_\_\_\_\_ Card Electro Number \_\_\_\_\_  
 Programmer \_\_\_\_\_ Page 1 2 of \_\_\_\_\_ Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (L, O, L, S, R, SR, AN, OR)	Indicators						Factor 1	Operation	Factor 2	Result Field		Resulting Indicators			Comments		
			And	And	Not	Not	Not	Not				Name	Length	Decimal Positions	Half Adjust (H)	High		Low	Equal
01	C																		
02	C	OR	91																
03	C		92																
04	C																		
05	C																		
06	C																		
07	C	SR																	
08	C																		
09	C																		
10	C																		
11	C																		
12	C	SR																	
13	C	SR																	
14	C																		

(Detailed return code checking.)

Figure 6-5. Using Reserved Indicators 91 and 92 (SPECIAL Only)



## Main File Description for SPECIAL

*Page and Line (1-5):* See *RPG II Reference Manual*, SC21-7504.

*Form Type (6):* The preprinted character F identifies this as a file description specification.

*Filename (7-14):* Enter a name for the CCP SPECIAL file. Use this name to refer to the SPECIAL file - *not* to a specific terminal - in subsequent Input, Calculation, and Output Specifications. Rules for forming filenames are given in the *RPG II Reference Manual*, SC21-7504.

*File Type (15):* I, O, and C are valid entries in this column. See combinations on Figure 6-6 for valid File Type/File Designation entries.

*File Designation (16):* P, S, D, and *blank* are valid entries in this column. See Figure 6-6 for valid combinations of File Type/File Designation entries.

*End of File (17):* E is a valid entry with Input and Combined files which are designated as primary or secondary. Leave this column blank if you need not process terminal input data to End of File. When using a false SPECIAL file, set End of File to on during the second and subsequent entries to SUBR93. When using a SPECIAL file with SUBR92, you must set the End of File condition by issuing the Force End of File operation, even though you may have entered an E in this column (see *CCP Operation Codes*, later in this chapter, for an explanation of the Force End of File operation).

*Sequence (18):* See *RPG II Reference Manual*.

*File Format (19):* Enter an F or leave blank (RPG II assumes F). Files must be defined as having fixed-length records, even though variable length records can be handled via the CCP (see *Record Length* and index entry *variable length records*).

*Block Length (20-23):* Enter the same value specified in columns 24-27 (record length), or leave these columns blank. If you specify a block length greater than record length, you will add unnecessary space to your program. Blocking is allowed with BSCA terminals (see index entry *blocking*), but is specified during CCP assignment (see *TERMATTR* statement in *CCP System Reference Manual*).

*Record Length (24-27):* Enter a value equal to the maximum size record or message you expect to receive or send, plus 14. The extra positions are required for the input/output parameters and terminal name in the first 14 positions of the record area. (Additional information must be provided in the record area for certain *3270 Display Format Facility* operations -- see index entry.) For SUBR93, a record length of 1 is suggested.

*Columns 28-39:* Leave these columns blank.

*Device (40-46):* Enter SPECIAL.

*Columns 47-53:* Leave these columns blank.

*Name of Label Exit (54-59):* Enter SUBR92 for all SPECIAL files to be used for communicating with terminals. Enter SUBR93 only if you are defining a *false* file in order to bypass RPG II record selection logic when using demand files (see Figure 6-6). See *Programming Examples* later in this chapter for an example of using SUBR93.

*Columns 60-74:* Leave these columns blank.

## Continuation Specification for SPECIAL

A continuation specification must follow each SPECIAL file description, except for a *false* SPECIAL file. The purpose of the continuation line is to associate an array with the SPECIAL file, to be used as a parameter list for communicating with terminals via the SPECIAL file.

*Form Type (6):* The preprinted character F identifies this as a file description specification.

*Columns 7-52:* Leave these columns blank.

*Column 53:* Enter K (continuation).

*Option (54-59):* Enter the name of the array that is to be used to pass parameters to the subroutine defined in the previous SPECIAL file description. This array must be defined in Extension Specifications. For a description of option (54-59), see *IBM System/3 RPG II Reference Manual*, SC21-7504 under the index heading *continuation lines option*.



Columns 43-74: Leave these columns blank. Note that the array must be defined as alphameric.

### CCP OPERATION CODES

The RPG II form of the CCP operation codes for SPECIAL and EXIT/RLABL communications operations is composed of two parts. The rightmost two characters of the operation code define the basic operation to be performed. The valid RPG II operation code/modifier combinations are summarized in Appendix D. See *Chapter 8: 3270 Display Format Facility* for additional operations and operation codes that are used only with the DFF.

The valid CCP operation codes for RPG II are:

<i>Code</i>	<i>Meaning</i>	<i>Code</i>	<i>Meaning</i>
00	Shutdown Inquiry	DD	Accept No-Wait (5704-SC2 only)
0A	Get (For Stop Invite Input see modifier Code D below)	DE	Invite Input
AA	Get and Reverse Interrupt	DF	Put-No-Wait
AD	Wait	BF	Put-No-Wait Block
GA	Force End of File	CF	Put-No-Wait Message
0B	Put	GF	Put-No-Wait Message – PRUF
BB	Put Block	0H	Get Terminal Attributes
DB	Copy (DFF only)	0I	Acquire Terminal (with present attributes)
EB	Erase All Unprotected (DFF only)	AI	Acquire Terminal (with new attributes)
FB	Put Block – PRUF	BI	Acquire a command-mode non-PRUF terminal with present attributes (5704-SC2 only)
CB	Put Message	CI	Acquire a command-mode non-PRUF terminal with new attributes (5704-SC2 only)
GB	Put Message – PRUF	FI	Generic Acquire (5704-SC2 only)
0C	Combined Put (Record)-Then-Get	GI	Generic Acquire and Set Attributes (5704-SC2 only)
CC	Put (Message)-Then-Get	0K	Release Terminal (and release line)
0D	Accept Input	AK	Release Terminal (keep line)
AD	Wait	BK	Task Chain Request (5704-SC2 only)
		CK	Port Command Request (5704-SC2 only)
		DK	Release and Task Chain (5704-SC2 only)
		CS	Put (Message) with Invite Input
		0W	Put-No-Wait (Message) with Invite Input

The valid operation modifiers for RPG II are:

<i>Code</i>	<i>Meaning</i>
␣␣	Perform carriage-returns at start of data and at end of data for MLTA typewriter terminals. (See index entries <i>new line</i> and <i>end line</i> )
␣A	Shutdown delay inquiry modifier used with shutdown inquiry.
␣A	Perform a carriage-return at start of data, not at end of data.
␣B	Perform a carriage-return at end of data, not at start of data.
␣C	Do not perform carriage-return.
␣D	Stop an Invite Input for this terminal. This modifier is only valid with the Get operation code, that is '␣D␣A'. (See index entry <i>stop invite input</i> .)
␣H	Override (modify) format fields (DFF only -- used with operation codes CB, GB, CF, GF, CS, and ␣W).

Descriptions of the operations listed above are given under *Operations* in Chapter 2 and under *Display Format Facility Operations* in Chapter 8, except for the following three operations, which are used only with the RPG II SPECIAL interface: Force End of File, Put with Invite Input, and Put-No-Wait with Invite Input.

#### **Force End of File (SPECIAL Only)**

When using a primary or secondary SPECIAL file, for communications operations, you may need to inform the communications service subroutine (SUBR92) when to set on an end of file indication in order to cause another secondary file to be processed or to cause your RPG II program to perform LR logic. To issue a Force End of File operation, you need only set the operation code in the second array element (if issued as an input operation) or in the output record area (if issued as an output operation).

CCP does not return a return code for this operation. The information in the parameter list is not changed by this operation.

Force End of File cannot be issued using EXIT/RLABL and cannot be issued to the "False" SPECIAL file (SUBR93). The "False" SPECIAL file will be set at end of file after the second access of the file.

### Put With Invite Input (SPECIAL Only)

Put with Invite Input is the equivalent of a Put Message operation (treated as Put Record for MLTA terminals) followed by Invite Input to the same terminal. This operation allows you to issue a Put and an Invite Input via H, D, T, or E output specifications in a single operation. To issue this operation, place the maximum input length for the Invite Input into the fourth element of the parameter array (if not there already) and place the following parameters in the output record area:

- Operation code '~~WHCS~~' ('~~WHCS~~', if DFF overrides are being put)
- Output length for the Put
- Symbolic terminal name

If Put with Invite Input is being issued at detail output time, the operation code for Accept Input ('~~WD~~') must be placed in the second element of the parameter array prior to issuing the Put. This allows the program to accept data at the next input time in the RPG II cycle.

The CCP places the return code for this operation in the parameter list and sets on indicator 91 or 92 if an error or exception condition results from the operation. The return code and indicators reflect only the result of the Put portion of the operation. If indicator 91 is on after the operation, the Invite Input (and Accept Input, if issued) will have been ignored and a -91 return code returned to your program.

When you want to Put a message to a terminal and Invite additional input from the terminal, you can save coding and possibly save an unnecessary pass through the RPG II logic cycle by using Put with Invite Input.

### Put-No-Wait With Invite Input (SPECIAL Only)

The function and use of Put-No-Wait with Invite Input are the same as Put with Invite Input, except that after the CCP has accepted the operation, your program does not wait for, and does not receive a return code. The operation code is '~~WHW~~' ('~~WHW~~', if DFF overrides are being put).

If indicator 91 is on after the operation, the Invite Input (and Accept Input, if issued) will have been ignored and a -91 return code returned to your program.

## PERFORMING CCP OPERATIONS WITH SPECIAL

Performing a CCP communications operation using SPECIAL involves one or more of the following steps. The specific steps required depend on which operation is being issued and whether the operation is issued as an input operation (primary, secondary, or demand input) or an output operation (H, D, T, or E output):

1. Set the parameters for the operation in the proper location in the parameter array (input) or record area (output).
2. Specify the name of the program for a chain task request or the symbolic terminal name for the operation, if required, in the parameter array (input) or record area (output).
3. Place any data to be transmitted in the record area (output operations).
4. Issue the operation during output or input time in the RPG II program cycle (via Output or Input specifications) or, if demand input or exception output are used, issue a READ or EXCPT operation in calculations to perform the operation.
5. Check the result of the operation by testing the return code and/or using indicators 91 and 92.
6. Examine and/or process other returned information, such as the effective input length (if a data truncated return code is received), symbolic terminal name, and input data.



Operations that involve transfer of either input or output data are restricted to being issued either as input or output operations, since either an input or output record area must be provided. The following operations must be issued as primary, secondary, or demand input operations via a SPECIAL input or combined file:

- Get
- Accept Input
- Accept No-Wait (5704-SC2)
- Stop Invite Input (may result in a Get)
- Get Attributes

The following operations must be issued as output operations, using H, D, T, or E Output Specifications via a SPECIAL output or combined file:

- Put
- Put-No-Wait
- Put with Invite Input
- Put-No-Wait with Invite Input
- Copy (DFF)
- Chain Task Request (5704-SC2)
- Erase All Unprotected (DFF)
- Release and Task Chain (5704-SC2)
- Wait

The following operations can be issued as either input or output operations, since they do not result in transfer of data:

- Acquire Terminal
- Invite Input
- Shutdown Inquiry
- Force End of File
- Release Terminal

Of these operations, Acquire Terminal, Invite Input, and Shutdown Inquiry are most logically issued as output operations; Force End of File is most logically issued as an input operation. If you set new attributes using the Acquire Terminal and Set Attributes operation, the attribute identifier must be given. If done as an input operation, the attribute identifier is placed on the third CCP array element. If done as an output operation, the identifier is placed in the output record. (See *Non-I/O Operations*, later in this section.)

The Put-Then-Get operation, which can be used only with combined SPECIAL files, is both an input and output operation and requires both an input and an output record area (see *Put-Then-Get Operation*, later in this section).

### Performing CCP Operations Using Primary, Secondary, or Demand Input

To cause an operation to be performed during primary, secondary, or demand input time in the RPG II program cycle, provide an input record area in the appropriate format (Figure 6-2 and 6-3) and place the following information in the parameter array:

1							
2							Operation
3							
4							Maximum Input Length
5							Symbolic Terminal Name

When using a primary or secondary SPECIAL file, this information must be placed in the array prior to the first read from the file. Therefore, you should load the array at either compilation time or pre-execution time with the appropriate information for the first operation. For subsequent operations from the file, if any of the array contents must be changed, they can be reset in calculations, as shown in Figure 6-8.

IBM		RPG CALCULATION SPECIFICATIONS										Form GX21-9093 Printed in U.S.A.	
Program		Punching Instruction		Graphic		Card Electro Number		Page 1 2 of		Program Identification 75 76 77 78 79 80			
Programmer		Date		Punch									
C	Line	Form Type	Control Level (C=CP, L=LP, SR, ANV, OR)	Indicators			Factor 1	Operation	Factor 2	Result Field		Resulting Indicators	Comments
				And	And	Not				Name	Length		
0	1	C											
0	2	C											
0	3	C											
0	4	C											
0	5	C											
0	6	C											
0	7	C											
0	8	C											
0	9	C											

These MOVE operations set up the parameter array for a Get operation from the terminal named in the field TNAME, with a maximum input length of 44.

Figure 6-8. Setting the Parameter Array Contents for a Primary, Secondary, or Demand Input Operation

In some applications, it is not convenient to perform input operations at primary or secondary input time in the RPG II cycle. Using a demand file, you have more control over when the file is read, since you cause the file to be read whenever you issue the READ operation in calculations. Therefore, demand files are more flexible for communicating interactively with terminals.

If the file is used for a series of operations, you may not need to modify all parameter array elements for every operation.

#### Operation Code

In setting or modifying the operation code, you can avoid having an invalid operation code/modifier combination by always moving both parts of the operation code into the parameter array, as is done in Figure 6-8. (See *Put-Then-Get Operation*, later in this section, for special considerations.)

#### Maximum Input Data Length

For operations that can result in input data being received in the input record area (Get, Accept Input, Accept No-Wait, Get Attributes, and Stop Invite Input), set the fourth element of the parameter array to indicate the maximum amount of data you expect from a terminal, including device control characters your program handles (see index entry *device control characters*) and including the 14 characters reserved in the record area for CCP use. The maximum number you can enter is 4096.

For the Get Attributes operation, set the value of the fourth array element to 35, the length of the attributes

data (21) plus the required information in the first 14 positions of the record area.

This element of the parameter array is never modified by the CCP; therefore, as long as your maximum input length remains the same, you need not reset this element.

#### Symbolic Terminal Name

You must place the symbolic terminal name (or blanks) in the fifth array element when you issue one of the following operations at primary, secondary, or demand input time:

- Get
- Invite Input
- Stop Invite Input
- Get Attributes

You can use blanks in the fifth array element for these operations only in *single requesting terminal (SRT) programs* (see index entry). When you use a blank terminal name, the CCP returns the name of the terminal that requested the program in the input record area and in the parameter array after the operation.

You need not specify a name for the Accept Input operation; however, after the operation, the CCP places the name of the program or terminal from which the input data was received in the fifth element of the parameter array and in the record area.

See *Put-Then-Get Operation*, later in this section, for special considerations.

### Testing the Return Code

After any CCP operation, you should test the return code to determine the result of the operation, since the result may require special actions in your program. If the operation resulted in an error condition (negative return code) or exception condition (positive return code), you might either take specific actions in your program or inform the system operator of the condition by issuing a message to the console.

The CCP places the return code for input operations issued at primary, secondary, or demand input time in the first two positions of the input record area as well as in the middle two positions of the first element of the parameter array (see Figures 6-1 and 6-2). The return code for the last output operation is available in positions 5-6 of the first array element and in positions 3-4 of the input record area. Also, the CCP sets on indicators 91 and 92 for error and exception conditions, respectively (see *Indicators Reserved for CCP Use*). Thus, you can test return codes in a variety of ways:

- Using Calculations to test the contents of the return code field in the input record area.
- Using indicators 91 and 92 to condition operations in Calculations and to condition output lines (Figure 6-5).
- Using record identification codes or field indicators on Input Specifications.

Figure 6-9 shows an example of testing the return code for plus, minus, or zero by means of a compare operation and resulting indicators. If you wish, you can test for the exact return code to determine which exception or error condition occurred and take appropriate action. Return codes and recommended program actions are summarized in Appendix E.

If indicator 91 is on when any input operation is requested, the operation is ignored by SUBR92 and a return code of -91 is returned (appears as 9J). This return code may be used to prevent blanking out of previous input fields (see Figure 6-10).

Figure 6-10 shows how indicators may be set on for input operations with SPECIAL files to indicate record type, normal completion, error or exception return code. In this example, the record *NAM* is read with indicator 01; the record *ADR* is read with indicator 02. If there is not a normal completion of the input operation, indicator 03 is on, the return code is moved into a two-character numeric field (RTNCOD) for more detailed testing in calculations.

Special considerations are involved when you use the input return code from positions 3 and 4 of the first element of the array. If the correct sequence of operations is not followed, the sign of the input return code can be lost. The procedure requires that the first element be moved into a four-position alphameric field via a MOVE instruction, and that the same four-position field be moved into a two-position field via a MOVE instruction. The two-position field then contains the input return code with the correct sign.

Although it is possible to do preliminary return code checking using the method shown in Figure 6-10, that method is not recommended, since it is much easier to detect positive and negative return codes using indicators 91 and 92.

### Examining the Effective Input Length

On SPECIAL input operations, the CCP returns the effective input length in positions 5-8 of the input record area (Figure 6-2). This value includes only the actual length of data available to your program and does not include the 14 characters preceding the data in the record area or the length of truncated data. The input data begins in position 15 of the record area.

After you have issued an input operation, you can examine the effective input length, for example, to verify that the terminal entered a required number of characters, in order to detect erroneous input. You might also use the value of the effective input length as the limiting value in a routine that scans the input data.

The effective input length may be equal to or less than the maximum input length you specified for the operation, depending on the actual amount of data transmitted to your program by a terminal. If the terminal transmits more data than the maximum you specified, the excess data is truncated and your program receives a "data truncated" return code. (See index entry *data truncated, special use in DFF*, for a special meaning of the data truncated return code.)

The effective input length contains the number of current outstanding Invite Inputs if a return code of 08 is returned after an Accept Input operation.

For 5704-SC2 only, the effective input length contains the number of current outstanding Invite Inputs if a return code of 04 or 16 is returned after an Accept No-Wait operation.

Contents of the effective input length field for each operation code and return code is summarized in Appendix E (Table E-7).

**RPG CALCULATION SPECIFICATIONS**

Form GX21 9093  
Printed in U.S.A.

IBM International Business Machine Corporation

Program \_\_\_\_\_ Punching Instruction \_\_\_\_\_ Graphic \_\_\_\_\_ Card Electro Number \_\_\_\_\_  
 Programmer \_\_\_\_\_ Date \_\_\_\_\_ Punch \_\_\_\_\_ Page 1 of 2 Program Identification 75 76 77 78 79 80

C	(U.S. OR)	Indicators	Operation	Factor 2	Result Field		Resulting Indicators			Comments
					Name	Length	Arithmetic	Plus	Minus	
0 3	C									
0 4	C									
0 5	C									
0 6	C									
0 7	C									
0 8	C									
0 9	C									
1 0	C									
1 1	C									

Assume RTNCOD is defined in Input Specifications as the first two positions of the input record area (numeric field).

In this example, if the return code is greater than zero, a subroutine that handles exception conditions is executed; if the return code is negative, an error-handling subroutine is executed.

Figure 6-9. Using Calculations to Test the Return Code in the SPECIAL Input Record Area

### Examining a Returned Name

For the Accept Input operation and for operations issued with a blank name field, CCP returns the program name or a symbolic terminal name to your program (see index entry *operations*). For all operations issued at primary, secondary, or demand input time, CCP places this name in positions 9-14 of the input record area. You may need to examine this name to determine which program or terminal is communicating with your program.

For example, in a *multiple requesting terminal (MRT) program* (see index entry), you should examine the terminal name you receive from an Accept Input operation in order to determine if the terminal is a new requester or if it is already attached to your program. If the terminal is a new requester, you might move the terminal name to a save area. If the terminal has previously provided input to your program, you may need to associate new data with data previously received by comparing the terminal name with a saved terminal name. *Example 2* under *Programming Examples* shows an example of examining the terminal name after an Accept Input operation.

If a program can be requested from both a terminal and another program using the Task Chain Request operation, you may want to determine how the program was requested. This can be accomplished by checking for a 14 return code, indicating a Task Chain Request operation. This information is useful if a program communicates with the requestor since your program cannot communicate with a task chain requesting program.

### Performing CCP Operations Using Heading, Detail, Total or Exception Output

When issuing CCP output operations at H, D, T, or E output time via a SPECIAL output or combined file, you must place the operation code, output length, and the program or terminal name in the output record area, as shown in Figures 6-2 and 6-4.

The output length for SPECIAL output operations includes the 14 positions required in the record area for the operation code, output length, and terminal name. This length also includes device control characters you are inserting yourself and the special information required for certain 3270 DFF operations. Thus, the value of the highest end position used on the Output Specifications should be used as the output length. The maximum length is 4096.

After an operation is performed at H, D, T, or E output time, CCP places the return code in positions 5-6 of the first element of the parameter array (Figure 6-1). You can easily test the return code for a positive or negative value by using indicators 91 and 92 (see *Indicators Reserved for CCP Use*, earlier in this chapter). In order to test for a specific return code value, however, you must move the contents of the first array element to a two-position numeric field. Figure 6-11 shows how these methods can be used in combination to test the return code. See *Programming Examples*, later in this chapter, for additional examples of testing return codes.



RPG CALCULATION SPECIFICATIONS

Form GX21-9093  
Printed in U.S.A.

IBM International Business Machine Corporation

Program	Punching Instruction	Graphic	Card Electro Number
Programmer	Date	Punch	

Page 1 2 of Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (L, O, L, R, SR, AN, OR)	Indicators			Factor 1	Operation	Factor 2	Result Field		Resulting Indicators			Comments
			And	And	And				Name	Length	Arithmetic	Plus	Minus	
01	C		91			SETON							LR	
02	C		LR			GOTO END								
03	C		92			EXSR RETCHK								
04	C					\$								
05	C													
06	C				END	TAG								
07	C	SR			RETCHK	BEGSR								
08	C	SR				MOVE ARY, 1	RTNCOD	20						
09	C	SR			RTNCOD	COMP 04								05 SHUTDOWN REQST-
10	C	SR	05			GOTO ENDRET								TERM CANCELLED
11	C					\$								
12	C					\$								(Other return code testing.)
13	C	SR			ENDRET	TAG								
14	C	SR				SETOF								9192
15	C	SR				ENDSR								
16	C													
17	C													
18	C													
19	C													
20	C													

Figure 6-11. Example of Return Code Testing After a SPECIAL Output Operation

Exception Output

To perform exception output to a specific terminal (via a SPECIAL output or combined file), place the appropriate operation code, output length, and program or terminal name in the output record area (Figure 6-2 and 6-4), and issue the EXCPT operation in calculations. If the program or terminal name is less than six characters long, it must be left-justified and padded with blanks in the name field of the output record area.

Exception output enables you to check the return code in calculations immediately after performing the EXCPT operation.

If you are sending multiple lines of output to one terminal, or single lines to more than one terminal during any one EXCPT operation, you can check the return code for only the last operation (last line of output). Therefore, it is suggested that one output record be sent to only one terminal by each EXCPT operation. This can be controlled through the use of indicators.

See *Programming Examples, Example 2*, later in this chapter, for examples of exception output.

### Heading, Detail or Total Output

In general, using H, D, or T output to issue CCP operations is similar to using exception output, in that the operation code, output length, and terminal name are placed in the output record area rather than in the parameter array and several lines of data can be sent to the same or different terminals on each program cycle.

You must be aware that when you send several lines of output to the same SPECIAL output file using H, D, or T output, the return code in the parameter list applies only to the last output line. Therefore, you have no way of knowing when an exception condition occurred on output lines previous to the last output line. Also, when an error occurs, you have no way of knowing on which output line the error occurred.

See *Programming Examples, Example 1*, and the *MRT Programming Example Using the Display Format Facility* in Chapter 8 for examples of H, D, and T output lines.

### Put-Then-Get Operation

The Put-Then-Get operation must be issued with a combined SPECIAL file. The operation is not actually performed via SUBR92 until the input part of the combined operation is performed.

### Combined Demand Input/Exception Output File

Perform the Put-Then-Get operation as follows:

1. Place the maximum input length in the parameter array.
2. Place the following information in the output record area:
  - Put-Then-Get operation code (positions 1-4)
  - Output length (positions 5-8)
  - Symbolic terminal name (positions 9-14)
3. Issue an EXCPT operation.
4. Issue a READ operation.

After you issue the READ operation, the input data is available in the input record area. You need not provide a terminal name or operation code in the parameter array for the READ, since SUBR92 assumes that the first input operation for a combined file following the Put part of a Put-Then-Get to the file is the Get part of the operation from the same terminal.

When issuing a Put-Then-Get to a terminal using EXCPT and READ, you may do output to other terminals after doing the output part of the combined operation. If successive Put-Then-Get operations are performed to the same or different terminals using the same SPECIAL file before doing the input part of the combined operation, then only the last output will be sent to the terminal.

See *Programming Examples, Example 2*, for an example of using READ and EXCPT to issue a Put-Then-Get operation.

### Combined Primary or Secondary Input/H, D, or T Output File

You can also issue a Put-Then-Get operation using H, D, or T output and primary or secondary input with a combined SPECIAL file. Place the maximum input length in the parameter array for the combined SPECIAL file. Place the following information in the output record area:

- Put-Then-Get operation code (positions 1-4)
- Output length (positions 5-8)
- Symbolic terminal name (positions 9-14)

The input data from the Put-Then-Get is available in the input record area after Primary or Secondary input in the RPG II cycle. You need not provide a terminal name or operation code parameter array for the input part of the combined operation, since SUBR92 assumes that the first input from the file is the input part of the operation for the same terminal.

Considerations for performing other output operations after the output part of the combined operation and for performing successive Put-Then-Get operations are the same as described for EXCPT and READ.

#### *Testing the Return Code After a Put-Then-Get*

The return code from a Put-Then-Get is placed in positions 3-4 of the first array element. Positions 5-6 of the first array element will be zero if the last output operation was the Put part of this combined operation. If an error condition resulted from the operation (this can be determined using indicator 91) you can determine whether the error occurred on the Put or on the Get by testing the value in the third array element. If this value is the same as the output length specified for the Put-Then-Get, then the error occurred on the Put portion of the operation (the Get was not performed). If the value of the third array element is different from the output length specified for the operation, then the error occurred on the Get.

#### *Note Concerning the Use of Put-Then-Get*

The typical use for Put-Then-Get is to issue a message to the system operator (CONSOL) and receive instructions. In an MRT program, the need to modify the parameter list can be minimized by initializing the array for the combined file with the 'WWD' operation code (Accept Input), a blank terminal name, and the maximum input length used by the program. Thus, when the Put-Then-Get operation is issued, the array need not be modified in order to continue with normal operation, since the Put-Then-Get operation code was given in the output record area.

## **Non-I/O Operations**

Non-I/O operations can be issued at either input time (primary, secondary, or demand) or at output time (H, D, T, or E), since they do not result in transfer of data to or from your program. The non-I/O operations are:

- Acquire Terminal
- Release Terminal
- Invite Input
- Shutdown Inquiry
- Force End of File

When issued via SPECIAL files, Acquire Terminal, Release Terminal, Invite Input, and Shutdown Inquiry are most logically issued as output operations; Force End of File is most logically issued as an input operation.

When you issue non-I/O operations at input time using SPECIAL files, you must provide a "dummy" input specification to avoid getting an UNIDENTIFIED RECORD halt, since RPG II assumes that valid data is contained in your input record area after the operation. See *Programming Examples, Example 2*, for an example of a "dummy" input specification.

While all of the non-I/O operations can be issued using SPECIAL files, it is also convenient to use EXIT/RLABL for these operations, since no input or output record area is required and there is no requirement for RPG II to move input or output fields. You must be aware, however, that using both SPECIAL and EXIT/RLABL for CCP operations increases the main storage requirement of your program. This can be an important consideration in some cases.



**SUMMARY – PERFORMING CCP OPERATIONS USING RPG II SPECIAL FILES**

(see notes on opposite page)

**INPUT RECORD AREA**

Input Return Code	Output Return Code	1. Effective Input Length 2. #Outstanding Invite Inputs	Program or Symbolic Terminal Name	Data
I <sub>4</sub>	I <sub>5</sub>	I <sub>3</sub>	I <sub>7</sub>	I <sub>6</sub>
1 2	3 4	5 8	9 14	15

**PARAMETER ARRAY**

Return Code	Operation Code/ Modifier	1. Effective Input Length 2. Attributes ID 3. #Outstanding Invite Inputs	Maximum Expected Input Length	Program or Symbolic Terminal Name
I <sub>4</sub> I <sub>5</sub>	O <sub>4</sub> O <sub>3</sub>	I <sub>3</sub> O <sub>4</sub>	I <sub>1</sub> I <sub>2</sub>	O <sub>4</sub> O <sub>6</sub>

**OUTPUT RECORD AREA**

Operation Modifier	Operation Code	1. Output Length 2. Attributes ID	Program or Symbolic Terminal Name	Data
O	O <sub>1</sub>	O <sub>2</sub>	O <sub>7</sub>	
1	4	5 8	9 14	15

**Operations Issued at Input Time**

*Information Supplied by Programmer:*

- 1 Place the operation code here before issuing the operation. *Exception:* On a Put-Then-Get operation, this element is disregarded on the Get portion of the operation. Upon completion of the operation, this element remains unchanged.
- 1 Place the maximum expected input length here before issuing the operation (includes the first 14 positions in the record area). Upon completion of the operation, this element remains unchanged.

- I<sub>2</sub> Place the symbolic terminal name here prior to issuing the operation. *Exception:*
1. Accept Input -- contents are ignored by the CCP
  2. Blanks - CCP assumes the requesting terminal (SRT programs)
  3. Shutdown Inquiry -- contents are ignored by the CCP

This element is changed after Accept Input or a blank terminal name is used. *Exception:* When a Put-Then-Get is issued with a blank terminal name, the name of the requesting terminal is returned only in positions 9-14 of the input record area.

*Information Returned by CCP:*

- I<sub>3</sub>
1. Contains the effective input length (not including the first 14 positions), after an input operation (Get, Accept Input, Put-Then-Get, Get Terminal Attributes, and Stop Invite Input that fails to stop input.)
  2. Contains the count of outstanding Invite Inputs when:
    - a. A Release Terminal operation is issued
    - b. A 08 return code was received
    - c. Shutdown Inquiry operation (5704-SC2 only)

I<sub>4</sub> Input operation return code.

I<sub>5</sub> Return code from the last output operation using this array.

I<sub>6</sub> If the 3270 Display Format Facility is used, position 15 contains the AID character, except when data is received with the program request on systems *without* Program Request Under Format (PRUF).

I<sub>7</sub> After an Accept Input operation, Put-Then-Get operation, or an operation issued with a blank terminal name, the CCP returns a symbolic terminal name or for an Accept Input in a program loaded by a Chain Task Request, the name of the program that issued the Chain Task Request operation in these positions.

**Operations Issued at Output Time**

*Information Supplied by Programmer:*

O Place the operation code here prior to issuing the operation.

O<sub>1</sub> Place the output length here for output operations (include the first 14 positions of the record area). Place the attributes identifier here for Acquire terminal with Set Terminal Attributes modifier. CCP ignores the contents of this field for Release Terminal, Invite Input, Shutdown Inquiry, and Acquire Terminal without Set Terminal Attributes modifier.

O<sub>2</sub> Place the requested program name, the symbolic terminal name, blanks, or CONSOL here. This entry is ignored for Shutdown Inquiry.

O<sub>7</sub> If the Display Format Facility is used, positions 15-20 contain the format name for the Put Message operation; position 15 contains the Write Control Character for the Put Overrides operation.

*Information Returned by CCP:*

O<sub>3</sub> Output operation return code.

O<sub>4</sub> These elements are unchanged after the output operation. Place the maximum input length in the fourth array element prior to issuing Invite Input, Put with Invite Input, Put-No-Wait with Invite Input, and Put-Then-Get.

O<sub>5</sub> After the Release Terminal operation, this element contains the count of outstanding Invite Inputs.

O<sub>6</sub> CCP returns the symbolic terminal name in this element after operations issued with a blank terminal name, except Put-Then-Get. On Put-Then-Get operations issued with a blank terminal name, CCP returns the symbolic terminal name in positions 9-14 of the input record area after the Get portion of the operation is performed.

**EXIT/RLABL COMMUNICATIONS INTERFACE**

All of the CCP operations described under *Operations* in Chapter 2 can be performed using EXIT/RLABL. CCP provides four subroutines for use with EXIT: SUBR91, which can be used for all communications operations including task chain request operations; SUBR90, which can be used to move fields to or from a record area; SUBR87, which is used to issue Chain Task Request operations only; and SUBR88, which is used to accept both program request and chain task data. SUBR87 and SUBR88 are best suited for programs that are primarily used for task chaining operations and do not communicate directly with terminals.

*Note:* You must not use SPECIAL subroutines SUBR92 and SUBR93 with EXIT/RLABL.

### Parameter Array

As with the SPECIAL interface, you must place the parameters for an operation in a parameter array. The format of the parameter array for EXIT/RLABL (Figure 6-12) is the same as for SPECIAL, except as follows:

- CCP always places the return code from an operation in positions 5-6 of the first array element, not in the record area.
- The program name or the symbolic terminal name is always in the record area. Therefore, you may define an array of four elements if you are using the array only for EXIT/RLABL. You can use the same array for both SPECIAL and EXIT/RLABL if the array has five elements.

The Extension Specifications for defining parameter arrays are described in *Defining the Parameter Array*, previously in this chapter.

### Record Area

The record area for communications operations using EXIT/RLABL must be defined by an RLABL in calculations (see *EXIT to SUBR91*). The record area (Figure 6-13) consists of the name field and a data area. Note that the format is different from the record area for SPECIAL.

### EXIT to SUBR91

The linkage to SUBR91 to perform a CCP operation involves an EXIT operation followed by two RLABLs, as follows:

RPG CALCULATION SPECIFICATIONS													
Punching Instruction		Graphic Punch		Card Electro Number									
Factor 1	Operation	Factor 2	Result Field										
			Name	Length	Decimal Positions Half Adjust (H)								
18 19 20 21 22 23 24 25 26 27	28 29 30 31 32	33 34 35 36 37 38 39 40 41 42	43 44 45 46 47 48	49 50 51	52	53							
EXIT SUBR91		RLABL		RLABL		Parameter array		ARY		RA,1		Record area	
Z													

The first RLABL is the name of an array (ARY, for example) which is used as a parameter list by SUBR91. It may be the same array as used for a SPECIAL file, except that the fifth element is not used. See *Defining the Parameter Array*, earlier in this chapter.

The second RLABL is the record area. The first six positions of this area must contain the program name or the symbolic terminal name for which an operation is intended, followed immediately by the data to be sent or received to or from a terminal. Since this data may consist of many fields, the CCP provides another EXIT subroutine, SUBR90, to move fields to or from the record area (see *EXIT to SUBR90*).

The size of the record area should be large enough to contain the maximum amount of data to be sent or received plus the six characters for the name field. If the largest amount of data to be sent or received is greater than 256 (including the name field, then an array must be defined for this area, since the maximum size of an RPG II alpha field is 256. Thus, if 800 characters are needed (for example with a 3270 Display) plus the symbolic name (a total of 806), an array could be defined containing four elements of 202 characters each. This would create a contiguous area of core of 808 bytes, which is large enough. If an array is defined as the record area for EXIT/RLABL linkage, the second RLABL must be a reference to the first element of the array:

RPG CALCULATION SPECIFICATIONS													
Punching Instruction		Graphic Punch		Card Electro Number									
Factor 1	Operation	Factor 2	Result Field										
			Name	Length	Decimal Positions Half Adjust (H)								
18 19 20 21 22 23 24 25 26 27	28 29 30 31 32	33 34 35 36 37 38 39 40 41 42	43 44 45 46 47 48	49 50 51	52	53							
EXIT SUBR91		RLABL		RLABL		Parameter array		ARY		RA,1		Record area array	

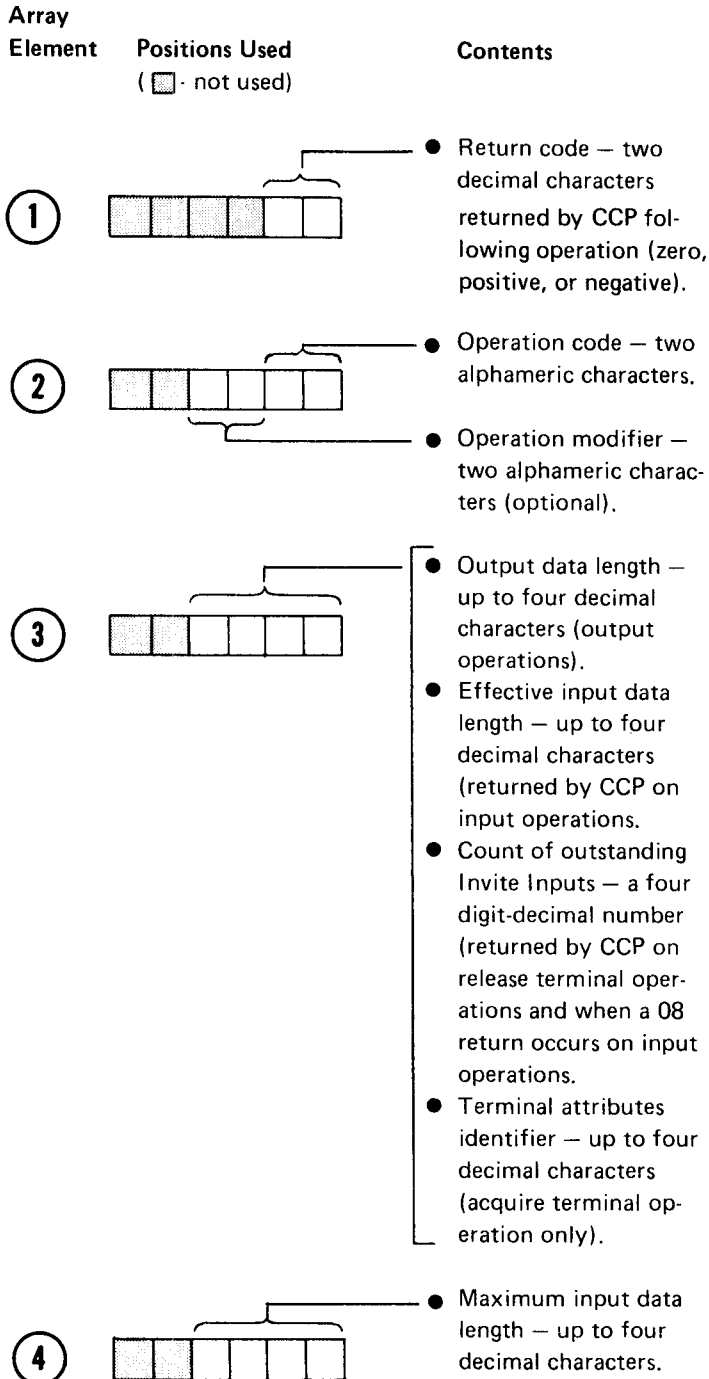
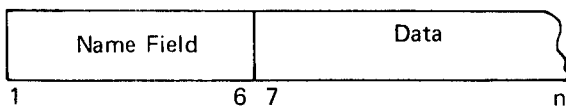


Figure 6-12. RPG II Parameter Array for EXIT/RLABL



(4096 maximum, including the terminal name)

Figure 6-13. Record Area Format for EXIT/RLABL CCP Interface

### Setting the Parameters for an Operation

Issuing a non-I/O operation (such as Release Terminal) involves simply setting the contents of the parameter array with MOVE instructions and placing the terminal name in the record area. Since only the terminal name is required in the record area for non-I/O operations, you can issue the operation as shown in the following example, specifying a field that contains the terminal name as the second RLABL.

RPG CALCULATION SPECIFICATIONS					
	Punching Instruction	Graphic Punch		Card Electro Number	
Factor 1	Operation	Factor 2	Result Field		
			Name	Length	
18 19 20 21 22 23 24 25 26 27	28 29 30 31 32	33 34 35 36 37 38 39 40 41 42	43 44 45 46 47 48	49 50 51	Decimal Position Half Adjust (H)
	MOVE ' K'		ARY,2		
	EXIT SUBR91				
	RLABL		ARY		
	RLABL		TNAME		

If, however, you are issuing an operation that involves a number of fields of input or output data, you will need to use SUBR90 to move fields to or from the record area, since there is no facility in RPG II Calculations to move fields to or from various section of another data field. (A series of MOVE or MOVEL operations could be used to move a small number of fields.)

### Setting the Parameters for EXIT/RLABL Operations

All parameters for CCP operations are placed in the parameter array. The program name or the symbolic terminal name is always placed in the record area (see EXIT to SUBR91).

### Operation Code

In setting the operation code, you can avoid having an invalid operation/modifier combination by always moving both parts of the operation code into the second array element. The valid operation and modifier codes are given in CCP Operation Codes, earlier in this chapter.

### *Output Length*

For EXIT/RLABL output operations, place the length of the output data in the third array element. The output length includes device control characters which you are inserting into the data in your program (see index entry *Device Control Characters*). The output length should not include the six characters for the program name or the symbolic terminal name.

The maximum output length you can specify is 4090. See *RPG II Programming Considerations*, later in this chapter, for information about using the third array element with the Acquire Terminal operation.

### *Maximum Input Length*

The maximum input length in the fourth array element for an EXIT/RLABL operation must not include the six characters for the program name or the symbolic terminal name, but must include the length of any device control characters you are handling in your program. You must enter a maximum input length for the following operations:

- Get
- Invite Input
- Put-Then-Get
- Get Attributes

If you also specify a maximum input length for the Accept Input operation, it will override the length given for the Invite Input operation. For the Get Attributes operation, set the value of the fourth array element to 21.

### **Examining Returned Information**

After you have issued a CCP operation using EXIT/RLABL, you should examine information returned to your program by the CCP:

- Return Code (*Note:* Indicators 91 and 92 are not reserved or used with SUBR91.)
- Effective input length (if an input operation was performed)
- The program name or the symbolic terminal name (if it was set by the CCP, such as for an Accept Input operation or a Put issued with a blank terminal name)

### *Testing the Return Code*

After an EXIT/RLABL operation, the CCP places the return code in positions 5-6 of first parameter array element. Figure 6-16 shows an example of testing the return code for plus, minus, or zero by means of a compare operation and resulting indicators. In this example, a subroutine that checks for specific return codes is executed when a positive or negative return code is encountered after a Stop Invite Input operation. Before checking the return code after an EXIT/RLABL operation, you must move the return code from the first array element to a 2-position numeric field.

*Note:* Notice the use of a MOVE instruction to move the terminal name from the field TNAME to the first six positions of the record area, INREC. This technique is useful when the record area consists of the symbolic terminal name and a single field (in this case assume the input record is a single 10-character field).

### *Examining the Effective Input Length*

After operations that return input data to your program, you may need to know the actual length of input data returned to your program, that is, the effective input length. This value does not include the six characters for the program name or the symbolic terminal name and does not include the length of truncated data when a terminal sends more data than you specified in your maximum input length for the operation. Special considerations are involved when you use the 3270 Display Format Facility (DFF) input operations; see *3270 Display Operations*.

In order to use the effective input length, you must first move the contents of the third array element to a numeric field.

### *Symbolic Terminal Name*

On EXIT/RLABL operations for which the CCP returns a terminal name (see index entry *Operations*), you may need to examine or otherwise use the terminal name (for example, to associate the input data with data previously received from the same terminal). The CCP places the returned symbolic terminal name in the first six positions of the record area.

See *Programming Examples, Example 2*, later in this chapter, for examples of saving and examining terminal names in an MRT program.

## Exit to SUBR90

You must prepare the record area before doing an output operation using SUBR91. Therefore, you must move fields into this area of storage before requesting the I/O operation. For input operations, you are presented with a mass of data from which you must extract specific fields. When you use SPECIAL file processing, the input and output specifications provide this capability. However, in RPG II Calculations there is no facility for selecting fields from various sections of another data field. Thus, SUBR90 is provided to serve this purpose.

Figure 6-14 illustrates the coding required to link to SUBR90.

The first four RLABL's following an EXIT to SUBR90 provide information to SUBR90 about the fields to be moved.

- A** The first RLABL is an indicator. If you "set on" this indicator prior to exiting to SUBR90, the sub-routine moves the data from the fields specified in subsequent RLABL's **E** to the record area. If the indicator is off when control is given to the routine, data is moved from the record area to the fields specified.
- B** The second RLABL is a field which must contain either the total length of data to be moved, if moving data to the output area, or the total length of the record area, if moving data from the input

area to fields. The total length of all the fields which are to be moved from or to the record area must be less than or equal to the value contained in the field named by this RLABL.

- C** The third RLABL must name the data area used with SUBR91. If the data area is larger than 256, an alpha array may be defined which is at least as large as the data area needed. In this case, the third RLABL passed to SUBR90 should be the first element of the array (indexed by the integer 1).
- D** The fourth RLABL must name a field which contains the number of fields to be moved to or from the data area. It also defines the number of RLABL's which follow the fourth RLABL.
- E** The fifth and succeeding RLABL's name the fields which are to be moved to or from the data area. These names may refer to fields, or an array indexed by a constant. The RLABL's may not name an array without an index. Fields are moved into the data area in the order you specify them as RLABL's; therefore, the fifth RLABL must identify a six-position terminal name field as the first field in the record area. There must be a character-for-character correspondence between the "from" and "to" fields; if blanks are to occur in the data area, fields containing blanks must be provided in the list of RLABL's.

IBM International Business Machine Corporation																																																												RPG CALCULATION SPECIFICATIONS										Form GK21 9093 Printed in U.S.A.									
Program															Punching Instruction					Graphic					Card Electro Number					Page 1 2 of					Program Identification 75 76 77 78 79 80																																												
Programmer															Date					Punch																																																											
Line	Form Type	Control Level (L, O, Lg, Lf, SR, AN, OR)	Indicators						Factor 1	Operation	Factor 2	Result Field		Decimal Positions	High Adjust (H)	Resulting Indicators			Comments																																																												
			Not	And	And	Not	Not	Not				Name	Length			Plus	Minus	Zero																																																													
01	C																																																																														
02	C																																																																														
03	C																																																																														
04	C								EXIT SUBR90																																																																						
05	C								A - RLABL																																																																						
06	C								B - RLABL																																																																						
07	C								C - RLABL																																																																						
08	C								D - RLABL																																																																						
09	C								E - RLABL																																																																						
10	C																																																																														
11	C																																																																														
12	C																																																																														
13	C																																																																														
14	C																																																																														
15	C																																																																														
16	C																																																																														

INDIC  
RECLN  
REC (OR REC, L)  
NFLDS  
FLDL  
FLD2  
FLDN

These field names are used as examples; they are not required names.

These field can be defined here or elsewhere to be the length you desire.

Figure 6-14. Coding Required to Use SUBR90

## Alphameric and Numeric Fields

Alphameric and numeric fields are properly left-justified or right-justified by your RPG II program prior to being moved to the record area by SUBR90 (SUBR90 treats all fields as alphameric fields). On input operations, however, the terminal operator must know whether he should enter data into a field left-justified (alphameric) or right-justified (numeric). These instructions may have been given previously to the terminal operator by means of a published procedure, preprinted typewriter form, or a display format (3270). If not, your program should put out a format to the terminal before any data is requested from the terminal to indicate to the operator where the data should appear in the record and how fields should be justified. For batch terminals, such as other central processing units, this step is unnecessary, since the communicating programs will be written to expect data in a certain format.

## Editing with SUBR90

The standard RPG II editing facilities available with the SPECIAL file interface to the CCP are not available with the EXIT/RLABL interface. Therefore, SPECIAL is recommended if edited fields are to be used.

If editing must be performed when using SUBR90, either you must provide coding in your program to insert and remove editing characters or edit characters must be shown on preprinted fields at the terminal. For 3270 terminals, edit characters may be placed on the display as part of a display format.

If the 3270 Display Format Facility is used, negative integer numeric fields (no decimal positions) are automatically converted so that the minus sign is printed to the right of the value in the field. Likewise, on input operation, the field is automatically converted by 3270 DFF to its internal negative form (D zone in the right-most position of the field -- see *negative numbers* in the *RPG II Reference Manual*).

## TASK CHAINING (5704-SC2)

### Exit to SUBR87 and SUBR88

SUBR87, an entry in SUBR88, is used to issue Chain Task Request operations from RPG II programs. SUBR88 is used to accept both program-request and chain-task data by RPG II programs. These subroutines are smaller than SUBR91 and should be used in programs that perform task chaining operations only to reduce main storage requirements.

Unlike SUBR91, SUBR87 and SUBR88 do not require a parameter list; instead, you specify two RLABLEs. The first RLABLE specifies an output data length (for SUBR87) or an input data length (for SUBR88). The second RLABLE specifies the record area that consists of a name field and a data area. The name field is six characters long. When using SUBR87, you must place the name of the program to be requested, left-justified and padded with blanks if not six characters long, in the name field. When using SUBR88 to accept task chaining data, SUBR88 places the name of the program that requested the task chain into the name field before returning control to the chained routine. The data field is used to pass data between programs.

SUBR87 and SUBR88 set on indicator 91 if the accept operation resulted in a negative return code, and indicator 92 if the operation resulted in a positive return code other than a 14 return code. The 14 return code indicates a successful accept operation of the chain request data.

Figure 6-15 shows an example program (PGM1) that uses SUBR87 to issue a chain request and an example program (PGM2) that uses SUBR88 to accept the task chain data. SUBR93 is used in both examples to bypass the record selection in the RPG II program cycle. Input and output occurs in calculations using SUBR87 and SUBR88.

If the Overlay Linkage Editor is entered directly from the RPG II Compiler, and the program being compiled contains exits to SUBR87 but does not contain at least one exit to SUBR88, a halt (E0 'P 25) will occur. This halt occurs because the linkage editor cannot resolve internal entry points of subroutines (see the *IBM System/3 Overlay Linkage Editor Reference Manual*, GC21-7561 for additional information on the linkage editor). One method to avoid this halt is to code a dummy exit to SUBR88 as shown in the following example:

```
EXIT  SUBR87
RLABL LEN
RLABL DAREA
GOTO  BYPASS
EXIT  SUBR88
BYPASS TAG
```

### File Description Specification

Line	Form Type	File Type											Mode of Processing											Device	Symbolic Device	Extent Exit			File Addition/Unordered			
		File Designation											Length of Key Field or of Record Address Field													Name of Label Exit			Number of Tracks for Cylinder Overflow			
		End of File											Record Address Type													Storage Index			Number of Extents			
		Sequence											Type of File Organization or Additional Area													Continuation Lines			Tape Rewind			
File Format											Overflow Indicator											Option			File Condition U1-U8							
		I/O/U/C/D	P/S/C/R/T/D	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/D/T or 2	Key Field Starting Location	Extension Code E..L			K	Option	Entry	A/U	R/U/N													
02	F	DUMMY	IP	F	I	I							SPECIAL			93																

Line	Form Type	Indicators			Factor 1	Operation	Factor 2	Result Field		Resulting Indicators	Comments
		Control Level (L, O, L, B, L, R, SR, AN, OR)	And	And				Name	Length		
		Not	Not	Not							
04	C				MOVE '0005'	OFIELD	4		1 > 2   1 < 2   1 = 2	Set up chain request.	
05	C				MOVE 'PGM265'	TNAM0	11		LookupFactor 2)is		
06	C				MOVE 'DATA1'	TNAM0			High Low Equal		
07	C				EXIT SUBR87						
08	C				RLABL	OFIELD					Execute chain request.
09	C				RLABL	TNAM0					
10	C	92			SETOM					LR) If unsuccessful, exit from program.	
11	C										
12	C										

Figure 6-15 (Part 1 of 2). Coding for SUBR87 and SUBR88





Another method to avoid this halt is to compile the program and place the non-link-edited program in a library (see *object output* in the *IBM System/3 RPG II Reference Manual*, SC21-7504, for additional information). Then call the linkage editor (\$OLINK). The OCL for this procedure is shown in the following example:

```
// CALL      RPG,R1
// COMPILE  SOURCE-rpgchn,OBJECT-nn,UNIT-nn
// RUN
// CALL      OLINK,R1
// RUN
// INCLUDE  NAME-rpgchn,UNIT-nn
// INCLUDE  NAME-SUBR88,UNIT-nn
// INCLUDE  NAME-SUBR93,UNIT-nn
// END
```

*Notes:*

1. Rpgchn is the name of the program containing the task chain exit.
2. The FILE statements are handled by default in the example.
3. As with SUBR91, SUBR93 must be included as a dummy input file when using SUBR88.

### 3270 DISPLAY FORMAT FACILITY

You can use the 3270 Display Format Facility (DFF) of the CCP to aid you in formatting and using the 3270 terminal. *Chapter 8. 3270 Display Format Facility* describes the programming requirements that are unique to using the DFF, including the unique DFF operations, additional information that must be placed in the record area for certain operations, field types that are unique to the 3270, and other information.

You can compare the programming necessary to use the 3270 without DFF to programming with the DFF by comparing *Example 1* under *Programming Examples*, later in this chapter, to the example in Chapter 8.

### RPG II PROGRAMMING CONSIDERATIONS

This section contains some reminders, suggestions, and restrictions pertaining to writing RPG II programs to run under the CCP.

### *Data Mode Escape and Release Terminal Operation*

If data mode escape is allowed in your CCP system (specified by the ESCAPE keyword in the \$EFAC generation statement—see *CCP System Reference Manual*) you must include coding in your program to test for the 08 return code (terminal has entered data mode escape and released itself from your program). When the CCP returns a 08 return code after an input operation (Accept Input, Get, or Stop Invite Input), a count of outstanding Invite Input operations is returned in the third element of the parameter array (effective input length). In an MRT program, you should check this count to determine how many terminals are still attached to your program. If you do not check this count, you might do an Accept Input operation without having an outstanding Invite Input, causing the CCP to cancel your program (unless it is a *never-ending program*—see index entry).

**RPG CALCULATION SPECIFICATIONS**

Form GX21-9093  
Printed in U.S.A.

IBM International Business Machine Corporation

Program \_\_\_\_\_ Card Elect. No. \_\_\_\_\_  
 Programmer \_\_\_\_\_ Date \_\_\_\_\_ PUNCHING Instruction \_\_\_\_\_ Graphic \_\_\_\_\_ Punch \_\_\_\_\_

Page 1 of 2 Program Identification 75 76 77 78 79 80

C Line	Form Type Control Level (LQ, LB, LR, SR, AN/OR)	Indicators						Factor 1	Operation	Factor 2	Result Field		Resulting Indicators			Comments
		And		And		Name	Length				Arithmetic	Compare				
		Not	Not	Plus	Minus						Zero	1 > 2	1 < 2	1 = 2		
01	C															
02	C															
03	C															
04	C								MOVE 'D A'	ARY, 2					STOP INV INPUT	
05	C								MOVE 'LØ'	ARY, 4					MAX INPUT LEN	
06	C								MOVELTNAME	INREC	.16					
07	C								EXIT SUBR9L							
08	C								RLABL	ARY						
09	C								RLABL	INREC						
10	C								MOVE ARY, L	RTNCOD	2Ø					
11	C							RTNCOD	COMP Ø			1Ø2Ø				
12	C			1Ø												
13	C			OR	2Ø				EXSR RETCHK							
14	C								⚡							
15	C															
16	C															
17	C			SR				RETCHK	BEGSR							
18	C			⚡												
19	C														(subroutine checks for specific return codes and takes appropriate action)	
20	C															

**Figure 6-16. Checking the Return Code After an EXIT/RLABL Operation**

Following a Release Terminal operation in an MRT program, the CCP also returns a count of outstanding Invite Input operations in the third array element. You should check this count to avoid doing an Accept Input operation without an outstanding Invite Input. (The Release Terminal operation always returns a 00 return code.)

**Using Both SPECIAL and EXIT/RLABL**

You should be aware that when you use both SPECIAL and EXIT/RLABL for CCP operations you are adding additional code to your program, since at least two sub-routines must be included. Therefore, if you are concerned about the size of your program, you can reduce its size by using only one of these methods of performing CCP operations.

You must be sure to specify the correct communications service subroutines for SPECIAL and EXIT/RLABL. For example, if you mistakenly use SUBR91 with SPECIAL, the results are unpredictable.

**Multiple Output Lines**

When sending multiple H, D, or T output lines in a single program cycle or when sending multiple output lines with a single EXCPT operation, you should condition the output lines with indicator N91 so that you can react to output errors when they occur. However, even if you use this technique, it is difficult to determine which output lines were or were not successfully sent in this case.

## End of File with SUBR92

Remember, when using SPECIAL files with SUBR92 you will never get end of file on a SPECIAL file unless you issue the Force End of File operation to the file (see *CCP operation codes*, earlier in this chapter). Therefore, the end of file indicator in positions 58-59 of the Calculation specification for the READ operation is meaningless unless Force End of File is used.

## Host/Subhost Communications

Considerations for communicating with host and subhost systems via BSCA are given in *Appendix A*.

## Communicating With the Console

You can communicate with the system operator console only by means of Put and Put-Then-Get operations. The symbolic terminal name for the console is CONSOL.

You cannot issue an Acquire Terminal operation to the console (there is no need, since any program can communicate with the console at any time).

You cannot communicate with the system operator console by specifying CONSOLE (Model 10 Disk System and Model 12) or CRT77 (Model 15) as the device in columns 40-46 of File Description Specifications.

## Master Index

CCP builds an in-storage index based on either the MSTRINDX or MIXSIZE keyword of the DISKFILE assignment statement (see *CCP System Reference Manual*). Specify a master index for disk files in the application program as follows:

- On a Model 10 or 12, specify at least one master index entry (key length plus 2) in the application program to make use of a master track index. This improves performance for large index files.
- On a Model 15, a master index entry is not required in the application program to make use of the master track index.

## Disk File Usage

You should be aware of how other programs running under the CCP are using disk files that you also use in your program, especially if records are being added to the files. See index entry *disk file considerations* for detailed information.

## Specific Restrictions

- Do not use magnetic tape files.
- Do not use look-ahead fields.
- Do not describe a SPECIAL file for CCP operations as a table file.
- Do not use the RPG II inquiry feature.
- (Model 10 and Model 12 CCP) Do not use RPG II halt indicators. If programs running under the CCP issue halts, they will be cancelled.
- (Models 10, 12, and 15 A, B, and C CCP) Do not use multivolume files. Model 15D must use random access files (place an R in column 28).

## PROGRAMMING EXAMPLES

The following programming examples are explained in this section.

*Example 1*—an RPG II program that supports a single requesting 3270 without using the Display Format Facility.

*Example 2*—an RPG II program that supports multiple requesting terminals.

See Chapter 8 for examples of RPG II programs that use the 3270 Display Format Facility.

Before attempting to use these examples, you should read and understand the description of the RPG II CCP interface in this chapter.

### Example 1

Figures 6-17, 6-18, and 6-19 show the flowcharts, messages, and listing for a single requesting terminal (SRT) RPG II program (see index entry *SRT program*). This program transmits two messages to a 3270 Model 1 Display System (480 character screen). The first message from the program requests the terminal operator to enter a room number. The program uses the room number entered by the terminal operator as the relative record number to access a disk file whose records contain guest and rate information about the room. This information is then formatted and displayed as the second message transmitted to the 3270 terminal. The program then goes to end of job. Figure 6-17 also shows how these messages appear on the 3270 terminal.

Because this program is a *single requesting terminal* (SRT) program (see index entry), it can receive data from and transmit data to only one 3270 terminal. However, multiple copies of this program could be in main storage at the same time (if the system has sufficient main storage), each communicating with a different 3270 Display System. (If multiple copies are in core at the same time, the disk file must be specified as sharable during the Assignment stage—see index entry *disk file sharing*.)

### *Formatting the Messages for the 3270 Display*

Because this sample program does not use the Display Format Facility, this sample program must provide formatting control characters for the display screen in the data portion of the record area and transmit them as part of the messages to be displayed. Figure 6-19 shows the messages and the 3270 control commands and orders as they are transmitted to the 3270. See *IBM 3270 Information Display System Component Description, GA27-2749*, for a description of 3270 system components, concepts, control commands, and orders.

The printable format characters are set by defining them as part of the message in the RPG II Output Specifications definition.

The unprintable format characters (hexadecimal values that have no corresponding printable character in 96-column card code) are in a pre-execution time array (ORDERS) that was loaded by a previous RPG II program (see comments in File Description Specifications). Elements of the ORDERS array are then used in appropriate positions in the output messages.

### *Notes Concerning this Sample Program*

- Message Mode was defined during CCP assignment for the 3270 terminal used by this program. (See TERMATTR statement in *CCP System Reference Manual*.) This eliminates the need to do repetitive input operations until EOT is received.
- To run this program using a terminal other than the 3270, you must remove all coding dependent on the 3270. This includes all screen formatting specifications and 3270 screen control characters within the data.
- This program will not accept data with the program request.

- This program specifies a Put with Invite Input operation using a blank terminal name as the first operation (1P output time in the RPG II program cycle). In an SRT program such as this, the blank terminal name refers to the current requester of the program. The name of the requester is received in the TNAME field of the input record area after the Accept Input operation.
- To keep this sample program simple, return code checking is kept to a minimum. You should do more return code checking in your application programs. For example, when you issue Accept Input you should check for the Shutdown Requested return code (04). Also, if data mode escape is allowed in the CCP system, programs should check for return code 08. It is recommended that each installation design its own return code checking and/or console communication routines so that a satisfactory standard is established for the installation that can be used in all application programs.
- This program does not check the length of the input data because the terminal operator is requested to enter a three-digit room number. However, you may want to check the input data length in your application programs.
- Since there are only two different screen formats used by this program, they are both contained within the program. For more complete applications, you might store the screen formats on disk and read them when they are needed by your program.
- You could also use the Get Attributes operation in this program. For example, if you do not know whether the 3270 Model 1 or the 3270 Model 2 will request the program, you can issue a Get Attributes operation to find out which type of terminal requested the program.
- If this program were coded and specified as a multiple requesting terminal (MRT) program with a MRTMAX=1 keyword on the PROGRAM assignment statement (see *CCP System Reference Manual*), multiple copies of the program would not be allowed in main storage at the same time. As the program is written, multiple copies could be in main storage at the same time and the disk file must be specified as sharable (FILES keyword of PROGRAM assignment statement).

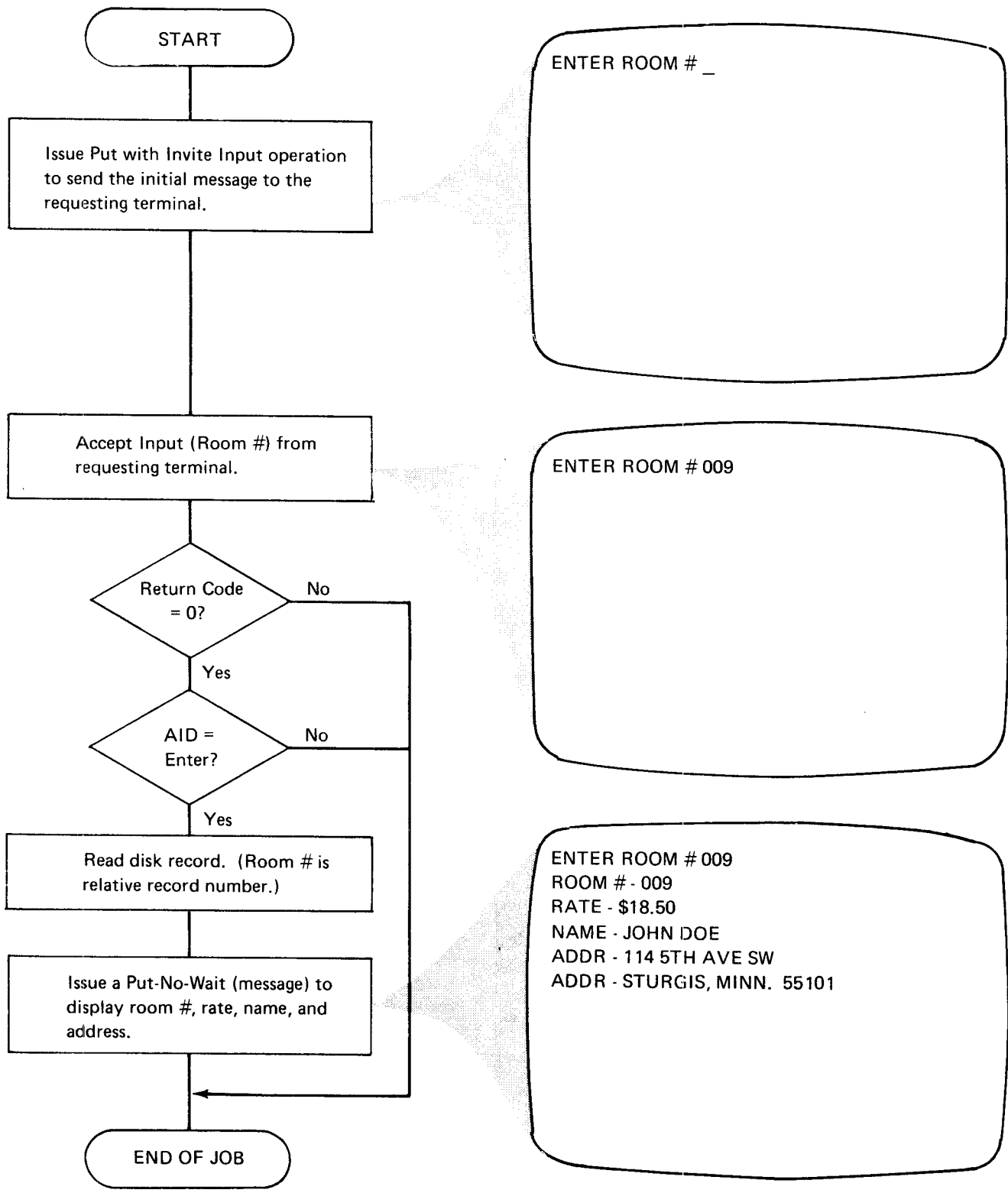
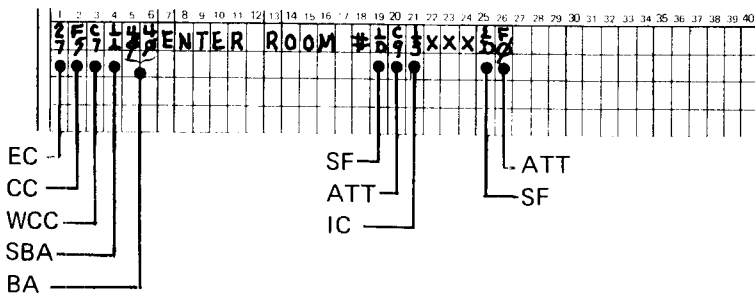
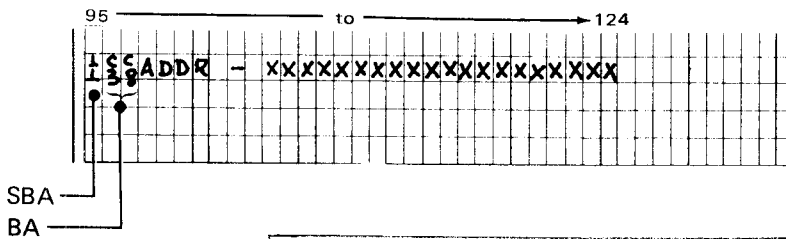
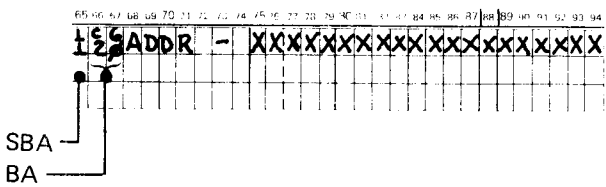
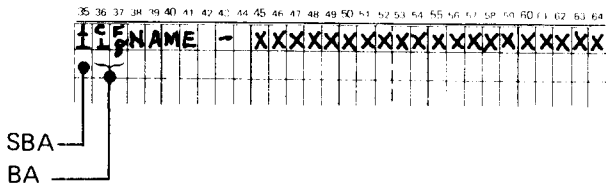
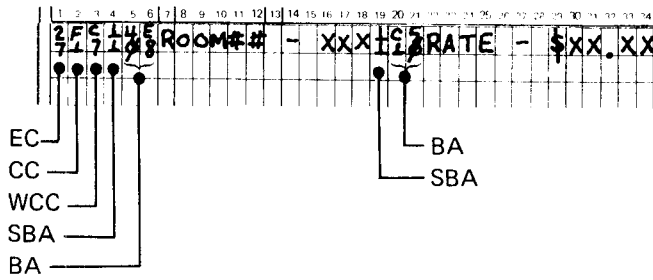


Figure 6-17. Program Logic of Example 1 (RPG II SRT Program)

First Message



Second Message



SF	- Start Field	CC	- Command Code
ATT	- Attribute Character	WCC	- Write Control Character
IC	- Insert Cursor	SBA	- Set Buffer Address
x	- Data Character	BA	- Buffer Address of first character position in the field
EC	- Escape Character		

Figure 6-18. Message Formats for Example 1 (RPG II SRT Program)

```

01010H R 4
0001 0102 FTPIN IP 25 SPECIAL SUBR92
0002 0103 F KIARR
0003 0104 FTPOUT 0 138 138 SPECIAL SUBR92
0004 0104 F KIARR
0005 0106 FGUEST IC 70 70R DISK
0006 0108 FORDERS IT 4 4 EDISK
0109 F* THIS FILE WAS LOADED BY A PREVIOUS RPG PROG SINCE IT CONTAINS HEX-
0110 F*ADECIMAL DATA THAT MUST BE AVAILABLE AT 1P OUTPUT. THAT RPG
0111 F*PROGRAM COULD HAVE CREATED THESE CHARS WITH THE SET BIT
0112 F*INSTRUCTION OR FILE TRANSLATION.
0007 0201 E IARR 0050005 06 INOUT CCP ARRAY
0008 0203 E ORDERS ORDR 4 4 01 3270 ORDERS
0204 E** THE FOLLOWING ORDERS (1 CHARACTER EACH) ARE CONTAINED IN THE
0205 E** ORDR ARRAY ORDR,1 -ESCAPE CHAR
0206 E** ORDR,2 -SET BUFFER ADDRESS
0207 E** ORDR,3 -START FIELD
0208 E** ORDR,4 -INSERT CURSOR
0009 0301 ITPIN AA 01 17 C'
0010 03010I 5 80EFL
0011 03020I 9 14 TNAME
0012 03021I 23 25ORDOM
0013 03030I AB 02
03040I** CATCH ALL FOR UNIDENTIFIED RECORD
0014 0308 IQUEST BA 03 1 CG
0015 0309 I 2 4 GROOM
0016 0310 I 5 82GRATE
0017 0311 I 9 28 GNAME
0018 0312 I 29 48 GADDR1
0019 0313 I 49 68 GADDR2
0020 0314 I BB 05
0315 I** CATCH ALL
S0305 I*NOTE: POSITION 17 IS AN AID CHAR. A SINGLE QUOTE MEANS THE
0306 I*ENTER KEY WAS KEYED. ANY OTHER AID CHARACTER OR NON-ZERO
0307 I*RETURN CODE ON INPUT CAUSES THIS PROGRAM TO GO TO END OF JOB.
0021 04010C 91
0022 04011COR 92
0023 04012COR 02 SETON LR BAD AID,RTN COD
0024 0402 C LR GOTO END LR ROOM # ENTERED
0025 0403 C 01RDOM COMP 1
0026 0405 C LR GOTO END
0027 0406 C 01RDOM COMP 10 LR
0028 0408 C LR GOTO END
0029 0409 C 01RDOM CHAINGUEST
0030 04100C 05 SETON LR
0031 0411 C 03 SETON LR04
0032 04111C 03 SETOF 9291
0033 0412 C END TAG
050100*FORMAT THE SCREEN FOR DATA ENTRY.
050110TPOUT H 1P
0034 0502 0 14 ' '
0035 0503 0 4 'CS'
0036 050310 8 '40'
0037 0504 0 ORDR,1 15
0038 0505 0* HEX'27' - ESCAPE CHAR
0039 0506 0 16 '5'
0507 0* HEX'F5' - ERASE/WRITE
J040 0508 0 17 'G'
0509 0* HEX'C7' - WRITE CONTROL CHAR (SOUND ALARM,RESTORE KEYBOARD,
0510 0* RESET MODIFIED DATA TAGS)
0041 0511 0 CRDR,2 18
0512 0* HEX'11' - SET BUFFER ADDRESS COMMAND
0042 0513 0 20 ' '
0514 0* HEX'4040' ROW1 COL1 3270 MOD 1
0043 0515 0 32 'ENTER ROOM #'
0044 0601 0 ORDR,3 33
0602 0*HEX'1D'--START OF FIELD
0045 0603 0 34 'I'
0604 0* HEX'C9'--UNPROTECTED ALPHAMERIC INTENSIFIED
0046 0605 0 ORDR,4 35
0606 0* HEX'13'--INSERT CURSOR
0047 0607 0 38 ' '
0608 0* AREA FOR ROOM NUMBER
0048 0609 0 ORDR,3 39
0049 0610 0 40 '0'
0611 0* PROTECT REMAINDER OF SCREEN
0050 0701 0 T LR 04
0051 0702 0 4 'CF'
0052 070210 8 '138'
0053 0703 0 TNAME 14
0054 0706 0 ORDR,1 15
0055 0707 0 16 '1'
070710*WRITE ONLY
0056 0708 0 17 'G'
0057 0709 0 ORDR,2 18
0058 0710 0 20 ' Y'

```

In this example, the same array is used for the input and output SPECIAL files since it is convenient and saves space in the program.

The parameter array is a compile time array in this example, so the array contents are set when 1P output is performed (see the first message).

At input time in the program cycle an Accept Input is issued. The Invite Input has been issued at 1P output time.

The program expects a room number between 1 and 10. If a different number is entered, the program does not attempt to access the GUEST file for rate, name, and address. Only the room number is printed on the T output line. If the operator enters an incorrect room number such as 100, the room number is displayed as 100.

A Put with Invite Input operation with a blank terminal name issued at 1P output time sends the initial message to the terminal and invites input from the terminal. The maximum input length (25) for the Invite Input is in the parameter array.

Figure 6-19 (Part 1 of 2). Example 1 – RPG II SRT Program



```

0711 0* 3270 M1 ADDRESS ROW2 COL1
0059 0712 0                                29 'ROOM # - '
0060 0712 0                                GROOM 32
0061 0714 0                                ORDER,2 33
0062 0715 0                                35 'A&'
0716 0* 3270 M1 ADDRESS ROW3 COL1
0063 0801 0                                42 'RATE - '
0064 0802 0                                GRATE 3 48 '$'
0065 0803 0                                ORDER,2 49
0066 0804 0                                51 'A8'
0805 0* 3270 M1 ADDRESS ROW4 CCL1
0067 0806 0                                58 'NAME - '
0068 0807 0                                GNAME 78
0069 0808 0                                CRDR,2 79
0070 0809 0                                81 'B-'
0810 0* 3270 M1 ADDRESS ROW5 COL1
0071 0811 0                                88 'ACCR - '
0072 0812 0                                GADDR1 108
0073 0813 0                                ORDER,2 109
0074 0814 0                                111 'CH'
0815 0* 3270 M1 ADDRESS ROW6 COL1
0075 0901 0                                118 'ADDR - '
0076 0902 0                                GADDR2 138
0903 0** END OF PRDG **

```

```

**
D          25

```

INDICATORS USED

LR 1P 01 02 03 04 05 91 92

RG 314 UNREFERENCED TABLE/ARRAY NAMES

STMT# NAME  
0007 IARR

EXECUTION TIME TABLES AND ARRAYS

STMT#	TABLE/	DEC	ENTRY	NUMBER OF	DTT	T/A
DEFINED	ARRAY	POS	LENGTH	ENTRIES	DISP	DISP
0008	ORDR		001	00004	0100	0000

RG 314 UNREFERENCED FIELD NAMES

STMT# NAME  
0010 EFL

FIELD NAMES USED

STMT#	NAME	DEC	LGTH	DISP
0011	TNAME		006	0009
0012	ROOM	0	003	0048
0015	GROOM		003	000C
0016	GRATE	2	004	004F
0017	GNAME		020	0020
0018	GADDR1		020	0034
0019	GADDR2		020	0048

LABELS USED

STMT# NAME TYPE  
0033 END TAG

COMPILE TIME TABLES AND ARRAYS

STMT#	TABLE/	DEC	ENTRY	NUMBER OF	DTT	T/A
DEFINED	ARRAY	POS	LENGTH	ENTRIES	DISP	DISP
0007	IARR		006	0005	0000	0000

IARR

D 25  
END OF TABLE/ARRAY - LAST ENTRY WAS BLANK

001110

ERROR NUMBER STATEMENT NUMBER

RG 097 0014  
RG 097 0020  
RG 558 0029

ERROR SEVERITY

TEXT

RG 097 W NO FIELDS DESCRIBED FOR THIS OR PREVIOUS RECORD.  
RG 314 W FIELD, TABLE OR ARRAY NAME DEFINED BUT NEVER USED.  
RG 392 W LAST ENTRY IN ONE OR MORE COMPILE TIME TABLE/ARRAYS WAS BLANK.  
RG 558 W INVALID USE OF, OR MISSING, RESULTING INDICATORS WITH THIS OP CODE. ASSUME  
INVALID RESULTING INDICATORS BLANK.

Figure 6-19 (Part 2 of 2). Example 1 — RPG II SRT Program

## Example 2

Figures 6-20, 6-21 and 6-22 show the flowchart, input/output messages, and listing for a sample RPG II multiple requesting terminal (MRT) program designed to run under the CCP (see index entry *MRT program*). This program handles up to four MLTA requesting terminals. The terminal operator enters a seven-digit number preceded by a +, -, or N. The CCP transmits this signed number to the RPG II program. The RPG II program:

- Adds the number to the value in the accumulator associated with the terminal that transmitted the data if the first position is +.
- Subtracts the number from the accumulator if the first position is -.
- Releases the terminal if the first position is N.

If a value was either added or subtracted, the new value accumulated for the terminal is inserted into the message *CURRENT VAL = sXXXXXXXXX ENTER DATA* and the message is sent to the terminal.

This sample program also checks for several error conditions and transmits the appropriate error message to the terminal requesting the operation.

This sample program is not designed to show the most effective way of performing operations. Instead, it shows a variety of ways to do things. It uses a variety of operation codes that show how data can be associated with a terminal by defining a save area for the terminal names and accumulated data. It frequently checks return codes; but you can do even more return code checking if you wish. Data entered by the terminal operator must be fixed length. To allow variable length input fields, you could include a subroutine in your program to check the effective input length returned in the parameter list and align the data correctly. This program communicates with the console in addition to the requesting terminals.

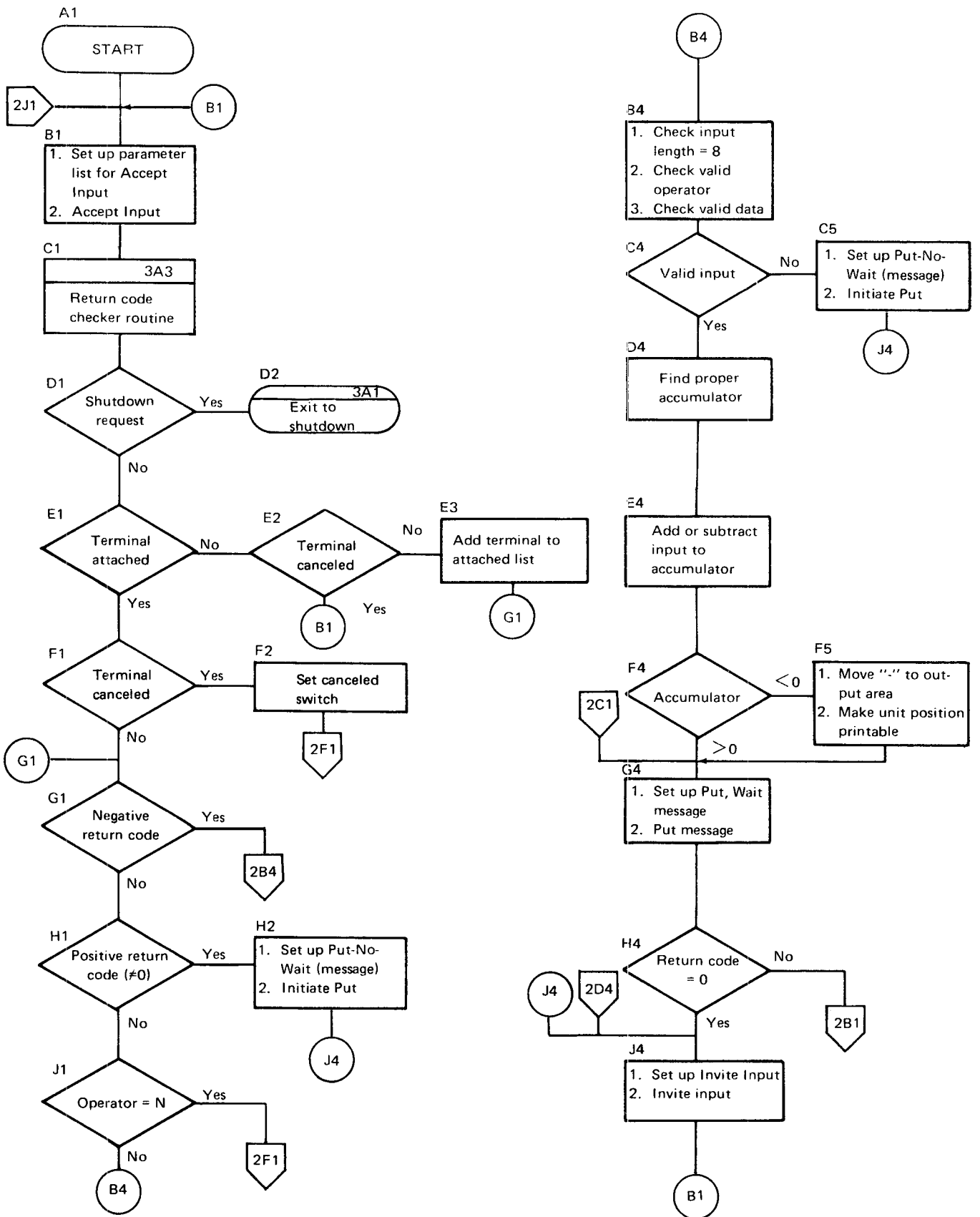


Figure 6-20 (Part 1 of 3). Program Logic of Example 2 (RPG II MRT Program)

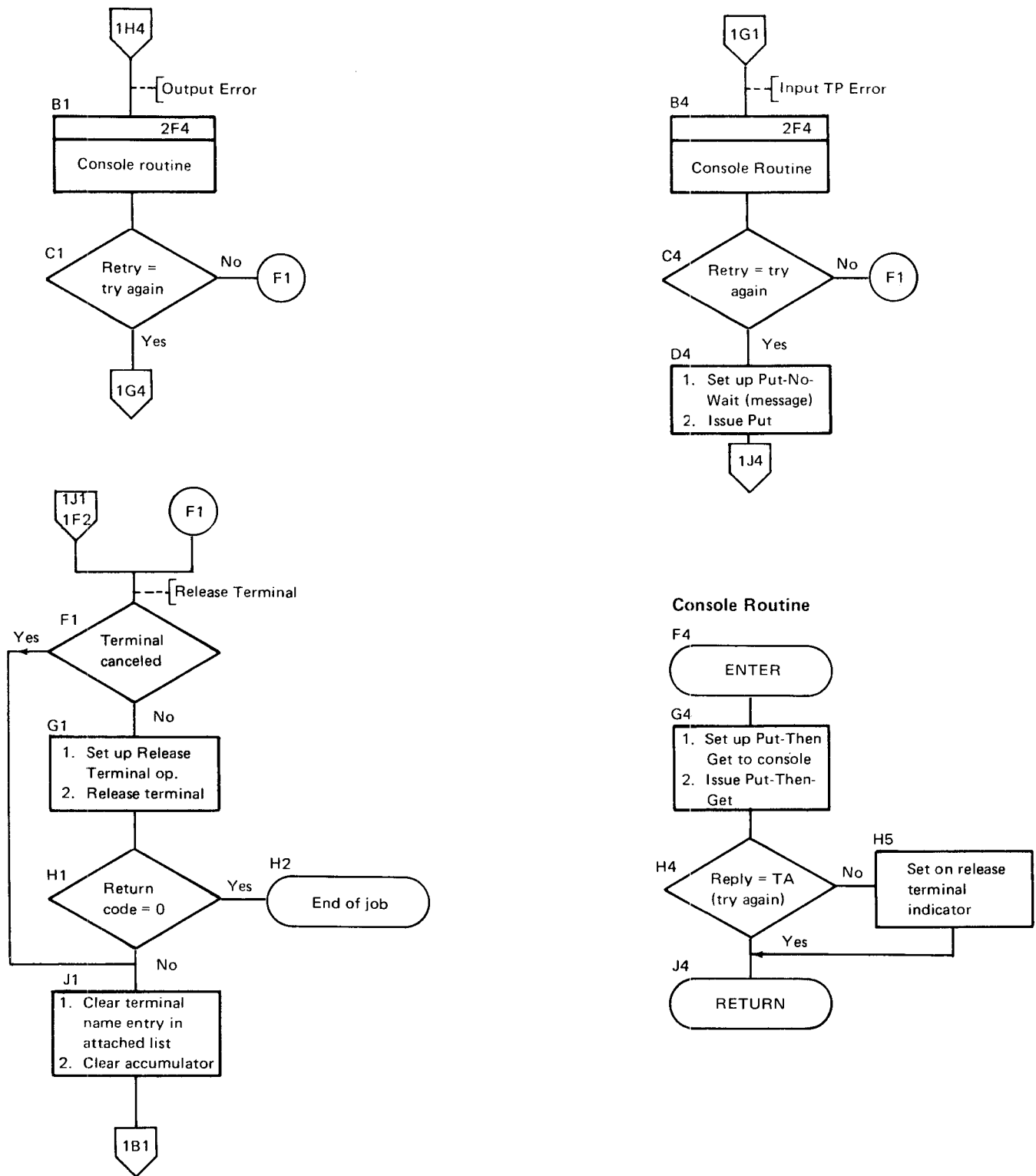
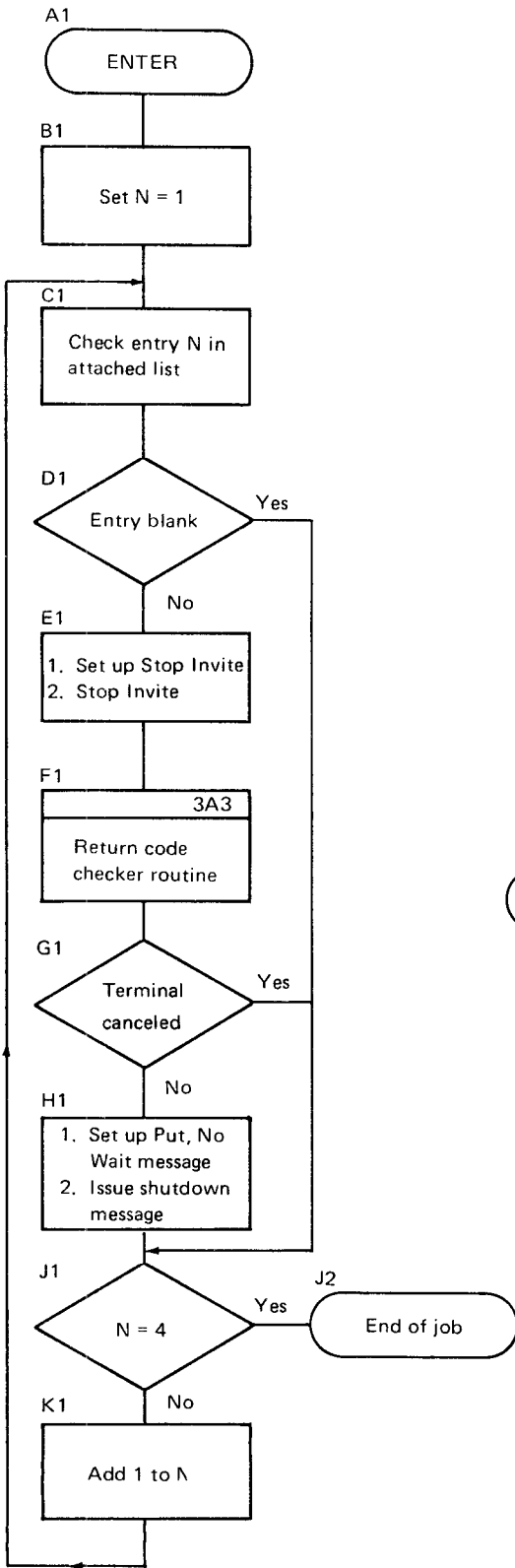


Figure 6-20 (Part 2 of 3). Program Logic of Example 2 (RPG II MRT Program)

**Shutdown Routine**



**Return Code Checker Routine**

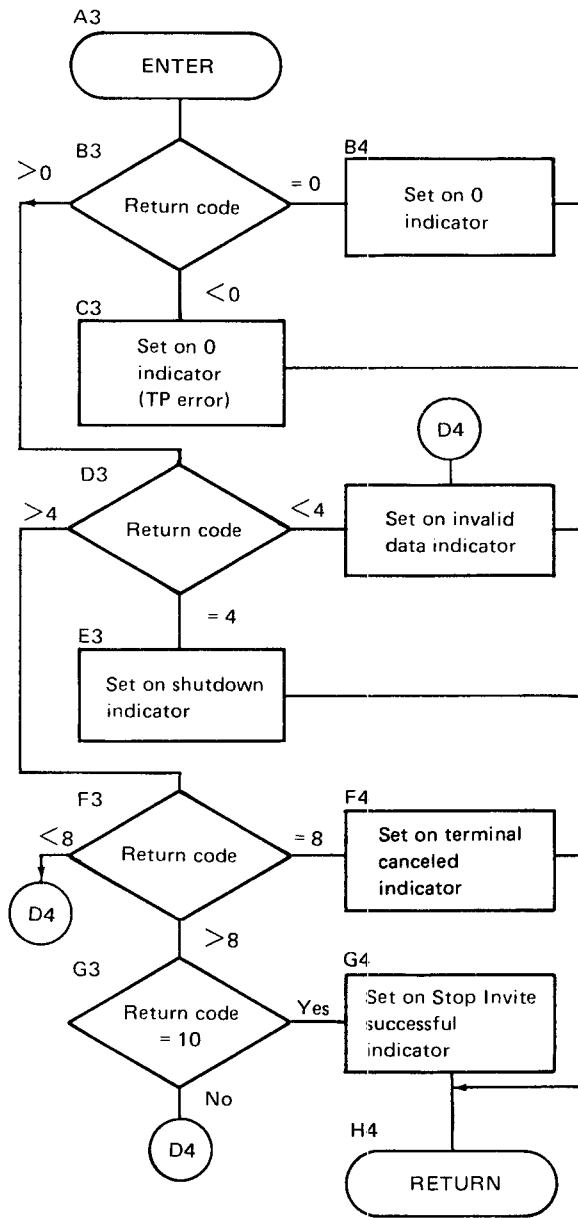


Figure 6-20 (Part 3 of 3). Program Logic of Example 2 (RPG II MRT Program)

**Input Data Entered by Terminal Operator**

1	2	3	4	5	6	7	8	9	10	11	12
S	X	X	X	X	X	X					

A fixed length numeric field where S is at a +, -, or N and X is a numeric digit. All eight positions must be entered, except when N is entered in the first position.

**Data Entered by System Operator on 5471 Printer/Keyboard (Models 10 and 12) or CRT/Keyboard (Model 15)**

1	2	3	4	5	6	7	8	9	10	11	12
T	A										
C	C										

In response to the messages INPUT TP ERROR TNAME-cccccc and OUTPUT TP ERROR TNAME-cccccc to the console, the system operator replies TA if he wants to Try Again. Any other reply (cc) causes the terminal to be released.

**Output to the Console**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30		
I	N	P	U	T		T	P		E	R	R	O	R		T	N	A	M	E	-	C	C	C	C	C	C					
O	U	T	P	U	T		T	P		E	R	R	O	R		T	N	A	M	E	-	C	C	C	C	C					

These messages are transmitted to the console (cccccc = terminal name).

**Output to Terminal**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39			
C	U	R	R	E	N	T		V	A	L	=	S	X	X	X	X	X	X	X	X		E	N	T	E	R		D	A	T	A										
T	R	Y		A	G	A	I	N		I	N	V		D	A	T	A																								
T	R	Y		A	G	A	I	N		T	P		E	R	R	O	R																								
C	C	P		S	H	T	D	W	N		L	A	S	T		R	E	C	-		T	P		E	R	R	O	R													
C	C	P		S	H	T	D	W	N		L	A	S	T		R	E	C	-		B	A	D		D	A	T	A													
C	C	P		S	H	T	D	W	N		L	A	S	T		R	E	C	-		S	X	X	X	X	X	X														
C	C	P		S	H	T	D	W	N		L	A	S	T		R	E	C	-		N	O		D	A	T	A														

- Transmitted with value in accumulator associated with the terminal.
- Issued if data is invalid
- Issued if system operator replies TA (Negative return code on Accept Input)
- Issued for negative return code on Stop Invite Input
- Issued for positive return code other than 10 on Stop Invite Input
- Issued for return code of 0 on Stop Invite Input
- Issued for return code of 10 on Stop Invite Input

Figure 6-21. Input and Output Message Formats for Example 2 (RPG II MRT Program)

```

0101OH R 4
0001 02010FDUMMY IP 1 1 SPECIAL SUBR93 MRRPG1
0002 02020FTPCONSIOD 52 52 SPECIAL SUBR92 MRRPG1
0003 02030F KTPRY MRRPG1
0004 03010E TPRY 5 6 CCP PARM LIST MRRPG1
0005 03020E ATRY 4 6 ATTACHD TERM LIST MRRPG1
0006 03030E ACMY 4 10 0 ACCUMULATOR LIST MRRPG1
0007 04010IDUMMY AA 01 MRRPG1
04020I* DUMMY SPEC FOR PRIMARY FILE (RPG REQUIRES A PRIMARY FILE) MRRPG1
0008 04030ITPCONSI0BB 02 MRRPG1
0009 04040I 1 20RETCDD MRRPG1
0010 04050I 5 80EFFL MRRPG1
0011 04060I 9 14 TNAME MRRPG1
0012 04070I 15 15 OPRATR MRRPG1
0013 04080I 16 220TPIPT MRRPG1
0014 04090I 15 17 CONSIPT MRRPG1
05010C*** CHECK THE 3RD ELEMENT OF THE PARAMETER ARRY TO DETERMINE IF MRRPG1
05020C** IF ANY MORE TERMINALS TO BE SERVICED. IF NOT GO TO EOJ. MRRPG1
0015 05030C ACPIPT TAG MRRPG1
0016 05040C 09 MRRPG1
0017 05050CDR 13TPRY,3 COMP ' 0000' LR NO MOR IF 0 MRRPG1
0018 05060C LR GOTO END GO TO EJ MRRPG1
0019 05070C 01N SUB N N 10 CLEAR INDEX MRRPG1
0020 05080C 01 MOVE ' D' TPRY,2 ACP IPT OP CODMRRPG1
0021 05090C 01 MOVE '22' TPRY,4 MAX XPECT LEN MRRPG1
05100C*****MRRPG1
05110C* MRRPG1
05120C* ACCEPT INPUT MRRPG1
05130C* MRRPG1
05140C*****MRRPG1
0022 05150C 01 READ TPCONSIO ACCEPT INPUT MRRPG1
05160C*****MRRPG1
05170C* MRRPG1
05180C* GO TO RETURN CODE CHECKER ROUTINE MRRPG1
05190C* MRRPG1
05200C*****MRRPG1
0023 06010C 02 EXSR RETCHK CHECK RTN CODE MRRPG1
06020C*****MRRPG1
06030C* MRRPG1
06040C* IF SHUTDOWN REQUEST RETCOD = 04,IND ON IS 06 MRRPG1
06050C* MRRPG1
06060C*****MRRPG1
0024 06070C 06 GDTO SHTDWN SHUTDOWN CODE MRRPG1
06080C*****MRRPG1
06090C* MRRPG1
06100C* ROUTINE TO DETERMINE IF TNAME IS AN ATTACHED TERMINAL MRRPG1
06110C* IF ATTACHED IND 12 IS ON,IF NOT IND 13 IS ON MRRPG1
06120C* MRRPG1
06130C*****MRRPG1
0025 06140C LUP1 TAG MRRPG1
0026 06150C 01 N ADD 1 N ADD TO INDEX MRRPG1
0027 06160C 01 ATRY,N COMP TNAME 12NAME FOUND MRRPG1
0028 06170C 01 12 GOTO G01 MRRPG1
0029 06180C 01 N COMP 4 13NAME NOT IN LST MRRPG1
0030 06190C 01N13 GOTO LUP1 KEEP LOOKING MRRPG1
06200C*****MRRPG1
07010C* MRRPG1
07020C* TERMINAL NOT ATTACHED AND CANCELED MRRPG1
07030C* MRRPG1
07040C*****MRRPG1
0031 07050C 09 13 GOTO ACPIPT TERM CANCELLED MRRPG1
07060C*****MRRPG1
07070C* MRRPG1
07080C* IF TERMINAL NOT ATTACHED AND NOT CANCELED MRRPG1
07090C* ADD TERMINAL TO ATTACHED LIST MRRPG1
07100C* MRRPG1
07110C*****MRRPG1

```

Figure 6-22 (Part 1 of 6). Example 2 – RPG II MRT Program

```

0032 07120C 13 N SUB N N CLEAR INDEX MRRPG1
0033 07130C LUP2 TAG MRRPG1
0034 07140C 13 N ADD 1 N ADD TO INDEX MRRPG1
0035 07150C 13 ATRY,N COMP ' .4THIS NTRY AVAILMRRPG1
0036 07160C 13N14 GOTO LUP2 NO TRY AGAIN MRRPG1
0037 07170C 13 14 MOVE TNAME ATRY,N YES ENTER NAME MRRPG1
0038 07180C 13 GOTO GO2 MRRPG1
07190C*****MRRPG1
07200C* MRRPG1
08010C* IF TERMINAL WAS IN THE ATTACHED LIST AND RETURN CODE MRRPG1
08020C* IS CANCEL,GO TO RELEASE TERMINAL ROUTIN MRRPG1
08030C* MRRPG1
08040C*****MRRPG1
0039 08050C GO1 TAG MRRPG1
0040 08060C 12 09 GOTO RELEAS CLEAR TERM MRRPG1
08070C*****MRRPG1
08080C* MRRPG1
08090C* IF NEGATIVE RETURN CODE-IND ON IS 03,GO TO MRRPG1
08100C* INPUT TP ERROR ROUTINE MRRPG1
08110C* MRRPG1
08120C*****MRRPG1
0041 08130C GO2 TAG MRRPG1
0042 08140C 03 GOTO IPTER NEG RET CODE MRRPG1
08150C*****MRRPG1
08160C* MRRPG1
08170C* IF POSITIVE RETURN CODE(GREATER THAN ZERO) BUT MRRPG1
08180C* NOT EQUAL TO 10 GO TO INVALID DATA ROUTINE MRRPG1
08190C* MRRPG1
08200C*****MRRPG1
0043 09010C 07 GOTO INV DAT INV DATA ROUTINMRRPG1
09020C*****MRRPG1
09030C* MRRPG1
09040C* IF RETURN CODE=0,IND 04 IS ON AND IF MRRPG1
09050C* OPERATOR=N,GO TO RELAEASE TERMINAL ROUTINE MRRPG1
09060C* MRRPG1
09070C*****MRRPG1
0044 09080C 04 OPRATR COMP 'N' 13ANY MORE DATA MRRPG1
0045 09090C 04 13 GOTO RELEAS NO,RELEASE MRRPG1
09100C*****MRRPG1
09110C* 1 CHECK INPUT LENGTH = 8,IF NOT 8 TURN IND 07 ON MRRPG1
09120C* 2 CHECK VALID OPERATOR ,IF OPERATOR IS + TURN IND MRRPG1
09130C* 14 ON,IF OPERATOR IS - TURN ON IND 15,IF OPERATOR MRRPG1
09140C* 5 IS ANY OTHER VALUE TURN ON IND 07 MRRPG1
09150C* 3 CHECK VALID DATA IF NOT VALID TURN IND 07 ON MRRPG1
09160C*****MRRPG1
0046 09170C 04 EFFL COMP 8 0707 LEN XACTLY 8 MRRPG1
0047 09180C 07 GOTO INV DAT MRRPG1
0048 09190C 04 OPRATR COMP '+' 151514IS ADD WANTED MRRPG1
0049 09200C 04 15 OPRATR COMP '- ' 070715IS SUBTRACT MRRPG1
0050 10010C 04N07 TPIPT COMP 0 07 VALID NUMBER MRRPG1
10020C*****MRRPG1
10030C* MRRPG1
10040C* IF INPUT IS NOT VALID GO TO INVALID DATA ROUTINE MRRPG1
10050C* MRRPG1
10060C*****MRRPG1
0051 10070C 04 07 GOTO INV DAT MRRPG1
10080C*****MRRPG1
10090C* MRRPG1
10100C* 1 FIND PROPER ACCUMULATOR MRRPG1
10110C* 2 ADD OR SUBTRACT INPUT TO ACCUMULATOR MRRPG1
10120C* MRRPG1
10130C*****MRRPG1
0052 10140C 04 14 ACMY,N ADD TPIPT ACMY,N ADD TO ACMLATR MRRPG1
0053 10150C 04 15 ACMY,N SUB TPIPT ACMY,N SUB FROM ACMLTRMRRPG1
10160C*****MRRPG1
10170C* MRRPG1
10180C* DETERMINE IF THE ACCUMULATOR IS NEGATIVE-LESS THAN 0- MRRPG1
10190C* TURN ON IND 16 WHICH ALLOWS A MINUS SIGN TO MRRPG1
10200C* OVERRIDE THE PLUS SIGN ON OUTPUT SPECIFICATION MRRPG1
11010C* MRRPG1
11020C*****MRRPG1
0054 11030C 04ACMY,N COMP 0 16 RESULT NEG MRRPG1
11040C*****MRRPG1
11050C* MRRPG1

```

Figure 6-22 (Part 2 of 6). Example 2 – RPG II MRT Program



```

11060C*
11070C*****MRRPG1
0055 11080C          OPTPUT   TAG          MRRPG1
0056 11090C          18      SETOF          18      MRRPG1
0057 11100C          04      MOVE '52'      TPRY,3      OUT LEN      MRRPG1
0058 11110C          04      EXCPT          SEND CURNT VAL MRRPG1
11120C*****MRRPG1
11130C*
11140C*
11150C*****MRRPG1
0059 11160C          91      GOTO OPTERR          NO GO TO ER RTNMRRPG1
11170C*****MRRPG1
11180C*
11190C** IF SUCCESSFUL DO INVITE INPUT FOR THIS TERM AND THEN GO TO ACCEPTMRRPG1
11200C** INPUT FROM ANY ONE
12010C*
12020C*****MRRPG1
0060 12030C          INVIT     TAG          MRRPG1
0061 12040C          01      MOVE ' E'      TPRY,2      INVITE OP CODE MRRPG1
0062 12050C          01      MOVE '22'      TPRY,4      XPECTED MAX LENMRRPG1
0063 12060C          01      EXIT SUBR91          INVITE WITHOUT MRRPG1
0064 12070C          RLABL     TPRY          SPECIAL      MRRPG1
0065 12080C          RLABL     TNAME         MRRPG1
12090C*****MRRPG1
12100C*
12110C**ONLY NAME AND LENGTH NEEDED FOR INVITE.THIS COULD HAVE BEEN DONE MRRPG1
12120C**VIA EXCPT. IF SO SUBR91 WOULD NOT HAVE BEEN NEEDED.
12130C*
12140C*****MRRPG1
0066 12150C          01      GOTO ACPIPT          GO ACCEPT INPUTMRRPG1
12160C*****MRRPG1
12170C*
12180C****END OF MAIN LINE.SPECIAL ROUTINES FOLLOW.
12190C*
12200C*****MRRPG1
13010C*
13020C** INPUT ERROR ROUTINE(IPTER)-MINUS RETURN CODE. ENTERED ON INDICA- MRRPG1
13030C** TOR 03
13040C** SUBROUTINE
13050C*
13060C*****MRRPG1
0067 13070C          IPTER     TAG          MRRPG1
0068 13080C          03      EXSR CONSB          TALK TO OPERATRMRRPG1
0069 13090C          13      GOTO RELEAS          UPON RETURN 13 MRRPG1
13100C*****MRRPG1
13110C*
13120C**MEANS RELEASE THIS TERMINAL. NOT 13 SAYS SEND MESSAGE AND INVITE.
13130C*
13140C*****MRRPG1
0070 13150C          18      MOVE '36'      TPRY,3      OUT LEN      MRRPG1
0071 13160C          18      EXCPT          SEND MSG      MRRPG1
13170C*****MRRPG1
13180C*
13190C* SINCE NOWAIT WAS SPECIFIED NO RETURN CODE IS AVAILABLE TO CHECK.
13200C*
14010C*****MRRPG1
0072 14020C          18      SETOF          18      MRRPG1
0073 14030C          03      GOTO INVIT          MRRPG1
14040C** END OF IPTER
14050C*****MRRPG1
14060C*
14070C** OUTPUT ERROR ROUTINE(OPTERR)-NDN 0 RET CODE
14080C*
14090C*****MRRPG1
0074 14100C          OPTERR     TAG          MRRPG1
0075 14110C          01      SETON          17      MRRPG1
0076 14120C          17      EXSR CONSB          TALK TO OPERATRMRRPG1
0077 14130C          17      SETOF          17      MRRPG1
0078 14140C          13      GOTO RELEAS          13,RELS TERM MRRPG1
0079 14150C          01      GOTO OPTPUT          TRY AGAIN    MRRPG1
14160C** END OF OPTERR
14170C*****MRRPG1
14180C*
14190C** INVALID DATA ROUTINE(INVDAT)-INPUT LEN NOT 8,INVALID OPERATOR,
14200C** (NOT N,+,-),INVALID DIGITS ENTERED,ETC
MRRPG1

```

Figure 6-22 (Part 3 of 6). Example 2 – RPG II MRT Program

```

15010C* MRRPG1
15020C*****MRRPG1
0080 15030C      INVDAT      TAG      MRRPG1
0081 15040C          07      SETON          18      MRRPG1
0082 15050C          07      MOVE '36'      TPRY,3      OUT LEN 18      MRRPG1
0083 15060C          07      EXCPT          TELL TERMINAL      MRRPG1
0084 15070C          07      SETOF          18      MRRPG1
0085 15080C          07      GOTO INVIT      INVITE MORE      MRRPG1
15090C*      INPUT FROM THISMRRPG1
15100C*      TERMINAL      MRRPG1
15110C** END OF INVDAT ROUTINE      MRRPG1
15120C*****MRRPG1
15130C* MRRPG1
15140C**RELEASE TERMINAL ROUTINE(RELEAS) RELEASES TERMS, CLEARS ATTACHED      MRRPG1
15150C**LIST AND ACCUMULATOR LIST ENTRIES AND DETERMINES WHEN TO GO TO EOJMRRPG1
15160C* MRRPG1
15170C*****MRRPG1
0086 15180C      RELEAS      TAG      MRRPG1
0087 15190C          09      GOTO RELCLR          CLEAR LISTS      MRRPG1
15200C* MRRPG1
16010C* NOTE 09 MEANS A TERMINAL ATTACHED TO THIS PROGRAM HAS ENTERED      MRRPG1
16020C* COMMAND INTERRUPT MODE AND ENTERED THE RELEASE COMMAND.      MRRPG1
16030C* MRRPG1
0088 16040C          13      MOVE ' K'      TPRY,2      REL TERM OP CODMRRPG1
0089 16050C          13      EXIT SUBR91      MRRPG1
0090 16060C          RLABL      TPRY      MRRPG1
0091 16070C          RLABL      TNAME      MRRPG1
0092 16080C          RELCLR      TAG      MRRPG1
0093 16090C          09      MRRPG1
0094 16100C          13      MOVE ' '      ATRY,N      CLEAR THIS NTRYMRRPG1
0095 16110C          09      MRRPG1
0096 16120C          13      MOVE '000000'      ACMY,N      MRRPG1
0097 16130C          09      MRRPG1
0098 16140C          13      GOTO ACPIPT      MRRPG1
16150C*****MRRPG1
16160C* MRRPG1
16170C**SHUTDOWN ROUTINE (SHTDWN) 04 RET CODE. ISSUES STOP INVITES AND      MRRPG1
16180C**SHUTDOWN MESSAGE TO ALL ATTACHED TERMS BEFORE GOING TO EDJ      MRRPG1
16190C* MRRPG1
16200C*****MRRPG1
0099 17010C      SHTDWN      TAG      MRRPG1
0100 17020C          06      SETON          11SHUTDOWN IND      MRRPG1
0101 17030C      SHDWNO      TAG      MRRPG1
0102 17040C          11N      ADD 1          N      ADD TO INDEX      MRRPG1
0103 17050C          11ATRY,N      COMP ' '      19TERMINAL HERE      MRRPG1
0104 17060C          19      GOTO SHDWN1      NO CHECK NEXT      MRRPG1
0105 17070C          11      MOVE ATRY,N      TPRY,5      TNAME TO ARY      MRRPG1
0106 17080C          11      MOVE ' D A'      TPRY,2      STOP INVITE      MRRPG1
0107 17090C          11      MOVE '22'      TPRY,4      MAX XPECTO LEN      MRRPG1
0108 17100C          11      READ TPCONSIO      MRRPG1
0109 17110C          11      EXSR RETCHK      CHECK TRN CODE      MRRPG1
0110 17120C          09      GOTO SHDWN1      TERM CANCELLED      MRRPG1
0111 17130C          11      MOVE '48'      TPRY,3      OUT LEN      MRRPG1
0112 17140C          11      EXCPT          APPROPRIATE MSGMRRPG1
0113 17150C          SHDWN1      TAG      MRRPG1

```

*Note:* When the last terminal attached to an MRT program is processed, issue a Release Terminal operation to that terminal in order to check the count of outstanding Invite Inputs. If the count is greater than zero, the program can issue an Accept Input operation. For example, suppose an MRT program is servicing the maximum number of requestors and one or more additional requests are queued to the program. If the program receives a shutdown-requested return code (04) and goes to end of job without checking the count of outstanding Invite Inputs, the program terminates with a 2C termination code (going to end of job with outstanding Invite Inputs) and each of the queued terminals receives an S06 message (program cancelled – shutdown).

**Figure 6-22 (Part 4 of 6). Example 2 – RPG II MRT Program**

```

0114 17160C          11N          COMP 4          LRDONE          MRRPG1
0115 17170C          NLR          GDT0 SHDWNO        NO,NEXT TERM    MRRPG1
0116 17180C          LR          GOTO END          YES,E0J        MRRPG1
0117 17190C          END          TAG          MRRPG1
17200C*****MRRPG1
18010C*              MRRPG1
18020C** RETURN CODE CHECKING ROUTINE(RETCHK) MRRPG1
18030C*              MRRPG1
18040C*****MRRPG1
0118 18050CSR          RETCHK    BEGSR          MRRPG1
0119 18060CSR          01        SETOF          MRRPG1
0120 18070CSR          01        SETOF          060708        MRRPG1
0121 18080CSR          01RETCOD COMP 00          0910          MRRPG1
0122 18090CSR          03          MRRPG1
0123 18100CSR          04          GOTO ENDRET    MRRPG1
0124 18110CSR          05RETCOD COMP 04          080706SHUT DOWN MRRPG1
0125 18120CSR          06          MRRPG1
0126 18130CSR          07          GOTO ENDRET    MRRPG1
0127 18140CSR          08RETCOD COMP 08          09TERM CANCELLED MRRPG1
0128 18150CSR          N09RETCOD COMP 10          10STOP INVITE OKMRRPG1
0129 18160CSR          N09          MRRPG1
0130 18170CSR          N10        SETON          07CATCHALL    MRRPG1
0131 18180CSR          ENDRET    TAG          MRRPG1
0132 18190CSR          ENDSR          MRRPG1
18200C**END OF RETCHK MRRPG1
19010C*****MRRPG1
19020C*              MRRPG1
19030C** CONSOLE I/O ROUTINE. MRRPG1
19040C*              MRRPG1
19050C*****MRRPG1
0133 19060CSR          CONSB     BEGSR          MRRPG1
0134 19070CSR          01        MOVE TNAME    SAVE 6        SAVE TNAME    MRRPG1
0135 19080CSR          91        SETON          81        CONSLC EXCPT MRRPG1
0136 19090CSR          91        SETOF          91        CLEAR NEG RETCDMRRPG1
0137 19100CSR          01        MOVE '47'     TPRY,3      OUT LEN       MRRPG1
0138 19110CSR          01        MOVE '16'     TPRY,4      MAX INPUT LEN MRRPG1
0139 19120CSR          01        EXCPT          PUT CONSOLE   MRRPG1
0140 19130CSR          01        READ TPCONSIO GET CONSOLE   MRRPG1
0141 19140CSR          01        MOVE SAVE    TNAME       RESTORE TNAME MRRPG1
0142 19150CSR          01CONSP   COMP 'TA'    13131800ES OPERATOR -MRRPG1
0143 19160CSR          81        SETOF          81          MRRPG1
0144 19170CSR          CONSB     ENDSR          WANT TO      MRRPG1
19180C**END OF CONSB SUBROUTINE TRY AGAIN MRRPG1
0145 200100TPCONSIOE 81          MRRPG1
0146 200200          TPRY,3    08          MRRPG1
0147 200300          04 'C'      MRRPG1
0148 200400          14 'CONSOL' MRRPG1
0149 200500          03        40 'INPUT TP ERROR TNAME-' MRRPG1
0150 200600          17        40 'OUTPUT TP ERROR TNAME-' MRRPG1
0151 200700          TNAME    47          MRRPG1
0152 200800          E          18          MRRPG1
0153 200900          TPRY,3    08          MRRPG1
0154 201000          04 'CF'    MRRPG1
0155 201100          TNAME    14          MRRPG1
0156 201200          36 'TRY AGAIN' MRRPG1
0157 201300          07        26 'INV DATA' MRRPG1
0158 201400          E          03        26 'TP ERROR' MRRPG1
0159 201500          04N17N07 MRRPG1
0160 201600          TPRY,3    08          MRRPG1
0161 201700          04 'CB'    MRRPG1
0162 201800          TNAME    14          MRRPG1
0163 201900          30 'CURRENT VAL-' MRRPG1
0164 202000          31 '+'     MRRPG1
0165 210100          16        31 '-'     MRRPG1
0166 210200          52 'ENTER DATA' MRRPG1
0167 210300          ACMY,N3 41          MRRPG1
0168 210400          E          11          MRRPG1
0169 210500          TPRY,3    08          MRRPG1
0170 210600          4 'CF'    MRRPG1
0171 210700          ATRY,N   14          MRRPG1
0172 210800          28 'CCP SHTDWN' MRRPG1
0173 210900          38 'LAST REC-' MRRPG1
0174 211000          03        48 'TP ERROR' MRRPG1
0175 211100          07        48 'BAD DATA' MRRPG1
0176 211200          04OPRATR 41          MRRPG1
0177 211300          04TIPT   48          MRRPG1
0178 211400          10        48 ' NO DATA' MRRPG1
211500* END OF PROG MRRPG1

```

Figure 6-22 (Part 5 of 6). Example 2 – RPG II MRT Program

## INDICATORS USED

LR 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 81 91

EXECUTION TIME TABLES AND ARRAYS						
STMT#	TABLE/ DEFINED	DEC	ENTRY ARRAY	NUMBER OF LENGTH	DTT ENTRIES	T/A DISP
0004	TPRY			006	00005	0100 002D
0005	ATRY			006	00004	0108 004B
0006	ACHY	0		010	00004	0110 0009

## FIELD NAMES USED

STMT#	NAME	DEC	LGTH	DISP
0009	RETCOD	0	002	006F
0010	EFFL	0	004	0073
0011	TNAME		006	0063
0012	OPRATR		001	0064
0013	TPIPT	0	007	007A
0014	CONSHIP		003	0067
0019	N	0	001	007B
0134	SAVE		006	006D

## LABELS USED

STMT#	NAME	TYPE
0015	ACPIPT	TAG
0025	LUP1	TAG
0033	LUP2	TAG
0039	GO1	TAG
0041	GO2	TAG
0055	OPTPUT	TAG
0060	INVIT	TAG
0067	IPTR	TAG
0074	OPTERR	TAG
0080	INVDAT	TAG
0086	RELEAS	TAG
0092	RELCLR	TAG
0099	SHTDWN	TAG
0101	SHDWN0	TAG
0113	SHDWN1	TAG
0117	END	TAG
0118	RETCHK	BEGSR
0131	ENDRET	TAG
0133	CONSB	BEGSR
0144	CONSE	ENDSR

## ERROR NUMBER STATEMENT NUMBER

RG 097	0008
RG 558	0022
RG 558	0108
RG 558	0140

ERROR SEVERITY	TEXT
RG 097 W	NO FIELDS DESCRIBED FOR THIS OR PREVIOUS RECORD.
RG 558 W	INVALID USE OF, OR MISSING, RESULTING INDICATORS WITH THIS OP CODE. ASSUME INVALID RESULTING INDICATORS BLANK.

Figure 6-22 (Part 6 of 6). Example 2 – RPG II MRT Program

As an Assembler programmer, you must be familiar with the information included in *Chapter 2: Standard Application Program Interface to the CCP*. To assist you in supplying codes needed for your source program, the CCP provides six macro instructions, each of which generates information needed by the source program.

Four of these macros generate equates for values significant to communications under CCP: \$NCOM, \$NPLO, \$NOPV, and \$NRTV. The fifth macro — \$NPL — generates a parameter list, and the sixth macro — \$NCIO — generates a request for a communications operation.

Each of the supplied macro-definitions is a member of the Source Library on your program-preparation pack.

*Note:* Because several of these macros use macro processor facilities not available prior to version 08, modification level 00 of System/3 Model 10 Disk System Management, they should not be used for program preparation with an earlier version of the Macro Processor on that system.

### Symbols Used in Defining Macro Instructions

The symbols [ ] and { } are used in this publication to help define the macro instructions. You do not code these symbols; they are only used to indicate how a macro instruction may be written.

[ ] indicates an optional operand. The operand enclosed in the brackets may or may not be coded, depending on whether or not the associated option is desired. If more than one item is enclosed in brackets, for example  $\left[ \begin{array}{l} \text{GET} \\ \text{PUT} \end{array} \right]$ , either one or none of the items may be coded.

{ } indicates that a choice must be made. One of the operands from the vertical stack within braces, for example  $\left\{ \begin{array}{l} \text{YES} \\ \text{NO} \end{array} \right\}$ , must be coded, depending on which of the associated services is desired.

Options that are underlined are the default values used by the CCP if you do not provide an operand. For example, for  $\left\{ \begin{array}{l} \text{YES} \\ \text{NO} \end{array} \right\}$ , YES is the assumed value.

For cases in which YES is appropriate, Y may be coded; for cases in which NO is appropriate, N may be coded.

For more information concerning the coding of macro instructions, see the following manuals:

- *IBM System/3 Models 10 and 12 Control Programming Macros Reference Manual*, GC21-7562
- *IBM System/3 Model 15 System Control Programming Macros Reference Manual*, GC21-7608

### Mnotes

Mnotes applicable to the supported Assembler macros are described at the end of this chapter. Each mnote description addresses the severity of the problem, the issuing macro, an explanation of the problem, system action, and suggested programmer action.

### GENERATE EQUATES FOR COMMON VALUES (\$NCOM)

The \$NCOM macro generates a set of symbolic equates, each of which represents a value commonly used in CCP Assembler language communications programming. The symbols generated and the values they represent are:

Symbol	Decimal Value	Value Represented
\$NIXR1	1	Index register 1
\$NIXR2	2	Index register 2
\$NSENT	4	Address of General Entry
\$NSCCR	1	DSM RIB for CCP
\$NSCCS	0	CCP sub-RIB for communications operation
\$NLPL	16	Length of a parameter list
\$NLPLF	2	Length of a parameter list field
\$NLSTN	6	Length of a symbolic terminal name

This macro need be used only once in an assembly to generate the desired symbols. However, if this macro is used more than once in an assembly, duplicate symbols are not to be generated and no assembly error or error mnote is generated. A warning mnote is issued, however, each time this macro is used after the first use.

The symbols generated by this macro are used by the \$NCIO macro. If the \$NCIO macro is issued in an assembly and the \$NCOM macro has not previously been issued, the \$NCIO macro generates the symbols. However, each subsequent time the \$NCOM macro is issued, a warning mnote is issued.

The format of the \$NCOM macro is:

1	8	14
	\$NCOM	

Notes:

- This macro has no operands.
- The name field of this macro is not used; any symbol entered in this field is ignored.

### GENERATE EQUATES FOR PARAMETER LIST OFFSETS (\$NPLO)

The \$NPLO macro generates a set of symbolic equates. Each symbol generated represents a field in a parameter list, and is equated to the displacement (offset from the first byte of the parameter list) of the rightmost byte of that field.

The symbols generated, and the fields they represent are:

<i>Symbol</i>	<i>Decimal Value</i>	<i>Field Represented</i>
\$NPRTC	1	Return code field
\$NPOPC	3	Operation code/modifier field
\$NPOUL	5	Output data length field (see note)
\$NPEFL	5	Effective (actual) input data length field (see note)
\$NPATI	5	Attributes identifier field (see note)
\$NPINL	7	Maximum input data length field
\$NPRAA	9	Record area address field
\$NPWKA	11	Internal work field
\$NPWKB	13	Internal work field
\$NPWKC	15	Internal work field

*Note:* Symbols \$NPOUL, \$NPEFL, and \$NPATI represent the same field within a parameter list and are equated to identical displacements.

This macro need be used only once in an assembly to generate the desired symbols. If used more than once, it does not cause duplicate symbols to be generated and no assembly error or error mnote is generated. A warning mnote is issued, however, each time this macro is issued after the first.

The symbols generated by this macro are used by the \$NCIO macro. If the \$NCIO macro is used in an assembly but the \$NPLO macro has not been previously issued, the \$NCIO macro generates the symbols itself. However any subsequent use of the \$NPLO macro only causes a warning mnote to be issued (duplicate symbols are not generated).

The format of the \$NPLO macro is:

1	8	14
	\$NPLO	

Notes:

- This macro has no operands.
- The name field of this macro is not used; any symbol entered in this field is ignored.

## GENERATE OPERATION CODE/MODIFIER VALUES (\$NOPV)

The \$NOPV macro generates a set of symbolic equates. Each symbol generated represents the value of a CCP communications operation code or operation modifier (see *Appendix E* for the hexadecimal values of the operation code/modifier combinations). The symbols generated, and the operation code or modifier each represents, are:

<i>Symbol</i>	<i>Operation defined</i>
\$NCGET	Get
\$NCPUT	Put
\$NCPTG	Put-Then-Get
\$NCPNW	Put-No-Wait
\$NCINV	Invite Input
\$NCACC	Accept Input
\$NCANW	Accept No-Wait
\$NCSPI	Stop Invite Get
\$NCGTA	Get Terminal Attributes
\$NCACQ	Acquire Terminal
\$NCAQC	Acquire Command-mode Terminal
\$NCAQG	Generic Acquire Terminal <sup>1</sup> (5704-SC2 only)
\$NCREL	Release Terminal
\$NCSHQ	Shutdown Inquiry
\$NCCPY	Copy (3270 DFF only)
\$NCEAU	Erase All Unprotected (3270 DFF only)
\$NCTCH	Chain Task Request (5704-SC2 only)
\$NCRTC	Release and Task Chain (5704-SC2 only)
\$NCPCR	Port Command Request <sup>1</sup> (5704-SC2 only)
 <i>Modifier defined</i>	
\$NMMSG	Send end-of-transmission
\$NMBLK	End of current data block
\$NMRVI	Send Reverse Interrupt
\$NMKPL	Keep control of communications line
\$NMSTA	Set Terminal attributes
\$NMCMD	Command-mode non-PRUF terminal
\$NMDLY	Shutdown Delay Modifier
\$NMNNL	Record does not start a new line (not New Line)
\$NMNEL	Record does not end the current line (not End Line)
\$NMOVR	Override/Selected Field List (3270 DFF only)
\$NMPRF	Program request under format
\$NMGEN	Generic modifier to acquire

*Note:* Acquire-command-mode-non-PRUF-terminal operation may be accomplished by a combination of the acquire terminal and command mode modifier. For example, \$NCACQ and \$NMCMD produces the same op code as \$NCAQC does.

<sup>1</sup> For SIOC see *IBM System/3 Model 15D Channel Connected Systems Program Reference and Logic Manual*, GC21-5199. For Portline see *operations* in the index.

The Generic Acquire Terminal operation may be accomplished by a combination of the acquire terminal and the generic modifier; for example, \$NCACQ and \$NMGEN produce the same operation code as \$NCAQG.

This macro need be used only once in an assembly to generate the desired symbols. If used more than once, this macro does not cause duplicate symbols to be generated and no assembly error or error mnote is generated. However, a warning mnote is issued each subsequent time this macro is issued in the assembly.

The symbols generated by this macro are used by the \$NCIO macro. If the \$NCIO macro is issued in an assembly but the \$NOPV macro has not previously been issued, the \$NCIO macro generates the symbols itself. However, any subsequent use of the \$NOPV macro only causes a warning mnote to be issued (duplicate symbols are not generated).

The format of the \$NOPV macro is:

1	8	14
	\$NOPV	

Notes:

- This macro has no operands.
- The label field of this macro is not used; if a symbol is specified in the label field, it is ignored.

## GENERATE EQUATES FOR RETURN CODE VALUES (\$NRTV)

This macro generates a set of symbolic equates. Each symbol generated represents the two-byte signed numeric value of a return code issued by CCP. The symbols generated, and the meaning and value of the return code specified by each, follow:

Type of Return Code	Symbol	Return Code Value	Meaning
No error or exception	\$NOK	0	Successful operation
	\$NACTC	14	Successful chain task data accept operation (5704-SC2 only)
Exception	\$NXDTR	1	Data truncated
	\$NXEOT	2	EOT received/non-PRUF data returned to PRUF program
	\$NXEDT	3	EOT received and data truncated
	\$NXSHD	4	Shutdown requested
	\$NXDPD	5	Data pending on BSCA line
	\$NXRVI	6	RVI/Terminal Interrupt received (5704-SC2 only)
	\$NXCLR	7	3720 CLEAR key
	\$NXNAV	8	Terminal no longer available
	\$NXOFF	9	Terminal offline
	\$NXSPI	10	Stop Invite Input successful
	\$NXNAQ	11	Acquire terminal failed
	\$NXMAX	12	Maximum number of chain task requests already queued (5704-SC2 only)
	\$NXTCP	13	Insufficient TP buffer available for this request
	\$NXADT	15	Chain task data was truncated (5704-SC2 only)
	\$NXNAC	16	No completed invite (5704-SC2 only)
	\$NXSHP	17	Shutdown delay entered
	Error	\$NRDCK	-1
\$NRTRN		-2	Invalid character
\$NRLST		-3	Lost data
\$NRPBS		-4	Permanent BSCA error
\$NRABN		-5	Abnormal response
\$NRXRA		-6	Transmit/Receive abort
\$NRATO		-7	No response to polling/addressing
\$NR TTO		-8	Text timeout
\$NRWTO		-9	Wait time exceeded
\$NRNOC		-10	No connection
\$NR IID		-11	Invalid ID
\$NRABD		-12	Abort, disconnect
\$NRADC		-13	Adapter check
\$NRNAK		-14	Negative response to addressing
Error (3270 only)	\$NR2DU	-20	Device unavailable or not ready
	\$NR2ED	-22	Equipment check, device end
	\$NR2TE	-23	Terminal control unit detection of BSCA error
	\$NR2CD	-24	Control check, data check
	\$NR2PD	-25	Data check on Copy command
	\$NR2PO	-26	Operation check on Copy command
	\$NR2PB	-27	Device busy on Copy command
	\$NR2PC	-28	Control check, operation check, data check on Copy command
	\$NR2PI	-29	Invalid data received from 3270 using DFF (Get, Accept Input, Stop Invite Input operations)



Type of Return Code	Symbol	Return Code Value	Meaning
Error (3735 only)	\$NR5SR	-40	Attempted send before receive
	\$NR5IC	-41	Invalid character
	\$NR5BF	-42	3735 buffer overflow
	\$NR5DF	-43	Disk full
	\$NR5RF	-44	Directory full
	\$NR5UH	-45	Undefined header
	\$NR5DE	-46	3735 disk error
Error (3741, 5234, and 5235)	\$NR7TE	- 50	Transparency error occurred
	\$NR7NA	- 51	No activity in 20 seconds
	\$NR7DC	- 52	Data check
	\$NR7LB	- 53	Received line bid error
	\$NR7WL	- 54	Wrong length error
	\$NR7RP	- 55	Reset was pressed on 3741
	\$NR7SC	- 56	Security check
	\$NR7DO	- 57	Disk overflow
	\$NR7BE	- 58	Bad extent error
	\$NR7BT	- 59	Both stations transmit
	\$NR7LE	- 60	Length error
	\$NR7NF	- 61	No record found on disk
	\$NR7SE	- 62	Seek error
	\$NR7RE	- 63	Read error
	\$NR7WE	- 64	Write error
	\$NR7NR	- 65	3741 not ready
\$NR7WP	- 66	Diskette is write-protected	

For more information concerning these return codes, and recommended user actions, see *Appendix E: Return Codes*.

This macro need only be used once in an assembly to generate the desired symbols. If it is used more than once, duplicate symbols are not generated and no assembly error is forced. However each subsequent time this macro is used, a warning mnote is issued.

The format of the \$NRTV macro is:

1	8	14
	\$NRTV	

*Notes:*

1. This macro has no operands.
2. The label field of this macro is not used; if a symbol is specified in the label field, it is ignored.

## GENERATE PARAMETER LIST (\$NPL)

The \$NPL macro generates a communications parameter list. You can specify the initial value of each field of this list. This macro may be used any number of times in an assembly; a separate parameter list is generated each time this macro is used. When the parameter list is generated, the operand values specified are not checked for validity; therefore, care must be taken that only valid operands are specified.

The format of the \$NPL macro is:

1	8	14
[ label ]	\$NPL	[ OP-valutex ] [ { ,OUTLEN-valutex } ] [ { ,ATTRID-valutex } ] [ ,INLEN-valutex ] [ ,RECA-addrx ]

See *Note* under \$NCIO macro for the meaning of *valutex* and *addrx*.

If you specify a label with the \$NPL macro, an EQUATE is generated. This EQUATE sets the specified symbol equal to the address of the first byte of the parameter list generated.

*OP-valutex* specifies an initial value for the operation/modifiers field of the parameter list. If this operand is omitted, a value of X'0000' is generated for the operation/modifiers field.

The operand value should represent a valid CCP operation. Any valid absolute expression may be used, but it is recommended that operation and modifier values generated by the \$NOPV macro be used.

*OUTLEN-valutex* specifies an initial value for the output data length field. If this operand is omitted (and the ATTRID operand is also omitted), a value of X'0000' is generated for the output data length field.

Because the output data length field (generated by OUTLEN) and the attributes identifier field (generated by ATTRID) occupy the same locations in the parameter list, both operands cannot be specified in one macro instruction. If both are specified, an error mnote is generated.

The operand value may be any valid absolute expression. Note that a data length entry does not include the six-byte symbolic terminal name found at the beginning of a record area.

*ATTRID-valutex* specifies an initial value for the attributes identifier field of the parameter list. If this operand is omitted and the OUTLEN operand is also omitted, a value of X'0000' is generated for the attributes identifier field.

Because the attributes identifier field generated by ATTRID and the output data length field (generated by OUTLEN) occupy the same locations in the parameter list, both operands cannot be specified in one macro instruction. If both are specified, an error mnote is generated. The operand value, which may be any valid absolute expression, should represent a positive number that corresponds to the identifier of a terminal-attributes set that has been defined for your installation. This definition is performed in a CCP Assignment Build run by a // TERMATTR statement.

*INLEN-valutex* specifies an initial value for the maximum input data length field of the parameter list; if this operand is omitted, a value of X'0000' is generated for that field.

This operand, which may be any valid absolute expression, should represent a positive number which is no greater than the length of the data portion of the record with which this parameter list will initially be used. The six-byte symbolic terminal name at the beginning of the record area is not included as part of the input data length.

*RECA-addrx* specifies an initial value for the record area address field in the parameter list; if this operand is omitted, a relocatable value equal to the beginning of the parameter list plus 16 is generated for that field.

This operand, which may be any valid relocatable expression, should represent the high-order (leftmost) byte of the record area. Note that the high-order byte of the record area is the leftmost byte of the name field in the record area, *not* the address of the first byte of the data portion of the record area.

Each time this macro is used, a 16-byte parameter list (composed of eight successive 2-byte DC fields) is generated. Any fields which will not be affected by the operands of this macro are set to X'0000'. If you specify a label in this macro instruction, the specified symbol is equated to the address of the high-order (leftmost) byte of the generated parameter list. No additional symbols are generated by this macro. You can cause symbols to be generated for offsets of fields within a parameter list by issuing a \$NPLO macro instruction or by issuing a \$NCIO macro instruction without issuing a \$NPLO macro instruction.



## SET CONTROL INFORMATION FOR COMMUNICATIONS OPERATION (\$NCIO)

The \$NCIO macro provides control information needed to identify a communications operation and request CCP to perform that operation.

The format of the \$NCIO macro is:

1	8	14
	\$NCIO	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>[PLIST— {  addrx  disp(,regx)  (1)  (2)  addrx  disp(,regx)} ]</p> <p>[OP — {  GET [,RVI]  {PUT} { {BLK} } [ ,NNL] [ ,NEL] [ ,OVR] [ ,PRF]  {PNW} { {MSG} }  PTG [,NNL] [ ,NEL]  INV  ACC  ANW  SPI  GTA  ACQ [,CMD] [ ,STA]  ACQ [,GEN]<sup>1</sup> [ ,STA]  REL [ ,KPL]  RTC  CPY  EAU  PCR<sup>1</sup>  SHQ [,DLY]  TCH  WAT  valuex  (1)  (address)  (dispx(,regx))</p> <p>[ , {  OUTLEN  ATTRID } — {  valuex  (1)  (addrx)  (dispx(,regx)) } ]</p> <p>[ , INLEN — {  valuex  (1)  (addrx)  (dispx(,regx)) } ]</p> <p>[ , RECA — {  addrx  (1)  (addrx)  (dispx(,regx)) } ]</p> <p>[ , TNAME — {  chars  (addrx)  (dispx(,regx)) } ]</p> <p>[ , EXEC— {  YES  NO } ]</p> </div> <div style="width: 50%; vertical-align: middle;"> <p style="text-align: center;">} ]</p> <p><i>Note:</i> The following operand forms must be enclosed in apostrophes:</p> <p>'operation, modifier' (OP—operand)  'chars' (TNAME—operand)  '(dispx(regx))'</p> </div> </div>

<sup>1</sup> For SIOC see *IBM System/3 Model 15D Channel Connected Systems Program Reference and Logic Manual*, GC21-5199. For Portline see *operations* in the index.

Notes:

- All operands in this macro are optional.
- Several syntactic classes are included in these descriptions, and are defined as follows:

valuex — an absolute expression representing a number whose assembled value will fall in the range  $0 \leq \text{value} \leq 32,767$

dispx — an absolute expression (representing a displacement from a base address) whose assembled value will fall in the range  $0 \leq \text{value} \leq 255$ .

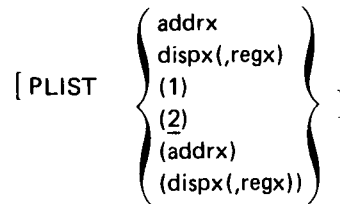
regx — an absolute expression (representing the number of an index register) whose assembled value will be either 1 or 2.

addrx — a relocatable expression representing a main-store address in your program.

chars — one to six alphameric characters.

- Certain operand values may be written with surrounding parentheses to specify indirect addressing. If the first character of such an operand value is a left-parenthesis, the last character of that value must be a right-parenthesis. If this is not the case, an error is issued and the macro instruction is not generated.
- If an operand value is written enclosed in parentheses, the total number of characters within the parentheses must not exceed 16; if the number of characters exceeds 16, an error is issued, and the macro instruction is not generated.
- The \$NCIO macro can be used any number of times within an assembly.
- If you specify a label in the \$NCIO macro instruction, an equate is generated. This equate sets the specified symbol equal to the address of the first byte of the first instruction generated.

- The first time a \$NCIO instruction is used in an assembly, any or all of three sets of symbolic equates may be generated. If any of three macros (\$NCOM, \$NPLO, or \$NOPV) had not been issued before in that assembly, any equates which would have been produced by that macro are generated by the \$NCIO macro. Any subsequent issuance of any of these macros produces a warning mnote.
- In the generation of the \$NCIO macro, no check is made that any expression is syntactically valid to the Assembler, nor that the value generated is valid for its intended use.



The *PLIST* operand specifies the address of the high-order (leftmost) byte of the communications parameter list being used.

When a request is made of CCP to perform a communications operation, the address of the parameter list used with this operation must be in index register 2. Therefore, you must either ensure that the address of your parameter list is in index register 2 when the instructions generated by this macro are executed, or code this operand in order to set index register 2 to the appropriate operand.

Omit this operand or code (2) to signify that index register 2 already contains the address of the parameter list. Specifying a value of other than (2) causes an instruction to be generated which loads index register 2 with the parameter list address you specify. The source of this address is determined by the form in which you specify the operand:

<i>Form</i>	<i>Index Register 2 set to:</i>
addrx	The value of the expression itself.
dispx(,regx)	An address equal to the value of the specified displacement added to the contents of the specified index register.
(1)	The contents of index register 1.
(addrx)	The contents of a 2-byte field in main storage whose rightmost byte is located at the specified address.
(dispx(,regx))	The contents of a 2-byte field in main storage whose rightmost byte is located at the specified displacement from an address contained in the specified index register.

}	[ ,OP ]	GET[,RVI]	}
		{PUT}{BLK}	
		{PNW}{MSG} [,NNL] [,NEL] [,OVR] [,PRF]	
		PTG [,NNL] [,NEL]	
		INV	
		ACC	
		ANW	
		SPI	
		GTA	
		ACQ[,CMD] [,STA]	
		ACQ[,GEN] <sup>1</sup> [,STA]	
		REL[,KPL]	
		CPY	
		EAU	
		PCR <sup>1</sup>	
		SHQ	
		TCH	
		WAT	
valuex			
(1)			
(address)			
(dispx(,regx))			

<sup>1</sup> For SIOC see *IBM System/3 Model 15D Channel Connected Systems Program Reference and Logic Manual*, GC21-5199. For Portline see *operations* in the index.

**Note:** If one (or more) modifiers is specified, the entire operand must be enclosed within apostrophes (for example, OP-'PUT,MSG').

The *OP* operand specifies the setting of the operation field of your parameter list. If this operand is omitted, no instructions are generated to set the operation field.

This operand can identify the actual value to which the field is to set, by specifying:

- an abbreviation of an operation code, and any desired associated modifiers (see Appendix D for meanings of the abbreviations).
- an absolute expression
- the name of a location from which the field's value can be obtained when the instruction is executed.

If the abbreviation of an operation code is used, the field is set to the proper numeric value for that operation (and associated modifier). If one or more modifiers are specified, each must be preceded by a comma; in addition, the entire operand must be enclosed within apostrophes to prevent the Macro Processor from interpreting the operation code and the (first) modifier as a delimiter of the operand. For example, if you specified NNL for the PTG operand, it would be coded OP-'PTG,NNL'.

If several modifiers are used they may be specified in any order after the operation code. In the generation of this macro, no check is made to determine if a valid modifier is used with the operation code.

If the operand value begins with a left-parenthesis, it indicates that the value to be set into the operation field is to come from a specified location when the generated instruction is executed. This location is determined as follows:

<i>If the form is</i>	<i>The field is set to</i>
(1)	The contents of index register 1.
(addrx)	The contents of 2-byte field in main storage the rightmost byte of which is located at the specified address.
(dispx(,regx))	The contents of a 2-byte field in main storage, the rightmost byte of which is located at the specified displacement from the specified index register.

If the operand value is neither a 3-character operation code abbreviation (with or without modifiers) nor an expression beginning with a left-parenthesis, it is assumed to be an absolute expression, the assembled numeric value of which is to be set into the operation field.

$$[ \left. \begin{array}{l} \{ \text{OUTLEN} \} \\ \{ \text{ATTRID} \} \end{array} \right\} - \left. \begin{array}{l} \text{valuex} \\ (1) \\ (\text{addrx}) \\ (\text{dispx}(\text{,regx})) \end{array} \right\} ]$$

These optional operands allow you to specify either the setting of the output data length field (if OUTLEN is specified) or the setting of the attributes identifier field (if ATTRID is specified). If both operands are omitted, no instructions are generated to set the field (the attributes identifier field and the output data length field are the same field in a parameter list). If both OUTLEN and ATTRID are specified in the same macro instruction, an mnote is issued, and the macro instruction is not generated.

The operand may specify either the actual value to be set into the field, or a location from which the value can be obtained at the time the instruction is executed. The source of the value is determined by the form in which the operand value is specified:

<i>If the form is</i>	<i>The field is set to</i>
valuex	The value of the expression itself.
(1)	The contents of Index Register 1.
(addrx)	The contents of a 2-byte field in main storage, whose rightmost byte is located at the specified address.
(dispx(,regx))	The contents of a 2-byte field in main storage, whose rightmost byte is located at the specified displacement from an address contained in the specified index register.

$$[ \text{,INLEN} - \left. \begin{array}{l} \text{valuex} \\ (1) \\ (\text{addrx}) \\ (\text{dispx}(\text{,regx})) \end{array} \right\} ]$$

This operand specifies the setting of the maximum input data length field of your parameter list. If this operand is omitted, no instructions are generated to set this field.

The operand may specify either the actual value to be set into the field, or a location from which the value can be obtained at the time the instruction is executed. The source of the value is determined by the form in which the operand value is specified. The possible forms and the resultant setting of each follow.

<i>Operand form</i>	<i>Resultant setting</i>
valuex	The value of the expression itself.
(1)	The contents of Index Register 1.
(addrx)	The contents of a 2-byte field in main storage, whose rightmost byte is located at the specified address.
(dispx(,regx))	The contents of a 2-byte field in main storage, whose rightmost byte is located at the specified displacement from an address contained in the specified index register.

$$[ \text{,RECA} - \left. \begin{array}{l} \text{addrx} \\ (1) \\ (\text{addrx}) \\ (\text{dispx}(\text{,regx})) \end{array} \right\} ]$$

This optional operand specifies the setting of the record area address field of the user's parameter list; if this operand is omitted, no instruction is generated to set that field.

The operand may specify either the actual address to be set into the field, or a location from which the address can be obtained at the time the instruction is executed. The source of the address is determined by the form in which the operand value is specified. The operand forms used, and the field setting for each follow.

<i>Operand form</i>	<i>Field set to</i>
addrx	The value of the expression itself.
(1)	The contents of Index Register 1.
(addrx)	The contents of a 2-byte field in main storage, whose rightmost byte is located at the specified address.
(dispx(,regx))	The contents of a 2-byte field in main storage, whose rightmost byte is located at the specified displacement from an address contained in the specified index register.

$$[,TNAME - \left. \begin{array}{l} \text{chars} \\ (\text{addrx}) \\ (\text{dispx} (, \text{regx}) ) \end{array} \right\} ]$$

$$[,EXEC - \left. \begin{array}{l} \text{YES} \\ \text{Y} \\ \text{NO} \\ \text{N} \end{array} \right\} ]$$

This optional operand specifies the setting of the name of the program to be requested or the symbolic terminal name in the name field of the user's record area. When this operand is specified, instructions are generated to set the field; these instructions make use of the record area address in the parameter list as a pointer to the leftmost byte to which the specified name should be moved. If a RECA operand was specified in this macro instruction, the record area address specified by that operand is used; if a RECA operand was not specified in this macro instruction, the address presently contained in the record area address field of the parameter list is used. If this operand is omitted, no instructions are generated to set the program or the symbolic terminal name in the name field.

The operand can specify the program or symbolic terminal name to be set into the field, or a location from which the name can be obtained at the time the instruction is executed. The source of the name is determined by the form in which the operand value is specified. The possible forms of these operands, and the field setting generated by each follow.

<i>Operand form</i>	<i>Field set to</i>
chars (addrx)	The character string specified. The contents of a 6-byte field in main storage, whose rightmost byte is located at the specified address.
(dispx(,regx))	The contents of a 6-byte field in main storage, whose rightmost byte is located at the specified displacement from an address contained in the specified index register.

This optional operand specifies whether or not (after setting any control fields specified in other operands of the \$NCIO macro instruction) instructions are generated to cause control to be transferred to CCP to perform the operation. If YES (or Y) is specified, or if this operand is omitted (YES is the default value), instructions are generated to request the CCP to perform the operation specified by the programmer, using information either set by this macro instruction or already available in your parameter list and record area.

If a NO (or N) is specified, no branch is made to General Entry, and no RIB and sub-RIB generated. You might choose to specify this option when you only want to set certain fields in your parameter list or record area, but you won't be performing the operation until a later point in your program.

#### Examples of Using \$NCIO

Figure 7-1 shows several examples of valid \$NCIO macro instructions as you might code them in a program. The examples show various ways of specifying keyword operands for the macro. After issuing the \$NCIO macro instruction, your program should test the return code to determine whether the operation was successful or whether it resulted in an error or exception condition. Return codes are summarized in Appendix E.

Use of the \$EOJ system macro instruction is also shown in Figure 7-1. You can use non-CCP System/3 macro instructions in your CCP programs. These System/3 macro instructions are described in:

- *IBM System/3 Models 10 and 12 Control Programming Macros Reference Manual, GC21-7562*
- *IBM System/3 Model 15 System Control Programming Macros Reference Manual, GC21-7608*



PROGRAM		DATE	PUNCHING INSTRUCTIONS	GRAPHIC PUNCH	PAGE	OF
PROGRAMMER						
Name		Operation	STATEMENT		Remarks	Identification Sequence
1	2	3	4	5	6	7
PROGX	START					
	\$NCOM					
	\$NPLO					
	\$NOPY					
	\$NRTY					
PLIST1	\$NPL					
**			ACCEPT INPUT FROM A TERMINAL (MAX LENGTH 8)			
ACPIP	\$NCIO		PLIST-PLIST1,OP-ACC,INLEN-8,RECA-INPUT			
**			ISSUE PUT MESSAGE TO TERMINAL FROM WHICH INPUT WAS RECEIVED			
MVC			OUT+39(34),MESSGL MESSAGE LENGTH=34			
MVC			OUT+5(6),INPUT+5			
OUTPT1	\$NCIO		PLIST-PLIST1,OP-'PUT,MSG',OUTLEN-34,RECA-OUT			
**			INVITE MORE INPUT FROM THE TERMINAL			
INVT	\$NCIO		PLIST-PLIST1,OP-INV,INLEN-8,RECA-INPUT			
**			ISSUE A MESSAGE TO THE CONSOLE (PUT-THEN-GET)			
\$NCIO			PLIST-PLIST1,OP-PTG,INLEN-3,OUTLEN-29,RECA-CONSLIO		X	
			TNAME-'CONSL'			
**			PUT A MESSAGE TO TERMINAL (PUT-NO-WAIT MESSAGE)			
\$NCIO			PLIST-PLIST1,OP-'PNW,MSG',RECA-OUT,OUTLEN-18		X	
			TNAME-'(INPUT+5)'			
**			PTG TO CONSOLE (NOTE USE OF DISPLACEMENT/REGISTER IN RECA)			
\$NCIO			PLIST-PLIST1,OP-'PTG',INLEN-2,OUTLEN-28,RECA-'(R(,L))',		X	
			TNAME-'CONSL'			
**			PNW TO TERMINAL (USE MOVES TO LOAD PARAMETERS)			
MVC			PLIST+5(2),BL18			
MVC			PLIST+3(2),PNWMSG			
\$NCIO			PLIST-PLIST1,RECA-OUT,TNAME-'INPUT+5'			
**			ISSUE RELEASE TERMINAL (NO PLIST PARAMETER ON \$NCIO)			
LA			PLIST1,2			
\$NCIO			OP-REL,RECA-INPUT,TNAME-'(INPUT+5)'			
**			ISSUE SYSTEM END OF JOB MACRO			
\$EOJ						
INPUT	EQU	*				
DC			14C11' DEFINE INPUT RECORD AREA			
OUT	EQU	*				
DC			40C11' DEFINE OUTPUT RECORD AREA			
MESSGL	DC		CL34'YOU MAY NOW ENTER DATA TO PROGX'			
CONSLIO	EQU	*				
DS			CL34			
BL18	DC		1L2'18' CONSTANTS USED IN STATEMENTS ABOVE			
PNWMSG	DC		1L2'54'			
END			PROGX			

Figure 7-1. Examples of \$NCIO Macro Instructions

## PROGRAMMING RESTRICTIONS

The following restrictions apply when programming for the standard interface to CCP using Assembler Language.

- Input/Output cannot be performed on either the 5471 Printer/Keyboard (Model 10 Disk System and Model 12) or the CRT/Keyboard (Model 15) using the standard DSM Data Management; instead, the console must be addressed as the "terminal" CONSOL, governed by the standard CCP restrictions concerning operations that may be performed on that device. (See the *Standard Communications Interface* for a description of these restrictions.)
- If the *(dispx,regx)* operand format is used in the \$NCIO macro with index register 1 as *regx*, the original value of index register 1 is not returned after using the macro. Therefore, you should save and restore the register when using \$NCIO.
- User DTFs and IOBs must be placed in the first 24K of the user program.

## ASSEMBLER MACRO SUPPORT MNOTES

Mnotes are error messages pertaining to macro instruction formats. They are included in your assembly listing, printed beneath the macro instruction to which each applies. The Mnotes which follow are issued if an error is encountered when using the Assembler Language support macros.

### N2001 CONFLICTING OPERANDS—OUTLEN/ATTRID

<i>Severity:</i>	Error (08)
<i>Issuing macro(s):</i>	\$NPL,\$NCIO
<i>Explanation:</i>	If a \$NPL or \$NCIO macro instruction, you have specified both the OUTLEN and the ATTRID operands. Because the output data length field (specified by OUTLEN) and the attributes identifier field (specified by ATTRID) are the same field in a parameter list, they are mutually exclusive, and only one can be specified at a time.

*System Action:* This macro instruction is not generated.

*Programmer Action:* Correct the macro instruction by using only one of the two operands, and repeat the macro-generation/assembly run.

### N3001 INVALID OPERATION CODE SPECIFIED

<i>Severity:</i>	Error (08)
<i>Issuing macro:</i>	\$NCIO
<i>Explanation:</i>	The parameter specified in the OP operand is in the form which indicates an operation code followed by operation modifiers. However, the first element of the list is not a valid operation code abbreviation.

*System Action:* This macro instruction is not generated.

*Programmer Action:* Correct the parameter by using a valid operation code and repeat the macro-generation/assembly run.

### N3002 INVALID OPERATION MODIFIER SPECIFIED

<i>Severity:</i>	Error (08)
<i>Issuing macro:</i>	\$NCIO
<i>Explanation:</i>	The parameter of the OP operand is in the form which signifies an operation code followed by one or more operation modifier. However, an operation modifier specified is not a valid abbreviation.

*System Action:* This macro instruction is not generated.

*Programmer Action:* Correct the parameter by specifying a valid operation modifier abbreviation and repeat the macro-generation/assembly run.

**N3003 PARAMETER MISSING FINAL RIGHT-PAREN**

**Severity:** Error (08)

**Issuing macro:** \$NCIO

**Explanation:** An invalid form for a parameter was specified. The first character was a left parenthesis, but the last character was not a right parenthesis.

**System Action:** This macro instruction is not generated.

**Programmer Action:** Correct the parameter by enclosing it properly with a right parenthesis and repeat the macro-generation/assembly run.

**N3004 PARENTHESIZED PARAMETER TOO LONG**

**Severity:** Error (08)

**Issuing macro:** \$NCIO

**Explanation:** A parameter enclosed by parentheses was composed of more than 16 characters (excluding the parentheses).

**System Action:** This macro instruction is not generated.

**Programmer Action:** If the error was due to a key-punching error, correct the parameter and repeat the macro-generation/assembly run. If the error was due to an overly complex expression being coded within the parentheses, simplify that expression (by creating an equate for it), use the newly created symbol with the parentheses, and repeat the macro-generation/assembly run.

**N6001 OFFSET VALUES PREVIOUSLY GENERATED**

**Severity:** Warning (04)

**Issuing macro:** \$NPLO

**Explanation:** A \$NPLO macro instruction was used but the offset equate values to be generated from it had previously been generated in this assembly (either by another \$NPLO macro instruction or by a \$NCIO macro instruction).

**System Action:** The offset equates are not generated; to avoid creation of duplicate symbols in this assembly.

**Programmer Action:** None required.

**N6002 OPERATION VALUES PREVIOUSLY GENERATED**

**Severity:** Warning (04)

**Issuing macro:** \$NOPV

**Explanation:** A \$NOPV macro instruction was used, but the values to be generated by it had previously been generated in this assembly (either by a \$NOPV macro instruction or by a \$NCIO macro instruction).

**System Action:** Operation code and modifier value equates are not generated by this statement; to avoid creation of duplicate symbols in the assembly.

**Programmer Action:** None required.

## N6003 RETURN-CODE VALUES PREVIOUSLY GENERATED

<i>Severity</i>	Warning (04)
<i>Issuing macro:</i>	\$NRTV
<i>Explanation:</i>	A \$NRTV macro instruction was used, but the return code value equates to be generated from it had previously been generated in this assembly.
<i>System Action:</i>	Return code value equates are not generated; to avoid creation of duplicate symbols in this assembly.
<i>Programmer Action:</i>	None required.

## N6004 COMMON VALUES PREVIOUSLY GENERATED

<i>Severity:</i>	Warning (04)
<i>Issuing macro:</i>	\$NCOM
<i>Explanation:</i>	A \$NCOM macro instruction was used, but the common equate values to be generated from it had been generated in this assembly (either by a \$NCOM macro instruction or by a \$NCIO macro instruction).
<i>System Action:</i>	Common equate values are not generated by this macro instruction; to avoid creation of duplicate symbols in this assembly.
<i>Programmer Action:</i>	None required.

## PROGRAMMING A USER SECURITY ROUTINE – MODELS 10 AND 12

The user of CCP may, if he chooses, write his own terminal sign on security routines rather than use the CCP password facility supplied with his system. To implement and use such routines, the following must be considered.

- This option must be selected at generation time by the "SECURE-USER" operand of the \$ESEC statement.
- The security data which will be used at terminal sign on time by the security routines must have been previously written to the security module \$CC4Z9 by User Security Data Program, \$CCPAU (see *CCP System Reference Manual*).
- The user security routines must be written in Basic Assembler Language (or an equivalent machine-level language) and must be structured such that they comprise four basic parts (statement numbers refer to the sample program, Figure 7-2):
  1. System equates. They provide the necessary offsets and pointers to various system tables, as well as other common equates.
  2. The transient prologue and equates. The function of this code is to provide the following:
    - a. a temporary two-byte offset value pointing to the relocatable address constants (statement 1087)
    - b. a one-byte offset to the first executable instruction (1091)
    - c. a one-byte ID with a binary value of 1, which defines the CCP transient area in which this transient will be running (1092)
    - d. a two-byte field of hex 0000 to permit the passing of a parameter from this transient to another (1094)
    - e. a two-byte displacement from the beginning of this module pointing to the first character past the ID of the transient to be called next (1095)

- f. the two-character ID of the transient to be called next (last two characters of the module name) (1097)
  - g. three bytes for disk control which are used when this transient is read from disk (1098-1099); the first byte is always 7, the second and third bytes are set by CCP startup with the cylinder/sector address of the transient specified by the two-character transient ID. A one-character delimiter (\$) and the two-byte ID of this transient follow (1100-1101).
3. The main body of code, which in this example begins with statement 1108 and ends with statement 1191. This sample code will: determine the length of, and move the sign-on data to an area within this transient; determine which terminal is attempting to sign on and then verify the sign-on data for that terminal, as found in the user security work area; pass the result of the verification to the CCP sign-on transient, \$CC4SO, by setting a value in "TAXPRM" prior to exiting from this transient. The "TAXPRM" field of any CCP transient (at relative locations 4 and 5 of the transient) is the field in which one transient passes information to another. The total length of parts 2 and 3 cannot exceed hex length 1FF.
  4. Relocation Address Table.

Because any sign-on security routine you write is a CCP transient, your routine must conform to the special way in which the addresses in CCP transients are established and relocated.

In this special relocation method, any (two-byte) address, used in an instruction, which refers to a location within your transient itself requires an entry in a table, at the end of your transient, called the Relocatable Address Table. This method also allows you to address elements, outside your transient, which are in the CCP Communications Area; it also permits you to use any of a selected set of addresses which refer to routines or areas within the remainder of the CCP resident code. Any use of either of these types of addresses also requires an entry in the Relocatable Address Table. In effect, any use of a two-byte address which does not refer to an absolute location requires an entry in this table.

In this method, any address you code in an instruction is *entirely replaced*, when CCP begins to run, by an address derived from an entry in the Relocatable Address Table. There must be *exactly* as many entries in this table as there are relocatable addresses in your transient. Further the first relocatable address in your transient is replaced by an address derived from the first Relocatable Address Table entry, the second relocatable address by an address derived from the second entry, and so forth. Thus, if you wish to make any of the three kinds of address references described in the previous paragraph, it does not matter whether, in an instruction, you code an appropriate relocatable address, but only that you code an appropriate entry in the table.

Each Relocatable Address Table entry is of one of three types, depending on the type of address you need to use in the corresponding instruction:

- a. Address within the transient
- b. Address within the CCP Communications Area
- c. Address of a special routine or area

Each entry is two bytes long. The form of each entry is as follows:

*Address within transient:* The entry must contain the *displacement* of the location within the transient being referred to. For example, the entry at statement 1198, corresponding to the address within the instruction at statement 1144, represents a reference to the location 0042 within the transient. The two-byte displacement is formed in table entry by subtracting the address value of the beginning of the transient (\$CC4YA) from the address within the transient being referred to (YA0100).

*Address within CCP Communications Area:* The entry must contain the hex value C0xx, where xx is the *displacement* within the Communications Area of the element being addressed. In this example, the symbol "COM" has been equated to the value X'C000', and the symbol "@TUSTG" has been defined (by the macro \$ECOM) as representing the displacement of that field in the Communications Area which contains the address of the Terminal Unit Block for the terminal attempting to sign on. Thus, the table entry at statement 1196, corresponding to the address used in statement 1108, is formed by defining a two-byte constant of AL2(COM+@TUSTG).

*Special Address of CCP Routine or Work Area:*

The entry must contain the hex value 80xx, where xx is a number used by CCP to signify which special address is to be referenced. Two such entries are used in this example. The symbol "PGM" has been equated to the value X'8000', and the symbols "#CC4TX" and "USI" have been generated by the macro \$ETRC, and represent the numbers for the special addresses, respectively, of:

- a. \$CC4TX -- the location to which you must branch at the completion of execution of your transient.
- b. The first byte of the user security information (from the module \$CC4Z9) which you will use for checking the validity of the terminal's sign on data.

The last entry in a Relocatable Address Table must be followed by a one-byte constant of hex FF (statement 1201).

The Relocatable Address Table is used only at the beginning of a CCP run to establish the addresses used in your transient. Therefore, it does not have to be present in main storage during each execution of your transient. Because of this, the table may extend beyond the 512 bytes which is the maximum size of the executable portion of a transient.

- The first user transient to which control will be passed must be named \$CC4YA. The first transient, (\$CC4YA), may pass control to other user written transients, providing such additional transients are written and named according to established conventions. Any user-written transient receiving control after \$CC4YA must be named such that the 1st five characters of the name are \$CC4Y. The sixth character of the name may be any character in the range B-Z.
- The last transient to be called, (or \$CC4YA if it is the only one used), must transfer control to the transient area scheduler after setting the desired parameter value in location "TAXPRM". The transfer of control is performed by branching to the CCP routine \$CC4TX. Reference sample program statements 1160, 1167, and 1176.
- The security data which was previously written in the modules \$CC4Z9, will be available for reference by any user written transient in a user security work area. The leftmost byte of this work area will contain the first byte from the module \$CC4Z9. The size of the work area and the size of \$CC4Z9 is specified at Generation by the operand "LUSI" in the \$ESEC statement. The address of the leftmost byte of this work area is available to the user's transient by the value "P + USI". See the sample program, statement 1129.

## Sample Program – Model 10 or Model 12

Figure 7-2 is an example of a user sign-on security routine. The following notes explain the logic of the example:

- A total of six macros must be included in the source code. These macros produce equates used elsewhere in the program. Their names are as follows:

1. \$EEQU
2. \$ECOM
3. \$ECPL
4. \$ETUB
5. \$ETNT
6. \$ETRC

These macros are available only on the CCP distribution disk cartridge (PID001). Either copy these to your own source library or perform the macro processing step by loading \$MPXDV from the distribution pack.

- This sample program expects sign-on data in the form /ON~~W~~SSSSSS, where SSSSSS is the security code to be checked by SCC4YA.
- As used by this program, the security data found in SCC4Z9 must be of the following form:

C<sub>1</sub>C<sub>1</sub>C<sub>1</sub>C<sub>1</sub>C<sub>1</sub>C<sub>1</sub>S<sub>1</sub>S<sub>1</sub>S<sub>1</sub>S<sub>1</sub>S<sub>1</sub>S<sub>1</sub>C<sub>2</sub>C<sub>2</sub>C<sub>2</sub>C<sub>2</sub>C<sub>2</sub>C<sub>2</sub>S<sub>2</sub>S<sub>2</sub>S<sub>2</sub>S<sub>2</sub>S<sub>2</sub>S<sub>2</sub>...00

CC ... represents a six-character symbolic terminal name and SS ... represents a six-position security field. There should be as many sets of CC ... SS ... data residing in SCC4Z9 as there are terminals to be used by an assignment set for CCP. The intent of this data is to provide a vehicle by which SCC4YA can verify a unique six-character sign-on code for each terminal so specified by the symbolic terminal name.

- Statement 1129 loads XR2 with the address of the left-most byte of the security data residing in the security data work area. Statement 1151 compares the data entered from the terminal after the 'ON', and the data in the security data work area that is associated with one symbolic terminal name.
- Statement 1140 compares the symbolic terminal name to the names in the security data area. The program loops between statements 1137 and 1144 and is exited when the end of the security data area is reached (1139), or when the name of the terminal signing on compares equal to one in the security data area (1142).
- Statement 1160 puts '01' in symbolic location 'TAXPRM'. This value indicates that the security data for the selected terminal verified correctly. After control is returned to SCC4TX and given to the CCP transient SCC4SO, a message will be sent to the selected terminal which will indicate that the sign on was successful.
- Statement 1167 puts 'FF' in symbolic location 'TAXPRM'. This value indicates that the security data for the selected terminal did *not* verify correctly. After control is returned to SCC4TX and given to the CCP transient SCC4SO, a message will be sent to the selected terminal which will say that the sign on was unsuccessful.
- Statement 1176 causes the control to be passed to the CCP transient SCC4SO. This is accomplished as follows:
  - a. The field TAXTID (at relative location 6 in the transient) contains the displacement from the beginning of the transient of the first three bytes (location 0A-0C of the transient) which address the transient SCC4SO. At CCP startup, the bytes at locations 0B and 0C are set to the disk address of that transient.
  - b. The branch at statement 1176 gives control to the CCP Transient Area Scheduler, requesting it to pass control to the transient (SCC4SO) pointed to by the TAXTID field.

\$CC4YA

ERR	LOC	OBJECT CODE	ADDR	STMT	SOURCE	STATEMENT	
			1072	*		CCP TRANSIENT ROUTINE EQUATES	
			0001	1074	TARLD@	EQU 1	OFFSET TO TRANSIENT
				1075	*		CCP RELOCATION CONSTANTS
			0001	1076	TAJUMP	EQU TARLD@	DISPLACEMENT FOR JUMP OP + Q
			0003	1077	TAID	EQU TARLD@+2	OFFSET TO TRANSIENT PROGRAM
			0005	1078	TAXPRM	EQU TAID+2	DISPLACEMENT TO PARM BYTES
			0007	1079	TAXTID	EQU TAXPRM+2	OFFSET TO NCS VALUE@
			0008	1080	TAXCID	EQU TAXTID+1	OFFSET TO TRANSIENT XCTL TABLE
			000A	1081	TAXNCS	EQU TAXCID+2	OFFSET TO FIRST NCS PARAMETER
			0005	1082	TAXCLN	EQU 5	LENGTH OF AN XCTL TABLE ENTRY
			1084	*		TRANSIENT PROLOG	
			0000	1086	USING	\$CC4YA,XR1	DEFINE BASE REGISTER
0000	0084		0001	1087	DC	AL2(YA@RLC-\$CC4YA)	DISPL OF CCP RELOCATE ADCONS
				1088	*		NOTE, THESE BYTES ARE OVERLAID
				1089	*		WITH A JUMP OP CODE AND X'87'
				1090	*		Q-BYTE BY \$CC4TA
0002	0E		0002	1091	DC	AL1(YAGO-\$CC4YA-3)	OFFSET TO FIRST INSTRUCTION
0003	01		0003	1092	DC	XL1'1'	TRANSIENT AREA ID, USED BY
				1093	*		TRANSIENT RELOCATION ROUTINE
0004	0000		0005	1094	DC	XL2'0'	PARAMETER BYTES
0006	000A		0007	1095	DC	AL2(YASD-\$CC4YA)	DISPLACEMENT TO FIRST XCTLEE
0008	E2D6		0009	1097	DC	CL2'S0'	SPECIFIES XCTL TO \$CC4SD
000A	01		000A	1098	YASD	DC AL1(1)	N BYTE FOR DISK
000B	C3E2		000C	1099	DC	CL2'CS'	DISK C/S BYTES, SET BY CCP START
000D	5B		000D	1100	DC	CL1'\$'	END OF XCTL TABLES FIELD
000E	E8C1		000F	1101	DC	CL2'\$A'	\$CC4YA EYECATCHER CONSTANT
0010	03		0010	1102	DC	XL1'03'	LEVEL NUMBER
			0011	1103	YAGO	EQU *	BEGINNING OF EXECUTABLE CODE
			1105	*		CHECK THE LENGTH OF THE TERMINAL MESSAGE AND IF CORRECT	
			1106	*		THEN MOVE IT TO THE TRANSIENT DATA AREA.	
0011	35 02 003B		1108	##0010	L	C+@TUSTG,XR2	LOAD POINTER TO TUB
0015	74 02 83		1109	ST	YATUB(,XR1),XR2	SAVE TUB POINTER	
0018	6C 01 7B 09		1110	MVC	YAENDP(2,XR1),TPRECA(,XR2)	SAVE BEGINNING @ OF MSG.	
001C	BD 0A 05		1111	CLI	TPEFFL(,XR2),10	TEST FOR CORRECT INPUT LENGTH	
001F	F2 01 41		1112	JNE	YAFRR	JUMP IF NOT	
0022	6E 01 7B 05		1114	ALC	YAENDP(2,XR1),TPEFFL(,XR2)	ADD LENGTH OF MESSAGE	
0026	5F 01 7B 6F		1115	SLC	YAENDP(2,XR1),YAXONE(,XR1)	DECREMENT POINTER	
002A	75 02 7B		1116	L	YAENDP(,XR1),XR2	XR2 POINTS TO END OF INPUT	
002D	6C 09 79 00		1117	MVC	YAENDD(10,XR1),0(,XR2)	MOVE SIGN ON DATA TO TRANSIENT	
			1119	*		MOVE THE 6 CHARACTER NAME OF THE TERMINAL ATTEMPTING	
			1120	*		TO SIGN ON INTO AN AREA WITHIN THIS TRANSIENT.	
0031	75 02 83		1122	L	YATUB(,XR1),XR2	LOAD TUB POINTER	
0034	85 02 21		1123	L	TUBTNT(,XR2),XR2	LOAD ADDRESS OF TERMINAL NAME T.	
0037	6C 05 81 05		1124	MVC	YANAME(6,XR1),TNTNAM(,XR2)	MOVE TERMINAL NAME	
			1126	*		LOAD XR2 WITH THE ADDRESS OF THE LEFTMOST BYTE OF THE	
			1127	*		SECURITY DATA FOUND IN THE SECURITY DATA WORKAREA.	
0038	C2 02 001D		1129	##0020	LA	P+US1,XR2	LOAD POINTER TO SECURITY DATA
003F	E2 02 05		1130	LA	5(,XR2),XR2	BUMP POINTER	
			1132	*		COMPARE THE 6 BYTE TERMINAL NAME PREVIOUSLY SAVED AND THE	
			1133	*		6 BYTE TERMINAL NAMES FOUND IN THE SECURITY DATA WORKAREA.	
			1134	*		KEEP COMPARING TILL A MATCH IS FOUND OR TILL THE END OF	
			1135	*		THE SECURITY DATA AREA IS REACHED	

Figure 7-2 (Part 1 of 2). Sample User Security Routine for Models 10 and 12



```

$CC4YA
ERR LOC OBJECT CODE ADDR STMT SOURCE STATEMENT
0042 BD 00 00 0042 1137 YA0100 EQU *
0045 F2 81 18 1138 CLI 0(,XR2),0 TEST FOR END OF DATA
0048 6D 05 81 00 1139 JE YAERR JUMP IF END HAS BEEN REACHED
1140 CLC YANAME(6,XR1),0(,XR2) COMPARE NAME ENTERED & NAME IN
1141 * SECURITY DATA AREA
004C F2 81 07 1142 JE YA0110 JUMP IF A MATCH IS FOUND
004F E2 02 0C 1143 LA 12(,XR2),XR2 STEP POINTER TO NEXT NAME FIELD
0052 C0 87 0042 1144 ##0030 B YA0100 TO TEST NEXT NAME

1146 * COME HERE WHEN A POSITIVE MATCH HAS BEEN MADE BETWEEN THE
1147 * TERMINAL NAME AND THE NAME IN THE SECURITY DATA AREA.
1148 * NOW COMPARE THE SIGN ON DATA ASSOCIATED WITH THE TERMINAL NAME.

0056 9D 05 06 79 0056 1150 YA0110 EQU *
1151 CLC 6(6,XR2),YAENDD(,XR1) COMPARE SIGN ON CODE FOR
1152 * TERMINAL WHO'S NAME WAS JUST
1153 * FOUND AND CODE IN SECURITY
1154 * DATA FIELD
005A F2 01 06 1155 JNE YAERR JUMP TO ERROR EXIT IF CODE DOES
1156 * NOT COMPARE

1158 * THE SIGN ON DATA HAS COMPARED OK, SO PUT A X'01' IN 'TAXPRM'.

005D 7C 01 05 1160 MVI TAXPRM(,XR1),YAEQU1 PUT '01' IN PARAMETER
0060 F2 87 03 1161 J YAOUT TO EXIT FROM THIS TRANSIENT

1163 * THE SIGN ON DATA DID NOT COMPARE CORRECTLY,
1164 * SO PUT A X'FF' IN 'TAXPRM'.

0063 7C FF 05 0063 1166 YAERR EQU *
1167 MVI TAXPRM(,XR1),YAEQUF PUT 'FF' IN PARAMETER
1168 * THIS SIGNIFIES THAT THERE WAS
1169 * AN ERROR IN THE SIGN ON
1170 * VALIDATION .

1172 * COME HERE TO EXIT FROM THIS TRANSIENT

0066 3A C0 0500 0066 1174 YAOUT EQU *
006A C0 97 0009 1175 ##0260 SBN C+$CPWK+$CPFLG,$C.PII+$CPRF
1176 ##0270 BC P+##CC4TX,RR97 TO XCTL

1178 * CONSTANTS AND DATA AREAS

006E 0031 006E 1180 YAXONE DC XL2'01' CONSTANT
0070 0000000000000000 0079 1181 YAENDD DC XL10'0' INPUT DATA GOES HERE
0078 0000 1181 LENGTH OF MESSAGE
007A 0000 0078 1182 YAENDP DC XL2'0' TERMINAL NAME GOES HERE
007C 000000000000 0081 1183 YANAME DC XL6'0' SAVE AREA FOR TUB POINTER
0082 0000 0083 1184 YATUB DC XL2'0'

00FF 1186 YAEQUF EQU X'FF'
0001 1187 YAEQU1 EQU 1
8000 1188 PGM EQU X'8000'
C000 1189 COM EQU X'C000'
0000 1190 C EQU $CC4YA
0000 1191 P EQU $CC4YA

1193 * RELOCATION ADDRESS TABLE

0084 1195 YA@RLC EQU *
0084 C038 0085 1196 @@0010 DC AL2(COM+@TUSTG)
0086 801D 0087 1197 @@0020 DC AL2(PGM+UST)
0088 0042 0089 1198 @@0030 DC AL2(YA0100-P)
008A C500 008B 1199 @@0260 DC AL2(COM+$CPWK+$CPFLG)
008C 8009 008D 1200 @@0270 DC AL2(PGM+##CC4TX)
008E FF 008E 1201 DC XL1'FF' STOPPER BYTE
0000 1202 END $CC4YA

TOTAL STATEMENTS IN ERROR IN THIS ASSEMBLY = 0

```

Figure 7-2 (Part 2 of 2). Sample User Security Routine for Models 10 and 12

## PROGRAMMING A USER SECURITY ROUTINE – MODEL 15

The user of CCP may, if he chooses, write his own terminal sign on security routines rather than use the CCP password facility supplied with his system. To implement and use such routines, the following must be considered.

- This option must be selected at generation time by the "SECURE-USER" operand of the \$ESEC statement.
- The security data which will be used at terminal sign on time by the security routines must have been previously written to the security module \$CC4Z9 by User Security Data Program, \$CCPAU (see *CCP System Reference Manual*).
- The user security routines must be written in Basic Assembler Language (or an equivalent machine-level language) and must be structured such that they comprise three basic parts (statement numbers refer to the sample program, Figure 7-3):
  1. System equates. They provide the necessary offsets and pointers to various system tables, as well as other common equates.
  2. The transient prologue and equates. The function of this code is to provide the following:
    - a. JUMP instruction to the start of executable code (statement 8).
    - b. A two-byte field of hex 0000 to permit the passing of a parameter from this transient to another (statement 10).
    - c. A two-byte address pointing to the third character past the ID of the transient to be called next (statement 11).
    - d. The two-character ID of the transient to be called next, consisting of the last two characters of the module name (statement 12).
    - e. Three bytes for disk control which are used when this transient is read from disk (statement 12). A one-character delimiter (\$) and the two-byte ID of this transient follow (statement 13).

The macro \$EBEG provides this code. The following \$EBEG parameters are required:

N-YA This is the module ID. The last two characters identify the module as a user security module.

X1-SO This is the ID of the sign on transient. It will be called next.

TID-1 This indicates that this transient runs in CCP transient area 1.

US-Y This indicates that this is a transient written by the user.

3. The main body of code, which in this example begins with statement 836 and ends with statement 909. This sample code will: determine the length of, and move the sign-on data to an area within this transient; determine which terminal is attempting to sign on and then verify the sign-on data for that terminal, as found in the user security work area; pass the result of the verification to the CCP sign-on transient, \$CC4SO, by setting a value in "TAXPRM" prior to exiting from this transient. The "TAXPRM" field of any CCP transient (at relative locations 4 and 5 of the transient) is the field in which one transient passes information to another. The total length of parts 2 and 3 cannot exceed hex length 1FF.
- The first user transient to which control will be passed must be named \$CC4YA. The first transient, (\$CC4YA), may pass control to other user written transients, providing such additional transients are written and named according to established conventions. Any user-written transient receiving control after \$CC4YA must be named such that the 1st five characters of the name are \$CC4Y. The sixth character of the name may be any character in the range B-Z.
  - The last transient to be called, (or \$CC4YA if it is the only one used), must transfer control to the transient area scheduler after setting the desired parameter value in location "TAXPRM". The transfer of control is performed by branching to the CCP routine \$CC4TX. Reference sample program statements 883, 890, and 898.
  - The security data which was previously written in the modules \$CC4Z9 will be available for reference by any user-written transient in a user security work area. The leftmost byte of this work area will contain the first byte from the module \$CC4Z9. The size of the work area and the size of \$CC4Z9 is specified at Generation by the operand "LUSI" in the \$ESEC statement. The address of the leftmost byte of this work area is available to the user's transient by the value @USECW. See the sample program, statement 852.

## Sample Program – Model 15

Figure 7-3 is an example of a user sign-on security routine. The following notes explain the logic of the example:

- A total of five macros must be included in the source code. These macros produce equates used elsewhere in the program. Their names are as follows:

1. \$EEQU
2. \$ECOM
3. \$ECPL
4. \$ETUB
5. \$ETNT

These macros are available only on the CCP distribution disk cartridge (PID001). Either copy these to your own source library or perform the macro processing step by loading \$MPXDV from the distribution pack.

- This sample program expects sign-on data in the form /ON#SSSSSS, where SSSSSS is the security code to be checked by \$CC4YA.
- As used by this program, the security data found in \$CC4Z9 must be of the following form:

C<sub>1</sub> C<sub>1</sub> C<sub>1</sub> C<sub>1</sub> C<sub>1</sub> C<sub>1</sub> S<sub>1</sub> S<sub>1</sub> S<sub>1</sub> S<sub>1</sub> S<sub>1</sub> S<sub>1</sub> C<sub>2</sub> C<sub>2</sub> C<sub>2</sub> C<sub>2</sub> C<sub>2</sub> C<sub>2</sub> S<sub>2</sub> S<sub>2</sub> S<sub>2</sub> S<sub>2</sub> S<sub>2</sub> S<sub>2</sub> . . . 00

CC . . . represents a six-character symbolic terminal name and SS . . . represents a six-position security field. There should be as many sets of CC . . . SS . . . data residing in \$CC4Z9 as there are terminals to be used by an assignment set for CCP. The intent of this data is to provide a vehicle by which \$CC4YA can verify a unique six-character sign-on code for each terminal so specified by the symbolic terminal name.

- Statement 852 loads XR2 with the address of the left-most byte of the security data residing in the security data work area. Statement 874 compares the data entered from the terminal after the 'ON', and the data in the security data work area that is associated with one symbolic terminal name.
- Statement 863 compares the symbolic terminal name to the names in the security data area. The program loops between statements 860 and 867 and is exited when the end of the security data area is reached (862), or when the name of the terminal signing on compares equal to one in the security data area (865).
- Statement 883 puts '01' in symbolic location 'TAXPRM'. This value indicates that the security data for the selected terminal verified correctly. After control is returned to \$CC4TX and given to the CCP transient \$CC4SO, a message will be sent to the selected terminal which will indicate that the sign on was successful.
- Statement 890 puts 'FF' in symbolic location 'TAXPRM'. This value indicates that the security data for the selected terminal did *not* verify correctly. After control is returned to \$CC4TX and given to the CCP transient \$CC4SO, a message will be sent to the selected terminal which will say that the sign on was unsuccessful.
- Statement 898 causes the control to be passed to the CCP transient \$CC4SO. This is accomplished as follows:
  - a. The field TAXTID (at relative location 6 in the transient) contains the displacement from the beginning of the transient of the first three bytes (location 0A-0C of the transient) which address the transient \$CC4SO. At CCP startup, the bytes at locations 0B and 0C are set to the disk address of that transient.
  - b. The branch at statement 898 gives control to the CCP Transient Area Scheduler, requesting it to pass control to the transient (\$CC4SO) pointed to by the TAXTID field.
- Statement 908 extends the end of the module to a sector boundary to provide maintenance space.

ERR	LOC	OBJECT	CODE	ADDR	STMT	SOURCE	STATEMENT
					2	*	\$EBEG N-YA,X1-SO,TID-1,US-Y
	4000				3	\$CC4YA	START 16384
				4000	4	\$CC4\$\$	EQU * TRANSIENT START DEFINITION
					5	*	LEVEL 02
					6	*	RLD N
					7		
	4000	F2	87	0D	8	J	YAGO
					9		
	4003	0000			10	TAXPRM	DC XL2*0000'
	4005	4008			11	TAXTID	DC AL2(YASO)
	4007	E2D64040	40		12	YASO	DC CL5'SO'
	400C	58EBC1			13	DC	CL3'\$YA'
	400F	02			14	DC	XL1'02'
				4010	15	YAGO	EQU * END OF \$EBEG CONSTANTS
					833	*	COME CHECK THE LENGTH OF THE TERMINAL MESSAGE AND IF CORRECT
					834	*	THEN MOVE IT TO THE TRANSIENT DATA AREA.
					835		
	4010	0D	00	44E6	405C	CLC	\$CPEFL,YATEN
	4016	F2	01	3B	837	JNE	YAERR
					838		
	4019	35	02	44E8	839	L	\$CPRCA,XR2
	401D	6C	05	62	09	MVC	YAENDD(6,XR1),9(,XR2)
					840		
					841		
					842	*	MOVE THE 6 CHARACTER NAME OF THE TERMINAL ATTEMPTING
					843	*	TO SIGN ON INTO AN AREA WITHIN THIS TRANSIENT.
					844		
	4021	35	02	4467	845	L	@TUSTG,XR2
	4025	B5	02	17	846	L	TUBTNT(,XR2),XR2
	4028	6C	05	68	05	MVC	YANAME(6,XR1),TNTNAM(,XR2)
					847		
					848		
					849	*	LOAD XR2 WITH THE ADDRESS OF THE LEFTMOST BYTE OF THE
					850	*	SECURITY DATA FOUND IN THE SECURITY DATA WORKAREA.
					851		

Figure 7-3 (Part 1 of 2). Sample User Security Routine for Model 15

\$CC4YA

```

ERR LOC OBJECT CODE      ADDR STMT SOURCE STATEMENT
402C 35 02 4411         852      L    @USECW,XR2          LOAD POINTER TO SECURITY DATA
4030 E2 02 05          853      LA   5(,XR2),XR2        BUMP POINTER
                        854
                        855 *
                        856 *   COMPARE THE 6 BYTE TERMINAL NAME PREVIOUSLY SAVED AND THE
                        857 *   6 BYTE TERMINAL NAMES FOUND IN THE SECURITY DATA WORKAREA.
                        858 *   KEEP COMPARING TILL A MATCH IS FOUND OR TILL THE END OF
                        859 *   THE SECURITY DATA AREA IS REACHED.
4033 8D 00 00          4033 860 YA0100 EQU    *
4036 F2 81 1B          861      CLI  0(,XR2),0          TEST FOT END OF DATA
4039 6D 05 68 00       862      JE   YAERR             JUMP IF END HAS BEEN REACHED
                        863      CLC  YAAME(6,XR1),0(,XR2)  COMPARE NAME ENTERED & NAME IN
                        864 *   SECURITY DATA AREA
403D F2 81 07          865      JE   YA0110           JUMP IF A MATCH IS FOUND
4040 E2 02 0C          866      LA   12(,XR2),XR2     STEP POINTER TO NEXT NAME FIELD
4043 C0 87 4033        867      B    YA0100           TO TEST THE NEXT NAME
                        868
                        869 *
                        870 *   COME HERE WHEN A POSITIVE MATCH HAS BEEN MADE BETWEEN THE
                        871 *   TERMINAL NAME AND THE NAME IN THE SECURITY DATA AREA.
                        872 *   NOW COMPARE THE SIGN ON DATA ASSOCIATED WITH THE TERMINAL NAME.
4047 9D 05 06 62       4047 873 YA0110 EQU    *
                        874      CLC  6(6,XR2),YAENDD(,XR1)  COMPARE SIGN ON CODE FOR
                        875 *   TERMINAL WHO'S NAME JUST
                        876 *   FOUND AND CODE IN SECURITY
                        877 *   DATA FIELD
4048 F2 01 06          878      JNE  YAERR             JUMP TO ERROR EXIT IF CODE DOES
                        879 *   NOT COMPARE
                        880
                        881 *   THE SIGN ON DATA HAS COMPARED OK, SO PUT A X'01' IN 'TAXPRM'.
                        882
404E 7C 01 04          883      MVI  TAXPRM(,XR1),YAEQU1  PUT '01' IN PARAMETER
4051 F2 87 03          884      J    YAOUT             TO EXIT FROM THIS TRANAIENT
                        885
                        886 *   THE SIGN ON DATA DID NOT COMPARE CORRECTLY,
                        887 *   SO PUT A X'FF' IN 'TAXPRM'.
                        888
4054 7C FF 04          4054 889 YAERR EQU    *
                        890      MVI  TAXPRM(,XR1),YAEQUF  PUT 'FF' IN PARAMETER
                        891 *   THIS SIGNIFIES THAT THERE WAS
                        892 *   AN ERROR IN THE SIGN ON
                        893 *   VALIDATION.
                        894
                        895 *   COME HERE TO EXIT FROM THIS TRANSIENT
                        896
4057 35 10 4403        4057 897 YAOUT EQU    *
                        898      L    @CC4TX,IAR          TO XCTL
                        899
                        900 *   CONSTANTS AND DATA AREAS
                        901
4058 000A              405C 902 YATEN DC    XL2'000A'
405D 00000000000000    4062 903 YAENDD DC   XL6'0'
4063 00000000000000    4068 904 YANAME DC   XL6'0'
                        905
00FF 906 YAEQUF EQU    X'FF'
0001 907 YAEQU1 EQU    1
41FC                    908      ORG  $CC4YA+508
                        4000 909      END    $CC4YA

```

*Note:* The following names, used in this figure, are defined in macros; the macros are not shown in this figure:

<i>Name</i>	<i>Macro Defined in</i>
\$CPEFL	\$ECOM
\$CPRCA	\$ECOM
@TUSTG	\$ECOM
@CC4TX	\$ECOM
TUBTNT	\$ETUB
TNTNAM	\$ETNT
@USECW	\$ECOM

Figure 7-3 (Part 2 of 2). Sample User Security Routine for Model 15



## Chapter 8: 3270 Display Format Facility (DFF)

### GENERAL INFORMATION

The 3270 Display Format Facility is an optional feature of the Communications Control Program (CCP). This facility allows programs to control printer and display formats on the IBM 3270 Information Display System with a minimum of coding. Instead of requiring a special assembler language subroutine, programs can interface with the 3270 in a high level language (RPG II, COBOL, or FORTRAN IV). For those programs written in assembler language, CCP provides interfacing macros that can be used with DFF (see Chapter 7. *Basic Assembler Programming for CCP*).

### Overview

The 3270 Display Format Facility offers:

- An easy way to define printer and display formats (use special form for offline generation).
- Formats that are program independent (stored on disk, not in the application program).
- Formats that can be used by several programs concurrently (the format remains unmodified on disk, therefore can be used by several programs).
- An easy way to handle data (all data is arranged in fields).
- Two ways of supplying data—at generation time (as part of the field definition) or during execution time (by the application program).
- The capability to change the characteristics or contents of a field on the display screen.
- Testing formats offline before writing the application program.
- Testing a terminal without writing a teleprocessing program.

## Prerequisite Information

In order to fully utilize DFF, it is essential that you have a basic understanding of the concepts and operation of the IBM 3270 Display system. This includes understanding the uses of:

- Write Control Character (WCC)
- New Line, End of Message, and Forms Feed (NL,EM,FF)
- Selector Pen
- Operator Identification Card Reader
- Copy Control Character (CCC)
- Attention Identification (AID)
- Modified Data Tag (MDT)
- Keyboard keys and functions

For information on any of the above, refer to *IBM 3270 Information Display System Component Description*, GA27-2749.

## DFF Routines

The Display Format Facility provides four routines — the Display Format Generator, the Printer Format Generator, the Display Format Control, and the Display Format Test:

1. The Display Format Generator routine runs offline; that is, it does not run under control of CCP. The display format generated by this routine is used by the Display Format Control Routine (DFCR), which does run under CCP. The Display Format Generator can run in either level of a DPF system, and it can run while CCP is executing. If DFGR is running while CCP is executing in the other level, the format being processed cannot be placed on the pack from which CCP was loaded; if this is attempted, an EO F PP halt occurs.
2. The Printer Format Generator runs offline; that is, it does not run under control of CCP, although the printer format it generates is used by the Display Format Control Routine (DFCR), which does run under CCP. The Printer Format Generator can run in either level of a DPF system, and it can run while CCP is running. If PFGR is running while CCP is executing in the other level, the format being processed cannot be placed on the pack from which CCP was loaded; if this is attempted, an EO F PP halt occurs.
3. The Display Format Control routine helps the transfer of data to and from the 3270 terminal. This routine does run under control of CCP.
4. The Display Format test routine runs offline; that is, it does not run under control of CCP. This routine is used as a development tool for designing and testing printer and display formats.



**INPUT**

Two forms are used in preparing material for either the printer or the display format generators: the printer/display layout sheet, and either the printer format specifications sheet (for PFGR) or the display format specifications sheet (for DFGR). The following describes the procedure used to generate a format. Descriptions applicable only to a particular routine are contained solely within its column; descriptions applicable to both routines span both columns.

**Printer Format Generator**

Arrange the fields on the layout sheet just as they are to appear on the form or display.

If the format has more than 33 lines, more than one sheet is needed.

Use the completed layout sheet as a guide in completing the two parts of the format specifications sheet (control and field definition forms). Provide for each field the name, type, location, associated data, and other needed information.

Transfer the information from the specifications sheet to the system input device or to the source library using \$MAINT.

System input records or source library statements are read by the format generator routine.

The format generator routine:

- Produces a printout of the specifications forms.
- Analyzes specifications for errors.
- Logs diagnostic error messages.

The formats are cataloged into an object library under a user-designated name.

The format is built in two parts:

- Field descriptor table containing the symbolic field names and is used to identify those fields for which data must be supplied in the user's output record area, and to support overrides to fields within the format.
- Complete data stream including control characters.

The printed information includes:

- Information used by the application program:
  - Required output data and the order of output data fields that must be provided in the user's output record area
- Information used in preparing the CCP assignment set under which the application program will execute:
  - The decimal length of the field descriptor table
  - The decimal length of the output data stream

**Display Format Generator**

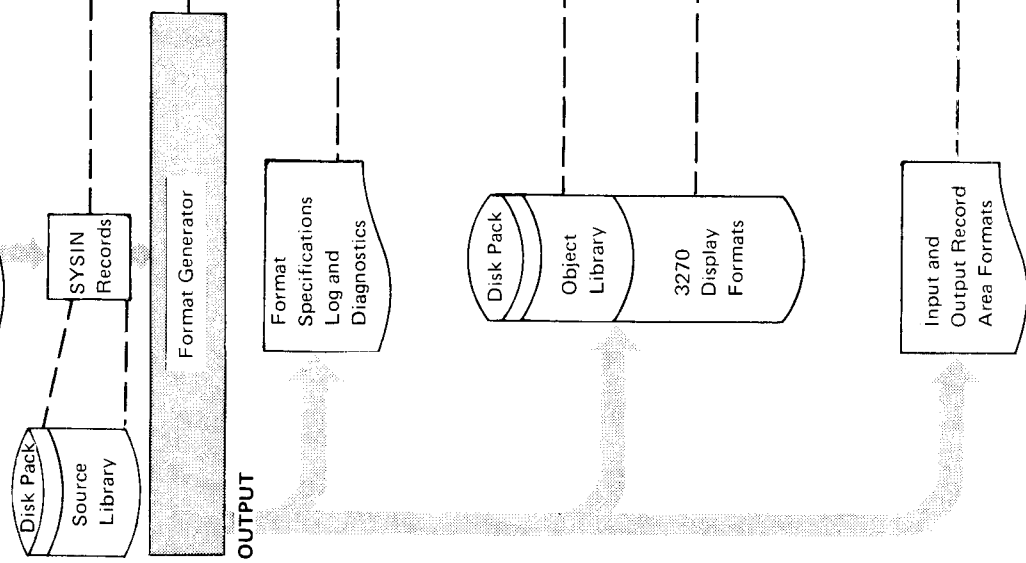
Include all fields that can be written to and/or received from the 3270.

The format generator routine:

- Produces a printout of the specification forms.
- Analyzes specifications for errors.
- Logs diagnostic error messages.

The printed information includes:

- Information used by the application program:
  - Required output data and the order of output data fields that must be provided in the user's output record area
  - Input data and the order of input data fields as they are passed to the application program in the user's input record area
- Information used in preparing the CCP assignment set under which the application program will execute:
  - The decimal length of the field descriptor table
  - The decimal length of the 3270 output data stream
  - The decimal length of the 3270 input data stream



Modifiers to existing operations and three additional operations (Copy, Put Override, and Erase) support the 3270 functions. The Copy operation is used to transfer a format between devices attached to the same 3271 control unit. The Put Override operation is used to (1) change the type, reposition the cursor, modify the data content (or any combination of these) of any field or fields defined in the format, and/or (2) request input from only selected fields within the display format. The Erase operation fills the data positions of all unprotected fields (input field types 1, 2, 3, 4 and output/input field types 1, 2, 3, 4) with null (hex 00) characters.

In all cases, additional entries in the CCP application program's output record area are required. For example, when requesting that a format be written to a 3270, the name of the format must be given in the record area. In the input record area, allowance must be made for the AID character.

Requests for display or printer format services use the same application interface as do other CCP requests; the application program passes a parameter list and a record area.

It is essential that the printer/display layout and specification sheets be completed accurately so that the control routine can operate exactly as intended. To do this, you need a working knowledge of such things as the *field concept* of handling data and the characteristics of each *type* of field within each of the *field classes*. You need to know how to determine *lengths of fields*, what *field attributes* are and how the *cursor* is positioned.

## FIELD CONCEPTS

### Definition

All 3270 display information is treated as a set of fields within a record rather than as a string of data. Each field has certain characteristics associated with it—such as whether it is alphameric or numeric. Field characteristics are defined by *type* and *class*.

*Class* describes whether the field is output, input, output/input, or selector pen detectable.

*Type* describes what group of characteristics are associated with the field.

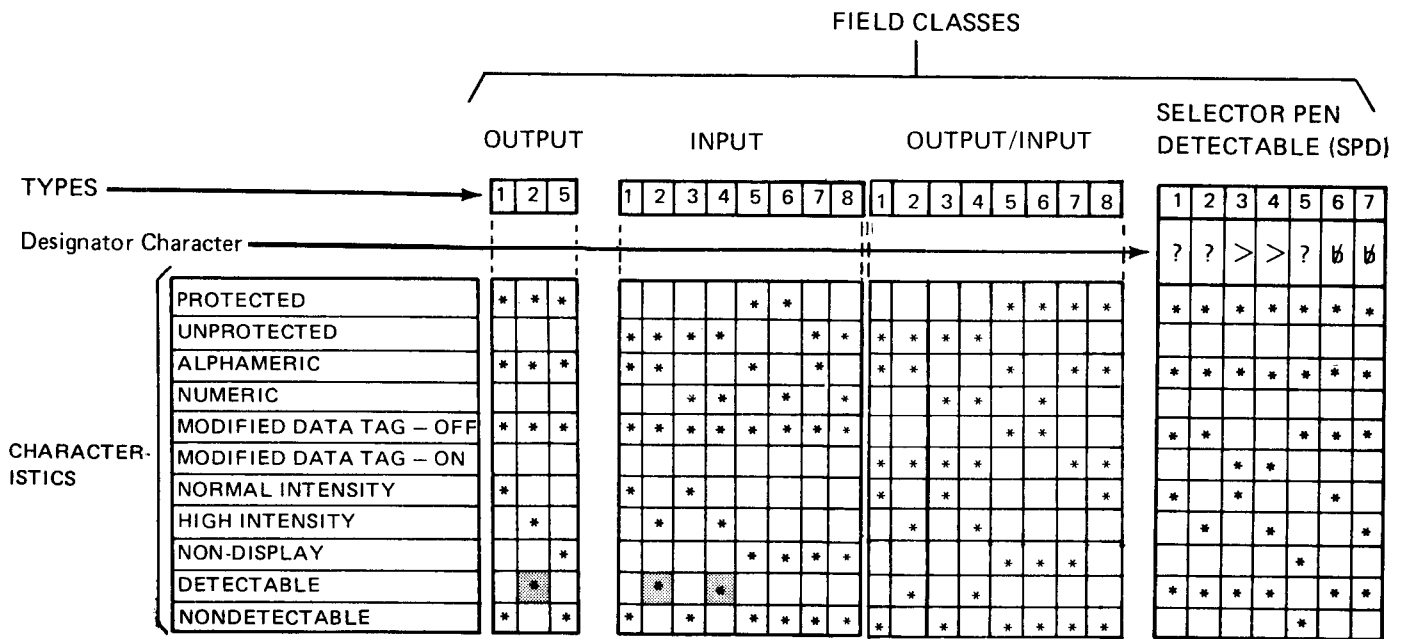
## Field Classes

The field class is selected at format generation time by the application programmer when using the Display Format Generator. When using the Printer Format Generator, the field class is assumed to be output. The field class determines what the terminal operator can or cannot do with that field (see Figure 8-1). For prompting or displaying messages to the terminal operator, an output class would be used. If data for a field is to be provided by the terminal operator, an input class would be used. If data for a field is to be initially provided by the application program but allowed to be changed (using the keyboard) by the terminal operator, an output/input class would be used. Lastly, a field class has been provided solely in support of the selector pen.

A description of the four field classes follows:

- Output field—contains data that cannot be changed by the terminal operator. The data for the field is supplied either during format generation or during execution of the application program.
- Input field—is an area reserved for keyboard entry; data for the field is entered by the terminal operator.
- Output/Input field—contains data which has been supplied either during format generation or during execution of the application program. This data can be changed by the terminal operator using the keyboard.
- Selector Pen Detectable (SPD) field—allows the terminal operator to select fields by using the selector pen.

All classes of fields may be used within the same display except when the SPD field is used as an attention field (see *Attention Fields*).



(This information also appears on the reverse side of the display layout sheet.)

*Note:* Shaded squares indicate that these fields are not intended to be used as detectable fields. To prevent accidental detection of such a field, users should avoid using one of the characters *b*, *?*, *>*, or null (X'00') as the first character in the field.

**Figure 8-1. Field Classes; Types Within Classes; and Characteristics Associated With Each Type**

The field class selected determines, to some extent, the characteristics of a field. Differences in characteristics by class are:

- *Protected or Unprotected* – A field is protected if the terminal operator cannot use the keyboard or operate the identification card reader to enter, modify, or erase data within that field. Output and selector pen detectable fields are protected. Input field types 5 and 6 and output/input field types 5-8 are protected.
- *Alphameric or Numeric* – All output and selector pen detectable fields are alphameric. Input and output/input fields may be either alphameric or numeric (see *Field Types*).
- *Modified Data Tag ON or OFF* – The modified data tag associated with each field causes the data from that field to be returned (MDT-ON) or not returned (MDT-OFF) during an input operation from the terminal. The

modified data tag may be set ON by (1) a keyboard input to the field, (2) a selector pen detection in the field, (3) a magnetic card read-in operation, (4) application program control, or (5) using the ERASE EOF key. The modified data tag can be set OFF by (1) a selector pen detection in the field, (2) application program control, or (3) use of the ERASE INPUT key.

When data for a particular field *is not received* from the terminal, blanks are passed to the application program.

When data for a field *is received* from the terminal, it is passed on to the application program.

Setting the modified data tag conditions what is received as input from the terminal and passed to the application program. Therefore, field classes affected by setting the modified data tag are: input, output/input, and selector pen detectable fields. For input fields, data will be received from the terminal only if the terminal operator has performed a keyboard input, ERASE EOF, or a magnetic card read-in operation. For output/input fields, data will normally be received from the terminal (see *Output/Input Field Types* for a discussion on output/input fields that have the MDT set to OFF, types 5 and 6). The terminal operator should not use the ERASE INPUT key with an output/input field. For selector pen detectable fields, data may or may not be received from the terminal, even though the field type selected (see *Field Types*) required that the modified data tag be set to ON initially. Detection by the terminal operator, on a selector pen detectable field can set the MDT from ON to OFF or from OFF to ON.

### Field Types

Within a field class, the field type selected also determines characteristics of a field (see Figure 8-1). The type is selected at format generation time by the application programmer when using the Display Format Generator. When the Printer Format Generator is used, the field type does not apply.

Choice of characteristics by type are:

- *Alphameric or Numeric* – Input and Output/input fields may be specified as either alphameric or numeric.

A field specified as numeric may be further defined concerning the decimal position. If the decimal position is specified, the numeric field is edited on input for decimal alignment and for removal of non-numeric characters.

- *Modified Data Tag ON or OFF* – An output/input field would normally have its modified data tag set to ON. For special use of an output/input field with its modified data tag set to OFF, refer to types 5 and 6 under *Output/Input Field Types*.

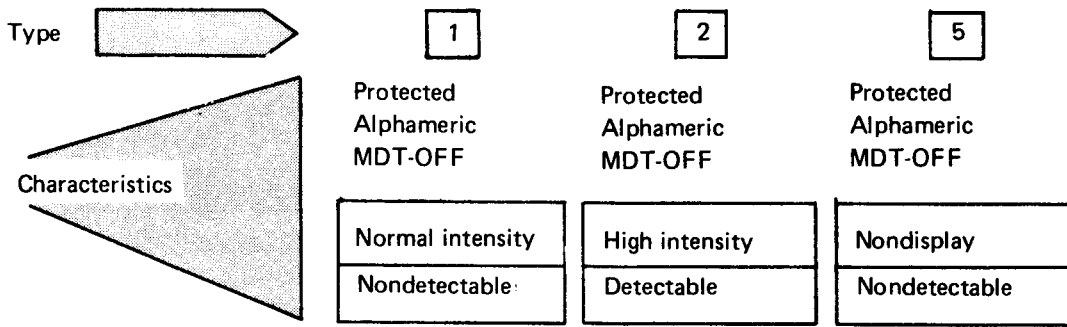
A selector pen detectable field may be specified with its modified data tag set to either ON or OFF. The initial setting can be changed by the terminal operator using the selector pen.

- *Intensity of Normal, High, or OFF (nondisplay)* – Nondisplay fields are not visible on the screen or printed on the printer. Field types are also provided which allow a field to be displayed with normal intensity or high intensity.

With the 3270 terminal, both the application programmer and the terminal operator should be aware that *all high-intensity fields, regardless of class*, have the potential of being detectable by the selector pen (see the note on Figure 8-1). To prevent an erroneous detection on a high-intensity non-SPD field, the first character in the field should not be a blank, null, question mark, or greater than character. This caution must be exercised by the application programmer when providing data for an output or output/input field class at format generation time or application program execution time. This condition can also be caused by the terminal operator when keying data or when using the cursor positioning keys.

If an erroneous detection on a high-intensity non-SPD field occurs, the interpretation of the data passed to the application program is the responsibility of the user (the designator character will not be separated from the data, and the data will be handled as usual for alphameric and numeric fields). Following is a breakdown of characteristics of each type under each class.

*Output Field Types*



All types of output fields have the characteristics of being protected, alphameric with the modified data tag set to OFF.

On input operations, data from output fields will not be passed to the application program.

The data for output fields is always handled as an alphameric value. Data will be displayed exactly as presented, whether supplied at format generation time or during execution time by the application program. If there are numeric values in the field, the application programmer is responsible for such editing of data as insertion of decimal points and other characters to properly display positive and negative quantities.

Type 1 (normal intensity) and type 2 (high intensity) output fields are used where data is to remain displayed. For example, headings and labels would appear on the screen and would not be subject to change by the terminal operator. Type 2 is indicated as being detectable only because it also has the high intensity characteristic; it is not intended that the selector pen be used. See the discussion of intensity under *Field Types*.

A type 5 field with the nondisplay characteristic can be changed using the override facility at execution time to become a type 1 or type 2 with the corresponding intensity characteristic. This is used to display appropriate error messages or comments as they are needed (see *Put Override*).

## Input Field Types

Type	1	2	3	4	5	6	7	8
Characteristics	Unprotected	Unprotected	Unprotected	Unprotected	Protected	Protected	Unprotected	Unprotected
	Alphameric	Alphameric	Numeric	Numeric	Alphameric	Numeric	Alphameric	Numeric
	MDT-OFF	MDT-OFF	MDT-OFF	MDT-OFF	MDT-OFF	MDT-OFF	MDT-OFF	MDT-OFF
	Normal intensity	High intensity	Normal intensity	High intensity	Non-display	Non-display	Non-display	Non-display
	Non-detectable	Detectable	Non-detectable	Detectable	Non-detectable	Non-detectable	Non-detectable	Non-detectable

All input fields have the modified data tag set to OFF.

Type 1 is the normal alphameric input field.

Type 2 allows for accenting an alphameric field (high intensity).

Type 3 is the normal numeric input field.

Type 4 allows for accenting a numeric field (high intensity).

A field specified as type 5 at format generation time can be changed using the override facility at execution time to become a type 1 with normal intensity or type 2 with high intensity.

A field specified as type 6 at format generation time can be changed using the override facility at execution time to become a type 3 with normal intensity or type 4 with high intensity. Note that type 5 is alphameric while type 6 is numeric.

Types 2 and 4 are indicated as being detectable only because they also have the high intensity characteristic; it is not intended that the selector pen be used. See the discussion of intensity under *Field Types*.

Type 7 input field is intended primarily for use as a security/authorization field. Data can be entered from the keyboard without displaying the data on the screen. Handling of the security/authorization data is left to the user. This field type may be changed using the override facility at execution time to a type 1, 2, or 5.

Type 8 input field is intended primarily for use as a security/authorization field. Data can be entered from the keyboard without displaying the data on the screen. Handling of the security/authorization data is left to the user. This field type may be changed using the override facility at execution time to a type 3, 4, or 6. Note that type 7 is alphameric while type 8 is numeric.

Types 3, 4, 6, and 8 may be specified with a decimal position for numeric input editing.

Output/Input Field Types

Type	1	2	3	4	5	6	7	8
Characteristics	Unprotected	Unprotected	Unprotected	Unprotected	Protected	Protected	Protected	Protected
	Alphameric	Alphameric	Numeric	Numeric	Alphameric	Numeric	Alphameric	Alphameric
	MDT-ON	MDT-ON	MDT-ON	MDT-ON	MDT-OFF	MDT-OFF	MDT-ON	MDT-ON
	Normal intensity	High intensity	Normal intensity	High intensity	Non-display	Non-display	Non-display	Normal intensity
	Non-detectable	Detectable	Non-detectable	Detectable	Non-detectable	Non-detectable	Non-detectable	Non-detectable

The word *Output* in the class designation indicates that the field contains data which has been supplied at generation time (see *Field Definition Form*) or by the application program at execution time. The *Input* indicates that the data may be changed by the terminal operator using the keyboard.

An example of using the output/input field would be in checking a "SHIP TO" address. The address is placed on the screen, is checked to see that it is still correct, is changed if necessary, and is returned to the application program.

Type 1 is the normal alphameric output/input field.

Type 2 allows for accenting an alphameric field in the display (high intensity).

Type 3 is the normal numeric output/input field.

Type 4 allows for accenting a numeric field (high intensity).

Types 2 and 4 are indicated as being detectable only because they also have the high intensity characteristic; it is not intended that the selector pen be used. See the discussion of intensity under *Field Types*.

A field specified as type 5 at format generation time can be changed using the override facility at execution time to become a type 1 with normal intensity or type 2 with high intensity.

A field specified as type 6 at format generation time can be changed using the override facility at execution time to become a type 3 with normal intensity or type 4 with high intensity. Note that type 5 is alphameric while type 6 is numeric.

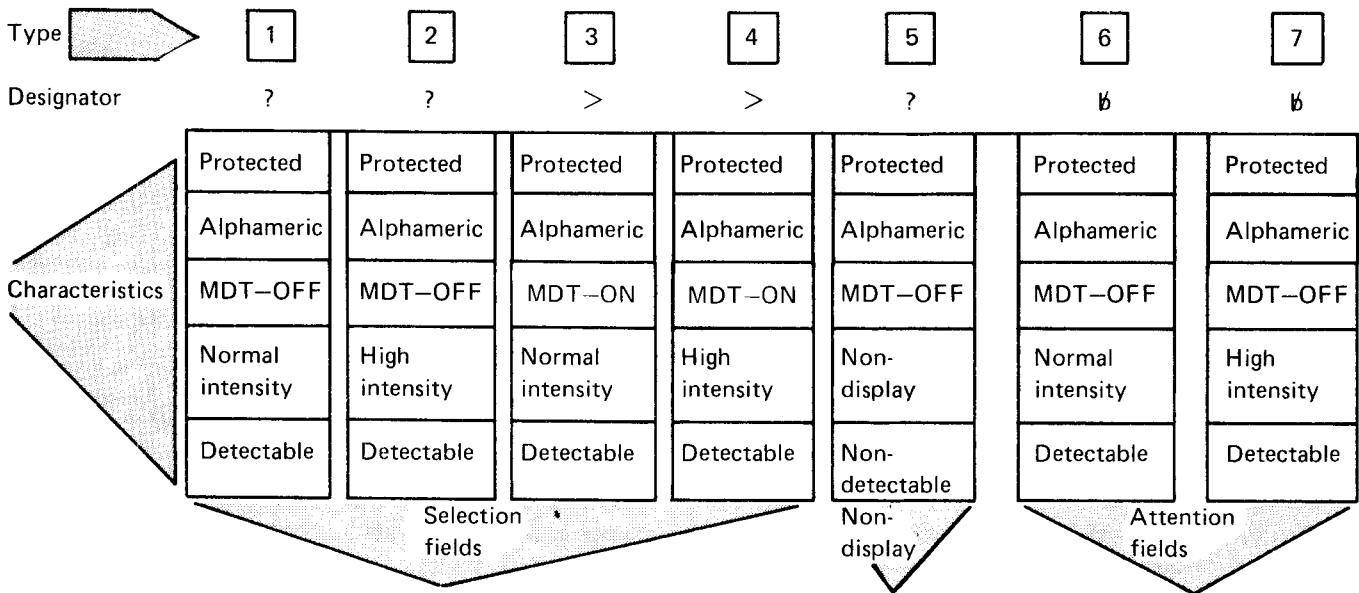
A field specified as type 7 or 8 at format generation time can be changed using the override facility at execution time to become a type 1 with normal intensity, a type 2 with high intensity, or a type 5 nondisplay. Note that types 1 and 2 have modified data tag on while type 5 has modified data tag off.

Type 3, 4, or 6 fields cannot be overridden to types 7 or 8. Also, type 7 and 8 fields are not interchangeable with types 3, 4, or 6 fields.

Types 3, 4, and 6, may be specified at input as being capable of input numeric editing.

*Note:* You will be unable to copy a display using the Copy operation if the field whose data begins in row 1 column 2 is a type 5, 7, or 8 output/input field.

*Selector Pen Detectable Field (SPD) Types*



As indicated, SPD fields are selection fields, nondisplay, or attention fields.

**Selection:** For fields specified as types 1, 2, 3, or 4 at format generation time, the designator character is a visible indicator on the screen and indicates whether the MDT is off or on. The designator is automatically assigned during generation of the display format based on type specified. Therefore, the user should not include the designator as part of the data for an SPD field. When sensed with a selector pen, the designator changes as shown below:

Designator	Selector Pen Detection Results
? (type 1)	Changes to > (type 3) and MDT set to ON
? (type 2)	Changes to > (type 4) and MDT set to ON
> (type 3)	Changes to ? (type 1) and MDT set to OFF
> (type 4)	Changes to ? (type 2) and MDT set to OFF

**Nondisplay:** Since type 5 is a nondetectable field, it cannot be changed by the use of the selector pen. A field specified as type 5 at format generation time can, however, be changed using the override facility at execution time. All SPD types are interchangeable by using the Put Overrides facility.







**Attention:** If an attention field (type 6 or 7) is present, *only output and SPD fields can be contained in the same display format.* Use of the selector pen to generate I/O pending will result in transmission of only the addresses of fields in which the modified data tag was set on (field data not included). Users who wish to combine SPD field input with keyed input must use the keyboard (ENTER or PF keys) to cause the data to be sent to the application program rather than using SPD attention.



## PLANNING THE PRINTER/DISPLAY LAYOUT

Before working with the printer/display layout sheet (Figure 8-2), the application programmer must be aware of the space requirements for each field class. The fields, as designed on the printer or display layout form, will have a direct correspondence with the information as it appears in the buffer of the 3270 screen or printer.

### LEGEND:

-  Defining attribute
-  Data field length
-  Terminating attribute
-  Starting location (first data position) of the field
-  Designator
-  Null character (X'00') appears as a blank space on the screen

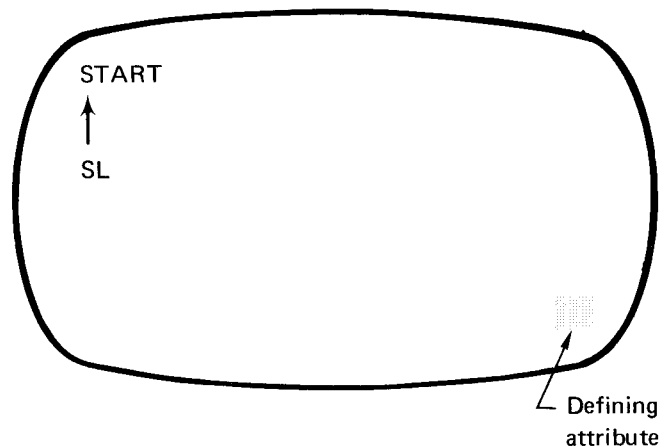
Items in the legend apply to the illustrations in the following discussion.

## Attributes

Attributes are generated only when the Display Format Generator is used; they are not generated when the Printer Format Generator is used.

The defining attribute is a nondisplay character located in the character position immediately preceding a data field and defines the characteristics of the display field that follows. The code for the attribute is developed by the format generation program based on field class and type. If data in an output, input, or output/input field must start at line 1, position 1, its defining attribute will be placed in the last display location (line 12, position 40 on Model 1 or line 24, position 80 on Model 2). Be sure this space is reserved when planning the rest of the display.

*Note:* When data for an output, input, or output/input field begins in line 1, position 1 of a display, that field is the *last* field transmitted in the 3270 text stream. The location of the field in the user program record area, however, corresponds to its location in the display.



The terminating attribute is a nondisplay character located in the character position immediately following a data field of an input or output/input class. This terminating attribute keeps the terminal operator from keying beyond the defined limits (data length) of the field and provides the function of autoskip/nonautoskip as defined at format generation time (see *Autoskip and Cursor Positioning*).

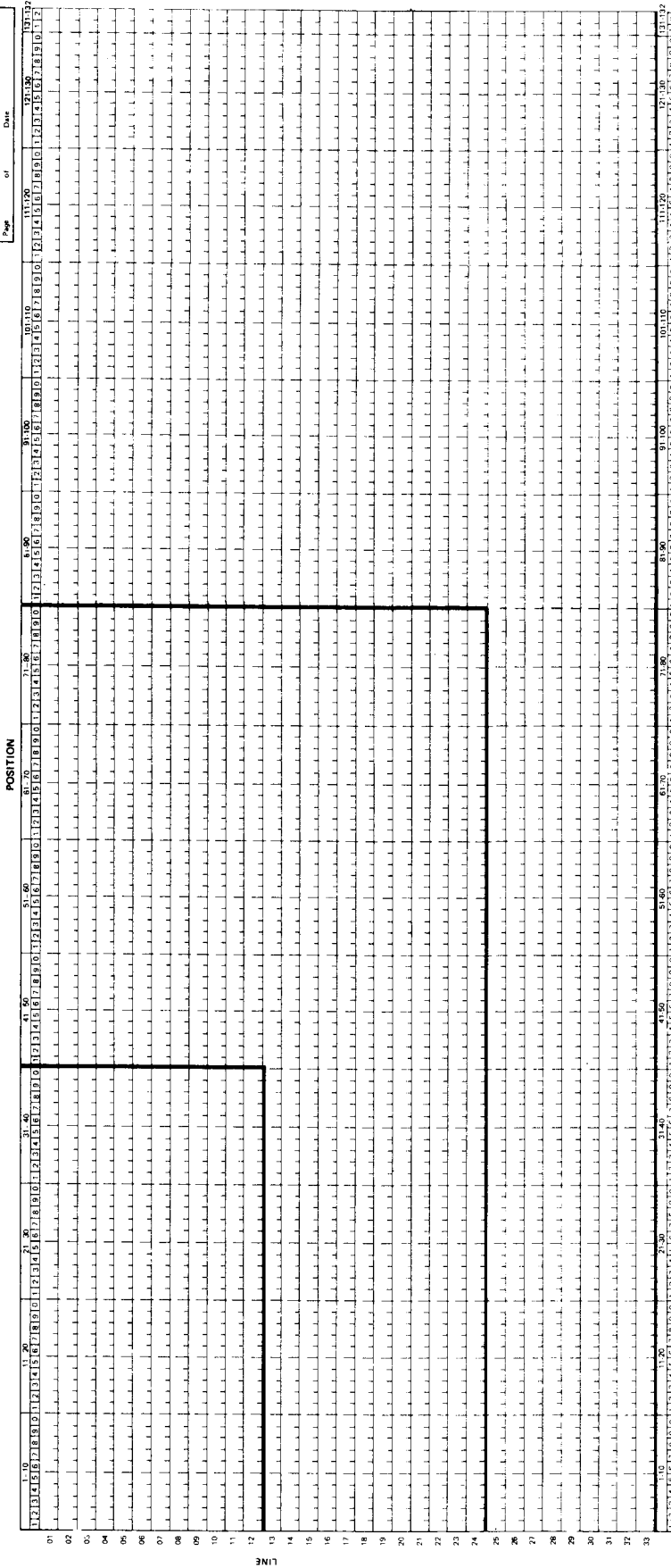


International Business Machines Corporation

Printer/Display Layout

GX21-9174 Printed in U.S.A.

Application \_\_\_\_\_  
 Completed by \_\_\_\_\_ of \_\_\_\_\_  
 Page \_\_\_\_\_ of \_\_\_\_\_ Date \_\_\_\_\_



KEY ASSIGNMENTS


AID		AID		AID		AID	
PF1	PF2	PF6	PF7	PF11	PF12	ENTER	ENTER


No. 31 sheets per page may vary slightly


## Output Class



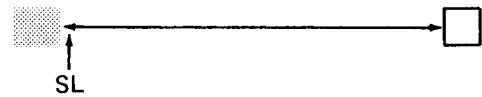
The total length of each output field on the printer/display layout sheet = data field length + one for the defining attribute when the Display Format Generator is used. When the Printer Format Generator is used, the total length of each field on the layout sheet equals the data field length only.

 The defining attribute determines the characteristics of the display field that follows. This attribute character is determined at format generation time based on the type as designated on the field definition entry. One position on the layout must be reserved for this nondisplayable character located immediately preceding the data field.


 SL The location of the first data position is entered in the field starting location on the field definition form. It must be the location immediately following the defining attribute character.


 Data field length (does not include defining attribute) is entered on the field definition form. Positions on the layout must be reserved equal to the data length.


## Input and Output/Input Classes




Total length on the printer/display layout sheet = field length + two for beginning and ending attributes (or field length + one if there is no allowance for a terminating attribute (see part three under *Autoskip and Cursor Positioning*)).

 The defining attribute determines the characteristics of the display field that follows. This attribute character is determined at format generation time based on the type as designated on the field definition entry. One position on the layout must be reserved for this nondisplayable character located immediately preceding the data field.

 SL The location of the first data position is entered in the field starting location on the field definition form. It must be the location immediately following the defining attribute character.

 The data field length is entered on the field definition form. Positions on the layout form must be reserved equal to the field length. The field length does not include either the defining or the terminating attribute.

 The terminating attribute limits the amount that can be keyed in. Based on autoskip/nonautoskip indication on the field definition form, this terminating attribute will reposition the cursor after a data character is entered into the last position of the field.

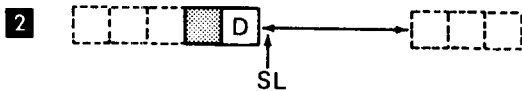
## SPD Class

For SPD fields, use the following guidelines for reserving space on the printer/display layout sheet.



SPD field is the first field on a line and not the last field on the line, or, the previous field on the line is SPD and this SPD field is not the last field on the line.

Total length allowed for this SPD field on the layout sheet is the field data length plus five.



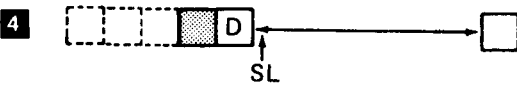
The preceding field on the line is another class and this SPD field is not the last field on the line.

Total length allowed for this SPD field on the layout sheet is the field data length plus eight.




The SPD field is the last field on the line and the previous field on the line is SPD, or, this SPD field is the only field on the line.


Total length allowed for this SPD field on the layout sheet is the field data length plus three.





The preceding field on the line is another class and this SPD field is the last field on the line.


Total length allowed for the SPD field on the layout sheet is the field data length plus six.


 The defining attribute determines the characteristics of the display field that follows. This attribute character is determined at format generation time, based on the type as designated on the field definition entry. One position on the layout must be reserved for this non-displayable character.

 The designator character must immediately follow the defining attribute character. The designator character is determined at format generation time, based on the field definition type entry. One position on the layout must be reserved for this character.

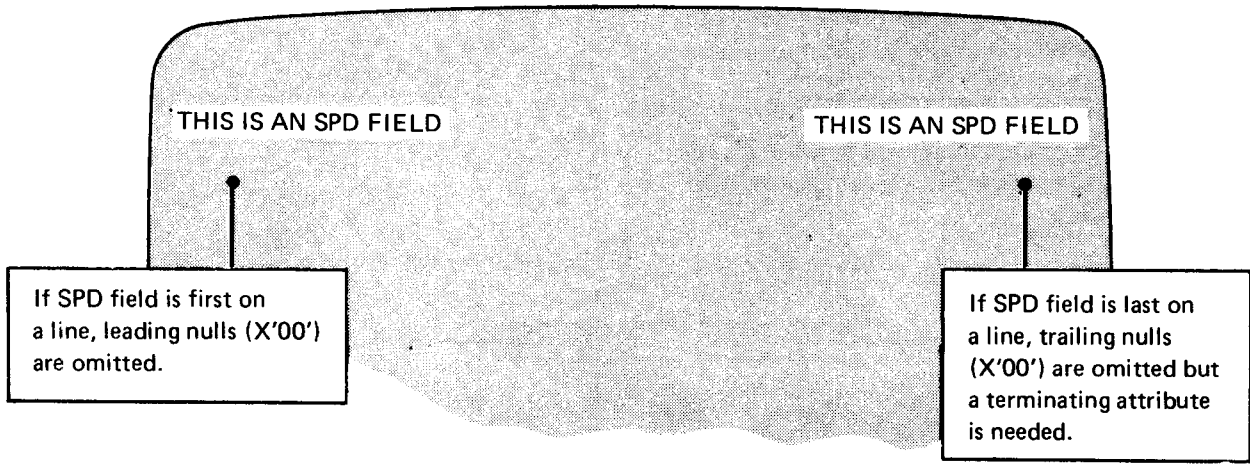
 The location of the first data position must be entered in the field starting location on the field definition form. This position must be located immediately following the designator character.

 The field data length is entered on the field definition form. Positions on the layout form must be reserved equal to the field length. The field length does not include leading nulls, defining attribute, designator character, trailing nulls, or terminating attribute.

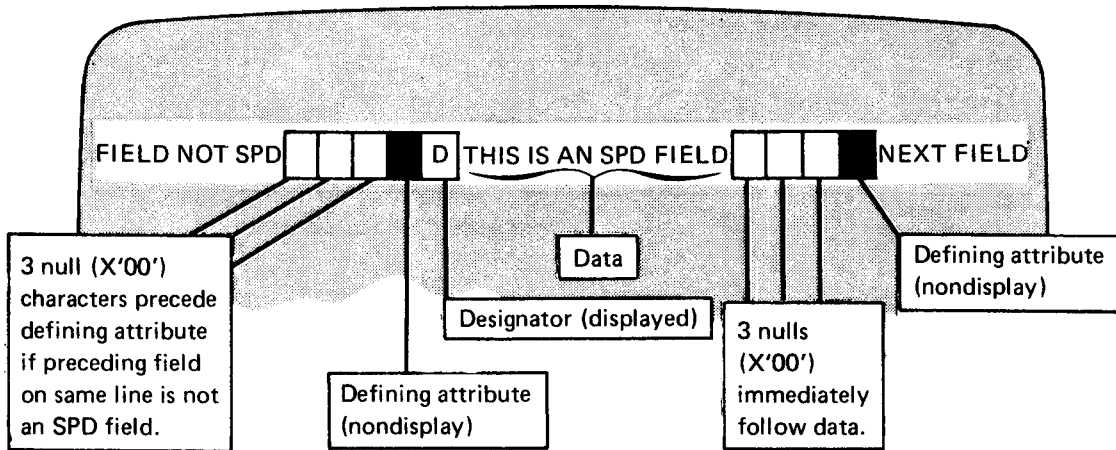
 Leading nulls are required when the preceding field on the same line is of another class. Trailing nulls are required unless the SPD field is the only, or last, field on the line. To reserve the correct number of positions on the layout form, refer to the guidelines above.

 A terminating attribute is required when the SPD field is the only, or last, field on the line. In either of these cases, one position on the layout must be reserved for this non-displayable character located immediately after the data field. The terminating attribute for an SPD field is used to prevent extraneous data, at the end of the current line or from the next line, from being transmitted along with the data for the SPD field during an input operation. Even though such a circumstance could occur only when using *overlay screens* (see index entry), all SPD fields ending a line are required to have the terminating attribute.

When SPD is first or last field on a line:



When SPD is not first or last field on a line:



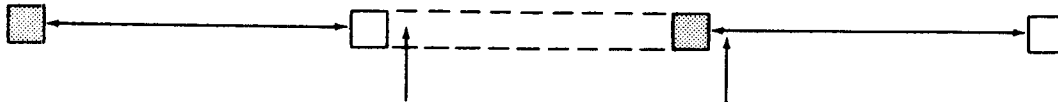
The entire field with its attributes and designator must be on the same line.

*Note:* For a type 5 SPD field, the designator and data are also not displayed.

## AUTOSKIP AND CURSOR POSITIONING

The way autoskip functions with input and output/input fields depends upon the relative positioning of successive fields.

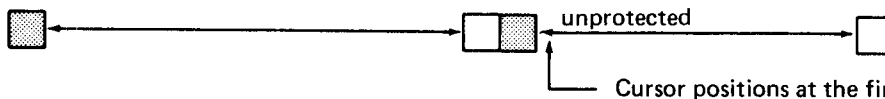
1. When there are one or more spaces between the trailing attribute of one field and the defining attribute of the next field:



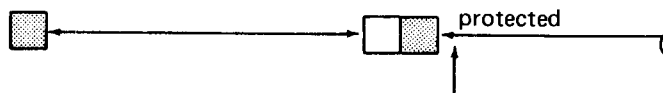
Without autoskip, the cursor is positioned after the trailing attribute. TAB or SKIP must then be pressed to move the cursor to the first character position of the next unprotected field.

With autoskip, the cursor is positioned at the first data character of the next unprotected field.

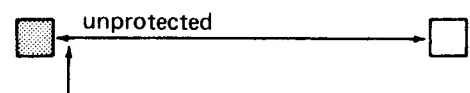
2. When the trailing and defining attributes are in adjacent positions:



Cursor positions at the first character of the unprotected field with or without autoskip.

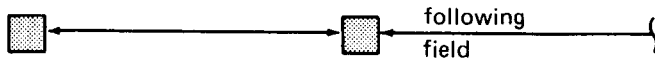


Without autoskip, the cursor positions at the first character of the protected field (unless this field is a type 6 input or output/input field). Use TAB or SKIP to move cursor to next unprotected field.

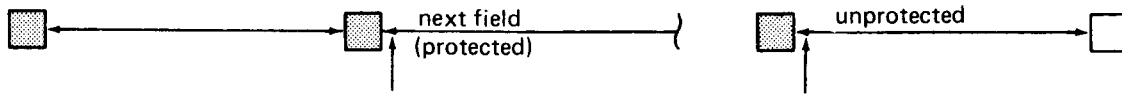


With autoskip or if protected field is a type 6 input or output/input, the cursor positions at the first character of the next unprotected field.

3. When two spaces were not allowed between fields:



If two spaces were not allowed by the application programmer during layout of the fields, the terminating attribute is dropped. The single space between the fields will be used for the attribute character of the next field.



The cursor positions at the first data character of the next field (unless this field is a type 6 input or output/input field). Use TAB or SKIP to move the cursor to the next unprotected field.

If the next field is a type 6 input or output/input field, the cursor positions at the first data character of the next unprotected field.

### Defining Data

#### Defining Data at Format Generation Time

During generation time, the user may provide data on the field definitions for output, output/input, and SPD fields. Such data is in the form of characters and/or spaces up to the specified field length.

The length of each field is defined at format generation time (see Figures 8-3 and 8-4). Each SPD field must be contained on one line. Therefore, the field length of SPD fields cannot exceed 37 for a Model 1 screen or 77 for a Model 2 screen. All other fields cannot exceed 240 characters. Minimum length for all fields is one.

See *Data in Numeric Fields* for techniques in handling numeric data.

#### Defining Data at Execution Time

Data can be provided for all but input fields during execution time. The length of data to be provided must equal the field length specified at format generation time. Less data than the amount specified by the field length cannot be provided.

The 3270 terminal does not support a negative 0 (hex D0) or a negative value with 0 in the unit position.

An application program that uses alphameric input/output fields to store negative numbers should display them as positive numbers and convert them when they are reread. The DFF numeric fields can also be used.

See *Data in Numeric Fields* for techniques in handling numeric data.

#### Data Entry at the Terminal

The terminal operator, when keying data for input fields, must start with the first character location of the field. Starting with any other position causes the control routine to return all blanks in the field.

See *Data in Numeric Fields* for techniques in handling numeric data.

#### Data Passed to the Application Program

On input from the 3270 terminal, alphameric data is padded on the right with spaces to the field length when less data than that required by the field length is entered by the terminal operator.

Numeric data, received from numeric fields, is right-adjusted and padded with blanks on the left to the field length when less data than that required by the field length is entered by the terminal operator.

Numeric data, with input edit fields, is padded with zeroes and adjusts to align decimal points. All non-numeric characters are removed. If no data is received for this field, the field is left blank.

For input and output/input fields, it is assumed that data will always be received from the terminal. If no data is received, the fields are returned with blanks.

When the terminal operator uses the ENTER or PF keys to initiate the input operation and data is not received from the terminal for SPD selection, the field (alphameric) will be filled with spaces. Since the designator is not passed as data, the field length should not include the designator character.



International Business Machines Corporation

# DISPLAY FORMAT SPECIFICATIONS

GX21-9175

Printed in U.S.A.

Application	Completed by	Punching Instructions	Graphic					Page	of
			Punch					Date	

## DISPLAY CONTROL

C														PRINTER CONTROL																																																																	
Form Type	Display Name	Field Name for Initial Cursor Position	Display Size (L, Z)	Clear Before Writing (Y/N)	Special	WCC	(Reserved)																														Continuation (X)	Identification Sequence																																									
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80

## FIELD DEFINITION

F														Data for Output, Output/Input and Detectable Fields																																																																	
Form Type	Field Name	Field Starting Location	Field Length	FIELD CLASS	(Reserved)																														Continuation (X)	Identification Sequence																																											
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80

Figure 8-3. Display Format Specifications Sheet



International Business Machines Corporation

# PRINTER FORMAT SPECIFICATIONS

GX21-9238

Printed in U.S.A.

Application	Completed by	Punching Instructions	Graphic					Page	of
			Punch					Date	

## PRINTER CONTROL

H														(Reserved)																																																																	
Form Type	Printer Format Name	Printer Size (L, Z)	Printer Copy (1, 2, 3)	Line Length	Line per Page (1-99)	Multiple Pages (X)	Vertical Form Feed (X)	Dark Storage Unit for Display Formats (KATAKANA (X) PUNCH PRINT (P N))	(Reserved)																														Continuation (X)	Identification Sequence																																							
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80

## FIELD DEFINITION

L														Data																																																																	
Form Type	Field Name	Field Starting Location	Field Length	FIELD CLASS	(Reserved)																														Continuation (X)	Identification Sequence																																											
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80

Figure 8-4. Printer Format Specifications Sheet



When the SPD attention field is used to generate the I/O pending condition, field data is not included with input from SPD selection and SPD attention fields from the terminal. The fields passed to the application program are filled with spaces except for a greater than (>) character in the leftmost position for any field in which the modified data tag was ON. By testing a field, the application programmer can determine those fields in which the modified data tag was or was not set.

Fields in the users input record area must be the same lengths as the field lengths specified on the field definition forms. Field lengths are used by the Display Format Control Routine in moving data into the user's input record area.

### *Data in Numeric Fields*

All data in numeric fields should be right-justified when entered at generation time. Negative values in numeric output/input fields are provided at generation time by entering the minus sign (-) after the rightmost numeric digit. For example, a negative thirty-five would be given as 35-. Space for the negative sign must be allocated in the generated format when defining the field length. If numeric data is to be later entered by the terminal operator, the numeric fields may be set to zeros so that the field is visible to the operator.

**Note:** Continuation cards must be provided if the numeric field is designated as being longer than forty digits (or thirty-nine digits and a minus sign).

Data provided (with Put Message or Put Override operations) in numeric fields at execution time must be the same length as the field length specified at generation time. Programs written in RPG II that use edit codes for numeric output fields, and use these fields in the Put Message or Put Override operations, must have the field length specified at generation time, and the field length must be equal to the edited length of the field. Numeric fields are always right justified. Negative values provided for numeric input and output/input fields must combine the minus sign (-) with the rightmost digit. For example, a negative thirty-five is given as 3N. Before sending the message to the 3270, DFCR will move the numeric digits one position to the left, convert the N to a five and place a negative sign in the right-most position of the field. Space for the negative sign must have been allocated at generation time.

**Note:** In a one-position numeric output field, the minus sign would be all that would be sent for a negative value.

To enter data in numeric fields, the terminal operator should either

- Right-justify data in the field and left pad with zeros or blanks, or
- First key in the numeric data left-justified; press ERASE EOF; and then press either SKIP or TAB. The cursor will be positioned at the next designated cursor location. On input, DFCR will right-justify the data in the field and pad with leading blanks if less data than the field length is received. DO NOT press the space bar after pressing ERASE EOF. See *Data in Numeric Input Edited Fields* later in this chapter.

Numeric fields are not examined by DFF for valid numeric characters. The application program is responsible for validation of its input data.

The terminal operator must enter negative numerics in a numeric field by entering a trailing negative sign in the field. For example, a 35- is entered. DFCR will right-adjust the numerics and pad with blanks on the left to the field length when less data than that required is entered. The five will be converted to an N and the application program receives a right-adjusted 3N as data input for the field.

**Note:** If the minus sign (-) was the only character received, an X'D0' (equivalent to a -0) is returned to the application program.

Since some languages are more restrictive than others, particularly in the form of numeric data supported, the application programmer must be aware of their differences and inform the terminal operator what characters can be used. If the data conventions of the language are not enforced by the compiler generated object code, the application programmer must validate the data keyed by the terminal operator.

### *Data in Numeric Input Edited Fields (5704-SC2 Only)*

CCP will provide some editing of input data from input or output/input numeric fields if a decimal position is specified on the Display Format Specifications field definition statement. For Put Message or Put Override operations, the data provided by the application is handled the same as the data for normal numeric fields. The maximum length of a numeric input edited field is 15.

Note that numeric editing is provided only upon input. For output operations the program should provide any editing.

The data entered into a numeric input edited field has all non-numeric characters removed and is aligned on a decimal position. The number is considered negative if a minus sign either precedes or follows the number. Embedded minus signs are removed. The application program is not notified if any syntax or truncation errors are encountered. The following is an example. All fields are five position fields.

Data Entered	Data Returned and Aligned in Decimal Position				
	0	1	2	3	4
Ø Ø Ø 56	00056	00560	05600	56000	60000
2.86Ø	00002	00028	00286	02860	28600
-34Ø Ø	0003M	0034Ø <sup>4</sup>	0340Ø	3400Ø	4000Ø
121-Ø	0012J	0121Ø	1210Ø	2100Ø	1000Ø
.1234	00000	00001	00012	00123	01234
3/2/5	00325	03250	32500	25000	50000
1A2BC	00012	00120	01200	12000	20000
1-2-3	00123	01230	12300	23000	30000
1-ABC <sup>1</sup>	00001	00010	00100	01000	10000
1Ø 2.3	00012	00123	01230	12300	23000
\$1.23	00001	00012	00123	01230	12300
1.2.3 <sup>2</sup>	00012	00123	01230	12300	23000
\$36CR <sup>3</sup>	00036	00360	03600	36000	60000

<sup>1</sup> Non-numeric characters are treated as if they did not exist.  
<sup>2</sup> The last decimal encountered in the one used.  
<sup>3</sup> Characters such as CR, DR, etc. do not make a negative number.  
<sup>4</sup> The character 'Ø' is a X'D0'.

### Number of Fields

A total of 256 fields can be defined for a printer or display format. This does not include F-type output fields. Any number of these fields may be defined. The combined total of Input, Output/Input, and SPD fields cannot exceed 200 fields.

A maximum of seven SPD fields in the 3277 or 3275 Model 1 or thirteen SPD fields in the 3277 or 3275 Model 2 may be on any given display line. This is, of course, because the minimum length of an SPD field is six with no trailing attribute or four if there is a trailing attribute. The minimum of six consists of: the defining attribute, the designator, one data character, and three trailing nulls. The last SPD field on a line can require as few as four positions since the trailing nulls are not required.

When mixing detectable and nondetectable fields, a maximum of 15 defining and/or terminating attributes may be on a given line. Whereas all fields generate a defining attribute, input and output/input fields generate a defining and terminating attribute.

*Note:* It is possible to define up to 15 fields on a line by positioning fields following the input and output/input fields so that their defining attribute overlays the terminating attribute of the previous field. However an SPD field following an input and output/input field will not overlay the previous terminating attribute. The leading nulls cannot overlay terminating attributes. See Chapter 8 for the use of nulls with SPD fields.

## RECORD CONCEPTS

Fields are received from and passed to the application program as data records. The user defines data record formats by defining the fields on the field definition forms.

### Display Output Record Format

When writing the initial display format, data can be supplied during execution time for output, output/input, and SPD fields. If data is to be supplied, the format of the associated output record is determined by the order and field lengths of the fields as they are defined on the field definition form. Field names, field lengths, field end positions (for RPG SUBR92 use), and length of output record area required are listed on the printed output from DFGR.

### Printer Output Record Format

When the initial printer format is written, data can be supplied during execution time. If data is to be supplied, the format of the associated output record is determined by the order and field lengths of the fields as they are defined on the field definition form. Field names, field lengths, field end positions (for RPG SUBR92 use), and length of output record area required are listed on the printed output from PFGR.

## Input Record Format

Since it is possible to receive data from each input, output/input and SPD field defined, the input record area must provide an assigned location for every field. The format of the associated input record area is determined by the order and field lengths of the fields as they are defined on the field definition form. The symbolic name, field length, assigned end position for each input field (for RPG SUBR92 use), and length of input record area required are listed on the printout from DFGR.

*Note:* With the Put Overrides operation, the application programmer can choose between receiving input from all fields or only from selected fields (see *Put Override*).

The order of processing an input record area is:

1. Examine the CCP return code.
2. Examine the attention identifier (AID).
3. Process the fields in the input record area.

The attention identifier (AID) is a single character immediately preceding the input fields in the user's input record area. The AID is set by the terminal when the operator takes any action that produces an I/O interruption. The AID identifies the action (such as using the selector pen on an SPD attention field) or key (program function or program access keys) that caused the condition to be generated. For a complete discussion on AID, refer to *IBM 3270 Information Display System Component Description*, GA27-2749. The return code and the AID must be examined before processing of fields in the input record area. The effective input length includes the AID character.

*Note:* AID characters are shown at the bottom of the printer/display layout sheet (Figure 8-2). The AID for TEST REQUEST (0) is received by the application program only when the block length specified in the terminal attributes set is not large enough for the test being run (see BLKL parameter of TERMATTR assignment statement in *CCP System Reference Manual*).

For CCP return code considerations, see Appendix E.

## DISPLAY FORMAT GENERATOR

The display format generator routine does not run under the control of CCP. DFGR generates a 3270 display format from display format specifications. DFGR performs the following functions:

1. Reads display format specification statements.
2. Produces a printout of the specification statements, analyzes the specifications for errors, and logs diagnostic error messages as required.
3. Builds the display format as a two-part table structure:
  - a. Field Descriptor Table (FDT) -- a table of descriptive field information including the symbolic name of the field.
  - b. 3270 Data Stream -- containing output data if provided, and 3270 device-dependent control information required for formatting all fields defined.
4. Provides a printout of the display format as it would appear on the 3270 display screen; the printout shows field relations and screen positions (5704-SC2 only).
5. Provides a printout of field names, in the order in which they must appear in the output record area if data from the Field Descriptor Table is required by the application program using the Display Format Facility.
6. Produces a printout of all fields defined for input and the order in which they will appear in the input record area.
7. Calculates and prints the following:
  - a. Length of the output record area required in DFF program
  - b. Length of the input record area required in DFF program
  - c. Decimal length of the field descriptor table
  - d. Decimal length of the 3270 output data stream
  - e. Decimal length of the 3270 input data stream
8. Places the display format in a work file (\$WORK) on disk and then catalogs the display format in an object library on disk.

*Note:* If printer control on DFGR is used for a Katakana printer, unpredictable results can occur because of new line and end-of-message considerations. For this reason, PFGR should be used to build a printer format for a Katakana printer.

### Printer/Display Layout Sheet

The printer/display layout sheet (Figure 8-2) is a planning device. The application programmer uses it to plan the format and layout of the fields on the display. The completed layout is then used as a guide when filling out the display control and field definition forms (see Figures 8-6 and 8-7).

Notice the heavy line after position 40 and line 12 which marks the boundary of the 480-character display. The heavy line after position 80 and line 24 marks the boundary of the 1920-character display. The entire page is used for the layout of printer formats.

### Display Control Form

The display control form provides special information about the display format which, in general, is unrelated to the fields being defined. One display control form is required and must precede all field definition forms. (Both forms appear on the Display Format Specifications sheet—see Figures 8-5 and 8-7. Additionally, both forms appear on the reverse of the sheet to be used as a template to verify the position of data in the DFGR listings.)



**6** WCC – Columns 16-18

Enter the write control character in column 16 or leave blank and place an entry (X) in either column 17 or 18. Use Figure 8-6 to select an entry for column 16. The WCC allows the user full control over certain device operations such as starting the printer and resetting the keyboard.

If there is an entry (X) in either column 17 or 18, a default write control character is assigned and any entry in column 16 will be ignored.

If columns 16, 17, and 18 are all blank, it is assumed that the space character (X'40') is to be used for the WCC.

Default WCC selections when column 17 or column 18 contain an X, and operations associated with each default are as follows:

Column	Default WCC (column 16)	Operations Performed
17 (Screen)	Character - C	Restore the keyboard. Reset the modified data tags.
18 (Printer)	Character - Q	Start the print operation. Set length of character line to 40.
	Character - 8	Start the print operation. Set length of character line to 80.

**7** Disk Storage Unit for Display Formats—Columns 19-20

Enter R1, F1, R2, or F2 to specify the location of the object library where the display format is to be stored. If columns 19 and 20 are blank, the display format will be placed on the disk from which the display format generation routine was loaded.

When executing with CCP (5704-SC2 only), all display formats must be stored either on the CCP program pack or on the DSM system pack.

When executing with CCP (5704-SC2 only), display formats may be stored in any simulation area. Note that all display formats must be stored in the same simulation area.

Operation				Output Device Format				
Sound Alarm	Restore Keyboard	Start Printer (No for displays)	Reset MDTs	Display or Printer NL/EM Control	40-Character Print Line	64-Character Print Line	80-Character Print Line	
Yes	Yes	Yes	Yes		⌋	?	"	
		No	No	+	;	>	=	
	No	Yes	Yes	G	P	X	7	
		No	No	F	O	W	6	
	No	Yes	Yes	Yes	(	)	—	'
			No	No	<	*	%	@
No		Yes	Yes	E	N	V	5	
		No	No	D	M	U	4	
No	Yes	Yes	Yes	.	\$	{	#	
		No	No	¢	!	} <sup>1</sup>	:	
	No	Yes	Yes	C	L	T	3	
		No	No	B	K	S	2	
	No	Yes	Yes	I	R	Z	9	
		No	No	H	Q	Y	8	
No	No	Yes	Yes	A	J	/	1	
		No	No	b	&	-	0	

<sup>1</sup>This character is converted internally to hex 6A for a WCC by DFGR when generating a format, and by DFCR when using the WCC in a Put Override operation.

Figure 8-6. Write Control Characters

**8** (US/World Trade Format)—Column 21 (5704-SC2 Only)

Entry	Explanation
blank	U.S. format
I	World Trade format

Use column 21 to describe the function of the decimal point and commas for numeric input editing. If this column is left blank, the decimal position is defined by the period. If an I is entered in this column, the decimal position is defined by the comma.

**9** (Reserved)—Columns 22-23 are reserved and must be left blank.

## 10 Printer Control—Columns 24-71

New line and end-of-message orders can be included in the display format for control of printouts. Each order occupies a data position in the display and is executed only when printing. The orders appear as the graphics '5' and '9' respectively on a display screen or during a printout when using a specified line length format in the WCC. New line and end-of-message orders are ignored and treated as spaces if the field in front of either of these orders is a non-display/nonprint field.

If WCC is specified in column 16, which will define a format for a printer in an unformatted mode (not 40 or 80 characters per line), then new line (NL) and end-of-message (EM) orders should be specified. This may be accomplished in one of two ways (or both); specify up to 23 NL orders and one EM in the control card, and/or specify any number of NL orders and one EM order in the field definition statement by use of the reserved keywords @@@NL and @@@EM. It is suggested that all NL and EM orders be specified using only one or the other form (control card or field definition statement using the keywords). If both forms are used and an NL is specified on the field definition statement (@@@NL) for a position in a format after the last new line specified on the control card (which is assumed to be an EM), that last control card entry will still be considered an EM order. Since the reserved keywords allow NL and EM orders to be specified in the same order and method as fields, it is suggested that the keyword method of specifying NL and EM orders be used for unformatted printer operations (See *Additional Functions for the Field Definition Statement* Chapter 8). For the 3288 printer equipped with the vertical forms control feature, a reserved keyword (@@@FF) will insert the forms feed order into the text stream. There is no way of specifying this order on the control card.

To specify control card NL and EM orders, each entry consists of a line (two columns) and a position (two columns). If the column 14 entry is 1 (for small screen), enter a number from 01 to 12 under line and a number from 01 to 40 under position. If the column 14 entry is 2 (for large screen), enter a number from 01 to 24 under line and a number from 01 to 80 under position.

Each entry, with the exception of the last entry, will cause an NL order character to be inserted into the specified location. The last entry will cause an EM order character to be inserted. If only one entry is specified, the EM order character will be inserted.

The new line and end-of-message orders specified on the display control form will not be diagnosed. An NL or EM may overlay another field, or an attribute with no error messages or halts given.

If NL or EM orders specified on the control card are used with the line/partial-line duplication function (@@@DP), a warning message and a halt (U-F1) is issued at the end of generation. See the Display Format Generator messages.

### Notes:

1. For a discussion of new line and end-of-message orders, refer to *IBM 3270 Information Display System Component Description, GA27-2749*.
2. If printer control on DFGR is used for a Katakana printer, unpredictable results can occur because of new line and end-of-message considerations. For this reason, PFGR should be used to build a printer format for a Katakana printer.

## 11 Continuation—Column 72

If more than twelve printer control entries are required, place an X in column 72 and make remaining entries in columns 24-71 of the second card. (Columns 2-23 of the second card must be blank.) One continuation card is allowed. This provides for NL control for up to 23 lines with EM control for the last line.

## 12 Identification-Sequence—Columns 73-80

Enter any character for sequential identification. These columns will be ignored other than to print them as part of the statement.

## Field Definition Form

An understanding of how to define fields (see *Field Concepts*) is necessary to correctly complete the field definition form. Refer to the printer/display layout sheet and the display control form while completing the field definition form. Each field in the display format must be specified on a field definition form. Enter an \* in column 1 to identify a comment statement. Comments can be used to include notes on the listing produced by the Display Format Generator program. Placement of the statements is not restricted (may appear before the display control form, between continuation cards, etc) during a single-format build. Placement of comments is not restricted when using multiple-format builds, but all comments before the display control form of the next format build are printed with the previous format build. See Figure 8-7 for location of items described below:

### 1 Form Identifier—Column 1

The preprinted character 'F' identifies this as a field definition form.

### 2 Field Name—Columns 2-7

Enter in columns 2-7 a unique field name (up to six characters in length). The first character must be alphabetic or one of the characters, \$, #, or @. Remaining characters can be any combination of alphabetic or numeric characters or the characters, \$, #, or @. Imbedded blanks are not allowed.

The field names as defined are the symbolic names used with Put Overrides applications.

*Note:* For more considerations, see *Additional Functions for the Field Definition Statement* in this chapter.

### 3 Field Starting Location—Columns 8-11

A line (two columns) and a position (two columns) entry define the leftmost character position of the field. Allowable entries are limited by the display size (see column 14 of the display control form). If the display size entry is 1 (for small screen), enter a number from 01 to 12 under line and a number from 01 to 40 under position. If the display size entry is 2 (for large screen), enter a number from 01 to 24 under line and a number from 01 to 80 under position. Field starting locations must be specified in numerically increasing order (see *Planning the Printer/Display Layout*).

### 4 Field Length—Columns 12-14

The minimum field length is one. The maximum length for an output, input, or an output/input field is 240 positions. An SPD field must be contained on one line and is therefore limited to a length of 37 or 77 depending upon the display size (see *Planning the Printer/Display Layout*). If the field is defined as a numeric input edit field, the length must not exceed 15. This includes all punctuation to be removed.

*Note:* For more considerations, see *Additional Functions for the Field Definition Statement* in this chapter.

### 5 Output, Type—Column 15

Enter 1, 2, or 5 (see *Field Concepts*, earlier in this chapter).

### 6 Output, Data Source—Column 16

Enter E if data is provided when writing the initial display format during execution by the application program.

Enter a G if data is generation-defined (data is entered in columns 32-71) and a Put Override is used to override the field during execution. Enter an F for an output field having generation-defined data (in columns 32-71 only). Entering an F eliminates the Field Descriptor Table (FDT) entry, which saves 14 characters of storage. However, the lack of this FDT entry eliminates the capability of overriding the field during execution.

Either a G or an E can be entered and a Put Override can still be used to override the field at execution time.

If this entry is blank, the default G is assumed.

### 7 Input, Type—Column 17

Enter 1, 2, 3, 4, 5, 6, 7, or 8 (see *Field Concepts*, earlier in this chapter).

### 8 Input, Automatic Skip—Column 18

This entry determines the terminating attribute. Enter Y in column 18 if automatic skip function is to be performed (see *Autoskip and Cursor Positioning*).

Enter N if automatic skip function is not to be performed.

Default is N if entry is left blank.





**14** *Decimal Position*—Column 24 (5704-SC2 Only)

Enter a number from 0 through 9 to indicate that this numeric field is to be scanned on input, all non-numeric characters are to be removed, and the resultant number is to be aligned on its decimal position.

- This number must not exceed the number of positions in the field or 9, whichever is smaller.
- The field may be defined only for input field types 3, 4, 6, and 8 and for output/input field types 3, 4, and 6.
- The length of the field must not exceed 15 characters.

**15** *(Reserved)*—Columns 25-31 must be left blank.

**16** *Data*—Columns 32-71

Data may be provided for Output, Output/Input, or SPD fields. Data must not exceed field length as specified in columns 12-14.

If data exceeds 40 characters in length, place an X in column 72 and continue entering data in column 32 of the next line (leave columns 2-31 blank).

**17** *Continuation*—Column 72

If the data entry must continue beyond column 71, place an X in column 72 and continue entering data in column 32 of the next line. A maximum of five continuation lines are allowed.

**18** *Identification-Sequence*—Columns 73-80

Since these columns are ignored other than to print them as part of the statement, they may contain any characters.

**Additional Functions for the Field Definition Statement**

*Printer Control*

An optional method of providing new line and end-of-message printer control is to request the generation of these orders on Field Definition statements. An additional control called forms feed can be specified for those printers having the Vertical Forms Control feature. The new line, end-of-message, and forms feed orders on field definition statements are checked for position. If there are control

characters overlapping other fields or attributes, an error message and halt are issued. The position of the new line and end-of-message orders on the display control form are not checked.

If printer control on DFGR is used for a Katakana printer, unpredictable results can occur because of new line and end-of-message considerations. For this reason PFGR should be used to build a printer format for a Katakana printer.

To specify printer control on a Field Definition statement:

*Form Type*—Column 1

Contains the preprinted character F.

*Name*—Columns 2-7

@@@NL — Signifies a new line order is to be generated in the display format (see note).

@@@EM — Signifies an end-of-message order is to be generated in the display format.

@@@FF — Signifies a forms-feed order is to be generated in the display format. This order is valid only for printers having the Vertical Forms Control feature. When valid, the forms-feed order is printed as a blank; when invalid, it is displayed as a graphic < (see note). See *IBM 3270 Information Display System Component Description*, GA27-2749.

*Note:* There is no restriction on the number of new line (@@@NL) or forms-feed (@@@FF) orders that can be generated.

*Field Starting Location*—Columns 8-11

Enter the line number in columns 8 and 9, the position number in columns 10 and 11. These numbers indicate where the order will be in the format. Each order occupies one position in a format. These orders should be provided in numerically increasing order.

## DFGR Line/Partial-Line Duplication

Line/partial-line duplication allows generation of duplicate fields on consecutive lines without requiring Field Definition statements for each duplicate field. Instead, one Field Definition statement is required to place the duplicate fields on the desired line. To allow duplication, enter the following on the Field Definition form:

### *Form Type*—Column 1

Contains the preprinted character F.

### *Field Name*—Columns 2-7

@@@DP identifies this as a duplication statement. This means that this is a request to duplicate a field(s) defined in the Field Definition statement(s) for the previous line.

### *Field Starting Location*—Columns 8-11

The line number, columns 8 and 9, identifies on which line to generate the duplicated fields. The fields will be duplicates of the fields for the last defined line. Any number of blank lines can be left between the original line and the line on which the duplicate fields are being generated. The position number, columns 10 and 11, is an optional entry. It gives the position number on the line where duplication is to begin. If no entry is made, the default is position 1. If the position specified is in the middle of a field, DFGR starts to duplicate at the start of the next field.

### *Field Length*—Columns 12-14

**Number of Fields:** The number entered specifies the number of fields to duplicate (see columns 8-11 for start position). If the number of fields is not specified or if the number of fields specified is greater than the number of fields available to be duplicated, all fields after the start position of that line (columns 8-11) are duplicated.

**Number of Lines:** If an @ is entered in column 12, followed by a number in 13 and 14, this number specifies the number of times to duplicate the last line with field definitions. If columns 13 and 14 are left blank it will duplicate the line once.

No other entries on the duplication statement are recognized.

## DFGR Considerations for the Duplication Function

- If a line being duplicated contains both fields defined by Field Definition statements and fields generated by a previous duplication statement, only the fields defined by Field Definition statements are duplicated.
- Field names are assigned to fields generated under the duplication process. The names have the format 'DPIlpp'. The 'DP' indicates it is a duplicate field and 'lpp' are the line and position where it is located. You should avoid defining other fields using this convention because you could get duplicate field names.
- If a field being duplicated extends to another line, or lines, the entire field is copied.
- If any DFGR warning messages were issued (but no termination messages), a halt (U-F1) is issued. The options available are:
  - 0 — Catalog the format and ignore the warnings.
  - 3 — Cancel the program and DFGR will go to end of job.

**Examples of Duplication Functions:** The following examples are some of the ways to use the line and field duplication functions. The duplication function is indicated by '@@@DP' in columns 2-6. The fields that are generated from this duplication function are between the lines of asterisks, starting with the line with the comment 'GENERATED DUPLICATION FIELDS' and ending with the line with the comment 'END OF GENERATED DUPLICATION FIELDS'.

C\$ZTSTL 2Y XR1  
FA 2 2 31E  
FB 3 5 402G DUPLICATED ON LINE 4  
F@@NL 350 DUPLICATED ON LINE 4

**A** F@@DP 4  
\*\*\*\*\* GENERATED DUPLICATION FIELDS  
FDP040504 5 402G DUPLICATED ON LINE 4  
F@@NL 0450 DUPLICATED ON LINE 4  
\*\*\*\*\* END OF GENERATED DUPLICATION FIELDS

FC 7 9 21G \*\*  
F@@FF 716 \*  
FD 718 205E \*\* DUPLICATED ON LINES 9,10,11  
FE 741 82G \*  
F@@NL 750 3 \*\*

**B** F@@DP 9 @ 3  
\*\*\*\*\* GENERATED DUPLICATION FIELDS  
FDP090909 9 21G \*\*  
F@@FF 0916 \*  
FDP09180918 205E \*\* DUPLICATED ON LINES 9,10,11  
FDP09410941 82G \*  
F@@NL 0950 3 \*\*  
FDP100910 9 21G \*\*  
F@@FF 1016 \*  
FDP10181018 205E \*\* DUPLICATED ON LINES 9,10,11  
FDP10411041 82G \*  
F@@NL 1050 3 \*\*  
FDP110911 9 21G \*\*  
F@@FF 1116 \*  
FDP11181118 205E \*\*  
FDP11411141 82G \*  
F@@NL 1150 3 \*\*  
\*\*\*\*\* END OF GENERATED DUPLICATION FIELDS

F@@FF 12 3  
FTESTA 14 2 3 7GY  
FTESTB 14 6 8 8GY

FTESTC 1420 30 7GN DUPLICATED ON LINE 15  
FTESTD 1454 22G DUPLICATED ON LINE 15

**C** F@@DP 1520@ 1  
\*\*\*\*\* GENERATED DUPLICATION FIELDS  
FDP15201520 30 7GN DUPLICATED ON LINE 15  
FDP15541554 22G DUPLICATED ON LINE 15  
\*\*\*\*\* END OF GENERATED DUPLICATION FIELDS

FTEST1 20 2 3 7GY  
FTEST2 20 6 8 8GY  
FTEST3 2020 30 7GN DUPLICATED ON LINE 22  
FTEST4 2054 22G DUPLICATED ON LINE 22

**D** F@@DP 2220 3  
\*\*\*\*\* GENERATED DUPLICATION FIELDS  
FDP22202220 30 7GN DUPLICATED ON LINE 22  
FDP22542254 22G DUPLICATED ON LINE 22  
\*\*\*\*\* END OF GENERATED DUPLICATION FIELDS

FTST1 23 1 20 7GY DUPLICATED ON LINE 24  
FTST2 2325 151G DUPLICATED ON LINE 24  
FTST3 2345 11F  
FTST4 2349 5 8GN

**E** F@@DP 24 1 2  
\*\*\*\*\* GENERATED DUPLICATION FIELDS  
FDP240124 1 20 7GY DUPLICATED ON LINE 24  
FDP24252425 151G DUPLICATED ON LINE 24  
\*\*\*\*\* END OF GENERATED DUPLICATION FIELDS

F@@EM 2525

**A** *Whole Line Duplication (Columns 12-14 blank)*

The duplication instruction (@@@DP) indicates duplication of fields on line 4. Columns 12-14 on the duplication card are blank, so the default is to duplicate one whole line. This would be the same as specifying @01 in columns 12-14. One field and one new line are defined on line 3, which is the line previous to line 4. This field and new line are duplicated on line 4.

**B** *Whole Line Duplication (Columns 12-14 non-blank)*

The duplication instruction (@@@DP) indicates duplication on lines 9, 10 and 11. Column 12 contains the character '@' which indicates line duplication. The character 3 in column 14 indicates the previously defined line is to be duplicated three times. Since nothing is specified in the position columns (columns 10 and 11), the whole line will be duplicated. There are three fields, a forms-feed order (@@FF) and a new line order (@@NL) on line 7. Since line 8 contains no fields or forms-feed orders, the entries on line 7 will be duplicated on lines 9, 10 and 11.

**C** *Partial Line Duplication*

The duplication instruction (@@@DP) indicates duplication on line 15 of everything on the previous nonblank line from position 20 to the end of that line. Column 12 contains the character '@' which indicates line duplication. The character 1 in column 14 indicates the line is to be duplicated one time. The characters 20 in columns 10 and 11 indicate everything from position 20 to the end of the line should be duplicated. In this case two fields, TESTC and TESTD, will be duplicated on line 15.

**D** *Field Duplication*

The duplication instruction (@@@DP) indicates duplication on line 22 of three fields from the previous nonblank line, beginning at position 20. Column 12 is blank, and column 14 contains a character 3 indicating field duplication. Three fields were specified to be duplicated but only two fields were specified on line 20 from position 20 to the end of the line. In this case only these two fields are duplicated.

**E** *Field Duplication*

The duplication instruction (@@@DP) indicates duplication on line 24 of the two fields from the previous nonblank line beginning at position 1. Column 12 is blank indicating field duplication, and column 14 contains a character 2, indicating two fields are to be duplicated. In this case the first two fields from line 23 are duplicated on line 24.

## OCL Considerations for the Display Format Generator

The following OCL statements are required when either single or multiple formats are built:

```
// LOAD $CCPDF,unit  
  
// FILE NAME-$WORK,UNIT-  $\left. \begin{matrix} \text{R1} \\ \text{F1} \\ \text{R2} \\ \text{F2} \end{matrix} \right\}$  ,PACK-packid,  
    RETAIN-S,TRACKS-2  
  
// RUN
```

When reading from the system input device, be sure that the display format statements immediately follow the RUN statement. A /\* follows the format statements. The statements are read from the system input device and written into a \$SOURCE file. If the display format statements require more than five tracks of a file, define a \$SOURCE file with an adequate number of tracks. If a \$SOURCE file OCL statement is not specified, a 5-track \$SOURCE file is automatically allocated on the unit from which \$CCPDF was loaded.

When reading from the source library, specify a \$SOURCE file with enough tracks to contain all the display format statements in the source library. A COMPILE is also required with the following parameters:

```
// COMPILE SOURCE-name of source data,UNIT-  $\left. \begin{matrix} \text{R1} \\ \text{F1} \\ \text{R2} \\ \text{F2} \end{matrix} \right\}$ 
```

where UNIT is the location of the source library !

### Notes:

1. For multiple format builds in a single run, the input stream must not contain delimiters between the display formats. Instead, the C in column 1 of the display control form indicates the start of another input.
2. Enter an asterisk (\*) in column 1 to identify a comment statement. Comments may be used to include notes on the listing produced by the Display Format Generator routine. Comments can be placed anywhere in the format build; however, on multiple format builds, all comments entered before the display control form of the second or later format builds are printed with the previous format build.

## Display Format Generator Diagnostic Messages

During its processing, the display format generator (DFGR) diagnoses errors in the Display Format Specifications and logs diagnostic messages on the system logging device. Refer to the *CCP Messages Manual*, GC21-5170, for a list of the diagnostic messages.

Since DFGR calls the Overlay Linkage Editor to put the formats into the object library, it is possible to get Overlay Linkage Editor halts when using the DFGR.

## PRINTER FORMAT GENERATOR ROUTINE (PFGR)

The Printer Format Generator routine does not run under the control of CCP. PFGR generates a 3270 printer format from printer specifications in the following functional steps:

1. Reads printer format specifications statements.
2. Produces a printout of the specification statements, analyzes the specifications for errors, and logs diagnostic error messages as required.
3. Builds the printer format as a two-part table structure:
  - a. Field Descriptor Table (FDT) — containing descriptive field information, including the symbolic name of the field
  - b. 3270 Data Stream — containing output data, if provided, and 3270 device-dependent control information required for formatting all fields defined
4. Provides a printout of field names, in the order in which they must appear in the output record area if data from the Field Descriptor Table is required by the application program using the Display Format Facility.
5. Calculates and prints:
  - a. Length of the output record area required in the DFF program
  - b. Decimal length of the Field Descriptor Table
  - c. Decimal length of the 3270 output data stream
6. Places the printer format in a work file (\$WORK) on disk, then catalogs the printer format in an object library on disk.

## Printer/Display Layout Sheet

The printer/display layout sheet is the same as that used for DFGR (Figure 8-2). The application programmer uses this form to plan the format and layout of the fields on the printer. The completed layout is then used as a guide to complete the printer control and field definition forms (see Figures 8-8 and 8-9).

The layout is 132 positions per line, the maximum platen length. Other platen lengths used are 120 and 126 positions per line.

## Printer Control Form

The printer control form provides special information about the printer format which, in general, is unrelated to the fields being defined. One printer control form is required and must precede all field definition forms. (Both forms appear on the Printer Format Specifications sheet—see Figures 8-4 and 8-8.)



Application		Completed by				Punching Instructions			Graphic		Punch		Page of	
												Date		

**PRINTER CONTROL**

Form Type 1	Printer Format Name 2-7	Printer Size (1, 2) Platen Lng (1, 2, 3) 8	Line Length 10-11	Lines per Page (1-99) 13-14	Multiple Pages (X) 15	Vertical Forms Feed (X) 16	Disk Storage Unit for Printer Formats 17-18	KATAKANA (X) 19	Print/Noprint (P, N) 20	(Reserved)													Identification Sequence 73-80																																				
										21	22	23	24	25	26	27	28	29	30	31	32	33		34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69

Figure 8-8. Printer Control Form is the First of Two Forms on the Printer Format Specifications Sheet

Entries on the printer control form are as follows:

**1** *Form Type*—Column 1

The preprinted character H identifies this as the printer control form.

**2** *Printer Format Name*—Columns 2-7

This entry is used to assign a unique name to a printer format. Entries in columns 2 and 3 are \$Z. The remaining characters in the name can consist of any combination of alphabetic and/or numeric characters plus any of the characters \$, #, or @. Blanks may not appear between characters in the name. The length of the name can be from three to six characters. All application programs using a particular printer format must refer to it by its assigned symbolic name, because each printer format is cataloged in the object library under its symbolic name.

**3** *Printer Size*—Column 8

Enter a 1 or leave this column blank if the printer format is to be used with a 480-character printer.

Enter a 2 in this column if the printer format is to be used with a 1920-character printer.

If more fields are defined than can be displayed, an error message is issued; and all remaining field definition forms are read but not processed.

**4** *Platen Length*—Column 9

Enter a 1 in this column if the printer used has a 120-character platen.

Enter a 2 in this column if the printer has a 126-character platen.

Enter a 3 or leave this column blank if the printer has a 132-character platen.

**5** *Line Length*—Columns 10-12

Enter in these columns the length of the line to be printed. Valid line length can be from 001 to the platen length specified (see column 9). A field cannot extend beyond the line length specified.

**6** *Lines Per Page*—Columns 13-14

Enter the total number of lines (page length), from 1 through 99, that can be printed on a single page. If columns 13-14 are left blank, the default value is 66 lines per page.



**7** *Multiple Pages*—Column 15

Enter an X in this column if the printer format requires more than one page. If this column is blank, all fields are printed on one page. Any field that has a *line/position* value less than the previous field specified is flagged with an error message, and the format is not built. If this column contains an X and a field has a *line/position* value less than the previous field specified, the field is placed on the next page at the *line/position* value given. Processing of the remaining field continues.

**8** *Vertical Forms Feed*—Column 16

Enter an X in this column if a 3288 printer is used with the option to support the forms-feed function. Forms-feed is not used if this column is left blank.

*Note:* The Katakana feature does not support vertical forms feed.

**9** *Disk Storage Unit for Printer Formats*—Columns 17-18

Enter R1, F1, R2, or F2 to specify the location of the object library in which the printer format is to be stored. If columns 14 and 15 are blank, the printer format is stored on the disk from which the printer format generation routine was loaded.

When CCP is executing, all printer formats must be stored on either the CCP program pack or the DSM system pack.

**10** *Katakana Printer Format*—Column 19

Enter an X in this column if the format to be generated is to be used on a Katakana printer other than a 3287 or 3289. (3287 and 3289 Katakana printers are to be treated the same as non-Katakana printers.) Leave this column blank for all other printers. Formats generated for a Katakana printer should not be used on a non-Katakana printer because unpredictable results will occur. Non-Katakana formats should not be run on Katakana printers.

**11** *Print/No-print*—Column 20

Enter P or leave this column blank if the initial format (not modified by an operation) is to be printed on the 3270 printer.

Enter N if the initial format is not to be printed.

**12** *(Reserved)*—Columns 21-72

These columns are reserved and are to be left blank.

**13** *Identification-Sequence*—Columns 73-80

Enter any characters for sequential identification. These columns are ignored other than to be printed as part of the statement.

FIELD DEFINITION

Figure 8-9. Field Definition Form on the Printer Format Specifications Sheet

**Field Definition Form**

Refer to the printer/display layout sheet and the printer control form while completing the field definition form. Each field in the printer format must be specified on a field definition form. When defining fields using PFGR, do not allow space for defining and terminating attributes, as PFGR does not generate field defining or terminating attributes. The amount of printer buffer space specified need only be the length of the data and the printer control characters (NL, EM, FF). See Figures 8-4 and 8-9 for the location of the following items:

**1 Form Type—Column 1**

The preprinted character L identifies this as a field definition form for the Printer Format Generator.

**2 Field Name—Columns 2-7**

Enter in these columns a unique field name that is from one to six characters in length. The first character must be alphabetic or a \$, #, or @. The remaining characters can consist of any combination of alphabetic and/or numeric characters plus any of the characters \$, #, or @. Avoid assigning names beginning with @@@ because these characters identify special function requests. Avoid field names starting with D followed by five numeric characters when using the field duplication function. Embedded blanks are not allowed.

**3 Field Starting Location—Columns 8-12**

A *line* entry (two columns) and a *position* entry (three columns) define the leftmost character of the field. Allowable entries are limited by the *printer size*, *line length*, and *lines per page* entries (see columns 8, 10-12, and 13-14 of the printer control form).

In the *line* field (columns 8-9), enter the number (01-99 not to exceed the value in the *lines per page* field) of the line that the field is to occupy in the output.

In the *position* field (columns 10-12), enter the number (001 to the value specified in the *line length* field) of the position on the line that the field is to occupy.

The field starting locations must be consecutive and must not exceed the length of the printer buffer. Determine the number of characters used in the printer buffer as follows:

- If the vertical forms-feed order is not active (blank in column 16 of the printer control form), add the following to calculate the buffer required to build the format:
  - a. Total length of all fields defined.
  - b. Number of all unused positions on a line that precede the last field in the line.

- c. One character for each new line (NL) order when building a format for a non-Katakana printer, or two characters for each NL order when building a Katakana format. An NL occurs each time that the last position in a line does not contain a character. Only the last line that contains a field and all previous lines are included in this count. If the last position in a line contains a character and the line length is equal to the platen length, the printer automatically advances to the next line and an NL is not required.
  - d. One character for an end-of-message (EM) order when building a format for a non-Katakana printer, or two characters for an EM order when building a format for a Katakana printer.
- If the vertical forms-feed order is active (an X in column 16 of the printer control statement), add the following to calculate the buffer required to build the format:
    - a. Total length of all fields defined.
    - b. Number of all unused positions that precede the last field in a line. Exclude line 01, position 001, from this count on all but the first page of the format (this position is unusable after a forms-feed order).
    - c. One character for each new line (NL) order. An NL occurs each time that the last position in a line does not contain a character. Only the last line to contain a field on each page and all previous lines on that page are included in this count. If the last position in a line contains a character and the line length is equal to the platen length, the printer automatically advances to the next line and an NL is not required.
    - d. One character for each page (excluding the last page) that does not contain any fields on the last line of that page.
    - e. One character for an end-of-message (EM) order.

*Note:* When a small print buffer is used (1 in column 8 of the printer control form), the preceding calculations must not exceed 480 characters. When a large printer buffer is used (2 in column 8 of the printer control form), the preceding calculations must not exceed 1920 characters.

#### **4** *Field Length*—Columns 13-15

The minimum field length is 1. The maximum field length is the line length value entered in columns 10-12 of the printer control form.

#### **5** *Data Source*—Column 16

Enter an E if data is provided when the initial display format is written during execution by the application program.

Enter a G if data is generation-defined (data is entered in columns 32-71) and a Put Override is used to override the field during execution.

Enter an F if an output field has only generation-defined data (in columns 32-71). Entering an F eliminates the Field Descriptor Table (FDT) entry; this elimination saves 14 characters of storage. However, when the FDT entry is eliminated, the field cannot be overridden during execution.

Either a G or an E can be entered, and a Put Override can still be used to override the field at execution time.

If this entry is blank, default G is assumed.

#### **6** *Repeat Last Character*—Column 17

Enter an X in this column if the last character entered in columns 32-71 is to be repeated to the end of the field. For example, if 20 asterisks (\*) are to be printed, define a field 20 characters long; place an asterisk in column 32; and enter an X in column 17. The result will be a field of 20 asterisks.

#### **7** *Reserved*—Columns 18-31

These columns are reserved and must be left blank.

#### **8** *Data*—Columns 32-71

Enter the data for the generation-defined fields. Data must not exceed the field length specified in columns 13-15. If the data exceeds 40 characters, place an X in column 72 and continue entering data in column 32 of the next line (leave columns 2-31 blank). If an execution-defined field is specified (E in column 16), columns 32-71 are ignored.

#### **9** *Continuation*—Column 72

If the data entry exceeds 40 characters, enter an X in column 72 and continue entering data in columns 32-71 of the next line. No more than three continuation lines are allowed.

#### **10** *Identification-Sequence*—Columns 73-80

Enter any characters for sequential identification. These columns are ignored other than to be printed as part of the statement.

## Printer Control on the Field Definition Statement

A method of supplying forms-feed printer control is to request the generation of these orders on field definition statements.

### Notes:

1. Multiple pages must be specified (an x in column 15 on the printer control form) if printer control is used.
2. The Katakana feature does not support vertical forms feed.

To specify printer control (forms feed) on a field definition statement, enter the following:

### 1 Form Type—Column 1

The preprinted character L identifies this as a field definition for the Printer Format Generator.

### 2 Field Name—Columns 2-7

To indicate either a forms-feed order (for 3288 printers with the Vertical Forms Control feature) or to generate multiple NL orders to advance the printer to line 01, position 001 of the next page, enter @@@FF.

*Note:* If a 3288 printer with the Vertical Forms Control feature is given the forms-feed order, the printer is advanced to line 01, position 002, of the next page. Placing the forms-feed order at the end of a print buffer is not recommended. When a valid FF order is placed at the end of a printer format, the 3288 printer will stop printing and will skip to line 02 of the next form.

## PFGR Line/Partial-Line Duplication

PFGR supports a line/partial-line duplication feature to allow a defined field (the master) to be duplicated on following output lines. The master line must be defined first; the duplicated lines must be defined immediately following the master line definition and must be entered in ascending order. Any number of blank lines can be left between the master and duplicated lines on the output form. The duplication request is entered as follows:

### 1 Form Type—Column 1

The preprinted character L identifies this as a field definition for the Printer Format Generator.

### 2 Field Name—Columns 2-7

Enter @@@DP to indicate the duplication request. The master field must immediately precede this entry. This request causes the field(s) defined in the master field definition statement to be duplicated in a new location.

### 3 Field Starting Location—Columns 8-12

A *line* entry (two columns) and a *position* entry (three columns) define the horizontal output line where the master line is to be duplicated and the position of the first character to be duplicated.

In the *line* entry (columns 8-9), enter the output line number that is to contain the duplicate of the master. Line numbers must be in ascending order, though any number of blank lines can be left between the output lines.

The *position* entry (columns 10-12), an optional entry, defines the first position of the master to be duplicated. If the position specified is not the starting position of the field within a line, duplication starts at the beginning of the next field. If the *position* entry is left blank, duplication begins at position 001.

### 4 Field Length—Columns 13-15

This entry, an optional entry, specifies the number of fields to be duplicated from the master. If no entry is made in these columns, all fields beginning at the point specified in the *position* entry are duplicated. If the number specified is greater than the number of master fields on the line, duplication begins with the first field specified and continues to the last field of the master.

If the master is to be duplicated on a number of consecutive output lines, enter the character @ in column 13 and the number of times the line is to be duplicated on the output in columns 14 and 15.

The remainder of the duplication statement must be left blank.

### Notes:

1. If a duplication request statement contains both field definition statements and fields generated by a previous duplication statement, only those fields defined directly by field definition statements are duplicated.
2. Field names are assigned to fields generated under the duplication process. Such names have the form Dllppp, where llppp indicates the line and position locations. Avoid defining other fields using this convention because duplicate field names could result.

**Example of a Format Written for the Printer Format Generator:** Following is an example of a format written for the Printer Format Generator. The *line* and *position* values (columns 8-12) on the field definition statements in the example (part **A**) are the placement of each field on the page when printed on the 3270 printer (part **B**).

When generation-defined data is specified (G in column 16), the data supplied in columns 32-71 for the length specified in columns 13-15 is printed on the 3270 printer.

When execution-time data is specified (E in column 16) and the Display Format Test routine (DFTR) is being run, asterisks are placed in the field and printed. The DFTR can be used for either printer or display formats. When execution-time data is used with the Display Format Control routine, the data supplied during execution is placed in these fields.

When the repeat-last-character function is specified, the last character supplied in columns 32-71 is repeated to the end of the field.

The duplicate line function for PFGR is the same as the duplicate line function for DFGR (see *Examples of Duplication Function*). The only variation is the method of supplying field names on the duplicated line statements; the *position value* is three characters instead of two characters.

**Note:** See 3284/3286 Printer Considerations in Chapter 2.

```

H$ZPF021311099X R1
A LB2 02010 18G          CUSTOMER NUMBER -
    LB3 02029 6E
    LB4 04001 93GX          *
    LB5 06010 10G          ADDRESS -
    LB6 06020 30E
    LB7 07020 30E

L000DP 08
***** GENERATED DUPLICATION FIELDS
LD0802008020 30E
***** END OF GENERATED DUPLICATION FIELDS

LB8 09040 10E
LB9 11010 11G          QUANTITY -
LB10 11021 3E
LB11 13010 14G          ITEM NUMBER
LB12 13024 2E
LB13 15001 93GX          *

                                $ZPF02 PRINTER FORMAT INFORMATION

EXECUTION TIME DATA - OUTPUT AREA FORMAT - END POSITIONS FOR RPG PROGRAMS
-----
* FIELD   FIELD   END   * FIELD   FIELD   END   * FIELD   FIELD   END   *
* NAME    LENGTH  POSITION * NAME    LENGTH  POSITION * NAME    LENGTH  POSITION *
* ----- * ----- * ----- * ----- * ----- * ----- *
* OPCODE  004    0004  * LENGTH  004    0008  * TMNAME  006    0014  *
* $CPF02  006    0020  * B3      006    0026  * B6      030    0056  *
* B10     003    0129  * B12     002    0131  *

LENGTH OF OUTPUT RECORD AREA REQUIRED IN DFF PROGRAM
RPG *SUBR92* - 0131  OTHER - 0123

INFORMATION FOR USE DURING CCP ASSIGNMENT STAGE
-----

THE DECIMAL LENGTH OF THE FIELD DESCRIPTOR TABLE IS 0112
THE DECIMAL LENGTH OF THE OUTPUT TEXT IS 0213

```

```

B
CUSTOMER NUMBER - *****
*****
ADDRESS - *****
*****
*****
*****

QUANTITY - ***
ITEM NUMBER - **
*****

```

## OCL Considerations for the Printer Format Generator

The following OCL statements are required when either single or multiple formats are built:

```
// LOAD $CCPPF,unit  
  
// FILE NAME-$WORK,UNIT- $\left. \begin{array}{l} (R1) \\ F1 \\ (R2) \\ F2 \end{array} \right\}$ ,PACK-packid,  
  
RETAIN-S,TRACKS-2  
  
// RUN
```

When reading from the system input device, be sure that the printer format statements immediately follow the RUN statement. A /\* follows the printer format statements. The statements are read from the system input device and written into a \$SOURCE file. If the printer format statements require more than five tracks of a file, define a \$SOURCE file with an adequate number of tracks. If a \$SOURCE file OCL statement is not specified, a 5-track \$SOURCE file is automatically allocated on the unit from which \$CCPPF was loaded.

When reading from the source library, specify a \$SOURCE file with enough tracks to contain all the printer format statements in the source library entry. A COMPILE is also required with the following parameters:

```
// COMPILE SOURCE-name of source data,UNIT- $\left. \begin{array}{l} (R1) \\ F1 \\ (R2) \\ F2 \end{array} \right\}$ 
```

where UNIT is the location of the source library.

### Notes:

1. For multiple format builds in a single run, the input stream must not contain delimiters between the printer formats. Instead, the H in column 1 of the printer control form indicates the start of another input.
2. Enter an asterisk (\*) in column 1 to identify a comment statement. Comments may be used to include notes on the listing produced by the Printer Format Generator routine. Comments can be placed anywhere in the format build; however, on multiple format builds, all comments entered before the printer control form of the second or later format builds are printed with the previous format build.

## Printer Format Generator Diagnostic Messages

During its processing, the printer format generator (PFGR) diagnoses errors in the Printer Format Specifications and logs diagnostic messages on the system logging device. Refer to the *CCP Messages Manual*, GC21-5170, for a list of the diagnostic messages.

Because the PFGR also calls the Overlay Linkage Editor to put the formats into the object library, it is possible to get Overlay Linkage Editor halts during PFGR processing.

## DISPLAY FORMAT CONTROL ROUTINE (DFCR)

The display format control routine operates under CCP to support formatted displays (screen or printer), as generated by DFGR and PFGR, for the IBM 3270 Information Display System.

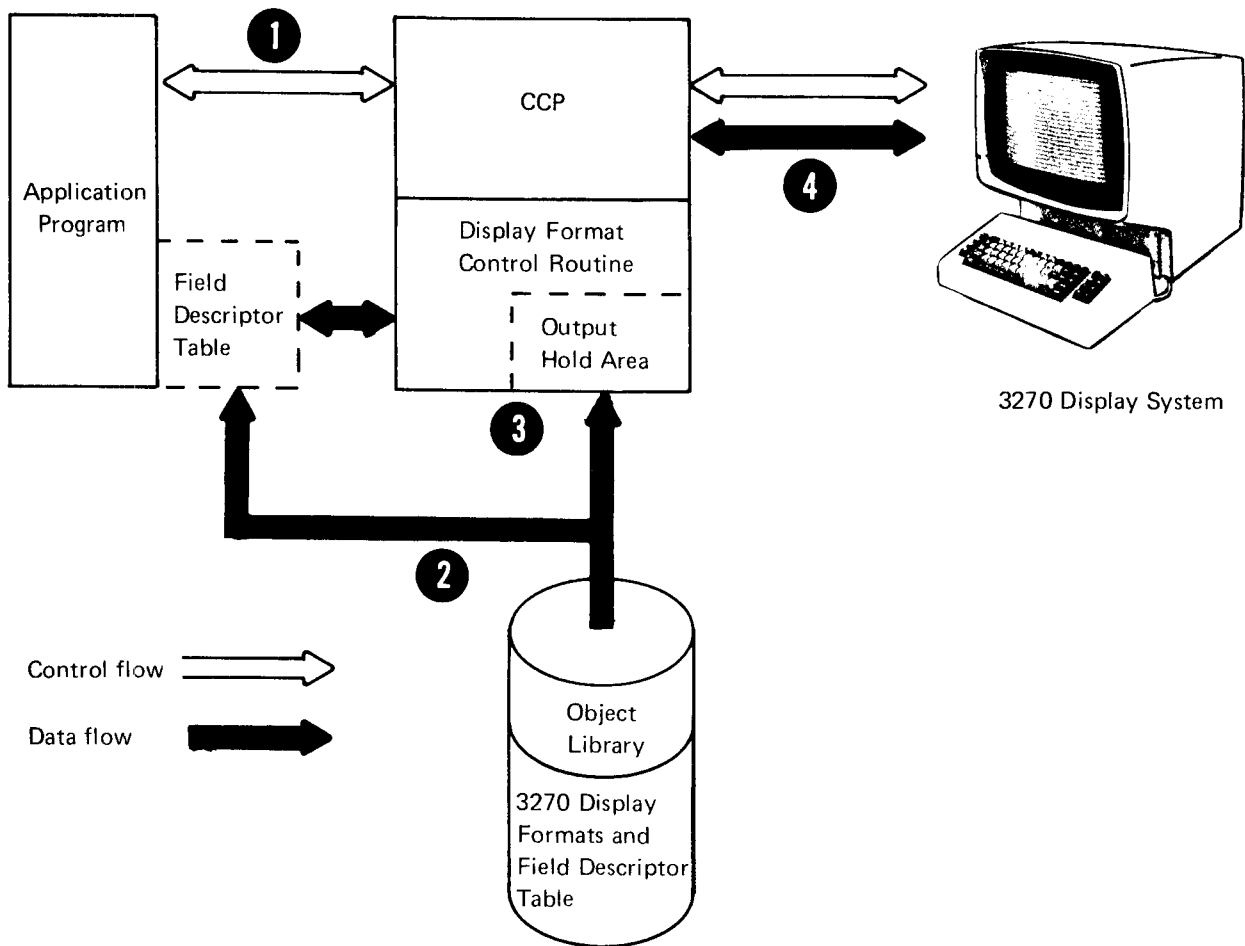
*Model 10 and Model 12 DFCR:* The control routine is loaded along with the first application program that requires DFF. In addition to the usual user allocation and initiation tests, CCP must make sure there is enough storage for DFCR before the program can start. The DFCR and the output hold area(s) are loaded at either the upper or lower boundary of the user program area and remain in main storage until the last application program using DFF terminates. The space occupied by the DFCR will then be made available for other uses.

*Model 15 DFCR:* The DFCR is loaded into the resident portion of CCP during Startup and remains resident in main storage throughout the CCP run.

The facilities can best be described in terms of output and input operations. Figure 8-10 shows an example of the operations performed by DFCR.

### *DFCR Functions for Output are:*

- Automatically retrieves the printer or display format requested by the application program.
- Merges execution time data supplied by the application program with the 3270 data stream at the proper field location; checks numeric fields for negative values.
- If necessary, splits the 3270 data stream into blocks and sends each block separately to the 3270. See *Hold Area* for a discussion of blocking.
- Provides application programs with the capability of modifying (overriding) the attributes, data content, or both of any field currently at the display. It also allows the application program to receive input from only selected fields.
- Provides the Copy operation which can be used to transfer a printer or display format between devices attached to the same control unit.
- Provides the Erase operation which can be used to clear all unprotected fields and reset the device for data entry for the terminal operator.



- 1 An application program running under CCP requests a particular format by issuing a Put Message operation, specifying the symbolic name of the format. (The symbolic name is defined in columns 2-7 of the display control form.)
- 2 The Display Format Control Routine (DFCR) reads the requested format from the object library. (Format was placed in the object library by the Display Format Generator or the Printer Format Generator.)  
The first part, the field descriptor table, is read into an area appended to the user program.
- 3 The second part, the 3270 data stream for this format, is read into the output hold area appended to the DFCR.
- 4 The format is passed from the output hold area to CCP for transmission to the display station.

Figure 8-10. DFCR Functions



### DFCR Functions for Input are:

- Creates a data record area based on fields received, or not received, from the display format at the terminal.
- Pads alphameric data on the right with spaces; right-justifies data in numeric fields and pads it on the left with spaces.
- Examines numeric fields for negative values.

### 3270 DISPLAY OPERATIONS

Requests for 3270 display operations are like other CCP communications I/O requests; that is, each request is issued through the communication service subroutine (by means of a macro in Basic Assembler) and must be accompanied by a parameter list and a record area. The parameter list contains information necessary for the operation requested as well as a location in which CCP will set a return code indicating the results of the operation. For certain operations, the programmer must supply additional information in the record area besides the terminal name and data. For example, when requesting that a display format be written to a 3270, the name of the display format is given in the record area following the terminal name.

See the sections for each language to determine such things as: the placement of op codes, return codes, input/output lengths, and symbolic terminal names.

DFF operations are always in *message mode* (see index entry). Record and block mode operations are not valid when using DFF.

#### Operation Considerations With DFF

Copy, Erase, and Put Override are operations provided specifically for DFF users. These operations are described in detail in this section. Use of the Put Message operation to place the initial format on the screen or printer is also described. Considerations for using the remaining CCP operations under DFF are given in this section and under *Operations in Chapter 2*. See *Examples* at the end of this chapter for examples of using CCP operations under DFF.

### Put Message

The Put Message is used to request that a particular printer or display format be retrieved from the object library and sent to the designated terminal. This operation is used to send either a complete new format or an overlay format to a terminal (see *Display Concepts for Overlay Screens*).

Hex	Dec	RPG II	Meaning
0032	50	WCB	PUT Message

One of the above values must be used depending upon the programming language. The name of the printer or display format to be sent to the terminal must appear in the first six positions of the record area immediately after the terminal name. If the name of the format is less than six characters in length, it must be left justified and padded with blanks. The output length must include the six-character format name. If there is no data to be added to the format (no fields defined as requiring execution time data), the output length is 20 (six for non-RPG SUBR92).

If data is to be added to the format at execution time, that data should start in the first data position after the format name. The amount of space to be provided for each field must be equal to the length of each field as defined on the format description sheet. The order of the fields provided must be the same as the order that they were defined to DFGR or PFGR. The output length specified with the operation is determined by using the format generator printout for this format. Use the end position of the last output field for RPG SUBR92 (less 14 for all others) as your length (the format name is included in this end position). The result of the Put Message is controlled also by the display format itself. Functions controlled by the display format are:

- Positioning of the cursor.
- Whether or not the display will be cleared before writing the requested display format.
- Device operations as initiated by the WCC.
- Printer control through new line and end-of-message orders.

The Put Message is changed automatically to Put Block by the control routine if blocking is required. (Refer to *Storage Areas – Output Hold Area*.)

### Program Request under Format Put Message

The PRUF Put message is used to format the screen with data to be used by the next program requested from the designated terminal. The PRUF-Put override or PRUF-Put message operation normally is the last operation before releasing the terminal or going to end of job.

Hex	Dec	RPG II	Meaning
0072	114	␣␣GB	PRUF Put Message

One of the above values must be used depending upon the programming language. The name of the format to be sent to the terminal must appear in the first six positions of the record area immediately after the terminal name. If the name of the format is less than six characters in length, it must be left justified and padded with blanks. The output length must include the six-character format name. If there is no data to be added to the format (no fields defined as requiring execution time data), the output length is 20 (six for non-RPG SUBR92).

As with the Put Message operation, DFF will automatically block this operation if blocking is required. See *Example 4* at the end of this chapter for more information on this operation.

### Put-No-Wait

This operation is defaulted automatically by DFF to a Put Message operation.

Hex	Dec	RPG II	Meaning
0036	54	␣␣CF	Put-No-Wait

### Put Override

The purpose of the Put Override operation is to (1) change the type, reposition the cursor, modify the data content, or any combination of these things for any fields defined in the format, or to (2) restructure the input record (only selected input fields are received).

### Operation Code

Hex	Dec	RPG II	Meaning
0832	2098	␣HCB	Put override

*Note:* RPG II allows the Put Override to be combined with an Invite Input operation (see index entry *Put with Invite Input*).

### Additional Requirements

- The format must have been previously sent to the printer or display by a Put Message or Copy operation.
- The output length specified by the user in the parameter list must define the exact length of the override list for RPG SUBR92 (less 14 for all others), starting with the WCC and including all field information in the list. (Can be only the WCC, if desired.)
- A DFF numeric field with a negative numeric content must have the sign over the units position. Space must be allocated in the format for the sign. DFCR moves the sign from over the units position to a position after the data (see *DATA in Numeric Fields*).
- Information for a Put Override must appear in the user's output record area. Fields are to be defined in the same order as defined at generation time.

### Information Returned

- Return codes (see explanation in *Appendix E*):
  - 0 Successful
  - 9 Terminal offline
  - n Negative return codes (I/O errors)

## Function and Use of Put Override

The Put Override allows the application programmer to change the field where the cursor is to be positioned, or to change the type, modify the data content, or do both for any field currently being displayed. Only the display is changed, not the display format in the object library or the field descriptor table in main storage. By specifying the appropriate WCC with the Put Override, the programmer controls device operations such as starting the printer, resetting the keyboard, and sounding the alarm (see *Selecting the WCC*).

The application programmer can also elect to receive input on the next input operation from the terminal from only those input, input/output, and SPD fields in the override list by specifying the appropriate WCC. The control routine checks the WCC. If the reset modified data tag bit is OFF in the WCC, the modified data tags for all fields are not changed and, on the next input request, data will be provided for all input fields, output/input fields, and for SPD fields that would have been received normally. If the reset modified data tag bit is ON in the WCC, the modified data tags for all fields, protected or unprotected, are turned to OFF. On the next input request, data from only those input, output/input, and SPD fields specified in the override list will be passed to the application program.

When selecting input fields using a Put Override, the reset modified data tag bit in the WCC must be ON (described previously). The user must be aware that this sets the modified data tag OFF for Input types 1, 2, 3, 4, 7, and 8, output/input field types 1, 2, 3, 4, 7, and 8, and SPD field types 3 and 4. Also, for the SPD fields, the designator (>) is not changed to the designator (?). If any of the selected input fields are one of these field types, the application programmer should not leave the type entry for that field blank but should instead provide the type entry for that field as override information even though the field type is not to be changed. This sets the modified data tag ON as required by the definition of these field types.

The format of the input record area, when selecting input fields, is determined by the order, and field length, of the fields as they are defined in the override list. The program logic must indicate the last output operation to the terminal and process the data on the next input from the terminal according to that indication. The layout of the input record area must be defined for each variation of input structure. Regardless of the input, the AID character always precedes the fields in the user's input record area and should be examined by the application programmer before the field data is processed.

## Program Request under Format Put Override

Hex	Dec	RPG II	Meaning
0872	2162	ØHGB	PRUF Put Override

The PRUF Put Override functions just like the normal Put Override operation with the added feature of formatting the screen with the data to be passed to a following program. The PRUF-Put Override or the PRUF-Put message operation is normally the last operation issued before releasing the terminal or going to end of job.

The information in the output record area, shown in Figure 8-11, consists of:

- **Field Name.** As defined in columns 1-6 of the field definition form. The name must be left justified and padded with blanks to a length of six characters. Use the same symbolic name as defined to PFGR or DFGR. Fields in an override list must be specified in the same order as during format generation.
- **Field Type.** If the field type is to be changed, this entry must contain the number of the type wanted. Leave blank if type is not to be changed. Allowable changes are:
  - Output Fields—all types are interchangeable.
  - Input Fields—types 1, 2, 5, and 7 are interchangeable; types 3, 4, 6, and 8 are interchangeable.
  - Output/Input Fields—types 1, 2, 5, 7 and 8 are interchangeable; types 3, 4, and 6 are interchangeable.
  - SPD Fields—all types are interchangeable.
  - The field type must be blank if the DFF format was generated using the printer format generator routine (PFGR).
- **Cursor Position.** A C entry positions the cursor in the first character position of this field. If the cursor is specified for more than one field, the last field with a C specified positions the cursor. A blank entry does not move the cursor.

This entry must be blank if the DFF format was generated using the printer format generator routine (PFGR).

- Erase/Modify Data Indicator. Enter an M if there is override data provided in subsequent positions. Such a change can be made in output, output/input, or SPD fields. A blank entry means data in the field is not to be changed.

Enter an E if the data input or output/input field at the terminal is to be erased to nulls. The erase indicator is valid only for input or output/input field types. If the E indicator is used, the next override field name begins in column 17.

- Data. Only include this field if M is specified for modify data indicator. The M is immediately followed by data equal in length to the field length as defined during format generation time (length of SPD fields does not include designator character). If M is not specified for the modify data indicator, the next field name begins in this position.

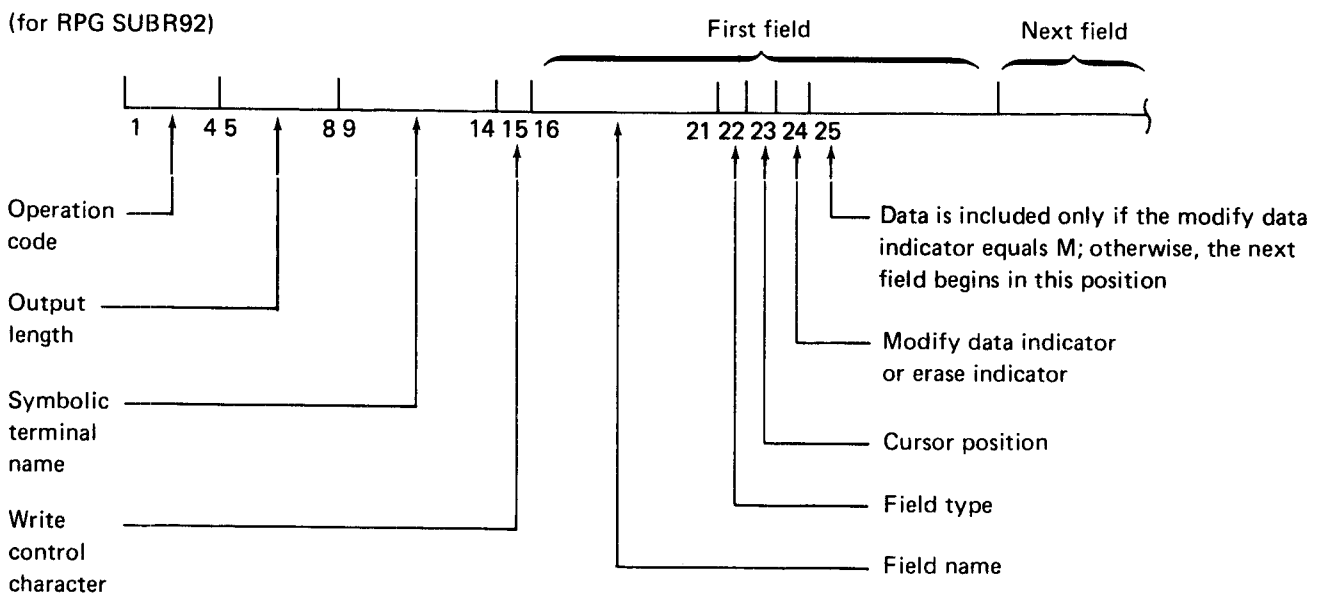
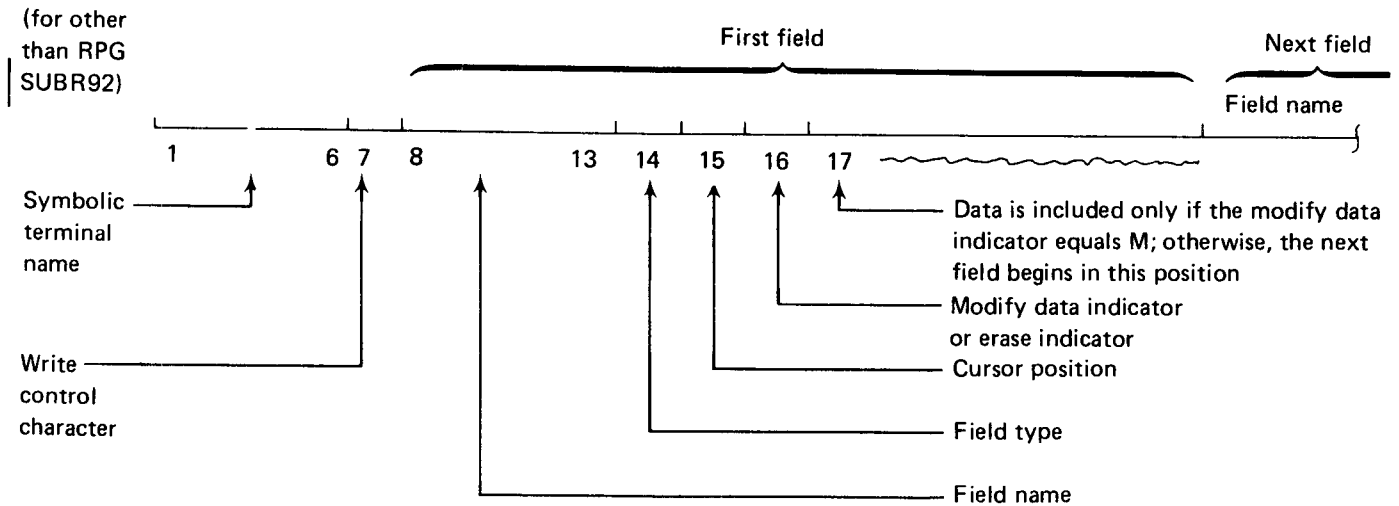


Figure 8-11. User Output Area

The output length consists of:

- One for the WCC
- Plus the sum of the following information for *all fields* to be overridden:
  - Six for the field name
  - Three (one space for each of the three possible entries for type, cursor, and modified data indicator)
  - Field data length as defined at generation time (if data is provided)
- An additional fourteen spaces if RPG SUBR92 is used.

#### Considerations

- With a Put Override operation, the application programmer can select to receive input from only those input, output/input and SPD fields specified in the override list. The field names must appear in the same order as they were given during format generation.
- To restore the display to its original condition, the application programmer could issue a Put Message or another Put Override.
- The override list can consist of a WCC only, if desired, (for example, to restore the keyboard). In this case, input received from the terminal may be affected (although there are no fields specified), if reset MDT is also ON in the WCC. A Put Override of the WCC should be used only to print the display currently on a 3275 on an attached 3284 Model 3 printer.
- An entry in a Put Override list may consist of a field name only, with blanks for type, cursor, and modify data indicator. In this case, if the WCC specifies reset MDT, then this field would be selected for input on the following input operations. If the WCC does not specify resetting of the MDT, this override entry is ignored and no 3270 text is generated.

#### Example 1

The structure of the record area should be as follows:

1. If RPG SUBR92 is used, the op code must be given in positions 1-4 and the output length must be given in positions 5-8.
2. The six-character symbolic name of the terminal.
3. The WCC. This field is mandatory. The WCC can be determined by following the directions in Figure 8-6.
4. For each field which is to be overridden, the following nine positions must be given or reserved in the output record area:
  - a. Six positions for a field name (left justified)
  - b. One position for type
  - c. One position for cursor (C)
  - d. One position to indicate modified data (M) or erase order (E)
5. The modified data indicator of M means that data is to be provided. If no data is to be provided (no M indicator) the record continues with the next symbolic field name.

#### Example 2

A primary use of the Put Override operation is interactive error detection/correction with input data being analyzed by the program logic. If any input fields are found to be in error, a Put Override operation is performed to the terminal with those fields in error displayed in intensified mode. Additionally, an error message can be displayed, the audible alarm sounded, and the cursor positioned at the first field in error. The RPG II programming example in Figure 8-12 illustrates the logic necessary to perform these functions.

The input fields are diagnosed in sequence, with the two arrays initialized to blanks and array index B advanced for each field. When an invalid input is diagnosed, the subroutine ERROR is executed, with indicator 63 set on for numeric fields. The cursor is positioned at the first field in error under the control of indicator 58.

When all the fields have been diagnosed, indicator 61 set on indicates that a Put Override operation should be performed.

**IBM** International Business Machine Corporation Form GX21-9093  
Printed in U.S.A.

**RPG CALCULATION SPECIFICATIONS**

Program		Punching Instruction	Graphic	Card Electro Number		Page 1 2 of	Program Identification 75 76 77 78 79 80
Programmer		Date	Punch				

Line	Form Type	Control Level (L, O, L9, LR, SR, AN, OR)	Indicators						Factor 1	Operation	Factor 2	Result Field		Resulting Indicators	Comments	
			Not	And	And	Not	Not	Not				Name	Length			
01	C	SR							ERROR	BEGSR						
02	C	SR								SETON						
03	C	SRN58								MOVE 'C'	\$U, B		61		ERROR FOUND	
04	C	SRN63								MOVE '2'	\$V, B				SET CURSOR	
05	C	SR 63								MOVE '4'	\$V, B				INTENSIFY ALPHA	
06	C	SRN58								SETON					INTENSIFY NUM	
07	C	SR								SETOF						
08	C	SR								ENDSR						

**IBM** International Business Machine Corporation GX21-9090  
Printed in U.S.A.

**RPG OUTPUT SPECIFICATIONS**

Program		Punching Instruction	Graphic	Card Electro Number		Page 1 2 of	Program Identification 75 76 77 78 79 80
Programmer		Date	Punch				

Line	Form Type	Filename	Type (I/D/T/E)	Stacker # (Fetch)	Space	Skip	Output Indicators						Field Name	End Position in Output Record	P/B/L/R	Constant or Edit Word
							Before	After	Not	And	And	Not				
01	O	TERMOUT	E									'AUTO				
02	O												4		'HCB'	
03	O												8		'0098'	
04	O											PL, 5	14			
05	O												15		'F'	
06	O												24		'ERRMSG2 M'	
07	O												48		'**ERROR** INTENSIFIED DA'	
08	O												62		'TA IN ERROR	
09	O												68		'FIELD1	
10	O											\$V, 1	69			
11	O											\$U, 1	70			
12	O												77		'FIELD2'	
13	O											\$V, 2	78			
14	O											\$U, 2	79			
15	O												86		'FIELD3'	
16	O											\$V, 3	87			
17	O											\$U, 3	88			
18	O												95		'FIELD4'	
19	O											\$V, 4	96			
20	O											\$U, 4	97			

Figure 8-12. Put Override Example

## Selecting the WCC

Figure 8-13 is provided to assist in selection of the Write Control Character (WCC). To use the figure, determine the characteristics of the output device and the operation to be performed, then select the code from the chart. For example, the write control code that restores the keyboard and sounds the alarm on a display but does not reset the MDTs is an F.

Operation				Output Device Format				
Sound Alarm	Restore Keyboard	Start Printer (No for displays)	Reset MDTs	Display or Printer NL/EM Control	40-Character Print Line	64-Character Print Line	80-Character Print Line	
Yes	Yes	Yes	Yes		┘	?	"	
			No	+	;	>	=	
		No	Yes	G	P	X	7	
			No	F	O	W	6	
	No	Yes	Yes	(	)	_	'	
			No	<	*	%	@	
		No	Yes	E	N	V	5	
			No	D	M	U	4	
No	Yes	Yes	Yes	.	\$	,	#	
			No	¢	!	} <sup>1</sup>	:	
		No	Yes	C	L	T	3	
			No	B	K	S	2	
		No	Yes	Yes	I	R	Z	9
				No	H	Q	Y	8
	No		Yes	A	J	/	1	
			No	¢	&	-	0	

<sup>1</sup>This character is converted internally to hex 6A for a WCC by DFGR when generating a format, and by DFCR when using the WCC in a Put Override operation.

Figure 8-13. Write Control Characters

## Copy

The purpose of the Copy operation is to transfer a format between devices attached to the same 3271 Control Unit. See *Selecting the Copy Control Character* in this chapter.

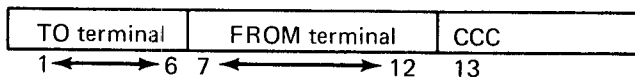
### Operation Code

Hex	Dec	RPG II	Meaning
0042	66	DDDB	Copy

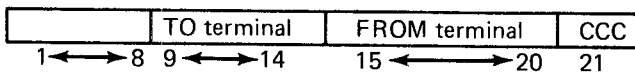
### Additional Requirements

The symbolic names of the terminal to receive the format (TO terminal) and the terminal to send the format (FROM terminal) appear in the output record area as follows:

Not RPG SUBR92:



RPG SUBR92:



The Copy Control Character (CCC) is optional and, if present, appears in the first position after the FROM terminal name. For a discussion of CCC, see *IBM 3270 Information Display System Component Description*, GA27-2749. A default CCC will be assumed if one is not given.

The output length should be 20 (six for other than RPG SUBR92) if the default CCC is to be assumed, or 21 (seven for other than RPG SUBR92) if a CCC is provided.

The FROM and TO terminals must be connected to the same 3271 control unit. A Put Message or a Copy operation must have been successfully issued to the FROM terminal prior to issuing a Copy operation.

This operation cannot be performed on a 3275. Data currently displayed on a 3275 screen can be printed on an attached 3286 Model 3 Printer using a Put Override with the proper WCC. Data can be simultaneously displayed on a 3275 screen and printed on a 3286 Model 3 Printer by using a Put Message.

### Information Returned

Return codes (see explanation in Appendix E):

- 0 Successful
- 9 Terminal offline
- n Negative return codes (I/O errors)

### Function and Use of Copy

This operation allows a format to be transferred between two devices attached to the same 3271 Control Unit. The application programmer may choose to use this operation instead of a Put Message for performance reasons. The advantage of this operation is that line transmission time is reduced because only a few characters are sent on the BSCA line.

### Considerations

- When a complete format image is received by the TO terminal and an overlay format (see *Display Concepts*) was the last format sent to the FROM terminal, that overlay format will be the only part processed by the Display Format Facility.
- Default CCC for a model 1 terminal is X'5B' (graphic character '\$'): default CCC for a model 2 terminal is X'7B' (graphic character '#').
- A display format can be locked (made incapable of being copied) by defining a protected field starting at line 1, position 2. DFGR will automatically place the attribute (specifying protected field) at line 1, position 1.
- The output length for Copy does not include the "copy to" symbolic terminal name but does include the space for the "copy from" symbolic terminal name and, if given, the copy control character.
- If a Put Override with select input fields was previously issued to the FROM terminal, it will be assumed that the same operation was performed at the TO terminal.





## Selecting the Copy Control Character

Figure 8-15 is provided to assist in selection of the Copy Control Character (CCC). To use the figure, determine the characteristics of the output device and the operation to be performed, then select the code from the chart. For example, to copy the entire buffer of a display and to sound the alarm, the code is G.

### Notes:

1. The default CCC for a Model 1 terminal is \$ (hex 5B) and a Model 2 terminal is # (hex 7B).
2. When copying a display format to a terminal and input is expected from the copy-to terminal, it is strongly recommended that either the entire buffer or the attributes and protected CCC be specified to enable DFGR to identify all input fields.

Operation		Output Device Format				
		Sound Alarm	Start Printer (No for displays)	Display or Printer NL/EM Control	40-Character Print Line	64-Character Print Line
Copy Attributes Only	Yes	Yes	<	*	%	@
		No	D	M	U	4
	No	Yes	H	Q	Y	8
		No	b	&	-	0
Copy Attributes and Unprotected Fields	Yes	Yes	( )	—	'	
		No	E	N	V	5
	No	Yes	I	R	Z	9
		No	A	J	/	1
Copy Attributes and Protected Fields	Yes	Yes	+	;	>	=
		No	F	O	W	6
	No	Yes	¢	!	} <sup>1</sup>	:
		No	B	K	S	2
Copy Attributes and All Fields (entire contents of buffer)	Yes	Yes		¬	?	"
		No	G	P	X	7
	No	Yes	.	\$	,	#
		No	C	L	T	3

<sup>1</sup>This character is converted internally to hex 6A for a WCC by DFGR when generating a format, and DFGR when using the WCC in a Put Override operation.

Figure 8-15. Copy Control Characters

## Erase

The purpose of the Erase operation is to clear the data portion of all unprotected fields in the current format.

### Operation Code

Hex	Dec	RPG II	Meaning
0052	82	ØØEB	Erase

### Additional Requirements

- A Put Message or Copy operation must have been issued to this terminal prior to the Erase operation.
- The symbolic terminal name must be specified in the users output record area.
- Output data length = 14 (or 0 if non-RPG SUBR92).

### Information Returned

- Return codes (see explanation in Appendix E):
  - 0 Successful
  - 9 Terminal offline
  - n negative return codes (I/O errors)

### Function and Use of Erase

The Erase operation:

- Clears the data portion of all unprotected fields to nulls (X'00').
- Sets the modified data tag on all unprotected fields to off.
- Unlocks the keyboard.
- Resets the AID byte.
- Repositions the cursor to the first character location in the first unprotected field.

If the entire display contains protected fields, the keyboard is unlocked, the AID is reset, and the cursor is positioned at line 1, position 1.

### Considerations

- An Erase operation is effective only if something has been placed on the display by a Put Message or Copy operation. When Erase is used, be aware that unprotected output/input fields containing data will have data erased and MDT tags set to OFF. It is suggested a Put Message be used to restore the display to its original condition if output/input fields are being used.

### Return Codes

CCP return codes, including the special meaning of the data truncated return code under 3270 DFF, are described in Appendix E.

If a negative return code is given for a Copy operation, Put Message operation, or Put Overrides operation with reset modified data tags, or if the CLEAR key exception return code is given for an input operation, DFCR treats the terminal as if no format were ever issued to it. Thus, if a negative return code is given for one of these operations (or CLEAR return code) and the next operation is a Get, the program is cancelled with a termination code of FF.

### Input Operations - Accept Input, Get, Stop Invite Input

Input operations are used to request input data (message) from the terminal whose name is specified in the record area (or in the parameter array for RPG II). When using display format services, the input operation codes that require processing by DFCR are:

Get  
Invite Input  
Accept Input  
Stop Invite Input

The control routine moves data into the input areas if data was received successfully. For the manner in which fields are handled, see *Field Concepts* and *Record Concepts*. A Put Message or Copy operation must have been successfully issued to the terminal prior to issuing one of these operations.

The structure of the input record area (beginning in position 15 of the data area when using RPG SUBR92 or position 7 for other programs) is given on the printout of the format generation. The record area must be long enough to contain the end position +14 (+6 for other than RPG SUBR92 use) specified in the format generation listing. The first character will be the AID character followed by data for all input, output/input and SPD fields, in the order defined during format generation.

If a Put Override was previously issued to the terminal, and that operation specified select input fields (reset MDT in the WCC), then the input record area will contain the AID character followed by data for all the input, output/input, and SPD fields whose names appeared in the last Put Override list.

If a Put Override was not issued to the terminal, the maximum expected input length you put in the fourth element of the parameter list or array should be the end position given for all uses, under *length of input record area required in DFF program* on the format generation printout.

If a Put Override with select input fields was issued to the terminal, the maximum expected input length to be specified in the parameter list or array must include one for the AID character plus the total data length of all input, output/input and SPD fields whose names appear in the override list.

If a maximum expected input length is given which is less than the actual length, a data truncated exception return code will be passed to the program. In this case, the truncated length reflects the full data length of all fields that could fit into the area as specified by the maximum expected input length entry in the parameter list or array. Thus, complete fields are returned; no incomplete fields are returned as in non-DFF operations.

*Note:* Program request data is processed by DFF if the program requested is a DFF PRUF program, and PRUF is active on the 3270 terminal at program request time. Only in this case is the AID character returned in the first position of the user's area (See Chapter 2 for more information on PRUF). The effective input length field in the input record area as filled in by RPG SUBR92 shows the data length of all fields that can fit into the area plus the AID.

#### Invite Input

Invite Input requires a previously issued Put Message or Copy operation. The maximum expected input length is disregarded by DFF for this operation.

#### Release Terminal

This operation is used by the Display Format Control routine to maintain, by program, a list of terminals using DFF.

#### USER PROGRAM RECORD AREA

The size of the input and output record areas (or combined input/output record area) in programs using DFF is calculated from information provided in the printed output of DFGR. Figures 8-16 and 8-17 show the printed output from DFGR for two formats -- \$ZOREN and \$Z0009. The sizes of the record areas for programs using these formats would be calculated as follows:

- Size of the output record area (programs using RPG SUBR92):

	\$ZOREN	\$Z0009
- Highest end position (from DFGR printout)	0020	0020
Total	0020	0020

- Size of the output record area (programs not using RPG SUBR92):

- Highest end position (from DFGR printout)	0020	0020
- Less 14 positions used in SUBR92 record area only	0014	0014
- Plus 6 positions for terminal name	0006	0006
Total	0012	0012

- Size of the input record area (programs using RPG SUBR92):

Highest end position (from DFGR printout)	0193	0547
Total	0193	0547

- Size of the input record area (programs not using RPG SUBR92 SPECIAL):

- Highest end position (from DFGR printout)	0193	0547
- Less 14 positions used in SUBR92 record area only	0014	0014
- Plus 6 positions for terminal name	0006	0006
Total	0185	0539

\$ZOREN DISPLAY FORMAT INFORMATION

EXECUTION TIME DATA - OUTPUT AREA FORMAT - END POSITIONS FOR RPG PROGRAMS

* FIELD NAME	FIELD LENGTH	END POSITION	* FIELD NAME	FIELD LENGTH	END POSITION	* FIELD NAME	FIELD LENGTH	END POSITION
* DPCODE	004	0034	* LENGTH	004	0008	* TMNAME	006	0014
* \$ZOREN	006	0020						

INPUT AREA FORMAT - END POSITIONS FOR RPG PROGRAMS

* FIELD NAME	FIELD LENGTH	END POSITION	* FIELD NAME	FIELD LENGTH	END POSITION	* FIELD NAME	FIELD LENGTH	END POSITION
* RTCODE	004	0004	* LENGTH	004	0008	* TMNAME	006	0014
* - AID-	001	0015	* CUSNO	006	0021	* SNAME	022	0043
* SADDR1	022	0065	* SADDR2	022	0087	* SADDR3	022	0109
* QTY1	004	0113	* ITEMN1	008	0121	* QTY2	004	0125
* ITEMN2	008	0133	* QTY3	004	0137	* ITEMN3	008	0145
* QTY4	004	0149	* ITEMN4	008	0157	* QTY5	004	0161
* ITEMN5	008	0169	* QTY6	004	0173	* ITEMN6	008	0181
* QTY7	004	0185	* ITEMN7	008	0193			

LENGTH OF OUTPUT RECORD AREA REQUIRED IN DFF PROGRAM  
RPG \*SUBR92\* - 0020 OTHER - 0012

LENGTH OF INPUT RECORD AREA REQUIRED IN DFF PROGRAM  
RPG \*SUBR92\* - 0193 OTHER - 0185

INFORMATION FOR USE DURING CCP ASSIGNMENT STAGE

THE DECIMAL LENGTH OF THE FIELD DESCRIPTOR TABLE IS 1150  
THE DECIMAL LENGTH OF THE OUTPUT TEXT IS 1146  
THE DECIMAL LENGTH OF THE INPUT TEXT IS 0240

Figure 8-16. Display Format Generation (\$ZOREN)

EXECUTION TIME DATA - OUTPUT AREA FORMAT - END POSITIONS FOR RPG PROGRAMS

* FIELD NAME	FIELD LENGTH	END POSITION	* FIELD NAME	FIELD LENGTH	END POSITION	* FIELD NAME	FIELD LENGTH	END POSITION
* DPCODE	004	0034	* LENGTH	004	0008	* TMNAME	006	0014
* \$Z0009	006	0020						

INPUT AREA FORMAT - END POSITIONS FOR RPG PROGRAMS

* FIELD NAME	FIELD LENGTH	END POSITION	* FIELD NAME	FIELD LENGTH	END POSITION	* FIELD NAME	FIELD LENGTH	END POSITION
* RTCODE	004	0004	* LENGTH	004	0008	* TMNAME	006	0014
* - AID-	001	0015	* NAME#	020	0035	* NAME#	020	0055
* FIRST#	038	0093	* SECOND#	038	0131	* THIRD#	038	0169
* FORTH#	038	0207	* FIFTH#	038	0245	* SIXTH#	038	0283
* ONE	044	0327	* TWO	044	0371	* THREE	044	0415
* FOUR	044	0459	* FIVE	044	0503	* SIX	044	0547

LENGTH OF OUTPUT RECORD AREA REQUIRED IN DFF PROGRAM  
RPG \*SUBR92\* - 0020 OTHER - 0012

LENGTH OF INPUT RECORD AREA REQUIRED IN DFF PROGRAM  
RPG \*SUBR92\* - 0547 OTHER - 0539

INFORMATION FOR USE DURING CCP ASSIGNMENT STAGE

THE DECIMAL LENGTH OF THE FIELD DESCRIPTOR TABLE IS 0396  
THE DECIMAL LENGTH OF THE OUTPUT TEXT IS 0463  
THE DECIMAL LENGTH OF THE INPUT TEXT IS 0579

Figure 8-17. Display Format Generation (\$Z0009)

A program using both of these formats could use a single input record area and a single output record area. The size of each would be equal to the size required for the larger format. In this case, the size of the output area would be 12 for programs not using RPG SUBR92 or 20 for programs using RPG SUBR92. The size of the input record area would be 539 for programs not using RPG SUBR92 or 547 for programs using RPG SUBR92.

The output record area in these two examples allows for only the format name. An additional consideration for a larger size output record would be if the Put Override operation is issued by the program or if the program provides output data to display format fields during program execution. The output record area would then need to be large enough for the Put Overrides list created during program execution or the total of the output fields, (value under heading *length of output record area required in DFF program* from DFGR printout) whichever is larger.

The program could also use a single combined input/output record area. The combined record area (created in RPG when SUBR92 file is used) would need to be as large as the larger of the input or output record areas.

## DISPLAY CONCEPTS

### New Screens

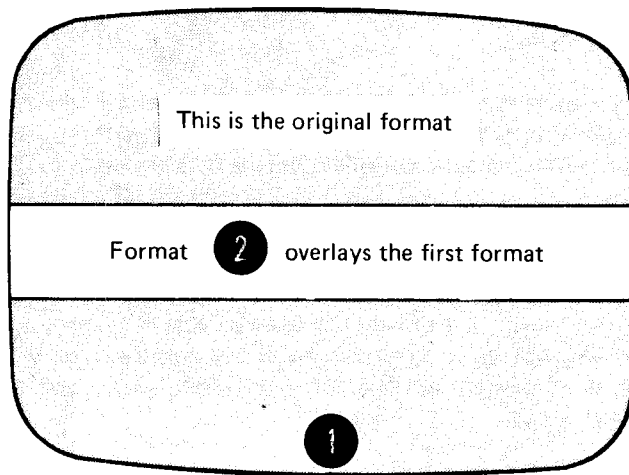
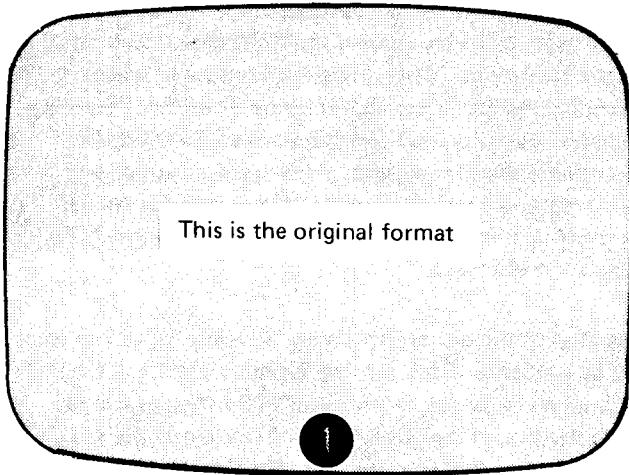
The Put Message is used to request that a particular display format be retrieved from the object library and written to the terminal. If the format is already at one terminal, a Copy operation can be used to duplicate the format from the first terminal to a second terminal attached to the same 3271 control unit. (*Note:* The format cannot be copied if a protected field starts in line 1, position 2.)

### Overlay Screens

Overlay screens are useful when only a portion of the screen is subject to change.

When multiple display formats are written to the same display, the integrity of the displayed data is the user's responsibility. The following must be considered if the user places a second format over a part of a format already being displayed without specifying 'clear before writing' on the display control form.

1. MDT should be set to OFF for all fields on Format 1 when Format 2 is placed over Format 1.



2. The control routine provides support for only the last format placed on the display. Only those input fields defined by the last display format will be accepted. All other field data will be ignored.

3. Entire fields or groups of fields should be overlaid to prevent residual data from appearing between fields in the overlay format. Also, residual data following an SPD field in the overlay format may be transmitted along with the data for the SPD field during an input operation. This can cause data for other fields in the overlay screen to be lost.
4. When using overlay screens, it is important that the terminal operator does not key into a field which belongs to a previous format. This could cause data for the fields in the overlay screen to be lost.
5. The overlay format need not define complete lines of data. An overlay format can define fields between fields of an existing format. It is important, however, that fields of an overlay format and their defining and delimiting attributes do not partially overlay an existing field or are not placed within the area of an existing field. Overlay fields should either be placed in an undefined area or should completely overlay existing fields.

## DFF, CCP CONSIDERATIONS

### Assignment Control Statements

During the assignment stage of CCP (see *CCP System Reference Manual*), the user must define those terminals which, on a program basis, require DFF support. The assignment control statements affected and the information to be specified are as follows:

**SYSTEM statement** – For DFF, a parameter on this statement identifies the disk unit and object library containing the display or printer formats.

**TERMATTR statement** – This statement defines terminal attributes. On this statement, the user specifies whether or not DFF will be used with all terminals referencing this terminal attributes set. Under Model 10 and Model 12 CCP, the BLKL parameter is used to define the size of the output hold area (in main storage). The output hold area is used to prepare the 3270 data stream for output operations. A separate hold area is supported for each BSCA line.

**BSCALINE statement** (5704-SC2 only)–DFFBUF parameter on this statement specifies whether or not a 2K buffer, separate from TP buffer, should be created for DFF puts. A yes option on this parameter would decrease (per BSCALINE statement) the size of the user program area by 2K but would potentially improve the data transmission and response time for multiple puts.

**PROGRAM statement** – This statement defines the logical structure and resource requirements of an application program. When using the Display Format Facility, the information required on a program basis is as follows:

- The maximum number of terminals, using the Display Format Facility, allocated concurrently to the application program.
- The maximum number of display or printer formats the program will use.
- The size of the largest field descriptor table used with this program. The character length of the field descriptor table is printed by the Format Generation programs (DFGR and PFGR) for each format generated.

### Storage Areas

#### *Output Hold Area—Model 10 and Model 12 CCP*

One output hold area for each BSCA line will be appended to the control routine to hold the 3270 data stream for output operations. When using Put Message or when overriding the attributes and/or contents of fields currently being displayed (see *Put Overrides* under *3270 Display Operations*) this area is used by the control routine to build a data stream based on the execution time data or the override information in the output record area of the application program.

The sizes of the output hold areas for each BSCA line need not be the same. Each may be the minimum of 256 bytes or as much larger (by multiples of 256 bytes) as needed up to the size of the output text of the largest display or printer format that uses this set. The size of the hold areas are determined by the BLKL parameter of the TERMATTR statement of the assignment set under which the program will execute.

If the assigned hold area space is less than the size of the output text of the largest display or printer format, the control routine automatically splits the data stream into output blocks and sends each block separately. If the control routine is to perform blocking, the hold area must contain at least 512 bytes in order to accommodate one 256-byte block plus information that cannot be sent in the current block. The user should be aware that if he uses blocking to achieve core economy, it will probably result in less efficient use of the BSCA line if the line has a high rate of activity.

#### *Output Hold Area—Model 15 CCP*

The output hold area is allocated dynamically by CCP. Program 5704-SC1 allocates the output hold area from the TP buffer area. Program 5704-SC2 allocates the output hold area from the TP buffer if optional DFF buffers are not specified. For Program 5704-SC2, if DFF buffers are specified, a separate 2K output hold area called a DFF buffer is allocated for each BSC line supporting a DFF buffer.

The size of the output hold area without DFF buffer support for Program 5704-SC1 and 5704-SC2 is determined by CCP. This size is based on the length of the output text of the format for each particular user program operation. There may be four output hold areas allocated at one time from the TP buffer; one for each BSC line.



## TP Buffer

Get, Invite Input, and (under Model 15 CCP) Put operations are not executed under DFF until buffer space is available. In calculating TP buffer area size at assignment time, the user should consider Invite Input, Get, and (under Model 15 CCP) Put operations that require DFF. See *Performance Tips* in the *IBM System/3 Communications Control Program System Design Guide*, GC21-5165 for information on estimating the size of this area.

## User Program Storage Considerations

In order for DFF to support multiple application programs concurrently, a storage area is appended to each application program when the program is loaded. The size and contents of this area are determined at assignment time.

The storage area contains a fixed constant area plus a variable area depending upon the following:

- Maximum number of 3270 terminals the program can service concurrently. (Take into account the maximum number of requesting terminals and program selected terminals that can be serviced at one time.)
- Number of display or printer formats used
- Length of the longest field descriptor table given on the PROGRAM statement.

The user must specify on the PROGRAM statement the size of the largest field descriptor table (information provided by the display or printer format generator). A particular field descriptor table remains in the storage area appended to the application program until an output or input request requires a different format. On output, the table identifies those fields to be supplied data by the user's output record area and also supports overrides to a field within the display. On input, the table aids in constructing the input record area from data received from a terminal. For storage size estimates, refer to *CCP System Reference Manual*.

## Terminal Operator Actions

### Keys

See *Operator's Guide for IBM 3270 Information Display Operator's Guide 3275, 3277, 3284, 3286, 3288*, GA27-2742 or *IBM 3270 Information Display System Component Description*, GA27-2749 for description of all 3270 keys and their actions.

Use of keys with DFF:

Key	Situation/Resulting Action
ENTER	When the operator completes an operation and presses ENTER, the fields with MDT-ON are read automatically in response to a poll request from CCP.
CLEAR	A 07 return code is returned to the application program when the operator of a terminal presses the CLEAR key (returned on the next input operation). An AID character indicating CLEAR is never returned to the user program, whether or not the program uses DFF. The 07 return code is given to the program if the terminal operator resumes communication with the application program after data mode escape.
Program Attention keys—ENTER, all Program Function (PF) keys, and the Program Access (PA) keys	When the operator presses any one of these keys, the terminal responds to an input request from the system and the Attention Identification (AID) character is generated (to identify which key was pressed). The AID is the first field in the input record. If a PA key is pressed, only the AID character appears in the input record area, no data.

If selector pen attention is caused by selection of an SPD field, each field that has the modified data tag ON will have a greater than (>) sign in the leftmost position of the field in the record area and the remainder of the field will be blank.

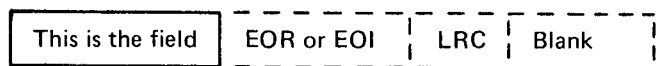
<i>Key</i>	<i>Situation/Resulting Action</i>
Character oriented keys	A cluster of four keys (located on the right of the keyboard) move the cursor one location at a time. These keys are: ↑ (up), ↓ (down), → (right), and ← (left). The cursor may be moved into any character location, including unprotected and protected alphameric character and attribute character locations. All four keys have typematic operation (keep repeating when key is held down); all four are capable of causing the cursor to 'wrap'.
Field oriented Keys	Four keys that move the cursor to the first position in a field. These are →  (Tab), ←  (Backtab), ←  (New Line) and the SKIP key. The tab key moves the cursor to the first character location of the next unprotected data field. If the cursor is located in other than the first location of an unprotected data field, the Backtab key causes the cursor to move to the first character location of that field. If the cursor is already located in the first position of a protected field, the cursor moves to the first alphameric character location of the first preceding unprotected data field. The New Line key moves the cursor to the first unprotected character location of the next line. The SKIP key is on the Data Entry Keyboard only and performs the same functions as the Tab Key.
ERASE INPUT	This key clears all unprotected fields to nulls and repositions the cursor to the first unprotected character location on the screen. It also resets all modified data tags associated with unprotected data fields.
ERASE EOF End of Field key	If the cursor is located in an alphameric character location in an unprotected data field, this key clears the position occupied by the cursor and all remaining character positions in that field. The cursor does not move. If this key is pressed when the cursor is located in a protected field, it disables the keyboard and no character locations are cleared.

<i>Key</i>	<i>Situation/Resulting Action</i>
FIELD MARK	This key provides a means of informing the application program of the end of a field in an unformatted display. If the key is used in entering data, no special handling is provided when using DFF.
DUP	This key provides a unique character code to be entered into the display buffer and a standard Tab key operation to be performed. The DUP character provides a means of informing the application program that a "duplicate" operation is indicated for the rest of the field in which it is located. If key is used in entering data, no special handling is provided when using DFF.

#### *Operator Identification Card Reader*

The Operator Identification Card Reader is a special feature for attachment to a 3277 or 3275 Display Station. It reads a small magnetic card encoded with a unique identification number. When the card is placed into a reader, the text on the card is read into the display buffer. For the card read operation, the cursor must be positioned at the first character location of the input field. The field should be an unprotected alphameric field. The AID is set to indicate the operator Identification Card Reader.

The field length should always be the length of the text plus three. The alphameric data which will be presented to the application program consists of text, an End of Record (EOR) or End of Inquiry (EOI) character, the LRC character, and one blank.



There is a hardware-generated attribute character entered automatically at the location of the cursor. This attribute character must be removed by the application program if this field is to be re-used. This can be accomplished by issuing either a Put Message or Put Override operation.

## DISPLAY FORMAT TEST ROUTINE (\$CCPDT)

\$CCPDT is a stand-alone program which displays DFF formats on an IBM 3270 Information Display System.

\$CCPDT operates on the IBM System/3 Model 10, 12, or 15. Communication between \$CCPDT and the 3270 is maintained using the multiline/multipoint (MLMP) binary synchronous communications (BSC) data link.

\$CCPDT, shown in Figure 8-19, uses the formats generated by the Display Format Generator routine (DFGR) or the Printer Format Generator routine (PFGR). The formats must be in the object library of the program pack from which \$CCPDT is loaded.

For Model 10 and Model 12, \$CCPDT will use SYSIN as the input device. When \$CCPDT is loaded the following option menu will be logged on SYSLOG:

### OPTION MENU

```
A. SEND FORMAT TO TERMINAL
B. READ FORMAT FROM TERMINAL AND PRINT IT
C. GO TO EOJ
D. SEND FORMAT TO TERMINAL, POLL, PRINT INPUT
E. SEND FORMAT TO SYSTEM PRINTER
ENTER ALL OPTIONS DESIRED THROUGH SYSIN IN ANY ONE OF TWO FORMS BELOW
O FFFFFFF PPPP AAAA
O FFFFFFF *
```

For Model 15, \$CCPDT will use the console as the input device. An option menu will appear on the screen describing each option. The input data may be entered starting at the cursor position.

```
A. SEND FORMAT TO TERMINAL
B. READ FORMAT FROM TERM.,AND PRINT IT
C. GO TO EOJ
D. SEND FORMAT TO TERM.,POLL,PRINT INPUT
E. SEND FORMAT TO SYSTEM PRINTER
1)O FFFFFFF PPPP AAAA      2)O FFFFFFF *
```

↑  
Cursor

\$CCPDT provides the user with two forms of input records and five options:

1. O FFFFFF PPPP AAAA

where:

O = A, B, C, D, or E (these options are described later in this chapter)

F = Format name, which must begin with the characters \$Z. The name must be at least three characters long but not more than six characters long.

P = Poll characters  
Enter immediately following the first blank after the format name.

A = Address characters

2. O FFFFFF \*

where:

O = A, B, C, D, or E (these options are described later in this chapter)

F = Format name, which must begin with the characters \$Z. The name must be at least three characters long but not more than six characters long.

\* = Last specified poll and address characters (see note). Enter immediately following the first blank after the format name.

*Note:* If the poll and address characters are not specified, \$CCPDT will use the last previous specified poll and address character. In this way the user can send multiple formats to the same terminal. If the user does not specify the poll and address characters at least once, \$CCPDT will default to the following characters:

Poll characters = 40404040

Address characters = 60604040

*Note:* \$CCPDT only supports EBCDIC on BSCA or DA.

## Options

### A. *Send Format to Terminal*

\$CCPDT reads the selected format from the program pack. The format is then transmitted to a terminal. This option allows the user to view the format. This option does not allow input or output data to be received or transmitted.

### B. *Read Format from Terminal and Print It*

This option allows the user to test for invalid characters within the text stream. \$CCPDT reads the entire format from the terminal (read buffer) and prints it. The user should compare the format on the terminal and the data portion (text stream) of the format printed. (See Figure 8-19 for example of printout.)

### C. *Go to End of Job*

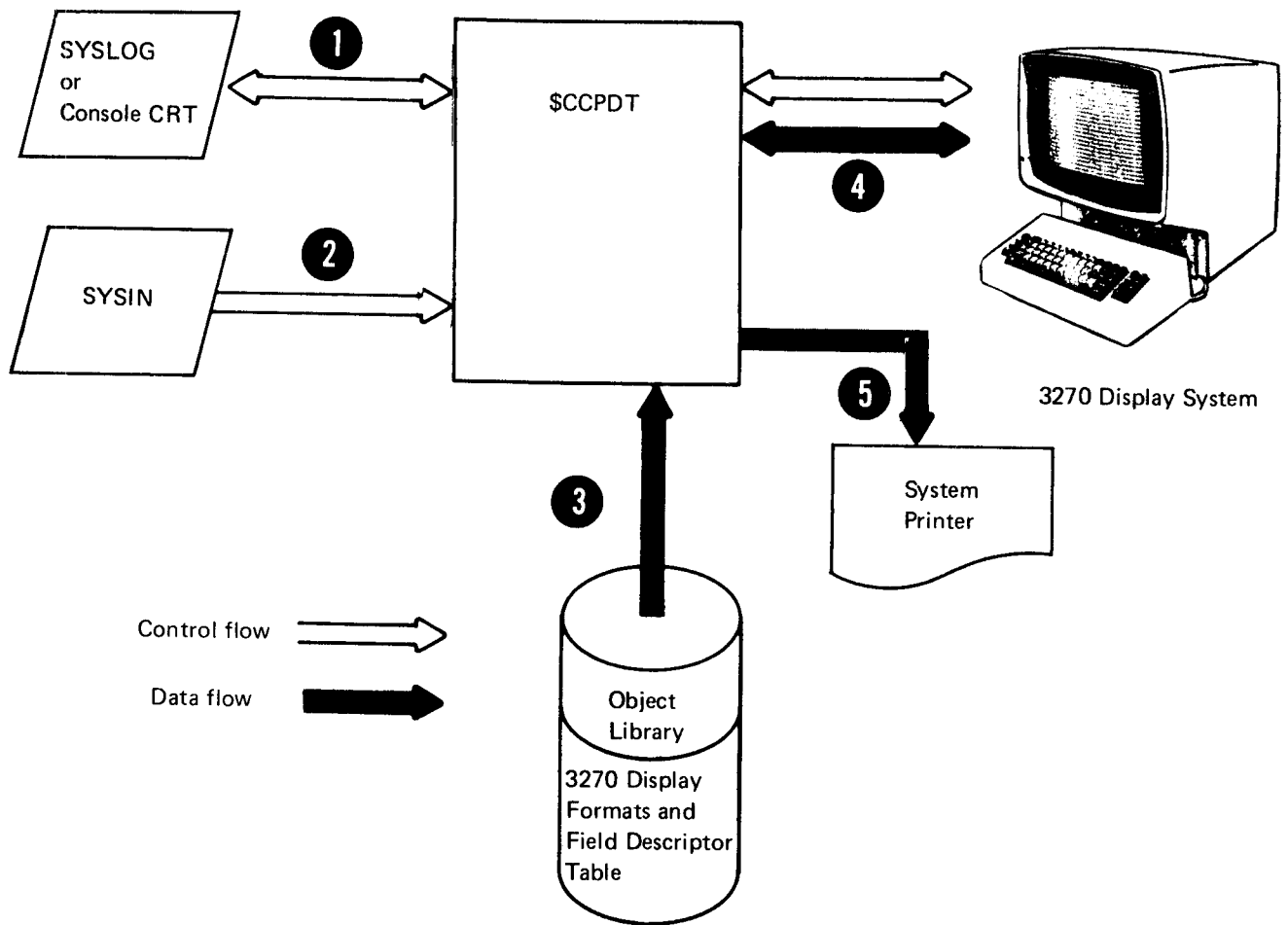
This option should be entered after the user has entered all other options to be used. \$CCPDT will then go to end of job.

### D. *Send Format to Terminal, Poll for Input and Print the Input Data Received*

\$CCPDT sends the format to the terminal and polls the terminal for input. The user enters data on the terminal keyboard. When attention is requested by the terminal operator by pressing either the ENTER, PA or PF key, \$CCPDT receives the modified data and prints it on the system printer. This is a means of verifying the modified data transmitted to \$CCPDT or what the text stream would look like if sent to a non-DFF application program. (See Figure 8-18 for example.)

### E. *Send Format to System Printer*

\$CCPDT reads the selected format from the program pack and prints the entire format on the system printer. This allows the user to view the 3270 text stream and provides a copy for documentation. (See Figure 8-19 for example of printout.)



- 1 \$CCPDT prints/displays the options menu on the SYSLOG device.
- 2 \$CCPDT reads an option from SYSIN.
- 3 \$CCPDT reads the format name specified on the option record from the object library of the program pack.
- 4 \$CCPDT sends the format to the terminal.
- 5 \$CCPDT sends the format to the system printer, or sends modified data from the format to the system printer.

Figure 8-18. \$CCPDT Functions

Address	Text of Format in Hex						Text of Format in Character
\$ZRPIQ							
0000	27F5C332	3211C6D4	1D60C3E4	E2E3D6D4	C5D940D5	D64811C6	5C FM -CUSTOMER NO. F
0018	F31DEA5C	5C5C5C5C	5C11C66D	1D60D5D6	F340D6D5	40C6C9D3	T Y***** F XNOT ON FIL
0030	C511C8F4	1D60C3E4	E2E3D6D4	C5D940D5	C1D4C511	C9C31DE8	F H4 -CUSTOMER NAME IC Y
0048	5C5C5C5C	5C5C5C5C	5C5C5C5C	5C5C5C5C	5C5C5C5C	5C5C1148	*****
0060	D41D60F2	C8C9D74C	E3D61148	E31DE85C	5C5C5C5C	5C5C5C5C	M -SHIP TO JT Y*****
0078	5C5C5C5C	5C5C5C5C	5C5C5C5C	5C114FC4	1D60C9E3	C5D440D5	***** D -ITEM N
0090	D648114F	D31DE85C	5C5C5C5C	5C5C5C11	4F5D1D6C	D5D6E340	O. L Y***** XNOT
00A8	D6D540C6	C9D3C511	D1E41D60	C9E3C5D4	40C4C5E7	C34811D1	ON FILE JU -ITEM CFSC. J
00C0	F31DE85C	5C5C5C5C	5C5C5C5C	5C5C5C5C	5C5C5C5C	5C5C5C	3 Y*****

Figure 8-19. Example of \$CCPDT Format Printout

### \$CCPDT Considerations

Any format that the user wants to test must be in the program pack's object library. Before a format is sent to a terminal or the printer it is scanned for nulls (00). In fields within formats defined as having execution time data containing nulls, the nulls are replaced by asterisks. Also in overlay formats with input fields containing nulls, these nulls are replaced by asterisks. Since nulls do not appear on the terminal, by replacing the nulls with asterisks, the user is able to see the null fields.

### OCL Statements

An initialized MLTERFIL must be on F1, for logging control station terminal error statistics. The OCL required to initialize MLTERFIL is:

```
// LOAD $$BSFI,unit
// FILE NAME-MLTERFIL,UNIT-F1,PACK-pack,
// TRACKS-1,LOCATION-track number (optional),
// RETAIN-P
// RUN
```

The following OCL is required to run \$CCPDT:

```
// LOAD $CCPDT,unit
[// BSCA LINE-2 ] see notes
// RUN
```

### Notes:

- \$CCPDT uses BSCA-1; if the user wants to use the BSCA LINE-2 instead of BSCA LINE-1, include the optional BSCA LINE-2 statement in the OCL. No recompilation of the program is required.
- When using the ICA or local display adapter, the user must enter the BSCA LINE-2 OCL statement.
- \$CCPDT supports 3270s only on BSCA lines, not on BSCC lines.

### Display Format Test Routine Diagnostic Messages

During processing, DFTR checks completion codes, status of terminals, and the various types of program intervention. Messages stating the reason the completion code was given, the status of terminals, and the type of program intervention received are logged on the system printer. For the DFTR diagnostic messages, see the *CCP Messages Manual*, GC21-5170.

### FORMAT FIND ROUTINE (5704-SC2 Only)

The format find routine (CCPFMT) makes formats that have been created or modified by DFGR or PFGR in another partition, available to programs in the current assignment set. The routine is executed under CCP and performs the following functions:

- Places the current object library C/S (cylinder/sector) location of the format into the format index in \$CCPFILE.
- Updates the DFFSFDT value of the PROGRAM assignment statement.

To use CCPFMT, make the following entries in the assignment set:

```
// TERMATTR ATTRID-04,DATAFORM-MESSAGE,
// DFF3270-NO
// PROGRAM NAME-CCPFMT,NEVEREND-YES,
// ENDMSG-NO
```

The routine is invoked from either the system operator's console or a command terminal. For additional information on the use of CCPFMT, refer to the *Model 15 CCP System Operator's Guide*, GC21-7619.

**Note:** When CCPFMT is included in an assignment set, ATTRID-04 required for format find.

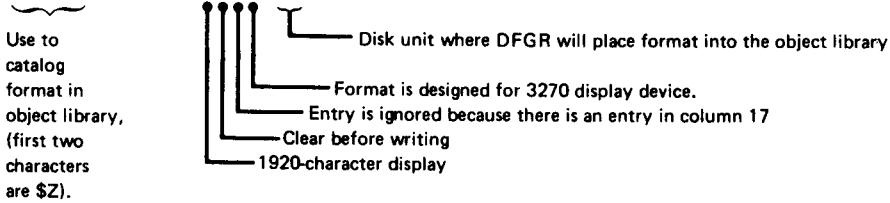
**EXAMPLES**

**Example 1—DFF Formatting Example**

**Problem:** Design a display format to be used on the 3270 display with an order entry program. Make the format usable with any programming language available under CCP.

**Solution:** Plan the format by laying out fields on a display layout sheet. This then serves as a visual guide in filling out the field definition forms.

<b>IBM</b> International Business Machines Corporation		<b>DISPLAY FORMAT SPECIFICATIONS</b>										GX21-9175 Printed in U.S.A.																			
Application <b>ORDER ENTRY</b>				Completed by				Punching Instructions				Page <b>1</b> of <b>4</b>																			
								Graphic				Date <b>3/13</b>																			
								Punch																							
<b>DISPLAY CONTROL</b>																															
<b>C</b>													<b>PRINTER CONTROL</b>																		
Form Type	Display Name	Field Name for Initial Cursor Position	Display Size (1,2)	Clear Before Writing (Y/N)	Special	Screen Default (X)	Printer Default (X)	Disk Storage Unit for Display Formats	(Reserved)	Line	Position	Line	Position	Line	Position	Line	Position	Line	Position	Line	Position	Line	Position	Line	Position	Line	Position	Line	Position	Continuation (X)	Identification Sequence
C	\$ZORENCUSNO		2Y	X			R2																								



LINE	1-10	11-20	21-30	31-40	41-50	51-60	61-70	71-80	81-90	91-100	101-110	111-120	121-130	131-140				
A	CUSTOMER NUMBER										CUSTOMER NAME				SHIP TO			
B	XXXXXXXXXX										XXXXXXXXXXXXXXXXXXXXXXXXXX				XXXXXXXXXXXXXXXXXXXXXXXXXX			
C	QUANT	TY	ITEM NO	DESCRIPTION							PRICE	QTY ON HAND	BACK ORDER					
D	XXXX	XX	XXXXXX	XXXXXXXXXXXXXXXXXXXXXXXXXX							XXXXXX	XXXX	XXXXXX					
E	XXXX	XX	XXXXXX	XXXXXXXXXXXXXXXXXXXXXXXXXX							XXXXXX	XXXX	XXXXXX					
F	XXXX	XX	XXXXXX	XXXXXXXXXXXXXXXXXXXXXXXXXX							XXXXXX	XXXX	XXXXXX					
	EXCISE TAX		STATE TAX		LOCAL TAX		TOTAL											
	OUT OF STOCK CUSTOMER NOT ON FILE ITEM DROPPED CREDIT OVER																	

**ORDER ENTRY**

Page 1 of 1 Date 3-2-73

KEY ASSIGNMENTS						
AID			AID		AID	
PF1	1	ENTER ORDER	PF6	6	PF11	#
PF2	2		PF7	7	PF12	@
PF3	3		PF8	8	PA1	%
PF4	4		PF9	9	PA2	>
PF5	5		PF10	:	PA3	column
						RELEASE TERMINAL
						CANCEL ORDER

- Shaded boxes indicate defining attribute positions.
- Open boxes indicate trailing attribute positions.



**Note:** The large letters identify a portion of the display screen with a segment of the field definition information as it appears on the field definition forms and on the listing.

**A** This heading is a type 1 output field which means it is to be displayed at normal intensity on the screen.

The entry on line 2 position 3 (under customer number) is a type 3 input field which means that it is a numeric field to be displayed at normal intensity.

The entry on line 2 position 21 (under customer name) is a type 1 output field which means that it is displayed at normal intensity.

The entry on line 2 position 46 (under ship to) is a type 1 output/input field. This entry is examined by the terminal operator and changed if necessary.

Line 07 contains headings (type 1 output fields with normal intensity). The items in the quantity column are type 3 (numeric) while the item numbers are type 1 input (alphanumeric). Automatic skip causes the cursor to move to the next quantity entry.

The fields in the rest of the line are defined as type 1 output/input fields. Note that there is a G in the data source column indicating information is provided at generation time yet no entries appear in columns 32-71. All output and output/input fields have been described as being generation-defined with no entries made in columns 32-71 because data will be provided using a Put Override during execution time.

**E** Display line 22 contains visible headings (type 1 output fields) and fields into which data can be placed by using Put Overrides at execution time (type 1 output fields).

**F** Display line 24 initially contains a series of type 5 output fields. Any one of these fields can be changed at execution time by the application program. They could, for example, be changed to displayed fields with high intensity.

The display control form is used to describe the format in such a way that it can be properly stored and retrieved.

All the information on the display control and field definition forms is transferred to cards and used as input to the display format generator.

Generation results include:

1. The display format consisting of the field descriptor table and the 3270 data stream with control information are entered into the object library under the symbolic format name.
2. Printout information:
  - a. Printout of the specification forms.
  - b. Log of diagnostic error messages.
  - G** c. Sequential list of output fields.
  - H** d. Input data fields and their order of appearance in the input record area.
  - e. The decimal length of the field descriptor table.
  - f. The decimal length of the 3270 output data stream.
  - J** g. The decimal length of the 3270 input data stream.





C\$ZURENCUSNO 2Y X RZ  
 FHEAD1 0103 501G  
 F  
 FCUSNO 0203 6 3N  
 FNAME 0221 221G  
 FSNAME 0246 22 1GN  
 FADDR1 0321 221G  
 FSADDR10346 22 1GN  
 FADDR2 0421 221G  
 FSADDR20446 22 1GN  
 FADDR3 0521 221G  
 FSADDR30546 22 1GN

CUSTOMER NUMBER CUSTOMER NAME X  
 SHIP TO

A  
 B

\* THE FIELD NAMES 'BLANK-' WILL CAUSE A BLANK LINE TO PRINT NOT BE  
 \* SKIPPED ON THE 3284 AND 3286 MODEL 1 AND 2 PRINTERS.

FBLANK10602 11G  
 FHEADR10703 81G  
 FHEADR20712 71G  
 FHEADR30728 111G  
 FHEADR40748 51G  
 FHEADR50757 111G  
 FHEADR60770 101G  
 FQTY1 0803 04 3Y  
 FITEMN10812 08 1Y  
 FDESCR10824 201G  
 FPRICE10848 71G  
 FQTYUH10861 41G  
 FBACKU10873 41G  
 FBLANK20902 11G  
 FQTY2 1003 4 3Y  
 FITEMN21012 8 1Y  
 FDESCR21024 201G  
 FPRICE21048 71G  
 FQTYUH21061 41G  
 FBACKU21073 41G  
 FBLANK31102 11G  
 FQTY3 1203 4 3Y  
 FITEMN31212 8 1Y  
 FDESCR31224 201G  
 FPRICE31248 71G  
 FQTYUH31261 41G  
 FBACKU31273 41G  
 FBLANK41302 11G  
 FQTY4 1403 4 3Y  
 FITEMN41412 8 1Y  
 FDESCR41424 201G  
 FPRICE41448 71G  
 FQTYUH41461 41G  
 FBACKU41473 41G  
 FBLANK51502 11G  
 FQTY5 1603 4 3Y  
 FITEMN51612 8 1Y  
 FDESCR51624 201G  
 FPRICE51648 71G  
 FQTYUH51661 41G  
 FBACKU51673 41G  
 FBLANK61702 11G  
 FQTY6 1803 4 3Y  
 FITEMN61812 8 1Y  
 FDESCR61824 201G  
 FPRICE61848 71G  
 FQTYUH61861 41G  
 FBACKU61873 41G  
 FBLANK71902 11G  
 FQTY7 2003 4 3Y  
 FITEMN72012 8 1N  
 FDESCR72024 201G  
 FPRICE72048 71G  
 FQTYUH72061 41G  
 FBACKU72073 41G  
 FBLANK82102 11G  
 FTOTEX 2206 101G  
 FTAXEX 2218 41G  
 FTOTST 2225 91G  
 FTAXST 2236 41G  
 FTOTLT 2243 91G  
 FTAXLT 2254 41G  
 FTOTAL 2262 51G  
 FTOTMON2269 61G  
 FBLANKA2302 11G  
 FBLANK92402 11G  
 FOUTOS 2404 125G  
 FCUSERR2417 205G  
 FITEMDR2439 125G  
 FCREDOV2453 115G  
 /\*

QUANTITY  
 ITEM NO  
 DESCRIPTION  
 PRICE  
 QTY ON HAND  
 BACK ORDER

C  
 D

EXCISE TAX  
 STATE TAX  
 LOCAL TAX  
 TOTAL

E

OUT OF STOCK  
 CUSTOMER NOT ON FILE  
 ITEM DROPPED  
 CREDIT OVER

F

\$ZOREN DISPLAY FORMAT INFORMATION

G

EXECUTION TIME DATA - OUTPUT AREA FORMAT - END POSITIONS FOR RPG PROGRAMS

* FIELD NAME	FIELD LENGTH	END POSITION	* FIELD NAME	FIELD LENGTH	END POSITION	* FIELD NAME	FIELD LENGTH	END POSITION
* OPCODE	004	0004	* LENGTH	004	0008	* TMNAME	006	0014
* \$ZOREN	006	0020	*			*		

H

INPUT AREA FORMAT - END POSITIONS FOR RPG PROGRAMS

* FIELD NAME	FIELD LENGTH	END POSITION	* FIELD NAME	FIELD LENGTH	END POSITION	* FIELD NAME	FIELD LENGTH	END POSITION
* RTCODE	004	0004	* LENGTH	004	0008	* TMNAME	006	0014
* - AID-	001	0015	* CUSNO	006	0021	* SNAME	022	0043
* SADDR1	022	0065	* SADDR2	022	0087	* SADDR3	022	0109
* QTY1	004	0113	* ITEMN1	008	0121	* QTY2	004	0125
* ITEMN2	008	0133	* QTY3	004	0137	* ITEMN3	008	0145
* QTY4	004	0149	* ITEMN4	008	0157	* QTY5	004	0161
* ITEMN5	008	0169	* QTY6	004	0173	* ITEMN6	008	0181
* QTY7	004	0185	* ITEMN7	008	0193	*		

LENGTH OF OUTPUT RECORD AREA REQUIRED IN DFF PROGRAM  
RPG \*SUBR92\* - 0020 OTHER - 0012

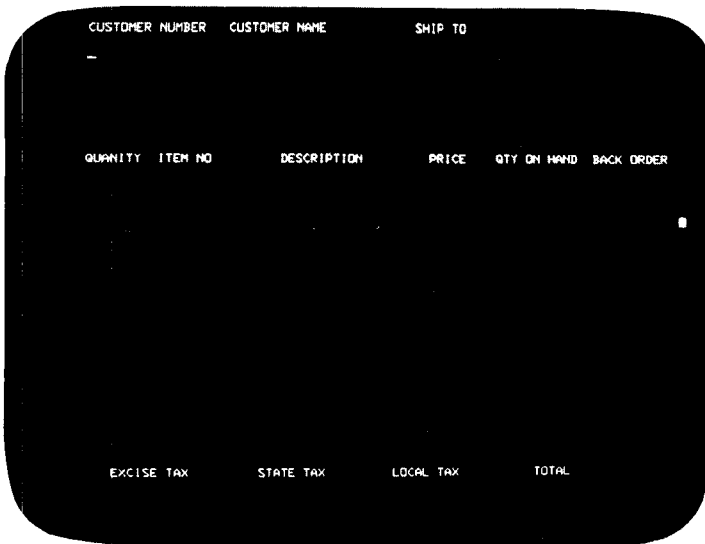
LENGTH OF INPUT RECORD AREA REQUIRED IN DFF PROGRAM  
RPG \*SUBR92\* - 0193 OTHER - 0185

J

INFORMATION FOR USE DURING CCP ASSIGNMENT STAGE

THE DECIMAL LENGTH OF THE FIELD DESCRIPTOR TABLE IS 1150  
THE DECIMAL LENGTH OF THE OUTPUT TEXT IS 1146  
THE DECIMAL LENGTH OF THE INPUT TEXT IS 0240

The generated display format appears as follows:



## Example 2 – RPG II MRT Program Using the Display Format Facility

This RPG II Multiple Requesting Terminal (MRT) program uses the Display Format Facility (DFF) to support the 3270 terminals. Included in the example are a program description, flowcharts, and RPG II coding.

The terminal operator is requested to press the CLEAR key before entering the program name so that unwanted data is not received with the PROGRAM REQUEST.

The application program will request the display format (by name) and cause it to be sent to the 3270 screen. The method of formatting the initial display for the order entry program is explained in *Example 1 – DFF Formatting Example*. The Order Entry display format was generated previously and stored in the object library with the name of \$ZOREN.

When the initial format is placed on the screen, the cursor is positioned at line 2, position 3.

The screenshot shows a terminal window with a dark background and white text. The text is organized into a table-like structure with the following headers:

CUSTOMER NUMBER	CUSTOMER NAME	SHIP TO				
QUANTITY	ITEM NO	DESCRIPTION	PRICE	QTY ON HAND	BACK ORDER	
EXCISE TAX	STATE TAX	LOCAL TAX	TOTAL			

The cursor is positioned at the beginning of the second line, under the 'CUSTOMER NUMBER' header.

The terminal operator enters the customer number and presses ENTER.

The screenshot shows the same terminal window as before, but now the customer number '135624' has been entered under the 'CUSTOMER NUMBER' header. The cursor is now positioned at the end of the first line, after the underscore character.

CUSTOMER NUMBER	CUSTOMER NAME	SHIP TO				
135624 _						
QUANTITY	ITEM NO	DESCRIPTION	PRICE	QTY ON HAND	BACK ORDER	
EXCISE TAX	STATE TAX	LOCAL TAX	TOTAL			

The program uses the number to retrieve the customer name, customer address and ship-to information from the customer master file.

CUSTOMER NUMBER	CUSTOMER NAME	SHIP TO					
135624	NATIONAL SPORTING CO DEPT 801 709 CHARLETON ST NE ROCHESTER, MINN 55901	NATIONAL SPORTING CO DEPT 76 409 N MAIN ST STEWARTVILLE, MINN					
QUANTITY	ITEM NO	DESCRIPTION	PRICE	QTY ON HAND	BACK ORDER		
-							
EXCISE TAX			STATE TAX			LOCAL TAX	
						TOTAL	

All data fields that have been modified are received by the application program. The items are verified by comparing against the inventory master file. If items are verified and sufficient stock is on hand, the description and price are returned to the terminal.

If an item is not found, the item number field is changed to high intensity and ITEM DROPPED is displayed at high intensity.

CUSTOMER NUMBER	CUSTOMER NAME	SHIP TO					
135624	NATIONAL SPORTING CO DEPT 801 709 CHARLETON ST NE ROCHESTER, MINN 55901	NATIONAL SPORTING CO DEPT 76 409 N MAIN ST STEWARTVILLE, MINN					
QUANTITY	ITEM NO	DESCRIPTION	PRICE	QTY ON HAND	BACK ORDER		
<u>36</u>	263146	HOCKEY STICK LEFT	2.48				
10	1646517						
12	37152681	SKI SUPREME GLASS	45.25				
4	46165	SNOW MOBILE 455C1	876.50				
EXCISE TAX			STATE TAX			LOCAL TAX	
						TOTAL	
ITEM DROPPED							

Customer name and address fields are output fields and cannot be altered, but the ship-to fields are output/input fields and can be changed by the operator if necessary. Next, the quantity and item number for each item are keyed in and ENTER is pressed.

CUSTOMER NUMBER	CUSTOMER NAME	SHIP TO					
135624	NATIONAL SPORTING CO DEPT 801 709 CHARLETON ST NE ROCHESTER, MINN 55901	NATIONAL SPORTING CO DEPT 76 409 N MAIN ST STEWARTVILLE, MINN					
QUANTITY	ITEM NO	DESCRIPTION	PRICE	QTY ON HAND	BACK ORDER		
36	263146						
10	1646517						
12	37152681						
4	46165						
EXCISE TAX			STATE TAX			LOCAL TAX	
						TOTAL	

If the item number is verified but there is insufficient stock on hand, the description, price, quantity on hand, and back ordered fields are displayed. OUT OF STOCK changes from nondisplay to high intensity and the item is entered in the back order file.

Once the item information is displayed, the operator can verify the status of the order and make any changes desired. For example, when OUT OF STOCK is indicated, the quantity field could be rekeyed for the intensified items.

For ITEM DROPPED, enter zeros in the quantity field to delete the entry. (Any item can be deleted by entering zeros in its quantity field.)

After all exceptions have been satisfied, ENTER is pressed. When the program recognizes that all items have been processed and there are no exceptions, it calculates all prices and taxes and verifies customer credit status.

CUSTOMER NUMBER	CUSTOMER NAME	SHIP TO
135624	NATIONAL SPORTING CO DEPT 801 709 CHARLETON ST NE ROCHESTER, MINN 55901	NATIONAL SPORTING CO DEPT 76 409 N MAIN ST STEWARTVILLE, MINN

QUANTITY	ITEM NO	DESCRIPTION	PRICE	QTY ON HAND	BACK ORDER
36	263146	HOCKEY STICK LEFT	2.48		
00	1646517				
12	371526B1	SKI SUPREME GLASS	45.25		
4	46165	SNOW MOBILE 455CI	876.50		

EXCISE TAX 41.38	STATE TAX 82.77	LOCAL TAX 82.77	TOTAL 4345.20
------------------	-----------------	-----------------	---------------

If the credit status is unacceptable, CREDIT OVER is displayed at high intensity. The display now includes the total fields.

The operator verifies the order including totals and credit. If acceptable, the PF1 key (enter order key) is pressed, and, if the credit check was good, the inventory file and customer account file are updated and the display is printed on the 3286. If the credit check was not good, the files are not updated but the display, including the credit over field, is printed on the 3286.

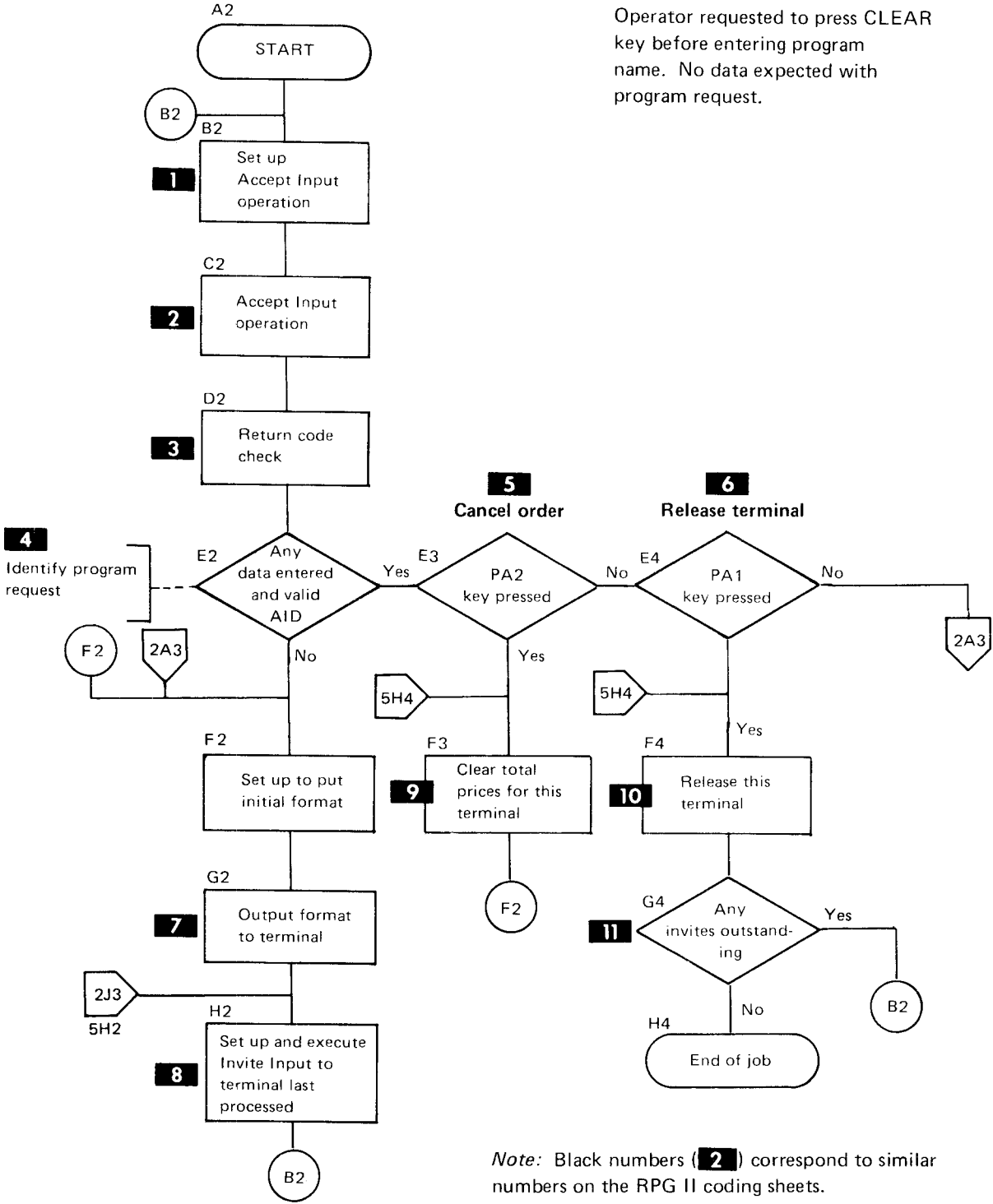
When the order has been completed and all files are updated, a new display format is sent to the terminal, ready for a new order.

Any order may be canceled at any time by pressing the PA2 key (cancel order key). The program then displays a new display format on the terminal.

The terminal may be released from the application program by pressing the PA1 key (release terminal key). The terminal would then be free to request another application and also may request the application program when desired by rekeying the program name.

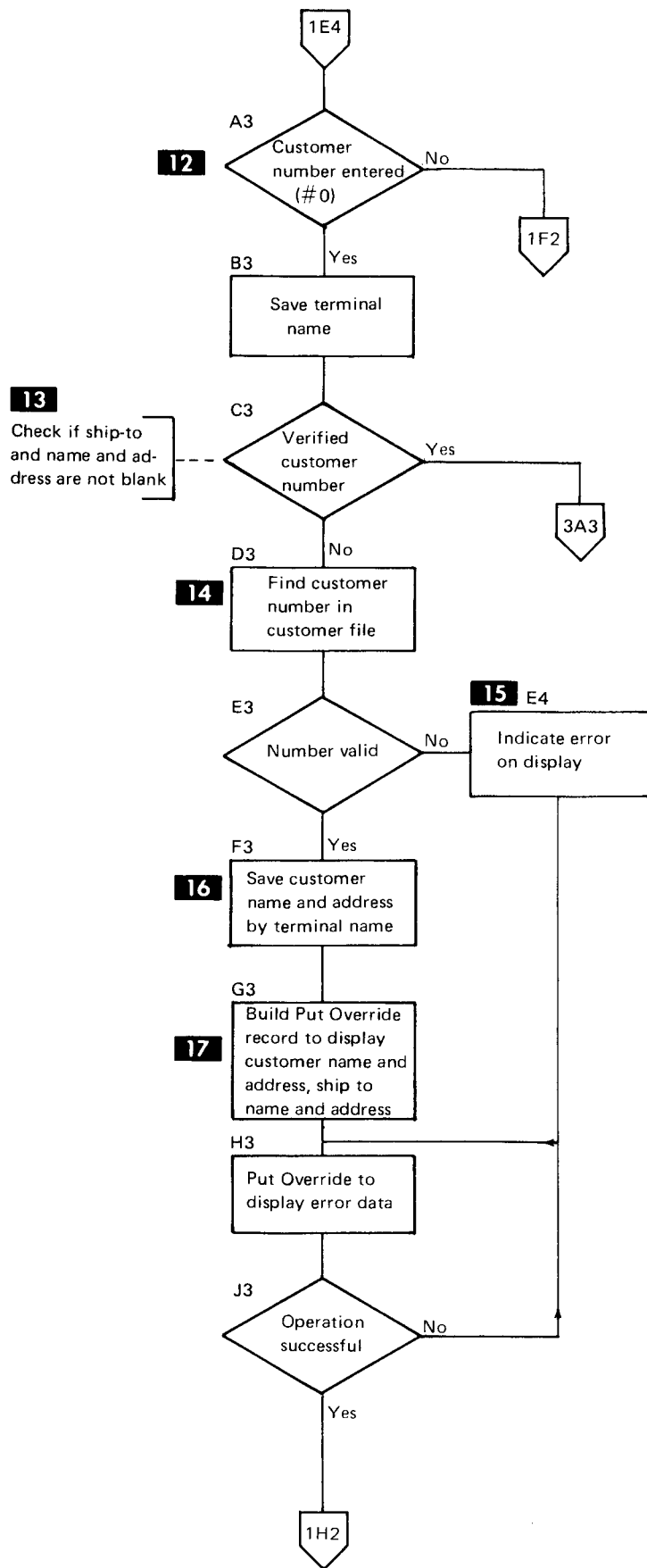


Operator requested to press CLEAR key before entering program name. No data expected with program request.

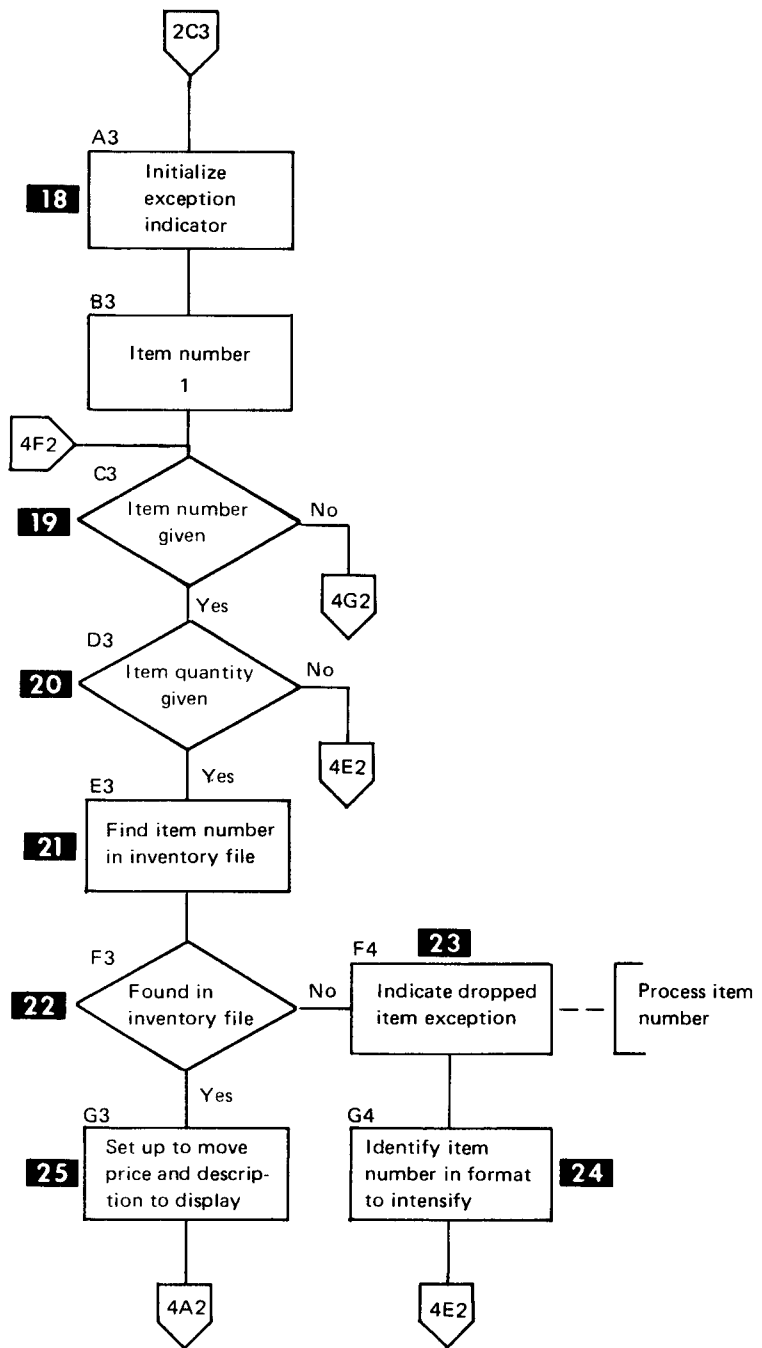


Note: Black numbers (2) correspond to similar numbers on the RPG II coding sheets.

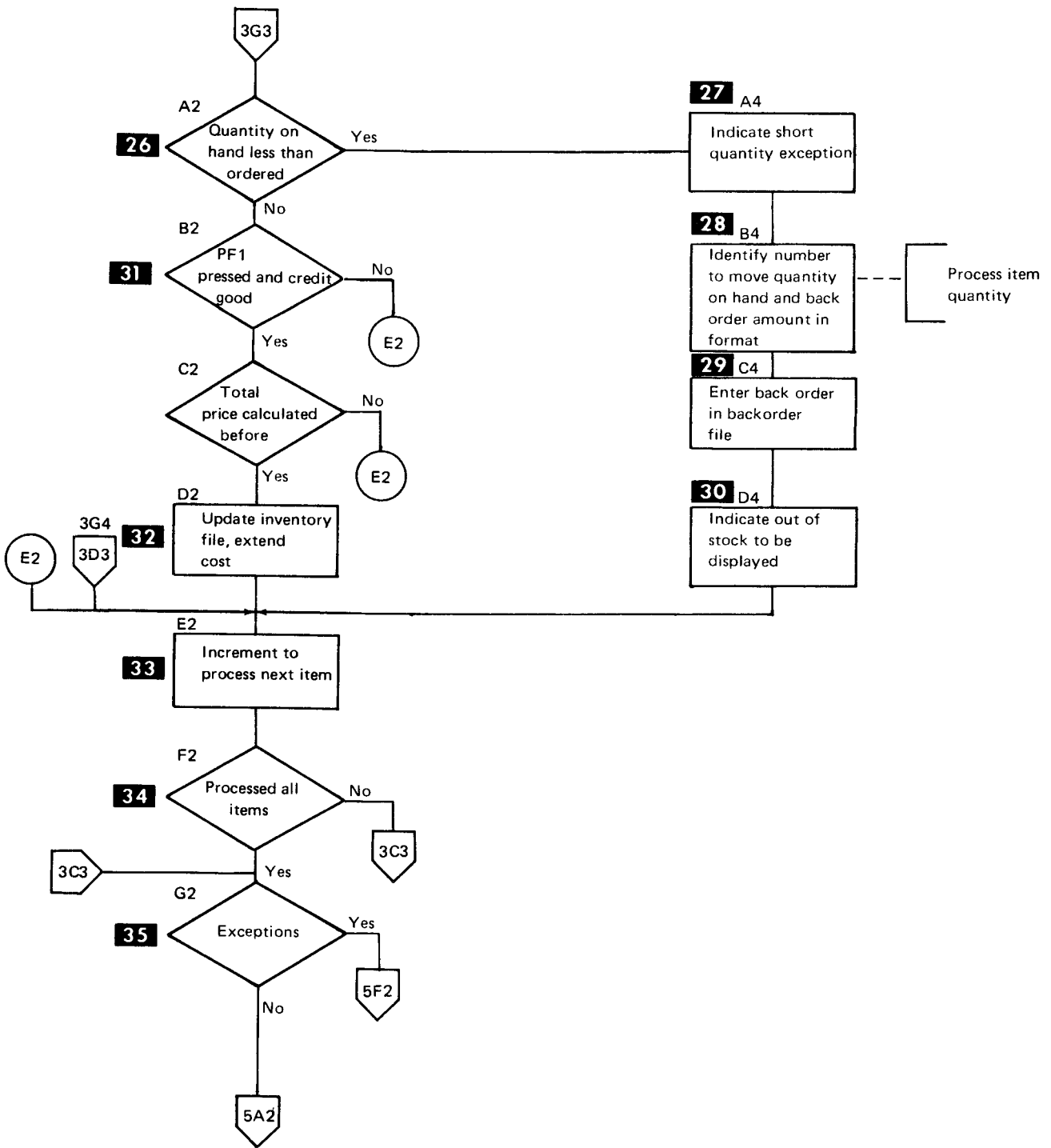
Order Entry Program Flow (Part 1 of 5)



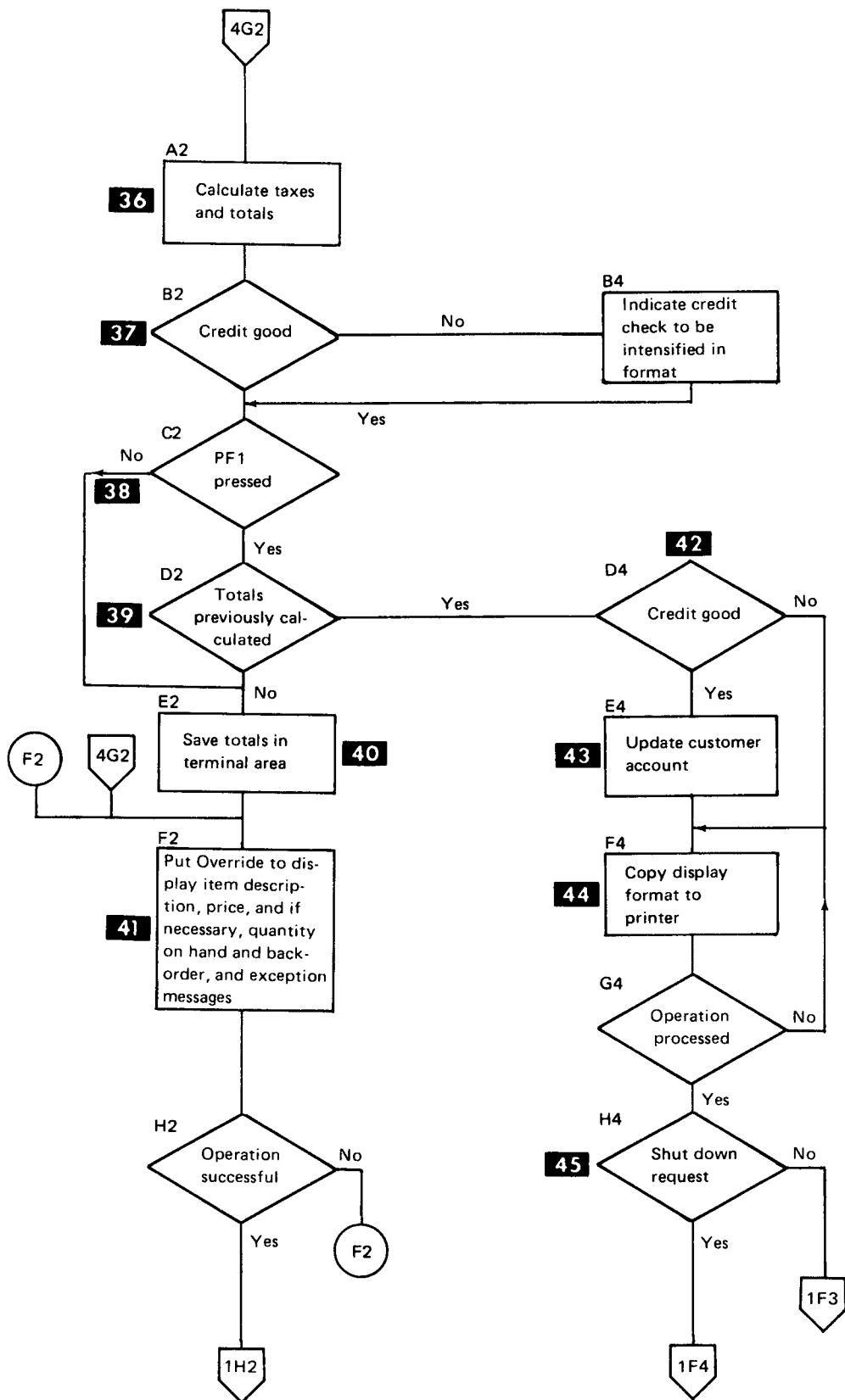
Order Entry Program Flow (Part 2 of 5)



Order Entry Program Flow (Part 3 of 5)



Order Entry Program Flow (Part 4 of 5)



Order Entry Program Flow (Part 5 of 5)



**RPG EXTENSION AND LINE COUNTER SPECIFICATIONS**

Program	Punching Instruction	Graphic	Card Electro Number
Programmer	Date	Punch	

**Extension Specifications**

Line	Form Type	Record Sequence of the Chaining File		To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions Sequence (A/D)	Comments
		Number of the Chaining Field	From Filename												
0 1	E				QTY		7	4							QUANTITY ORDERED
0 2	E				ITMN		7	8							ITEM NUMBER
0 3	E				CNM		6	22							CUSTOMER NAME
0 4	E				CA1		6	22							CUSTOMER ADDRESS1
0 5	E				CA2		6	22							CUSTOMER ADDRESS2
0 6	E				CA3		6	22							CUSTOMER ADDRESS3
0 7	E				C#		6	6							CUSTOMER NUMBER
0 8	E				INT		7	1							TYPE ITEM NUMBER
	E				ONH		7	4							QNTY ON HAND
	E				BOA		7	4							BACKORDER AMNT

**RPG EXTENSION AND LINE COUNTER SPECIFICATIONS**

Program	Punching Instruction	Graphic	Card Electro Number
Programmer	Date	Punch	

**Extension Specifications**

Line	Form Type	Record Sequence of the Chaining File		To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions Sequence (A/D)	Comments
		Number of the Chaining Field	From Filename												
0 1	E				PR		7	5							PRICE OF ITEM
0 2	E				DSC		7	20							DESCRIPTION
0 3	E				CROK		6	1							CREDIT OKAY/NOTOK
0 4	E														
0 5	E														
0 6	E														
0 7	E														
0 8	E														







RPG CALCULATION SPECIFICATIONS

IBM International Business Machine Corporation

Form GX21-9093  
Printed in U.S.A.

Program	Punching Instruction	Graphic	Card Electro Number
Programmer	Date	Punch	

Page 8 of 19 Program Identification **ORDENT**

Line	Form Type Control Unit (L, O, L, S, R, AN, OR)	Indicators					Factor 1	Operation	Factor 2	Result Field		Resulting Indicators	Comments
		And	And	Not	Not	Not				Name	Length		
01	C	*	*	*	*	*	HANDLE REQUEST TO CANCEL THE ORDER						
02	C	*	*	*	*	*							
03	C						96	GOTO CANCEL					CLEAR TOTAL
04	C	*	*	*	*	*		AND SEND NEW FORMAT TO TERMINAL					
05	C	*	*	*	*	*							
06	C	*	*	*	*	*							
07	C	*	*	*	*	*		HANDLE REQUEST TO RELEASE TERMINAL					
08	C						N95	GOTO CONT1					
09	C	*	*	*	*	*		IF NOT RELEASE, CONTINUE					
10	C							RELEAS TAG					
11	C							SETON			95		RELSE TERMINAL
12	C							EXCPT					
13	C	*	*	*	*	*		IF THE FOLLOWING COMPARE OPERATION					
14	C	*	*	*	*	*		WERE REPLACED WITH THE COMMENTED INSTRUCTION, THE					
15	C	*	*	*	*	*		PROGRAM WOULD BE A NEVER ENDING PROGRAM.					
16	C	*	*	*	*	*		MOVE PL,3	OUTSII	20			IF NO OUTSTAND-
17	C							COMP OUTSII					LRING INVITES, E
18	C							GOTO END					
19	C	*	*	*	*	*							
20	C	*	*	*	*	*		90	COMP OUTSII				LR
01	C							CONT1	TAG				
02	C	*	*	*	*	*							
03	C	*	*	*	*	*		IF THIS IS A PROGRAM REQUEST, SEND THE					
04	C	*	*	*	*	*		FORMAT TO THE TERMINAL. ALSO SEND INITIAL					
05	C	*	*	*	*	*		FORMAT FOR CLEAR KEY, AND UNIDENTIFIED					
06	C	*	*	*	*	*		INPUT RECORD					
07	C	*	*	*	*	*							
08	C						N97	GOTO CONT2					
09	C	*	*	*	*	*		CANCEL ORDER AND RESET FOR NEW ORDER					
10	C							RESET	TAG				
11	C							CANCEL	TAG				
12	C							MOVE '5'	CROK, N				REINIT CREDIT
13	C							Z-ADD	TPR, N				CLEAR TOT PRICE
14	C							SETON					
15	C							SETOF			80		
16	C						N99	GOTO INVITE			970102		
17	C						90	GOTO RELEAS					
18	C	*	*	*	*	*							IF SHUTDOWN
19	C												RELSE TERMINAL
20	C												

RPG CALCULATION SPECIFICATIONS

Form GK21-9093  
Printed in U.S.A.

IBM International Business Machine Corporation

Program	Punching Instruction	Graphic	Card Electro Number
Programmer	Date	Punch	

Page 9 of 19 Program Identification **ORDENT**

Line	C	Form Type	Control Level (L, O, L, S, L, R, SR, AN, OR)	Indicators			Factor 1	Operation	Factor 2	Result Field		Resulting Indicators			Comments	
				And	And	And				Name	Length	Plus	Minus	Zero		
01	C						CONT2	TAG								
02	C	*	*	*	*	*	*	*	*	*	*	*	*	*	*	
03	C	*					CHECK IF CUSTOMER NUMBER HAS BEEN VERIFIED									
04	C	*														
05	C						BLANKC	COMP SNAME						81		
06	C		N81					GOTO PROCESS								
07	C	*						IF SHIP TO NAME IS NOT BLANK, CUSTOMER NUMBER								
08	C	*						HAS BEEN VERIFIED								
09	C	*														
10	C	*						VERIFY CUSTOMER NUMBER								
11	C															
12	C															
13	C	*						SETON INDR 82 IF NUMBER NOT CORRECT								
14	C	*						SAVE CUSTOMER NAME, ADDRESS AND NUMBER								
15	C		N82					MOVE CNAME	CNM, N							
16	C		N82					MOVE CADDR1	CAL, N							
17	C		N82					MOVE CADDR2	CA2, N							
18	C		N82					MOVE CADDR3	CA3, N							
19	C		N82					MOVE CNUMB	C#, N							
20	C	*						EITHER A 5 (CREDIT OK) OR A 2 (NOT OK) IS MOVED INTO CROK, N								
21	C	*														
22	C															
23	C							GOTO INVITE								
24	C							PROCESS TAG								
25	C	*						START PROCESSING ITEM NUMBERS AND QUANTITY								
26	C	*														
27	C	*						REINITIALIZE EXCEPTION LOGIC FLAGS AND POINTER								
28	C	*						TO LINE NUMBER TO PROCESS								
29	C	*														
30	C							SETOF					8586			
31	C	*						85-ITEM NOT FOUND								
32	C	*						86-BACK ORDER ITEM								
33	C							MOVE L	1				L0			
34	C							MOVE ( )	INT					BLANK OUT TYPE		
35	C							MOVE L	ONH					BLANK OUT		
36	C							MOVE ( )	BOA					OUTPUT		
37	C							Z-ADD 00.00	PR					ARRAYS		
38	C							MOVE BLANKC	DISC							
39	C							SETON						73		
40	C															
41	C															
42	C															
43	C															
44	C															
45	C															
46	C															
47	C															
48	C															
49	C															
50	C															
51	C															
52	C															
53	C															
54	C															
55	C															
56	C															
57	C															
58	C															
59	C															
60	C															
61	C															
62	C															
63	C															
64	C															
65	C															
66	C															
67	C															
68	C															
69	C															
70	C															
71	C															
72	C															
73	C															
74	C															

13

14

15

16

18

RPG CALCULATION SPECIFICATIONS

Form GX21-9093  
Printed in U.S.A.

IBM International Business Machine Corporation

Program	Punching Instruction	Graphic	Card Electro Number
Programmer	Date	Punch	

Page 10 of 19 Program Identification 75 76 77 78 79 80  
ORDENT

Line	Form Type	Control Level (LO, L, R, SR, AN, OR)	Indicators						Factor 1	Operation	Factor 2	Result Field		Resulting Indicators	Comments			
			And	And	Not	Not	Not	Not				Name	Length			Arithmetic	Plus	Minus
01	C	XX																
02	C																	
03	C	XX																
04	C	XX																
05	C																	
06	C																	
07	C	XX																
08	C	XX																
09	C																	
10	C	XX																
11	C																	
12	C	XX																
13	C																	
14	C																	
15	C																	
16	C																	
17	C	XX																
18	C																	
19	C																	
20	C																	
21	C																	
22	C																	
23	C																	
24	C																	
25	C																	
26	C																	
27	C																	

RPG CALCULATION SPECIFICATIONS

Form GX21-9093  
Printed in U.S.A.

IBM International Business Machine Corporation

Program	Punching Instruction	Graphic	Card Electro Number
Programmer	Date	Punch	

Page 11 of 19 Program Identification **ORDENT**

Line	Form Type	Control Level (LD, LR, SR, AN/OR)	Indicators			Factor 1	Operation	Factor 2	Result Field		Resulting Indicators	Comments
			And	And	Not				Name	Length		
01	C	*				SETUP FOR NO BACK ORDER						
02	C											
03	C	*				UPDATE INVENTORY IF ORDER IS TO BE ENTERED AND						
04	C	*				CREDIT IS OKAY AND TOTAL PRICE CALCULATED						
05	C	*				PREVIOUSLY.						
06	C	*										
07	C					02	'5'	COMP CROK, N				72
08	C					02	<del>000000</del>	COMP TPR, N				73
09	C	*										
10	C	*				OK TO UPDATE INVENTORY FILE FOR THIS ITEM IF 72 IS ON, 73 OFF						
11	C	*										
12	C					72N73		EXCPT				
13	C							SETOF				12
14	C							GOTO NXTITM				
15	C	*										
01	C	*				CREATE BACKORDER						
02	C	*				INITIALIZE FIELDS FOR OVERRIDE TO TERMINAL						
03	C	*				BACKOR TAG						
04	C											
05	C											
06	C											
07	C							MOVE INVONH	ONH, 1			ON HAND AMOUNT
08	C							MOVE INVBOA	BOA, 1			BACK ORDER AMT
09	C							EXCPT				
10	C							SETOF				12
11	C	*										
12	C					NXTITM		TAG				
13	C	*				CHECK IF FINISHED WITH ALL 7 ITEMS OR IF MORE WORK TO DO						
14	C					1		ADD 1	1			
15	C					8		COMP 1				08
16	C					08		GOTO LOOP				
17	C											
18	C											
19	C											
20	C											

31

32

28

33

34

RPG CALCULATION SPECIFICATIONS

Form GX21-9093  
Printed in U.S.A.

IBM International Business Machine Corporation

Program	Punching Instruction	Graphic	Card Electro Number
Programmer	Date	Punch	

Page 12 of 19 Program Identification ORIENT

Line	C	I-Form Type	Control Level (LO, L, R, SR, AN/OR)	Indicators			Factor 1	Operation	Factor 2	Result Field		Decimal Positions Pair Adjust (H)	Resulting Indicators			Comments			
				And	And	And				Name	Length		Plus	Minus	Zero				
01	C	*																	
02	C	*																	
03	C	*																	
04	C																		
05	C																		
06	C	OR																	
07	C	*																	
08	C	*																	
09	C																		
10	C																		
11	C																		
12	C	*																	
13	C	*																	
14	C	*																	
15	C	*																	
16	C	*																	
17	C																		
18	C																		
19	C	OR																	
20	C	*																	
01	C	*																	
02	C																		
03	C	*																	
04	C																		
05	C																		
06	C																		
07	C																		
08	C																		
09	C																		
10	C																		
11	C																		
12	C																		
13	C																		
14	C																		
15	C																		
16	C																		
17	C																		
18	C																		
19	C	OR																	
20	C																		

35

37

38

39

40

42

43

44

45

8

RPG CALCULATION SPECIFICATIONS

Form GX21-9093  
Printed in U.S.A.

IBM International Business Machine Corporation

Program	Date	Punching Instruction	Graphic	Card Electro Number
Programmer		Punch		

Page 13 of 19 Program Identification **ORDENT**

Line	Form Type	Control Level (LD, LB, LR, SR, AN/OR)	Indicators			Factor 1	Operation	Factor 2	Result Field		Resulting Indicators	Comments
			And	And	Not				Name	Length		
01	C	X										
02	C	X										
03	C	X										
04	C	SR										
05	C	SR	91									
06	C	SR	92		04						98	IF ERROR
07	C	X									88	99 SHUTDOWN-99
08	C	SR	99								90	TRUNCATED-88
09	C	SR									9192	
10	C	SR										
11	C											
12	C											

RPG OUTPUT SPECIFICATIONS

Form GX21-9090  
Printed in U.S.A.

IBM International Business Machine Corporation

Program	Date	Punching Instruction	Graphic	Card Electro Number
Programmer		Punch		

Page 14 of 19 Program Identification **ORDENT**

Line	Form Type	Filename	Type (H/D/T/E)	Space	Skip	Output Indicators			Field Name	End Position in Output Record	Constant or Edit Word
						And	And	Not			
01	O	INVENTORYE									
02	O										
03	O										
04	O	CUSTOMER E									
05	O										
06	O										
07	O	HOLDORDRD									
08	O										
09	O										
10	O	ONORDER E									
11	O										
12	O										
13	X										
14	X	THIS IS UPDATING THE CUSTOMER'S ACCOUNT									
15	X										
16	O										
17	O										
18	O										
19	O										
20	O										

42  
43

Program \_\_\_\_\_ Puncturing Instruction \_\_\_\_\_ Graphic \_\_\_\_\_ Card Electro Number \_\_\_\_\_  
 Programmer \_\_\_\_\_ Date \_\_\_\_\_ Punch \_\_\_\_\_

Line	Form Type	Filename	Type (H D T E)		Space	Skip	Output Indicators			Field Name	Edit Codes	End Position in Output Record	Constant or Edit Word	Commas	Zero Balances to Print	No Sign	CR	-	X	Remove Plus Sign	Date	Field Edit	Zero Suppress
			Before	After			Not	And	And														
01	O	* FIRST OUTPUT IS FOR PUT WHOLE FORMAT TO TERMINAL																					
02	O	* TERMOUT D																					
03	O																						
04	O																						
05	O																						
06	O																						
07	O																						
08	O																						
17	O	* PUT OUT THE CUSTOMER NAME, ADDRESS, AND SHIP TO INFORMATION																					
02	O																						
03	O																						
04	O																						
05	O																						
06	O	* THE FOLLOWING WCC REFLECTS NO CHANGE TO MDT'S																					
07	O																						
08	O	* PUT OVERRIDE HEADER																					
09	O																						
10	O																						
11	O	* REPOSITION THE CURSOR																					
12	O																						
13	O																						
14	O																						
15	O																						
16	O																						
17	O																						
18	O																						
19	O																						
20	O																						
01	O																						
02	O																						
03	O																						
04	O																						
05	O																						
06	O																						
07	O																						
08	O																						
09	O																						
10	O																						
11	O																						
12	O	* NOTE THAT THE LENGTH GIVEN IN COLUMNS 1-4 GIVES A LENGTH FOR																					
13	O	* A FULL OVERRIDE LIST. THE TYPE WAS CHANGED FOR CUSERR, BUT																					
14	O	* ROOM MUST BE LEFT TO SHOW BLANK ENTRIES FOR CURSOR AND																					
15	O	* MODIFY DATA FLAGS.																					
01	O	* PUT OVERRIDES TO DISPLAY PROCESSING OF ITEMS, AND INDICATE																					
02	O	* TAXES, TOTALS, EXCEPTION CONDITIONS IF NECESSARY.																					
03	O																						
04	O																						
05	O																						
06	O																						
07	O																						
08	O																						
09	O																						
10	O																						
11	O																						



Program	Punching Instruction	Graphic	Card Electro Number
Programmer	Date	Punch	

Page 16 of 19  
Program Identification ORIENT

Line	Form Type	Filename	Type (H/D/T/E)	Space		Skip	Output Indicators			Field Name	End Position in Output Record	Constant or Edit Word	Commas	Zero Balances to Print	No Sign	CR	-	X	Remove Plus Sign	Date	Field Exit	Zero Suppress
			Before	After	And		And	And	Not													
01	O									INT, 1	30	'ITEMNL'										
02	O										31											
03	O	* MODIFY TYPE OF ITEMNL IF NOT FOUND TO HIGH INTENSITY																				
04	O	* ARRAY ELEMENT CONTAINS 2 OR 1																				
05	O	*																				
06	O																					
07	O										39	'DESCRL'										
08	O									DSC, 1	42	'M'										
09	O	* DSC ELEMENT IS BLANK IF ITEM IS NOT FOUND IN INVENTORY FILE									62											
10	O										68	'PRICEL'										
11	O										71	'M'										
12	O									PR, 1 2	78											
13	O	* EDIT CODE WILL ENTER DECIMAL POINT IF ITEM FOUND, BLANK OUT																				
14	O	* WHOLE FIELD IF NOT FOUND																				
15	O																					
16	O	* VALUE WILL BE BLANK IF ENOUGH IS IN STOCK, OR ACTUAL AMOUNT								ONH, 1	84	'QTYOHL'										
17	O	* WILL APPEAR IF SHORT OF ORDER QUANTITY									87	'M'										
18	O	*									91											
19	O																					
20	O																					
21	O																					
22	O																					
23	O																					
24	O																					
25	O																					
26	O																					
27	O																					
28	O																					
29	O																					
30	O																					
31	O																					
32	O																					
33	O																					
34	O																					
35	O																					
36	O																					
37	O																					
38	O																					
39	O																					
40	O																					
41	O																					
42	O																					
43	O																					
44	O																					
45	O																					
46	O																					
47	O																					
48	O																					
49	O																					
50	O																					
51	O																					
52	O																					
53	O																					
54	O																					
55	O																					
56	O																					
57	O																					
58	O																					
59	O																					
60	O																					
61	O																					
62	O																					
63	O																					
64	O																					
65	O																					
66	O																					
67	O																					
68	O																					
69	O																					
70	O																					
71	O																					
72	O																					
73	O																					
74	O																					

41

## RPG OUTPUT SPECIFICATIONS

GX21-9090  
Printed in U.S.A.

**IBM** International Business Machine Corporation

Program	Punching Instruction	Graphic	Punch	Card Electro Number	Page <span style="float: right;">1 2</span>	Program Identification	75 76 77 78 79 80
Programmer	Date				17 of 19	<b>ORDENT</b>	

**41**

Line	Form Type	Filename	Type (H/D/T/E)		Space	Skip	Output Indicators			Field Name	Edit Codes B/A/C/LS/R	End Position in Output Record	P/B/L/R	Constant or Edit Word	Commas	Zero Balances to Print	No Sign	CR	-	X	Remove Plus Sign	Date Field Edit Zero Suppress	
			A	D			Before	After	Not														Not
01	O									*AUTO													
02	O									INT, 2													
03	X	PROCESS SECOND ITEM																					
04	X									{													
05	O									(ADDITIONAL ITEMS)													
06	O									{													
07	O																						
08	O																						
09	O																						
10	O									BOA, 7													
11	O																						
01	X	THE FOLLOWING FIELDS WILL BE SENT ONLY IF TOTALS AND TAXES																					
02	X	WERE CALCULATED (NO EXCEPTION CONDITIONS)																					
03	O									N83													
04	O									N83													
05	O									N83													
06	O									TAXEX 2													
07	O									N83													
08	O									N83													
09	O									TAXST 2													
10	O									N83													
11	O									N83													
12	O									TAXLT 2													
13	O									N83													
14	O									N83													
15	O									TOTMON 2													
16	O									N83													
17	O																						
18	O																						
19	O																						
20	O																						

RPG OUTPUT SPECIFICATIONS

GX21 9090  
Printed in U.S.A.

IBM International Business Machine Corporation

Program	Punching Instruction	Graphic	Card Electro Number
Programmer	Date	Punch	

Page 18 of 19 Program Identification ORDENT

Line	Format	Filename	Type (H/D/T/E)			Space		Skip		Output Indicators			Field Name	Edit Codes	End Position in Output Record	Constant or Edit Word	Options			
			A	D	T	E	Before	After	Not	Not	Not	Commas					Zero Balances to Print	No Sign	CR	Remove Plus Sign
01	O													643	'OUTOS'					
02	O													645	'(5)'					
03	O													653	'CUSERR'					
04	O													654	'(5)'					
05	O													662	'ITEMDR'					
06	O													663	'(5)'					
07	O													671	'CREDOV'					
08	O													674	'(2)'					
09	O													674	'(5)'					
10	O	* THE ABOVE FIELD WILL OVERRIDE THE TYPE ONLY TO NONDISPLAY OR																		
11	O	* DISPLAY																		
01	O	* THE FOLLOWING FIELDS WILL BE PUT OUT IF EXCEPTIONS OCCURRED																		
02	O														589	'OUTOS'				
03	O														591	'(2)'				
04	O														591	'(5)'				
05	O														599	'CUSERR'				
06	O														600	'(5)'				
07	O														608	'ITEMDR'				
08	O														609	'(2)'				
09	O														609	'(5)'				
10	O														617	'CREDOV'				
11	O														620	'(2)'				
12	O														620	'(5)'				

41

41

**RPG OUTPUT SPECIFICATIONS**

Program \_\_\_\_\_ Punching Instruction \_\_\_\_\_ Graphic \_\_\_\_\_ Card Electro Number \_\_\_\_\_  
 Programmer \_\_\_\_\_ Date \_\_\_\_\_ Punch \_\_\_\_\_

Line	Form Type	Filename	Type (H/D/E)		Space		Skip		Output Indicators			Field Name	End Position in Output Record	PB L/R	Commas	Zero Balances to Print	No Sign	CR	-	X	Remove Plus Sign	Y	Date	Field Edit	Z	Zero Suppress
			O	A	N	D	Before	After	Before	After	Not															
8	01	O	* REINVITE IF NECESSARY CURRENT TERMINAL									*AUTO	8 (14)	4 ( )												
	02	O	D										14 ( )													
	03	O																								
	04	O																								
	05	O																								
10	06	O	* RELEASE A TERMINAL										8 (14)	4 ( )												
	07	O	E										14 ( )													
	08	O																								
	09	O																								
	10	O																								
	11	O																								
	12	O	*																							
	13	O	*																							
44	01	O	* COPY THE DISPLAY TO A 3284 OR 3286 PRINTER ON 3271 CONTROL UNIT										8 (20)	4 ( )												
	02	O	E										14 ( )													
	03	O																								
	04	O																								
	05	O																								
	06	O																								
	07	O	* DEFAULT CCC WILL BE USED.																							
	08	O	*																							
	09	O	* COPY THE DISPLAY TO A 3284 PRINTER ATTACHED TO A 3275 WITH WCC																							
	10	O	* FOR START PRINTER AND 80-COLUMN LINE.																							
	11	O	E																							
	12	O																								
	13	O																								
	14	O																								
	15	O																								
	16	O																								
	17	O																								
	18	O																								
	19	O																								
	20	O																								

### Example 3—SRT Inquiry Program

This example shows RPG II, COBOL, and FORTRAN IV coding for an inquiry program that uses the Display Format Facility to display information on the 3270. The three versions of the program use the same basic logic, inquire into the same three disk files, and use the same screen format. The logic flow, display layout, display format specifications, and DFGR printout are shown on the following pages.

This is a single requesting terminal (SRT) program; therefore, the program can handle only one inquiry on each execution. If sufficient main storage area is available, however, more than one copy of the program could be executing concurrently, serving different requestors.

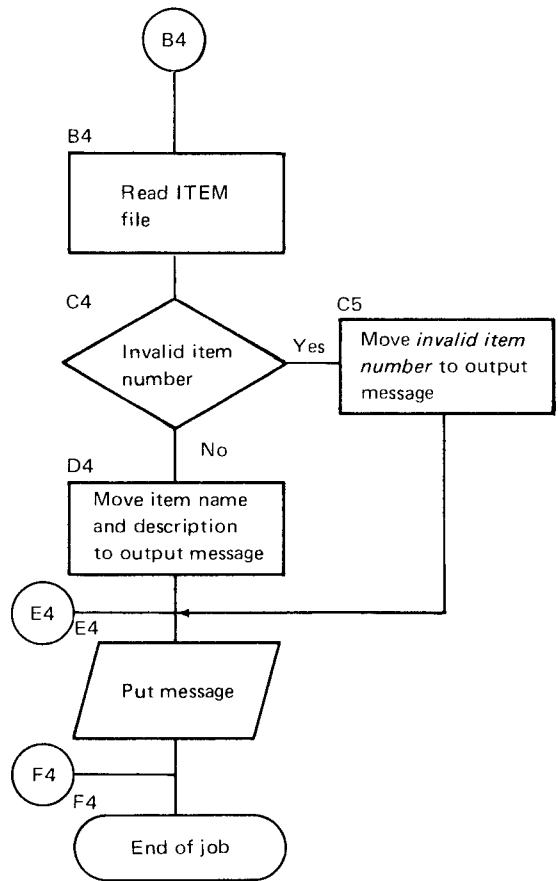
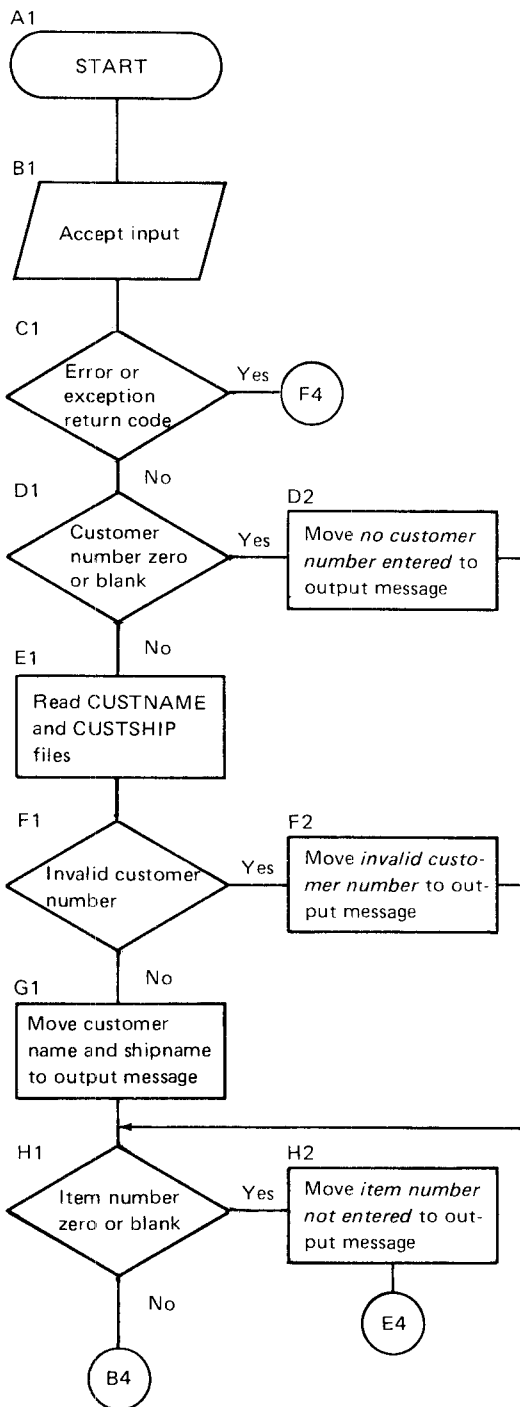
The terminal operator uses this inquiry program as follows:

1. Signs on, keying /ONb [password] , and presses the ENTER key.
2. Presses the ERASE INPUT key.
3. Enters a program request accompanied by input data, as follows:  
*COBOL:*  
COBINQbcccccc/IIIIIIII  
*RPG II:*  
RPGINQbcccccc/IIIIIIII  
*FORTRAN IV:*  
FORINQbcccccc/IIIIIIII  
where ccccc is a six-digit customer number and IIIIIII is an eight-digit item number. Presses the ENTER key.
4. Program returns a message to the terminal operator.
5. Program terminates.

If the input from the terminal operator is incomplete or invalid, the program issues an appropriate error message.

#### Notes:

- The three disk files used by the program are processed directly.
- In order to have the format remain on the screen after the program reaches end of job, ENDMSG-NO must be specified on the // PROGRAM assignment statement.



Example 3 – SRT Inquiry Program Flow



```

C$ZRPIQ      2Y X R2
FHEADR10606 1216      CUSTOMER NO.
FCUSNO 0621 62E
FHEADR20806 1316      CUSTOMER NAME
FCUSNM 0821 222E
FHEADR31006 716       SHIP TO
FSHPNM 1021 222E
FHEADR41306 816       ITEM NO.
FITMNO 1321 82E
FHEADR51506 1016      ITEM DESC.
FITDSC 1521 202E
/*

```

\$ZRPIQ DISPLAY FORMAT INFORMATION

-----  
EXECUTION TIME DATA - OUTPUT AREA FORMAT - END POSITIONS FOR RPG PROGRAMS  
-----

* FIELD NAME	FIELD LENGTH	END POSITION	* FIELD NAME	FIELD LENGTH	END POSITION	* FIELD NAME	FIELD LENGTH	END POSITION	*
* UPCODE	004	0004	* LENGTH	004	0008	* TMNAME	006	0014	*
* \$ZRPIQ	006	0020	* CUSNO	006	0026	* CUSNM	022	0048	*
* SHPNM	022	0070	* ITMNO	008	0078	* ITDSC	020	0098	*

-----  
INPUT AREA FORMAT - END POSITIONS FOR RPG PROGRAMS  
-----

* FIELD NAME	FIELD LENGTH	END POSITION	* FIELD NAME	FIELD LENGTH	END POSITION	* FIELD NAME	FIELD LENGTH	END POSITION	*
* RTCODE	004	0004	* LENGTH	004	0008	* TMNAME	006	0014	*
* - AID-	001	0015							

LENGTH OF OUTPUT RECORD AREA REQUIRED IN DFF PROGRAM  
RPG \*SUBR92\* - 0098 OTHER - 0090

LENGTH OF INPUT RECORD AREA REQUIRED IN DFF PROGRAM  
RPG \*SUBR92\* - 0015 OTHER - 0007

-----  
INFORMATION FOR USE DURING CCP ASSIGNMENT STAGE  
-----

THE DECIMAL LENGTH OF THE FIELD DESCRIPTOR TABLE IS 0132  
THE DECIMAL LENGTH OF THE OUTPUT TEXT IS 0216  
THE DECIMAL LENGTH OF THE INPUT TEXT IS 0005

Example 3 — Display Format Generation for SRT Inquiry Program



0101 H P  
RG 004

```

0201 F*
0202 F* SINGLE REQUESTOR INQUIRY PROGRAM WHICH USES THE
0203 F* DISPLAY FORMAT FACILITY IN CCP.
0204 F*
0001 0205 FTERMIN IP      029 029          SPECIAL      SUBR92
0002 0206 F
0003 0207 FTERMOUT D     098 098          SPECIAL      SUBR92
0004 0208 F
0005 0209 FCUSTNAMEIC    300 300R        DISK          KPL
0006 0210 FCUSTSHIPIC    256 128R        DISK
0007 0211 FINVENTRYIC    300 300R        DISK
    
```

0008 0301 E PL 5 5 6

COMPILE TIME — A compilation time array is used so that the array contains the accept input operation code and the maximum input length when the TERMIN file is read.

```

0401 I***** KEY CUST. NO. 6 / ITEM NO. 8 SUCH AS XXXXXX/YYYYYYYYY
0009 0402 ITERMIN AA 01
0010 0403 I
0011 0404 I
0012 0405 I
0013 0406 I
0014 0407 ICUSTNAMEAA 10 1 CM 2 CA
0015 0408 I
0016 0409 ICUSTSHIPBB 11 1 CM 2 CB
0017 0410 I
0018 0411 IINVENTRYCC 12 1 CM 2 CI
0019 0412 I
    
```

80 — Indicators 80 and 81 indicate when the customer number or item number are zero or blank.  
81 —

```

0020 0501 C
0021 0502 C N81 /* SETON LR
0022 0503 C N05 COMP SLASH 05
0023 0504 CLR N05 SETON 858687
0024 0505 C R 91
0025 0506 C R 92
0026 0507 CLR N9IN80 CUSNO GOTO END
0027 0508 CLR N9IN80 CUSNO CHAINCUSTNAME 85
0028 0509 CLR N9IN81 ITMNO CHAINCUSTSHIP 86
0029 0510 CLR END TAG 87
0030 0511 CLR SETOF 9192
    
```

```

0031 0601 ITERMOUT T LR
0032 0602 I
0033 0603 I
0034 0604 I PL,5
0605 I*
0035 0606 I
0607 I*
0036 0608 I CUSNO 026
0609 I*
0037 0610 I 10480N85CUSNM 043
0611 I N30 85 042 *NAME NOT ON FILE*
0038 0612 I 30 045 *NO CUST NO. ENTERED*
0613 I*
0039 0614 I 11480N36SHPNM 070
0615 I N80 85 064 *NAME NOT ON FILE*
0040 0616 I 80 057 *NO CUST NO. ENTERED*
0617 I*
0041 0618 I ITMNO 073
0619 I*
0042 0620 I 12481N87ITDSC 093
0621 I N31 87 092 *ITEM NOT ON FILE*
0043 0622 I 31 095 *NO ITEM NO. ENTERED*
    
```

\*\*

0 29

Example 3 – RPG II Coding for SRT Inquiry Program

IBM SYSTEM/3 AMERICAN NATIONAL STANDARD COBOL

```

STNO -A...B... COBOL SOURCE STATEMENTS .....IDENTFCN SEQ/NO S
1 IDENTIFICATION DIVISION.                                001010
2 PROGRAM-ID. COBINQ.                                     001020
3 REMARKS. THIS IS A SINGLE REQUESTOR INQUIRY PROGRAM THAT 001030
  USES THE DISPLAY FORMAT FACILITY OF CCP.                001040
4 ENVIRONMENT DIVISION.                                   002010
5 CONFIGURATION SECTION.                                  002020
6 SOURCE-COMPUTER. IBM-S3.                               002030
7 OBJECT-COMPUTER. IBM-S3.                               002040
8 INPUT-OUTPUT SECTION.                                  002050
9 FILE-CONTROL.                                          002060
10  SELECT CUSTNAME ASSIGN TO DA-5444-R-CUSTNAME          002070
    ACCESS IS RANDOM                                     002080
    ACTUAL KEY IS NAMEKEY.                               002090
11  SELECT CUSTSHIP ASSIGN TO DA-5444-R-CUSTSHIP          002100
    ACCESS IS RANDOM                                     002110
    ACTUAL KEY IS SHIPKEY.                               002120
12  SELECT INVENTORY ASSIGN TO DA-5444-R-INVENTORY        002130
    ACCESS IS RANDOM                                     002140
    ACTUAL KEY IS INVTKEY.                               002150
13 DATA DIVISION.                                       003010
14 FILE SECTION.                                         003020
15 FD CUSTNAME                                           003030
    DATA RECORD IS CUST-NAME                            003040
    LABEL RECORDS ARE STANDARD.                          003050
16 01 CUST-NAME.                                         003060
17 05 FILLER PICTURE X(8).                               003070
18 05 CUSNM1 PICTURE X(22).                              003080
19 05 FILLER PICTURE X(270).                             003085
20 FD CUSTSHIP                                           003090
    DATA RECORD IS CUST-SHIP                            003100
    LABEL RECORDS ARE STANDARD.                          003110
21 01 CUST-SHIP.                                         003120
22 05 FILLER PICTURE X(8).                               003130
23 05 SHPNM1 PICTURE X(22).                              003140
24 05 FILLER PICTURE X(98).                             003145
25 FD INVENTORY                                          003150
    DATA RECORD IS INVEN                                003160
    LABEL RECORDS ARE STANDARD.                          003170
26 01 INVEN.                                             003180
27 05 FILLER PICTURE X(13).                              003190
28 05 ITDSC1 PICTURE X(20).                              003200
29 05 FILLER PICTURE X(276).                            003205
30 WORKING-STORAGE SECTION.                              004010
31 77 NAMEKEY PICTURE S9(7) VALUE ZERO.                 004020
32 77 SHIPKEY PICTURE S9(7) VALUE ZERO.                 004030
33 77 INVTKEY PICTURE S9(7) VALUE ZERO.                 004040
34 77 NO-CUSNO PIC X(22) VALUE 'NO CUST NUMBER ENTERED'. 004041
35 77 NO-SHPNO PIC X(22) VALUE 'NO SHIP NUMBER ENTERED'. 004042
36 77 NO-ITMNO PIC X(22) VALUE 'NO ITEM NUMBER ENTERED'. 004043
37 77 INVALID-C PIC X(22) VALUE 'INVALID CUST NUMBER '. 004044
38 77 INVALID-S PIC X(22) VALUE 'INVALID SHIP NUMBER '. 004045
39 77 INVALID-I PIC X(22) VALUE 'INVALID ITEM NUMBER '. 004046
*****                                                    004050
* CCP-COBOL INTERFACE PARAMETER LIST.                    * 004060
*****                                                    004070
40 01 PARAMETER-LIST.                                    004080
41 05 PL-RTC PICTURE S9(4) COMP-4.                      004090
42 05 PL-OPC PICTURE S9(4) COMP-4 VALUE 4.              004100
43 05 PL-OUL PICTURE S9(4) COMP-4 VALUE 128.            004110
44 05 PL-EFL REDEFINES PL-OUL PIC S9(4) COMP-4.         004120
45 05 PL-ATI REDEFINES PL-OUL PIC S9(4) COMP-4.         004130
46 05 PL-INL PICTURE S9(4) COMP-4 VALUE 16.              004140
47 05 FILLER PICTURE X(8).                              004150
*****                                                    004160
* INPUT RECORD AREA.                                     * 004170
*****                                                    004180
48 01 INPUT-A.                                           004190
49 05 TERM-IN PICTURE X(6).                              004200
50 05 CUSNO PICTURE X(6) VALUE '000000'.                004220
51 05 SLASH PICTURE X(1).                                004230
52 05 ITMNO PICTURE X(8) VALUE '00000000'.              004240
*****                                                    004250
* PUT FORMAT OUTPUT AREA.                               * 004260
*****                                                    004270
53 01 OUTPUT-A.                                          004280
54 05 TERM-OUT PICTURE X(6).                             004290
55 05 SCREEN PICTURE X(6) VALUE '$ZRPIQ'.               004300
56 05 CUSNO PICTURE X(6) VALUE '000000'.                004310
57 05 CUSNM PICTURE X(22) VALUE SPACES.                 004320
58 05 SHPNM PICTURE X(22) VALUE SPACES.                 004330
59 05 ITMNO PICTURE X(8) VALUE '00000000'.              004340
60 05 ITDSC PICTURE X(20) VALUE SPACES.                 004350
61 PROCEDURE DIVISION.                                  005010
62 OPEN-THE-FILE.                                       005020

```

Example 3 - COBOL Coding for SRT Inquiry Program (Part 1 of 2)

63	OPEN INPUT CUSTNAME.	005030
64	OPEN INPUT CUSTSHIP.	005040
65	OPEN INPUT INVENTORY.	005050
66	ACCEPT-INPUT.	005060
67	MOVE 4 TO PL-OPC.	005070
68	MOVE 15 TO PL-INL.	005080
69	CALL 'CCPCIO' USING PARAMETER-LIST, INPUT-A.	005090
70	IF PL-RTC NOT EQUAL TO 0 GO TO END-OF-JOB.	005100
72	GET-DISKDATA.	005110
73	MOVE CUSNO OF INPUT-A TO NAMEKEY.	005120
74	MOVE CUSNO OF INPUT-A TO SHIPKEY.	005130
75	MOVE ITMNO OF INPUT-A TO INVTKEY.	005140
76	MOVE CUSNO OF INPUT-A TO CUSNO OF OUTPUT-A.	005142
77	MOVE ITMNO OF INPUT-A TO ITMNO OF OUTPUT-A.	005144
78	IF NAMEKEY IS EQUAL TO ZERO MOVE NO-CUSNO TO CUSNM	005150
80	ELSE PERFORM READ-C.	005160
82	TAG1.	005165
83	IF SHIPKEY IS EQUAL TO ZERO MOVE NC-SHPNO TO SHPNM	005170
85	ELSE PERFORM READ-S.	005180
87	TAG2.	005185
88	IF INVTKEY IS EQUAL TO ZERO MOVE NO-ITMNO TO ITDSC	005190
90	ELSE PERFORM READ-I.	005200
92	TAG3.	005201
93	GO TO CONTINUE-1.	005205
94	READ-C.	005210
95	READ CUSTNAME INVALID KEY GO TO ERROR-C.	005220
97	MOVE CUSNM1 TO CUSNM.	005230
98	GO TO TAG1.	005235
99	READ-S.	005240
100	READ CUSTSHIP INVALID KEY GO TO ERROR-S.	005250
102	MOVE SHPNM1 TO SHPNM.	005260
103	GO TO TAG2.	005265
104	READ-I.	005270
105	READ INVENTORY INVALID KEY GO TO ERROR-I.	005280
107	MOVE ITDSC1 TO ITDSC.	005290
108	GO TO TAG3.	005295
109	ERROR-C.	005300
110	MOVE INVALID-C TO CUSNM.	005310
111	GO TO TAG1.	005320
112	ERROR-S.	005330
113	MOVE INVALID-S TO SHPNM.	005340
114	GO TO TAG2.	005350
115	ERROR-I.	005360
116	MOVE INVALID-I TO ITDSC.	005370
117	GO TO TAG3.	005380
118	CONTINUE-1.	005390
119	PUT-ROUTINE.	005400
120	MOVE 50 TO PL-OPC.	005410
121	MOVE 84 TO PL-OUL.	005420
122	MOVE TERM-IN TO TERM-OUT.	005430
123	CALL 'CCPCIO' USING PARAMETER-LIST, OUTPUT-A.	005440
124	END-OF-JOB.	005500
125	CLOSE CUSTNAME.	005510
126	CLOSE CUSTSHIP.	005520
127	CLOSE INVENTORY.	005530
128	STOP RUN.	005540

Example 3 – COBOL Coding for SRT Inquiry Program (Part 2 of 2)

```

1      PROGRAM FORINQ
2      DEFINE FILE 8(100,300,L,IR8),9(100,128,L,IR9),10(100,309,L,IR10)
3      INTEGER * 2 PARM(8),RTC,OPC,OUTL,INL,ICUST,ISHIP,ITEM
4      INTEGER * 2 PUTBUF(92),BUFFER(309),FNAME(6),INBUF(22)
5      INTEGER * 2 MSG1(22),MSG2(22),MSG3(22),MSG4(22),MSG5(22)
6      INTEGER * 2 MSG6(22),SLASH
7      EQUIVALENCE (RTC,PARM(1)),(OPC,PARM(2))
8      EQUIVALENCE (OUTL,PARM(3)),(INL,PARM(4))
9      DATA FNAME /' $ Z R P I Q '/
10     DATA MSG1 /' N O   C U S T   N U M B E R   E N T E R E D '/
11     DATA MSG2 /' N O   S H I P   N U M B E R   E N T E R E D '/
12     DATA MSG3 /' N O   I T E M   N U M B E R   E N T E R E D '/
13     DATA MSG4 /' I N V A L I D   C U S T   N U M B E R           '/
14     DATA MSG5 /' I N V A L I D   S H I P   N U M B E R           '/
15     DATA MSG6 /' I N V A L I D   I T E M   N U M B E R           '/
16     DATA SLASH /' / ' /
17     OPC=4
18     OUTL=16
19     INL=16
20     CALL CCPFIO(PARM,INBUF)
21     IF (RTC .NE. 0) GO TO 140
22     XCUST=GET(INBUF,7,12,1.0)
23     ICUST=XCUST
24     ISHIP=ICUST
25     CALL MOVE(INBUF,1,6,PUTBUF,1)
26     CALL MOVE(FNAME,1,6,PUTBUF,7)
27     CALL MOVE(INBUF,7,12,PUTBUF,13)
28     CALL MOVE (INBUF,14,21,PUTBUF,63)
29     IF (ICUST) 10,10,20
30     10 CALL MOVE(MSG1,1,22,PUTBUF,19)
31     GO TO 30
32     20 READ (8*ICUST,1000,ERR=110)(BUFFER(I),I=1,300)
33     CALL MOVE(BUFFER,9,30,PUTBUF,19)
34     30 IF (ISHIP) 40,40,50
35     40 CALL MOVE(MSG2,1,22,PUTBUF,41)
36     GO TO 60
37     50 READ (9*ISHIP,1010,ERR=120)(BUFFER(I),I=1,128)
38     CALL MOVE(BUFFER,9,30,PUTBUF,41)
39     60 IF (INBUF(13)-SLASH) 80,70,80
40     70 XTEM=GET(INBUF,14,21,1.0)
41     ITEM=XTEM
42     IF (ITEM) 80,80,90
43     80 CALL MOVE(MSG3,1,22,PUTBUF,71)
44     GO TO 100
45     90 READ (10*ITEM,1020,ERR=130)(BUFFER(I),I=1,309)
46     CALL MOVE(BUFFER,14,33,PUTBUF,71)
47     100 OPC=50
48     OUTL=84
49     INL=84
50     CALL CCPFIO(PARM,PUTBUF)
51     GO TO 140
52     110 CALL MOVE(MSG4,1,22,PUTBUF,19)
53     GO TO 30
54     120 CALL MOVE(MSG5,1,22,PUTBUF,41)
55     GO TO 60
56     130 CALL MOVE(MSG6,1,22,PUTBUF,71)
57     GO TO 100
58     140 STOP
59     1000 FORMAT(150A1)
60     1010 FORMAT(128A1)
61     1020 FORMAT(103A1)
62     END

```

Example 3 – FORTRAN IV Coding for SRT Inquiry Program

**Example 4 – RPG II Order Entry Program (Using PRUF with the DFF)**

This illustrates the use of the PRUF capability with RPG II. The example is an order entry application program. Example 2 in this chapter illustrates an order entry application written as one large MRT program, and as can be seen, the logic becomes quite complicated. By using PRUF with DFF, the one program can be broken up into six smaller and logically much simpler programs. The six sample programs are described as follows:

*ORDERL-SRT Program Requested from the System Console*

ORDERL reads cards from the MFCU and formats the direct output HOLDORDR file (Figure 8-20), which will be used as a spool-type file to contain all orders entered from the 3270 terminals, with the master record for the file and all header records for the 3270 order entry terminals. The ORDERL program (Figure 8-21) requires input in the following format:

- Card 1, Columns 1 – A
- Columns 2-7 – Number of 64 character records per allocation unit
- Cards 2-n, Columns 1 – H
- Columns 2-7 – Symbolic terminal name of 3270 order entry terminal
- Columns 8-9 – Number of allocation units to be assigned this terminal

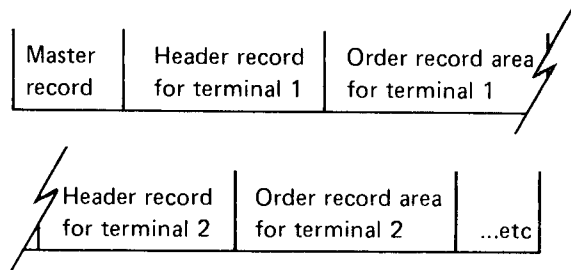
The advantage of this order entry application is that the inventory file is not updated until the terminal's PF1 key is pressed, which indicates that the order is complete. All orders are written to the HOLDORDR file, from which they can be printed via a stand-alone print program. They also can be printed at the time the orders are processed via a print program that runs as a CCP user task.

The format of the HOLDORDR file is as follows:

Master record	Order records for terminal 1	Order records for terminal 2	.....	Order records for terminal n
---------------	------------------------------	------------------------------	-------	------------------------------

For this example, *n* equals a maximum of six terminals.

Prior to running the order entry program, the HOLDORDR file must be created in the following format:



The master record is 64 characters long and is the first record of the HOLDORDR file. The format of the master record is as follows:

Column/ Positions	Value	Meaning
1	A	Identifies the master record
2-7	RRECT	A 6-digit number, representing the number of records between header records
8-13	TERMNAME,1	Symbolic name of terminal
14-19	TERMNAME,2	
.	.	
.	.	
.	.	
38-43	TERMNAME,6	

The format of the 64 character header record is as follows:

Column/ Positions	Value	Meaning
1	H	Identifies the header record
2-7	RRECDN	Relative record number of the next available record to start an order; initialized to relative record number of header record + 1
8-13	RRECDP	Relative record number of next order to print; initialized to the same value as RRECDN
14-19	TERMNAME	Symbolic name of terminal this space is allocated to

When the HOLDORDR file was created, space was allocated (RRECT) to contain a predetermined number of order records between header records. When a terminal has been found to have a high level of order entry, it is of great value to the user to create more than one header record for that terminal. This technique can multiply the space available for a particular terminal.

When an order has been successfully entered, RRECDN will be updated to point to the record where the next order will start. When an order has been printed, RRECDP is updated to point to the next order to be printed. The printing of an order will allow another order to be entered in that record area (Figure 8-26).

Prior to starting a new order, if RRECDP = RRECDN, both values are updated to point to the header record +1. This means that all the order record areas for that terminal are available.

The HOLDORDR file is dynamic and needs to be formatted only initially.

The order entry application outputs an order in the following format:

Record Number Relative to Start of Order	Position	Value	Meaning
1	1	N	Identifies order heading
	2	1	
	3-10	ORDNO	Number assigned to order
	11-16	CUSNO	Customer number
2	17-38	CUSNAM	Customer name
	39-60	CUSA1	Customer address field 1
	1	N	Identifies order heading
	2	2	
	3-24	CUSA2	Customer address field 2
	25-46	CUSA3	Customer address field 3
3	1	N	Identifies order heading
	2	3	
	3-24	SHPNAM	Ship-to name
4	25-46	SHPA1	Ship-to address field 1
	1	N	Identifies order heading
	2	4	
	3-24	SHPA2	Ship-to address field 2
	25-46	SHPA3	Ship-to address field 3
	47-51	ZIPCD	Zip code
5	1	D	First detail line of order
	2-3	LN	Detail line number
	4-9	QTY	Quantity
	10-17	ITMNO	Item number
	18-37	ITMDSC	Item description
R	38-48	PRICE	Price = Quantity* Price per item
	1	T	Total line
	2-12	Cost excl.	Tax
	13-22	State tax	
	23-32	Federal tax	
33-43	Total cost		

A detail record is written for every ITMNO entered by the order entry operator.

The order entry application functions as follows:

The order entry operator requests the order entry application by keying in ORDER1␣CUSNO/ORDNO where CUSNO is a 6-position customer number and ORDNO is an 8-position order number field. ORDER1 (Figure 8-22) reads the CUSTNAME and CUSTSHIP files, and if found performs a PRUF-Put message to the 3270 terminal of the data retrieved. The cursor is left positioned under the ship-to information. The operator can override this information if so desired. The first field of the screen is a type 7 output/input field which has the value of ORDER2␣. After the ship-to information has been updated, the operator presses the ENTER key. This is actually a program request for ORDER2 (Figure 8-23). ORDER2 inputs the fields using \$ZORD1 as the controlling format for inputting the program request data. ORDER2 writes the four N records to the HOLDORDR file and performs a PRUF-Put message operation to format the 3270 with header information for entering QTY and ITMNO. The first field on the screen is a type 7 output/input field of value ORDER3␣. The cursor is positioned to allow input of the QTY field. The operator keys in the QTY number and ITMNO number and presses the ENTER key. This is a program request for ORDER3 (Figure 8-24). ORDER3 inputs the QTY and ITMNO numbers using \$ZORD2 as the controlling format for the program request data. If the ITMNO number is found, the following information is displayed under the QTY and ITMNO headings:

```

LINE NO.  QTY  ITMNO  ITM DES  PRICE
02
01          xxxxxx xxxxxxxx . . . . . xxxx.xx

```

The cursor is positioned under the QTY field, however if line number 01 is found to be in error, the operator can backspace to the line number field and enter 01 followed by the correct QTY and ITMNO. ORDER3 will write out a D (detail) record in the HOLDORDR file if the line number field is not equal to the previous line number record. In this example, line number 01's D record is written following the successful entry of line number 02, or the PF1 key is pressed to indicate the order is complete.

When the PF1 key is pressed, ORDER3 issues a PRUF-Put message to the operator indicating to press the ENTER key which completes the order. Now a program request for ORDER4 is issued (Figure 8-25). ORDER4 inputs all D records written for this order, updates the inventory file, writes the T (total) record for the order to the HOLDORDR file, and updates the RRECDN value in the header record for that terminal. The terminal is now in command mode, and is available for the next order entry request or other program requests.

It should be noted that after issuing a PRUF-Put message, ORDER1 and ORDER2 go to end of job; that is, as the operator keys in data on the terminal, the terminal buffer is being used, not the user's program area in main storage. When the ENTER key is pressed, the next program is requested and all non-output fields with MDT-ON are passed under format control to the requested program. Figure 8-28 shows the assignment set requirements for the order entry application, and Figure 8-27 shows orders after they have been printed by ORDERP.

The formats used in this sample order entry program are illustrated in Figure 8-29. The following chart lists the programs and the formats they use:

Program	Formats
ORDER1	\$ZORD1 \$ZORD5
ORDER2	\$ZORD2 \$ZORD5
ORDER3	\$ZORD3 \$ZORD4 \$ZORD5
ORDER4	\$ZORD5 \$ZORD6

```

*A001000N32112N32112N32142N32102N*
*32002N32002
*H000018000003N32112
*
*N100000006000006BROWN WHOLESAL*
*SUPPLY SEVILLE
*N2PHILA., PENN
*
*N3SLAY BROS INC WELLS*
*
*N4PHILA., PENN TRUCK
*
* 00000
*
*D0100000100000001 BLUE FLUORES*
*CE 00000000013000
*D0200000200000001 BLUE FLUORES*
*CE 00000000026000
*
*T0000003900000000015600000001950*
*00000042510
*N100000010000010BLACK&JONES CHEM*
*ICAL C 19TH STREET
*N2PHILADELPHIA, PENNA.
*
*N3SAME
*
*N4 TRUCK
*
* 00000
*
*D0100000200000002 X02 PLUG
* 000000000003426
*D0200000300000003 X03 PLUG
* 000000000003600
*D0300000400000004 X04 PLUG
* 000000000006800
*
*T0000001382600000005530000000691*
*00000015070
*
*H002018002003N32142
*
*N100000005000005FAST ELECTRONICS*
* W GLENWOOD
*N2PITTSBURG, PENN
*
*N3SAME
*
*N4 AIR FREI*
*GHT 00000
*D0100000100000002 X02 PLUG
* 000000000001713
*D0200000100000001 BLUE FLUORES*
*CE 00000000013000
*
*D0300000400000001 BLUE FLUORES*
*CE 000000000052000
*T0000006671300000026680000003335*
*00000072716
*N100000009000009 JOSEPH CLARKE
* 1520 MAPLE STREET
*N2MAPLEWOOD, OHIO
*
*N3 SAM BLIZZARD SAME
*
*N4 SAME FIRST C*
*LASS MAIL 00000
*D0100000400000004 X04 PLUG
* 000000000010500
*
*T0000001730000000000692000000865*
*00000018857
*
*H003018003003N32102
*
*N100000007000007BENNETT INC
* CHESTNUT
*N2PHILA., PENN
*
*N3SAME
*
*N4 AIR FREI*
*GHT 00000
*D0100000100000001 BLUE FLUORES*
*CE 00000000013000
*D0200000200000002 X02 PLUG
* 000000000003426
*
*D0300000300000003 X03 PLUG
* 000000000003600
*T0000002002600000008010000001001*
*00000021828
*N100000008000008 STEPHANIE BOOKS*
* MAPLE STREET
*N2 GREENVILLE, NEW YORK
*
*N3 JOHN BOOKS CLEMENTS*
*ON COLLEGE
*N4 GETTYSBURG, PENNA. PONY EX*
*PRESS 00000
*D0100000100000001 BLUE FLUORES*
*CE 00000000013000
*D0200000400000004 X04 PLUG
* 000000000006800
*
*T0000001980000000007920000000990*
*00000021582
*
*H004017004003N32002
*
*N100000003000003D & C SHIPPERS
* FILBERT
*N2 PHILA., PENN
*
*N3SAME
*
*N4 AIR FREI*
*GHT 00000
*D0100000500000005 X05 PLUG
* 00000000010500
*D0200000600000006 XZ6 PLUG
* 00000000010800
*
*T0000002130000000008520000001056*
*00000023217
*N100000003000002CARLSON BYRNES C*
* N BROAD
*N2PHILA., PENN
*
*N3W K GARRISON CO SANS*
*OM
*
*N4PHILA., PENN TRUCK
*
* 00000
*D0100000200000003 X03 PLUG
* 00000000002400
*D0200000100000002 X02 PLUG
* 00000000001713
*T000000041130000000164000000205*
*00000004482
*
*

```

Figure 8-20. HOLDORDR File Organization



```

0101 H   R   4
0201 FCARDIN IPE      96 96      MFCU1
0202 FHOLDORD5OC     64 64R     DISK45
0203 FPRINTOUTC     132 132     PRINTER
0301 E
0401 ICARDIN AA 10 1 CA      6 6
0402 I
0403 I          AB 20 1 CH      2 7ORRECT      40
0404 I          GR 30 1NCA 1NCH
0405 I
0406 I          2 7 TRMM      8 90N      85
0407 I          1 96 CARD
0501 C 30
0502 CORN8ON10
0503 CORN80 10 40      SETON      LR50      IF INVALID CARD CRDERL
                                GOTO END      OR RRECT ZERO ORDERL
0504 C LR      SETON      LR50      SET ON LR AND ORDERL
0505 C 10      Z-ADDO      I 10      PRINT ERROR MSGORDERL
0506 C 10      SETON      80      CHECK FOR ORDERORDERL
0507 C 10      Z-ADD2      RRECDH 60      OF INPUT CARDS ORDERL
0508 C 10      GOTO END      READ NEXT CARD CRDERL
0509 C          MOVE RRECDH      RREC1 60
0510 C          ADD 1      RRECDD 60      HDR REC PLUS 1 ORDERL
0511 C          LOOP      TAG
0512 C          I      ADD 1      I
0513 C          7      COMP I      70MAX OF 6 TERMS ORDERL
0514 C 70      SETON      50LR
0515 C 50      GOTO END
0516 C          MOVE TRMM      TERM,I
0517 C          ADD RRECT      RRECDH      BUMP TO NXT HDRORDERL
0518 C N85      N      SUB 1      N      8585
0519 C N85      GOTO LOOP
0520 C          RREC1      CHAINHOLDORD5      60      GET HEADER REC CRDERL
0521 C 60      SETON      LR50      FOR THIS TERM CRDERL
0522 C          END      TAG
0523 C          SETOF      85
0524 CLRN50      1      CHAINHOLDORD5      GET MASTER REC CRDERL
0601 OPRINTOUTH 201 1P
0602 O
0603 O          13 'RECORD NUMBER'
0604 O          D 1 20      30 'TERMINAL NAME'
0605 O          RREC1 10
0606 O          TERM,I 25
0607 O          T 3 LR 50
0608 O          CARD 96
0609 O          10 40      125 'RECORDS/TERMINAL INVALID'
0610 O          30      126 'INVALID CARD TYPE'
0611 O          60      126 'END OF ORDER FILE'
0612 O          70      126 'MORE THAN 9 TERMINALS'
0613 OHOLDORD5D 20
0614 O          1 'H'
0615 O          RRECDD 7
0616 O          RRECDD 13
0617 O          TERM,I 19
0618 O          T LRN50
0619 O          1 'A'
0620 O          RRECT 7
0621 O          TERM,1 13
0622 O          TERM,2 19
0623 O          TERM,3 25
0624 O          TERM,4 31
0625 O          TERM,5 37
0626 O          TERM,6 43

```

Figure 8-21. ORDERL Program

```

0101 H R 4
0201 FTERMIN IP 29 29 SPECIAL SUBR92 ORDER1
0202 F KPL CRDER1
0203 FTERMOUT O 215 215 SPECIAL SUBR92 ORDER1
0204 F KPL CRDER1
0205 FCUSTNAMEIC 300 300R DISK CRDER1
0206 FCUSTSHIPIC 256 128R DISK CRDER1
0300 E* THE FOLLOWING COMPILE TIME ARRAY HAS ACCEPT INPUT LENGTH OF 29 ORDER1
0301 E PL 5 5 6 CRDER1
0400 I* THE FOLLOWING INPUT IS DATA WITH PROGRAM REQUEST ORDER1
0401 ITERMIN AA 01 ORDER1
0402 I 1 20RTNCOO ORDER1
0403 I 15 200CUSNO 80 CRDER1
0404 I 21 21 SLASH CRDER1
0405 I 22 290ORDERN 81 ORDER1
0406 ICUSTNAMEAA 10 1 CM 2 CA CRDER1
0407 I 9 30 CUSNM CRDER1
0408 I 34 55 CUSA1 ORDER1
0409 I 59 80 CUSA2 CRDER1
0410 I 84 105 CUSA3 ORDER1
0411 ICUSTSHIPAA 11 1 CM 2 CB ORDER1
0412 I 9 30 SHPNM ORDER1
0413 I 34 55 SHIP1 ORDER1
0414 I 59 80 SHIP2 CRDER1
0415 I 84 105 SHIP3 ORDER1
0416 I 109 1130ZIPCD CRDER1
0501 C SETON LR20 ORDER1
0502 C 91 CRDER1
0503 COR 92 SETON 30 CRDER1
0504 C 30 CRDER1
0505 COR 80 CRDER1
0506 COR 81 GOTO END CRDER1
0507 C '/' COMP SLASH 8181 VALID SEPARATORORDER1
0508 C 81 GOTO END CRDER1
0509 C CUSNO CHAINCUSTNAME 85 VALID NAME ORDER1
0510 C N85 10 CUSNO CHAINCUSTSHIP 86 VALID ORDER NUMORDER1
0511 C 85 FOUND INPUT ERRORORDER1
0512 COR 86 DO PUT OVERRIDEORDER1
0513 CORN10 ORDER1
0514 CORN11 GOTO END ORDER1
0515 C SETOF 20 NO PUT OVERRIDEORDER1
0516 C END TAG CRDER1
0600 O* IF PUT OVERRIDE OCCURS MUST MAKE ANOTHER PROGRAM REQUEST ORDER1
0517 C SETOF 9192 ORDER1
0601 OTERMOUT O LR 20 ORDER1
0602 O 4 ' CB' ORDER1
0603 O 8 '42' ORDER1
0604 O PL,5 14 ORDER1
0605 O 20 '$ZORD5' CRDER1
0606 O N11 40 'INVALID CUSSHIP FILE' CRDER1
0607 O N10 40 'INVALID CUSTMAS FILE' CRDER1
0608 O 86 40 'NO CUSSHIP RECORD ' CRDER1
0609 O 85 40 'NO CUSTMAS RECORD ' ORDER1
0610 O 81 40 'INVALID ORDER NUMBER' CRDER1
0611 O 80 40 'INVALID CUST. NUMBER' ORDER1
0612 O 30 40 'ERROR RETURN CODE = ' CRDER1
0613 O 30 RTNCOO 42 ORDER1
0614 O T LRN20 ORDER1
0615 O 4 ' GB' CRDER1
0616 O 8 '215' ORDER1
0617 O PL,5 14 CRDER1
0618 O 20 '$ZORD1' ORDER1
0619 O ORDERN 28 CRDER1
0620 O CUSNO 34 CRDER1
0621 O CUSNM 56 CRDER1
0622 O SHPNM 78 CRDER1
0623 O CUSA1 100 ORDER1
0624 O SHIP1 122 CRDER1
0625 O CUSA2 144 ORDER1
0626 O SHIP2 166 ORDER1
0627 O CUSA3 188 CRDER1
0628 O SHIP3 210 ORDER1
0629 O ZIPCD 215 ORDER1
**
D 29

```

Figure 8-22. ORDER1 Program

```

0101 H   R   4
0201 FTERMIN IP   217 217          SPECIAL      SUBR92      ORDER2
0202 F                                     KPL      ORDER2
0203 FTERMOUT 0   280 280          SPECIAL      SUBR92      ORDER2
0204 F                                     KPL      CRDR2
0205 FHOLDORD5UC   64 64R          DISK45      ORDER2
0301 E                                     PL   5   5 6      ORDER2
0302 E                                     TERM  6   6      ORDER2
0401 ITERMIN  AA  01
0402 I                                     1   20RTNCOD      ORDER2
0403 I                                     15  15 AID      ORDER2
0404 I                                     16  22 PGMNAM      CRDR2
0405 I                                     23 300ORDNUM      ORDER2
0406 I                                     31 360CUSNO      CRDR2
0407 I                                     37  58 CUSNM      CRDR2
0408 I                                     59  80 SHPNM      ORDER2
0409 I                                     81 102 CUSA1      ORDER2
0410 I                                     103 124 SHPA1      ORDER2
0411 I                                     125 146 CUSA2      CRDR2
0412 I                                     147 168 SHPA2      CRDR2
0413 I                                     169 190 CUSA3      ORDER2
0414 I                                     191 212 SHPA3      CRDR2
0415 I                                     213 217OZIPC0      ORDER2
0416 IHOLDORD5AA 30   1 CA
0417 I                                     2   7ORRECT      50      ORDER2
0418 I                                     8   13 TERM,1      ORDER2
0419 I                                     14  19 TERM,2      ORDER2
0420 I                                     20  25 TERM,3      ORDER2
0421 I                                     26  31 TERM,4      ORDER2
0422 I                                     32  37 TERM,5      ORDER2
0423 I                                     38  43 TERM,6      ORDER2
0424 I           AB  40   1 CH
0425 I           OR  80   INCH  1NCA
0426 I                                     2   7ORRECD      50      ORDER2
0427 I                                     8   13ORRECDP      55      ORDER2
0428 I                                     14  19 TRMM      ORDER2
0501 C                                     SETON      1020LR      ORDER2
0502 C   91
0503 COR 92          GOTO END      NONE ZERO RETURPRDR2
0504 C           SETOF      CODE PUT ERRMSGORDER2
0505 C           1          CHAINHOLDORD5      20      ORDER2
0506 C   50          60      GET MASTER REC      ORDER2
0507 COR 60          BAD ORDER FILE      CRDR2
0508 CORN30
0509 C           GOTO END      CRDR2
0510 C           Z-ADD2      RREC#   60      CRDR2
0511 C           MOVE 1      I   10      CRDR2
0512 C   N70   PL,5      LOKUPTERM,I      70FIND THIS TERM      CRDR2
0513 C   56          SETON      56      INVALID TERM      ORDER2
0514 C           TAG1      GOTO END      ORDER2
0515 C           I          TAG      CRDR2
0516 C   N71   RREC#      SUB 1      I   71      CRDR2
0517 C   N71          ADD RRECT      RREC#      BUMP REC NUM TCCRDR2
0518 C           GOTO TAG1      POINT TO HEADERCORDER2
0519 C           RREC#      MOVE RREC#      RRECDH  60      REC FOR THIS      ORDER2
0520 C   50          CHAINHOLDORD5      60      TERMINAL      ORDER2
0521 COR 55
0522 COR 60
0523 CORN40
0524 C           RRECDD      GOTO END      CRDR2
0525 C   65   RRECDH      COMP RRECDP      65IF ALL ORDERS      ORDER2
0526 C   65          ADD 1      RRECDD      FOR THIS TERM      ORDER2
0527 C   65          MOVE RRECD      RRECDP      PRINTED, RESET      CRDR2
0528 C           EXCPT      TO FIRST RECORD      ORDER2
0529 C           SETOF      MOVE RRECDD      RREC#   65      ORDER2
0530 C           EXSR READF      ORDER2
0531 C   60          GOTO END      CRDR2
0532 C           SETON      51      FIRST 'N' REC      ORDER2
0533 C           EXSR CHECK      ORDER2
0534 C   60          GOTO END      ORDER2
0535 C           SETON      52      SECOND 'N' REC      ORDER2
0536 C           EXSR CHECK      CRDR2
0537 C   60          GOTO END      ORDER2
0538 C           SETON      53      THIRD 'N' REC      CRDR2
0539 C           EXSR CHECK      ORDER2
0540 C   60          GOTO END      ORDER2

```

Figure 8-23 (Part 1 of 2). ORDER2 Program

```

0541 C          SETON          54  FOURTH 'N' REC ORDER2
0542 C          EXSR CHECK          ORDER2
0543 C    60          GOTO END          ORDER2
0544 C          SETOF          1020          ORDER2
0545 C          END          TAG          ORDER2
0546 C          SETOF          9192          ORDER2
0547 CSR          READF          BEGSR          ORDER2
0548 CSR          SETOF          304080          ORDER2
0549 CSR          RREC#          CHAINHOLDORD5 60  GET NEXT RECORD ORDER2
0550 CSR    60          SETON          40          ORDER2
0551 CSR          ENDSR          ORDER2
0552 CSR          CHECK          BEGSR          CRDER2
0553 CSR          EXCPT          ORDER2
0554 CSR          RREC#          ADD 1          RREC#          WRITE REC TO ORDER2
0555 CSR          EXSR READF          HOLDORD5 FILE ORDER2
0556 CSR          ENDSR          ORDER2
0601 OHOLDORD5E          51          ORDER2
0602 O          1 'N'          ORDER2
0603 O          N52N53N54          2 '1'          CRDER2
0604 O          N52N53N54ORDNUM          10          ORDER2
0605 O          N52N53N54CUSNO          16          ORDER2
0606 O          N52N53N54CUSNM          38          ORDER2
0607 O          N52N53N54CUSA1          60          CRDER2
0608 O          52N53N54          2 '2'          CRDER2
0609 O          52N53N54CUSA2          24          ORDER2
0610 O          52N53N54CUSA3          46          ORDER2
0611 O          53N54          2 '3'          ORDER2
0612 O          53N54  SHPNM          24          ORDER2
0613 O          53N54  SHPA1          46          ORDER2
0614 O          54          2 '4'          ORDER2
0615 O          54  SHPA2          24          ORDER2
0616 O          54  SHPA3          46          CRDER2
0617 O          54  ZIPCD          51          ORDER2
0618 O          E          65          CRDER2
0619 O          RRECD          7          ORDER2
0620 O          RRECDP          13          CRDER2
0621 OTERMOUT O          10          ORDER2
0622 O          4 ' CB'          ORDER2
0623 O          8 '42'          ORDER2
0624 O          PL,5          14          CRDER2
0625 O          20          20 '$ZORD5'          ORDER2
0626 O          N20N60          40 'ERROR RETURN CODE = '          CRDER2
0627 O          N20 56          40 'INVALID ORDER FILE '          ORDER2
0628 O          N20 60          40 'TERMINAL NOT IN FILE'          ORDER2
0629 O          20          40 'END OF ORDER FILE '          CRDER2
0630 O          RTNCO          42          ORDER2
0631 O          D          N10 LR          ORDER2
0632 O          4 ' GB'          ORDER2
0633 O          8 '280'          ORDER2
0634 O          PL,5          14          CRDER2
0635 O          20          20 '$ZORD2'          ORDER2
0636 O          RRECDH          26          CRDER2
0637 O          RREC#          32          ORDER2
0638 O          ORDNUM          40          ORDER2
0639 O          CUSNO          46          CRDER2
0640 O          CUSNM          68          ORDER2
0641 O          SHPNM          90          ORDER2
0642 O          CUSA1          112          ORDER2
0643 O          SHPA1          134          CRDER2
0644 O          CUSA2          156          ORDER2
0645 O          SHPA2          178          ORDER2
0646 O          CUSA3          200          CRDER2
0647 O          SHPA3          222          ORDER2
0648 O          ZIPCD          227          ORDER2
**

```

D 217

Figure 8-23 (Part 2 of 2). ORDER2 Program

```

0101 H R 4
0201 FTERMIN IP 103 103 SPECIAL SUBR92 ORDER3
0202 F KPL ORDER3
0203 FTERMOUT O 123 123 SPECIAL SUBR92 ORDER3
0204 F KPL ORDER3
0205 FHOLDORD5UC 64 64R DISK45 ORDER3
0206 FINVENTRYIC 300 300R DISK ORDER3
0301 E PL 5 5 6 ORDER3
0302 E TERM 6 6 ORDER3
0303 E PRI 6 15 2 ORDER3
0401 ITERMIN AA 10 15 C% ORDER3
0402 I BB 35 15 C' ORDER3
0403 I OR 36 15 C1 ORDER3
0404 I OR 60 15NC' 15NC1 15NC% ORDER3
0405 I 1 20RTNCOD ORDER3
0406 I 15 15 AID ORDER3
0407 I 16 22 PGMNAM CRDR3
0408 I 23 28ORRECDH ORDER3
0409 I 29 34ORRECD ORDER3
0410 I 35 360LIN1 79 ORDER3
0411 I 37 420QTY1 80 ORDER3
0412 I 43 500ITM1 81 ORDER3
0413 I 51 520LIN2 ORDER3
0414 I 53 580QTY2 ORDER3
0415 I 59 660ITM2 ORDER3
0416 I 67 88 DESC ORDER3
0417 I 89 1032PR CRDR3
0418 IHOLDORD5AA 65 1 CH CRDR3
0419 I OR 70 1NCH CRDR3
0420 I 14 19 TRMNM ORDER3
0421 IINVENTRYAA 75 1 CM 2 CI ORDER3
0422 I 13 32 ITMDSC CRDR3
0423 I 42 472PRICE ORDER3
0424 I 167 1720QTYOH CRDR3
0501 C SETOF 2582 CRDR3
0502 C SETOF 1518 CRDR3
0503 C SETON 302019 ORDER3
0504 C Z-ADD1 I 10 CRDR3
0505 C PL,5 LOKUPTERM,I 31 ORDER3
0506 C N31 ' ' LOKUPTERM,I 31 CRDR3
0507 C MOVE PL,5 TERM,I CRDR3
0508 C 91 CRDR3
0509 COR 92 GOTO END ORDER3
0510 C SETOF 30 CRDR3
0511 C 10 GOTO END PA1 CANCEL PGM CRDR3
0512 C LIN2 COMP O 82 LIN2 ZERO WITH ORDER3
0513 C N82 36 GOTO END PF1 KEY CRDR3
0514 C SETOF 20 LIN2 NONE ZERO CRDR3
0515 C 36 82 GOTO TAG3 WITH PF1 KEY CRDR3
0516 C SETOF 19 CRDR3
0517 C SETON 25 CRDR3
0518 C 79 NO LINE NUMBER ORDER3
0519 COR 80 NO QUANTITY ORDER3
0520 COR 81 SETON 18 NO ITEM NUMBER CRDR3
0521 C 18 GOTO END CRDR3
0522 C ITM1 CHAININVENTORY 16 ITEM INVENTORY ORDER3
0523 C 16 GOTO END ITEM NOT FOUND CRDR3
0524 C N75 SETON 15 INVALID RECORD ORDER3
0525 C 15 GOTO END CRDR3
0526 C QTYOH COMP QTY1 17 BACKORDER REQ'D ORDER3
0527 C LIN1 COMP LIN2 21 LIN1 = LIN2 OR ORDER3
0528 C 21 LIN2 = ZERO CRDR3
0529 CORN82 GOTO TAG2 ORDER3
0530 C TAG3 TAG CRDR3
0531 C RRECDD CHAINHOLDORD5 85 END OF ORD FILE CRDR3
0532 C 65 SETON 85 HEADER RECORD ORDER3
0533 C 85 SETON 652019RELEASE TERM CRDR3
0534 C 85 SETOF 823625AND CANCEL PGM CRDR3

```

Figure 8-24 (Part 1 of 2). ORDER3 Program

```

0535 C 20 GOTO END ORDER3
0536 C SETON 70 ORDER3
0537 C EXCPT PUT ORDER REC ORDER3
0538 C SETOF 70 ORDER3
0539 C RRECDD ADD 1 RRECDD ORDER3
0540 C 36 82 SETOF 25 ORDER3
0541 C 36 82 GOTO END ORDER3
0542 C TAG2 TAG ORDER3
0543 C MOVE LIN1 LIN2 MOVE FIELDS TO ORDER3
0544 C LIN1 ADD 1 LIN1 LINE TWO ON ORDER3
0545 C MOVE QTY1 QTY2 DISPLAY SCREEN ORDER3
0546 C MOVE ITM1 ITM2 ORDER3
0547 C MOVE ITMDSC DESC ORDER3
0548 C PRICE MULT QTY1 PR ORDER3
0549 C MOVE PR PRI,I ORDER3
0550 C END TAG ORDER3
0551 C SETOF 9192 ORDER3
0552 C 19 EXCPT ORDER3
0553 C 19 MOVE ' ' TERM,I ORDER3
0554 C 19 MOVE PL,3 INVC 20 ORDER3
0555 C 19 COMP 00 LR ORDER3
0601 Q TERMOUT E 20N70 ORDER3
0602 O 4 ' CB' ORDER3
0603 O 8 '42' ORDER3
0604 O PL,5 14 CRDR3
0605 O 20 '$ZORD5' CRDR3
0606 O 30 40 'ERROR RETURN CODE = ' CRDR3
0607 O 30 RTNCCD 42 CRDR3
0608 O 36 40 'NO ITEMS ENTERED ' CRDR3
0609 O 65 40 'END OF ORDER FILE ' CRDR3
0610 O D 25 ORDER3
0611 O 4 ' S' CRDR3
0612 O 8 '123' CRDR3
0613 O PL,5 14 ORDER3
0614 O 20 '$ZORD4' ORDER3
0615 O RRECDH 26 CRDR3
0616 O RRECDD 32 CRDR3
0617 O LIN1 34 CRDR3
0618 O 18 70 'ITEM ENTRIES INVALID' CRDR3
0619 O 17 70 'BACK ORDER ITEM ' CRDR3
0620 O 16 70 'ITEM NOT IN INVENTOR' CRDR3
0621 O 15 70 'INVALID INVEN. REC. ' CRDR3
0622 O LIN2 72 CRDR3
0623 O QTY2 78 CRDR3
0624 O ITM2 86 CRDR3
0625 O DESC 108 CRDR3
0626 O PR 2 123 CRDR3
0627 O E 36 82N70 ORDER3
0628 O 4 ' GB' CRDR3
0629 O 8 '32' CRDR3
0630 O PL,5 14 CRDR3
0631 O 20 '$ZORD3' CRDR3
0632 O RRECDH 26 CRDR3
0633 O RRECDD 32 CRDR3
0634 O E 19N70 ORDER3
0635 O 4 ' K' CRDR3
0636 O 8 '14' CRDR3
0637 O PL,5 14 CRDR3
0638 OHOLDORD5E 70 CRDR3
0639 O 1 'D' CRDR3
0640 O LIN2 3 CRDR3
0641 O QTY2 9 CRDR3
0642 O ITM2 17 CRDR3
0643 O DESC 39 CRDR3
0644 O PRI,I 54 CRDR3
**
D 103

```

Figure 8-24 (Part 2 of 2). ORDER3 Program

0101	H	R	4																			CRDR4
0201	F	TERMIN	IP	34	34			SPECIAL		SUBR92												CRDR4
0202	F									KPL												CRDR4
0203	F	TERMOUT	O	80	80			SPECIAL		SUBR92												CRDR4
0204	F									KPL												CRDR4
0205	F	HOLDORDSUC		64	64R			DISK45														CRDR4
0206	F	INVENTORYUC		300	300R			DISK														CRDR4
0301	E				PL			5	5	6												CRDR4
0401	I	TERMIN	AA	10																		CRDR4
0402	I								1		20RTNCD											CRDR4
0403	I								15	15	AID											CRDR4
0404	I								16	22	PGMNAM											CRDR4
0405	I								23	280RRECDH												CRDR4
0406	I								29	340RRECD												CRDR4
0407	I	HOLDORDSAA		20	1	CH																CRDR4
0408	I									2	70RRECD#											CRDR4
0409	I								14	19	TRMNM											CRDR4
0410	I		AB	25	1	CO																CRDR4
0411	I		OR	30	1	NCH	INCD															CRDR4
0412	I									4	90QTY#											CRDR4
0413	I								10	170ITM#												CRDR4
0414	I								40	542PR												CRDR4
0415	I	INVENTORYAA		60																		CRDR4
0416	I								161	1660QTY00												CRDR4
0417	I								167	1720QTY0H												CRDR4
0501	C						SETON				4050LR											CRDR4
0502	C	91																				CRDR4
0503	C	COR	92				GOTO END															CRDR4
0504	C						SETOF			50												CRDR4
0505	C			RRECD			CHAINHOLDORDS			83	CHECK IF NEXT											CRDR4
0506	C	N83	20	TRMNM			COMP PL,5			8383	REC IS HEADER											CRDR4
0507	C	83					GOTO END				FOR NEXT TERM											CRDR4
0508	C						SETOF			202530												CRDR4
0509	C			RRECDH			CHAINHOLDORDS			30	GET HEADER REC											CRDR4
0510	C	N20					SETON			30	FOR THIS TERM											CRDR4
0511	C	30					GOTO END															CRDR4
0512	C			RRECD			ADD 1	RREC1	60		UPDATE POINTER											CRDR4
0513	C						EXCPT				UPDATE HEADER											CRDR4
0514	C			RRECD#			ADD 4	RREC#	60		JUMP THE N RECS											CRDR4
0515	C						Z-ADDO	PRICE	112													CRDR4
0516	C			TAG1			TAG															CRDR4
0517	C						SETOF			202530												CRDR4
0518	C			RREC#			CHAINHOLDORDS				GET ITEM RECORD											CRDR4
0519	C	N25					SETON			30												CRDR4
0520	C	30					GOTO END															CRDR4
0521	C			PRICE			ADD PR	PRICE			ACCUMULAT PRICE											CRDR4
0522	C			RREC#			ADD 1	RREC#			BUMP RECORD CTR											CRDR4
0523	C			ITM#			CHAININVENTORY															CRDR4
0524	C			QTYOH			COMP QTY#			27	INCREMENT QUAN-											CRDR4
0525	C	27		QTY#			SUB QTYOH	QTYT	60		TITY ON ORDER											CRDR4
0526	C	27		QTYO			ADD QTYT	QTYCO														CRDR4
0527	C	27					Z-ADDO	QTYOH														CRDR4

Figure 8-25 (Part 1 of 2). ORDER4 Program

```

0528 C N27 QTYOH SUB QTY# QTYOH DECREMENT ONHANORDER4
0529 C SETON 27 ORDER4
0530 C EXCPT ORDER4
0531 C RREC# COMP RRECDD 26END OF ORDER? ORDER4
0532 C N26 GOTO TAG1 ORDER4
0533 C PRICE MULT .04 TAX1 102 CALC TAXES FOR ORDER4
0534 C PRICE MULT .05 TAX2 102 PRICE ON TOTAL ORDER4
0535 C PRICE ADD TAX1 PRT 112 ORDER4
0536 C PRT ADD TAX2 PRT ORDER4
0537 C RREC# CHAINHOLDORD5 GET NEXT RECORDORDER4
0538 C EXCPT WRITE TOTAL RECORDORDER4
0539 C SETOF 40 ORDER4
0540 C END TAG ORDER4
0541 C SETOF 9192 ORDER4
0601 OHOLDORD5E N26N27 ORDER4
0602 O RREC1 7 ORDER4
0603 O E 26 ORDER4
0604 O 1 'T' ORDER4
0605 O PRICE 12 ORDER4
0606 O TAX1 22 ORDER4
0607 O TAX2 32 ORDER4
0608 O PRT 43 ORDER4
0609 OINVENTRYE N26 27 ORDER4
0610 O QTYOD 166 ORDER4
0611 O QTYOH 172 ORDER4
0612 OTERMOUT D 40 LR ORDER4
0613 O 4 ' CB' ORDER4
0614 O 8 '42' ORDER4
0615 O PL,5 14 ORDER4
0616 O 20 '$ZORD5' ORDER4
0617 O 50 40 'ERROR RETURN CODE = ' ORDER4
0618 O 50 RTNCOD 42 ORDER4
0619 O 83 40 'END OF ORDER FILE ' ORDER4
0620 O 30 40 'ORDER FILE OVERLAYED' ORDER4
0621 O D N40 LR ORDER4
0622 O 4 ' CB' ORDER4
0623 O 8 '80' ORDER4
0624 O PL,5 14 ORDER4
0625 O 20 '$ZORD6' ORDER4
0626 O PRICE 2 35 ORDER4
0627 O TAX1 2 50 ORDER4
0628 O TAX2 2 65 ORDER4
0629 O PRT 2 80 ORDER4
**
D 34

```

Figure 8-25 (Part 2 of 2). ORDER4 Program



0101	H	R	4						1								ORDERP
0201	F	D	I	P	F	1	1		SPECIAL							SUBR93	ORDERP
0202	F	R	E	P	O		132	132	PRINTER								ORDERP
0203	F	H	O	L	D	O	R	D	DISK45								ORDERP
0301	E						64	64	TERM	7	6						ORDERP
0401	I	D	U	M	M	A		01									ORDERP
0402	I									1		1	DA				ORDERP
0403	I	H	O	L	D	O	R	D									ORDERP
0404	I									2			7ORRECT				ORDERP
0405	I									8		13	TERM,1				ORDERP
0406	I									14		19	TERM,2				ORDERP
0407	I									20		25	TERM,3				ORDERP
0408	I									26		31	TERM,4				ORDERP
0409	I									32		37	TERM,5				ORDERP
0410	I									38		43	TERM,6				ORDERP
0411	I					A	A	11	1								ORDERP
0412	I									2			7ORRECDD				ORDERP
0413	I									8			13ORRECDP				ORDERP
0414	I					A	A	12	1	C	N	2	C	1			ORDERP
0415	I									3			100ORDNUM				ORDERP
0416	I									11			160CUSNUM				ORDERP
0417	I									17			38 CUSNM				ORDERP
0418	I									39			60 CUSA1				ORDERP
0419	I					A	A	13	1	C	N	2	C	2			ORDERP
0420	I									3			24 CUSA2				ORDERP
0421	I									25			46 CUSA3				ORDERP
0422	I					A	A	14	1	C	N	2	C	3			ORDERP
0423	I									3			24 SHPNM				ORDERP
0424	I									25			46 SHPA1				ORDERP
0425	I					A	A	15	1	C	N	2	C	4			ORDERP
0426	I									3			24 SHPA2				ORDERP
0427	I									25			46 SHPA3				ORDERP
0428	I									47			51 ZIPCD				ORDERP
0429	I					A	A	16	1								ORDERP
0430	I									4			90QTY				ORDERP
0431	I									10			170ITEM				ORDERP
0432	I									18			39 ITMDS				ORDERP
0433	I									40			542PRICE				ORDERP
0434	I					A	A	17	1								ORDERP
0435	I									2			122PRT				ORDERP
0436	I									13			222TAX1				ORDERP
0437	I									23			322TAX2				ORDERP
0438	I									33			432COST				ORDERP
0439	I					A	A	18									ORDERP
0501	C																ORDERP
0502	C																ORDERP
0503	C																ORDERP
0504	C	N10															ORDERP
0505	C																ORDERP
0506	C																ORDERP
0507	C																ORDERP
0508	C																ORDERP
0509	C																ORDERP
0510	C																ORDERP
0511	C																ORDERP
0512	C																ORDERP
0513	C																ORDERP
0514	C																ORDERP
0515	C																ORDERP
0516	C	N23															ORDERP
0517	C	N23															ORDERP
0518	C																ORDERP
0519	C																ORDERP
0520	C																ORDERP
0521	C																ORDERP
0522	C																ORDERP
0523	C																ORDERP
0524	C	N15															ORDERP

Figure 8-26 (Part 1 of 2). ORDERP Program

```

0525 C          EXCPT          PRINT NAME REC ORDERP
0526 C          TAG          TAG          ORDERP
0527 C          EXSR READF    READ DETAIL REC ORDERP
0528 C          EXCPT          PRNT DETAIL REC ORDERP
0529 C N17      GOTO TAG3          ORDERP
0530 C          SETOF          17      ORDERP
0531 C          RRECDH    CHAINHOLDORD5  UPDATE HDR REC ORDERP
0532 C          EXCPT          ORDER PRINTED  ORDERP
0533 C          TAG5          TAG          ORDERP
0534 C          SETOF          1124     ORDERP
0535 C          RREC#      COMP RRECDD    24      ORDERP
0536 C N24      GOTO TAG2          ORDERP
0537 C          TAG6          TAG          ORDERP
0538 C          SETOF          25      ORDERP
0539 C          I          ADD 1          I          BUMP TERM INDEX ORDERP
0540 C          GOTO TAG4          ORDERP
0541 C          END          TAG          ORDERP
0542 C          SETON          LR          ORDERP
0543 CSR        READF          BEGSR          ORDERP
0544 CSR        SETOF          101112    ORDERP
0545 CSR        SETOF          131415    ORDERP
0546 CSR        SETOF          1617      ORDERP
0547 CSR        RREC#      CHAINHOLDORD5  ORDERP
0548 CSR        RREC#      ADD 1          RREC#      BUMP REL RECORD ORDERP
0549 CSR        ENDSR          ORDERP
0601 OREPORT   E 301 15          ORDERP
0602 O          ORDERP
0603 O          ORDNUM 61 'ORDER NUMBER'  ORDERP
0604 O          70      ORDERP
0605 O          UDATE Y 75 'DATE'         ORDERP
0606 O          84      ORDERP
0607 O          E 2 15          ORDERP
0608 O          38 'CUSTOMER'           ORDERP
0609 O          102 'SHIP TO'          ORDERP
0610 O          E 1 15          ORDERP
0611 O          CUSNM 44             ORDERP
0612 O          SHPNM 110            ORDERP
0613 O          E 1 15          ORDERP
0614 O          CUSA1 44             ORDERP
0615 O          SHPA1 110            ORDERP
0616 O          E 1 15          ORDERP
0617 O          CUSA2 44             ORDERP
0618 O          SHPA2 110            ORDERP
0619 O          E 2 15          ORDERP
0620 O          CUSA3 44             ORDERP
0621 O          SHPA3 110            ORDERP
0622 O          ZIPCD 116            ORDERP
0623 O          E 2 15          ORDERP
0624 O          20 'QUANTITY'           ORDERP
0625 O          51 'ITEM'              ORDERP
0626 O          88 'DESCRIPTION'        ORDERP
0627 O          117 'COST'             ORDERP
0628 O          E 1 16          ORDERP
0629 O          QTY 19              ORDERP
0630 O          ITEM 53             ORDERP
0631 O          ITMDSC 92            ORDERP
0632 O          PRICE 2 121          ORDERP
0633 O          E 11 17          ORDERP
0634 O          121 '-----'          ORDERP
0635 O          107 'TOTAL'            ORDERP
0636 O          PRT 2 121            ORDERP
0637 O          E 1 17          ORDERP
0638 O          107 'STATE TAX'        ORDERP
0639 O          TAX1 2 121            ORDERP
0640 O          E 1 17          ORDERP
0641 O          107 'FEDERAL TAX'      ORDERP
0642 O          TAX2 2 121            ORDERP
0643 O          E 1 17          ORDERP
0644 O          107 'TOTAL ORDER COST'  ORDERP
0645 O          COST 2 121           ORDERP
0646 OHOLDORD5E 11              ORDERP
0647 O          RREC# 13             ORDERP

```

Figure 8-26 (Part 2 of 2). ORDERP Program

ORDER NUMBER 0000016 DATE 2/16/75

CUSTOMER  
BROWN WHOLESALE SUPPLY  
SEVILLE  
PHILA., PENN

SHIP TO  
SLAY PROS INC  
WELLS  
PHILA., PENN  
TRUCK

QUANTITY	ITEM	DESCRIPTION	COST
000001	CCCCCCC1	BLUE FLUORESC	130.00
000002	CCCCCCC1	BLUE FLUORESC	260.00
			-----
			TOTAL 390.00
			STATE TAX 15.60
			FEDERAL TAX 15.50
			TOTAL ORDER COST 425.10

ORDER NUMBER 0000010 DATE 2/16/75

CUSTOMER  
BLACK&JONES CHEMICAL C  
19TH STREET  
PHILADELPHIA, PENNA.

SHIP TO  
SAME  
TRUCK

QUANTITY	ITEM	DESCRIPTION	COST
000002	CCCCCCC2	X02 PLUG	34.26
000003	CCCCCCC3	X03 PLUC	36.00
000004	CCCCCCC4	X04 PLUC	68.00
			-----
			TOTAL 138.26
			STATE TAX 5.53
			FEDERAL TAX 6.91
			TOTAL ORDER COST 150.70

ORDER NUMBER 0000005 DATE 2/16/75

CUSTOMER  
FAST ELECTRONICS  
2362 W GLENWOOD  
PITTSBURG, PENN

SHIP TO  
SAME  
AIR FREIGHT

QUANTITY	ITEM	DESCRIPTION	COST
000001	CCCCCCC2	X02 PLUG	17.13
000001	CCCCCCC1	BLUE FLUORESC	130.00
000004	CCCCCCC1	BLUE FLUORESC	520.00
			-----
			TOTAL 667.13
			STATE TAX 26.68
			FEDERAL TAX 33.35
			TOTAL ORDER COST 727.16

Figure 8-27 (Part 1 of 3). Sample Output of ORDERP Program

ORDER NUMBER 00000000 DATE 2/16/75

CUSTOMER  
JOSEPH CLARKE  
MAPLE STREET  
MAPLEWOOD, OHIO

SHIP TO  
SAM BLIZZARD  
SAME  
SAME  
FIRST CLASS MAIL

QUANTITY	ITEM	DESCRIPTION	COST
000004	CCCCCCC4	X04 PLUG	68.00
000005	CCCCCCC5	X05 PLUG	105.00
			-----
TOTAL			173.00
STATE TAX			6.92
FEDERAL TAX			8.65
TOTAL ORDER COST			188.57

ORDER NUMBER 00000007 DATE 2/16/75

CUSTOMER  
BENNETT INC  
CHESTNUT  
PHILA., PENN

SHIP TO  
SAME

QUANTITY	ITEM	DESCRIPTION	COST
000001	CCCCCCC1	BLUE FLUORESC	130.00
000002	CCCCCCC2	X02 PLUG	34.26
000003	CCCCCCC3	X03 PLUG	36.00
			-----
TOTAL			200.26
STATE TAX			8.01
FEDERAL TAX			10.01
TOTAL ORDER COST			218.28

ORDER NUMBER 00000008 DATE 2/16/75

CUSTOMER  
STEPHANIE BOOKS  
MAPLE STREET  
GREENVILLE, NEW YORK

SHIP TO  
JOHN BOOKS  
CLEMENTSON COLLEGE  
GETTYSBURG, PENNA.  
VIA BEST WAY

QUANTITY	ITEM	DESCRIPTION	COST
000001	CCCCCCC1	BLUE FLUORESC	130.00
000004	CCCCCCC4	X04 PLUG	68.00
			-----
TOTAL			198.00
STATE TAX			7.92
FEDERAL TAX			9.50
TOTAL ORDER COST			215.82

Figure 8-27 (Part 2 of 3). Sample Output of ORDERP Program

ORDER NUMBER 00000003 DATE 2/16/75

CUSTOMER

D & C SHIPPERS  
FILBERT  
PHILA., PENN

SHIP TO

SAME

AIR FREIGHT

CCCCC

QUANTITY	ITEM	DESCRIPTION	COST
000005	CCCCCCC5	X05 PLUG	105.00
000006	CCCCCCC6	X26 PLUG	108.00
			-----
TOTAL			213.00
STATE TAX			8.52
FEDERAL TAX			10.65
TOTAL ORDER COST			232.17

ORDER NUMBER 00000003 DATE 2/16/75

CUSTOMER

CARLSON BYRNES CO  
N BROAD  
PHILA., PENN

SHIP TO

W K GARRISON CO  
SANSOM  
PHILA., PENN  
TRUCK

CCCCC

QUANTITY	ITEM	DESCRIPTION	COST
000002	CCCCCCC3	X03 PLUG	24.00
000001	CCCCCCC2	X02 PLUG	17.13
			-----
TOTAL			41.13
STATE TAX			1.64
FEDERAL TAX			2.05
TOTAL ORDER COST			44.82

Figure 8-27 (Part 3 of 3). Sample Output of ORDERP Program

```

// DISKFILE NAME-HOLDORD5,ORG-D,RECL-64
// DISKFILE NAME-CUSTSHIP,ORG-D,RECL-128
// DISKFILE NAME-CUSTNAME,ORG-D,RECL-300
// DISKFILE NAME-INVENTORY,ORG-D,RECL-300
// PROGRAM NAME-ORDER1,ENDMSG=YES,PGMDATA-NO,MFCUI-R,PRINTER-SHR,
// FILES-HOLDORD5/DDE
// PROGRAM NAME-ORDER1,PGMDATA=YES,ENDMSG-NO,
// DFFMTERM-1,DFFNDF-1,DFFSFDT-182,
// FILES-DCUSTNAME/DG/SHR,CUSTSHIP/DG/SHR
// PROGRAM NAME-ORDER2,PGMDATA=YES,ENDMSG-NO,
// DFFMTERM-1,DFFNDF-1,DFFSFDT-312,
// PRUFLNG-243,PRUF$Z-$ZORD1,
// FILES-HOLDORD5/DU/SHR
// PROGRAM NAME-ORDER3,PGMDATA=YES,ENDMSG-NO,
// MRTMAX-4,DFMTERM-4,DFFNDF-2,DFFSFDT-312,
// PRUFLNG-123,PRUF$Z-$ZORD2,
// FILES-HOLDORD5/DU/SHR,INVENTORY/DG/SHR
// PROGRAM NAME-ORDER4,PGMDATA=YES,ENDMSG=YES,
// DFFMTERM-1,DFFNDF-1,DFFSFDT-70,
// PRUFLNG-30,PRUF$Z-$ZORD3,
// FILES-HOLDORD5/DU/SHR,INVENTORY/DU/SHR
// PROGRAM NAME-ORDERP,PGMDATA-NO,ENDMSG=YES,PRINTER=YES,
// FILES-HOLDORD5/DU/SHR

```

Figure 8-28. Assignment Set Considerations

**\$ZORD1**

```

C$ZORD1SHPNM 2Y X R1
FFGMNAM0102007 7CY ORDER2
FCRCDRHO3010121F ORDER NUMBER
FCRDNJHO316008 8EY
FCUSTH 05010151F CUSTOMER NUMBER
FCUSNO 0518006 8FY
FCUSNMHO7010131F CUSTOMER NAME
FSHPNMHO7360071F SHIP TO
FCUSNM 0901022 8EY
FSHPNM 0936022 1EY
FCUSA1 1001022 8FY
FSHPA1 1036022 1EY
FCUSA2 1101022 8EY
FSHPA2 1136022 1EY
FCUSA3 1201022 9FY
FSHPA3 1236022 1EY
FZIPCO 1260005 3EY

```

LENGTH OF OUTPUT RECORD AREA REQUIRED IN DFF PROGRAM  
RPG \*SUBR92\* - 0215 OTHER - 0207

LENGTH OF INPUT RECORD AREA REQUIRED IN DFF PROGRAM  
RPG \*SUBR92\* - 0217 OTHER - 0209

These lengths must be  
reduced by 6 for  
COBOL programs

-----  
INFORMATION FOR USE DURING CCP ASSIGNMENT STAGE  
-----

THE DECIMAL LENGTH OF THE FIELD DESCRIPTOR TABLE IS 0182  
THE DECIMAL LENGTH OF THE OUTPUT TEXT IS 0399  
THE DECIMAL LENGTH OF THE INPUT TEXT IS 0243

Figure 8-29 (Part 1 of 3). Formats

\$ZORD2

```

C#ZORD2QTY1 2N X R1
FPGMNAM0102007 7GY ORDER3
FRRECCO0111012 7EY
FCRDMJM03160081E
FCUSNC 05180061E
FCUSNM 09010221E
FSHPNM 09360221E
FCUSA1 10010221E
FSHPA1 10360221E
FCUSA2 11010221E
FSHPA2 11360221E
FCUSA3 12010221E
FSHPA3 12360221E
FZIPCD 12600051E
FLINEH 15010081F LINE NO.
FCTYH 15140081F QUANTITY
FITMH 15240081F ITEM NO.
FDESCH 15380111F DESCRIPTION
FPRICEH15660051F PRICE
FLIN1 1604002 3GY C1
FCTY1 1614006 3GY
FITM1 1624008 3GY
FLIN2 1704002 7EY
FCTY2 1714006 7EY
FITM2 1724008 7EY
FDESC 1734022 7EY
FPRICE 1759015 7EY

```

-----  
INFORMATION FOR USE DURING CCP ASSIGNMENT STAGE  
-----

THE DECIMAL LENGTH OF THE FIELD DESCRIPTOR TABLE IS C312  
THE DECIMAL LENGTH OF THE OUTPUT TEXT IS C513  
THE DECIMAL LENGTH OF THE INPUT TEXT IS C123

\$ZORD3

```

C#ZORD3 2Y X R1
FPGMNAM0102007 7GY ORDER4
FRRECCO0111012 7EY
FCRDERM03010291F PRESS ENTER TO COMPLETE ORDER

```

-----  
INFORMATION FOR USE DURING CCP ASSIGNMENT STAGE  
-----

THE DECIMAL LENGTH OF THE FIELD DESCRIPTOR TABLE IS CC56  
THE DECIMAL LENGTH OF THE OUTPUT TEXT IS CC79  
THE DECIMAL LENGTH OF THE INPUT TEXT IS CC3C

\$ZORD4

```

C#ZORD4QTY1 2N X R1
FPGMNAM0102007 7GY ORDER3
FRRECCO0111012 7EY
FLIN1 1604002 3EY
FCTY1 1614006 3EY
FITM1 1624008 3EY
FMESG 16340221F
FLIN2 1704002 8EY
FCTY2 1714006 8EY
FITM2 1724008 8EY
FDESC 1734022 8EY
FPRICE 1759015 8EY

```

-----  
INFORMATION FOR USE DURING CCP ASSIGNMENT STAGE  
-----

THE DECIMAL LENGTH OF THE FIELD DESCRIPTOR TABLE IS 0168  
THE DECIMAL LENGTH OF THE OUTPUT TEXT IS 0225  
THE DECIMAL LENGTH OF THE INPUT TEXT IS C123

Figure 8-29 (Part 2 of 3). Formats

**\$ZORD5**

C1ZORC5        2Y X R1  
FERRMS01020201E  
FERRMS201230021F  
FERRMS301270151F

ORDER CANCELED

-----  
INFORMATION FOR USE DURING CCP ASSIGNMENT STAGE  
-----

THE DECIMAL LENGTH OF THE FIELD DESCRIPTOR TABLE IS 0056  
THE DECIMAL LENGTH OF THE OUTPUT TEXT IS 0058  
THE DECIMAL LENGTH OF THE INPUT TEXT IS 0005

**\$ZORD6**

C1ZORC6        2Y X R1  
FHEAD1 03010181F  
FPRICH 05010061F  
FPRICE 05170151F  
FHEAD2 06010131F  
FTAX1 06170151F  
FHEAD3 07010131F  
FTAX2 07170151F  
FHEAD4 08170151F  
FTOTALH10010131F  
FTOTAL 10170152F

ORDER IS COMPLETED  
AMOUNT

STATE TAX

FEDERAL TAX

-----  
TOTAL

-----  
INFORMATION FOR USE DURING CCP ASSIGNMENT STAGE  
-----

THE DECIMAL LENGTH OF THE FIELD DESCRIPTOR TABLE IS 0070  
THE DECIMAL LENGTH OF THE OUTPUT TEXT IS 0164  
THE DECIMAL LENGTH OF THE INPUT TEXT IS 0005

**Figure 8-29 (Part 3 of 3). Formats**

**Example 5 – COBOL Order Entry Program (using PRUF with the DFF)**

This example shows the use of the PRUF capability with COBOL. The example uses the same order entry programs used in example 4. The RPG II programs are converted to COBOL, but the logic, where possible, and the labels are the same. The screen formats are the same, except that the item description field is increased to 24 bytes to accommodate the entire field from the inventory record.

*COBOL Considerations*

Because COBOL uses a parameter list that is defined in the data division to accomplish CCP program interface, the arrays in RPG II are not needed. The parameter list can be defined for the first operation (usually an accept) using the correct operation code and length. All that is required, then, is a call to CCPCIO as the first operation.

The display format generator gives the output and input record lengths required in DFF programs. One set of lengths is for the RPG II programs, the other for non-RPG programs. These lengths must be reduced by 6 (the length of the terminal name) for COBOL programs. See Figure 8-29 for an example.



STNO -A...B... C O B O L S O U R C E S T A T E M E N T S .....IDENTFCN SEQ/NO S

```

PROCESS MAP,LINK P,R1
1 IDENTIFICATION DIVISION.
2 PROGRAM-ID. ORDERL.
3 REMARKS. THIS PROGRAM LOADS THE HOLD ORDER FILE.
      INPUT IS THE A CARD, WHICH DEFINES THE SPACE FOR EACH
      TERMINAL ORDERS, AND THE H CARDS, ONE FOR EACH TERMINAL
      ALLOCATED ORDER SPACE IN THE HOLD ORDER FILE.
4 ENVIRONMENT DIVISION.
5 CONFIGURATION SECTION.
6 SOURCE-COMPUTER. IBM-S3.
7 OBJECT-COMPUTER. IBM-S3.
8 INPUT-OUTPUT SECTION.
9 FILE-CONTROL.
10 SELECT CARDIN ASSIGN TO UR-2501-RD.
11 SELECT HOLDORDS ASSIGN TO DA-5445-R-HOLDORDS
      ACCESS IS RANDOM
      ACTUAL KEY IS RREC.
12 SELECT PRINTOUT ASSIGN TO UR-1403-3-68.
13 DATA DIVISION.
14 FILE SECTION.
15 FD CARDIN
      LABEL RECORDS ARE OMITTED
      DATA RECORD IS CARD.
16 01 CARD.
17 02 CODES PIC X.
18 88 AREC VALUE 0A0.
19 88 HREC VALUE 0H0.
20 02 RRECT PIC X 6.
21 88 NO-RRECT VALUE SPACES.
22 02 TRMNM REDEFINES RRECT PIC X 6.
23 02 N PIC 99.
24 02 FILLER PIC X 71.
25 FD HOLDORDS
      LABEL RECORDS ARE STANDARD
      DATA RECORD IS HOLDORD.
26 01 HOLDORD.
27 02 CODE PIC X.
28 02 RRECD.
29 03 RRECD PIC 9 6.
30 03 RRECD PIC 9 6.
31 03 TERMI PIC X 6.
32 03 FILLER PIC X 45.
33 02 TERMS REDEFINES RRECD.
34 03 RRECT PIC 9 6.
35 03 TERM-TBL PIC X 36.
36 03 FILLER PIC X 21.
37 FD PRINTOUT
      LABEL RECORDS ARE OMITTED
      DATA RECORD IS PRINT.
38 01 PRINT PIC X 132.
39 WORKING-STORAGE SECTION.
40 77 RREC PIC S9 7 VALUE 0.
41 77 RRECTS PIC 9 6 VALUE 0.
42 77 RRECDH PIC 9 6 VALUE 2.
43 77 ERRI PIC X 24 VALUE 0RECORDS/TERMINAL INVALID0.
    
```

```

44 77 FRR2          PIC X 24 VALUE #INVALID CARD TYPE#.
45 77 ERR3         PIC X 24 VALUE #END OF ORDER FILE#.
46 77 ERR4         PIC X 24 VALUE #MORE THAN 6 TERMINALS#.
47 77 IND          PIC X VALUE # #.
48 88 AOK VALUE #A#.
49 01 TERM-TABLE.
50 02 TERM          PIC X 6 OCCURS 6 INDEXED BY 1.
51 01 HEADER.
52 02 FILLER        PIC X 13 VALUE #RECORD NUMBER#.
53 02 FILLER        PIC X 4 VALUE SPACES.
54 02 FILLER        PIC X 13 VALUE #TERMINAL NAME#.
55 02 FILLER        PIC X 102 VALUE SPACES.
56 01 DETAIL.
57 02 FILLER        PIC X 4 VALUE SPACES.
58 02 RRECIN        PIC 9 6 .
59 02 FILLER        PIC X 9 VALUE SPACES.
60 02 TERM1         PIC X 6 .
61 02 FILLER        PIC X 107 VALUE SPACES.
62 01 ERRORS.
63 02 CARDOUT       PIC X 80 .
64 02 FILLER        PIC X 22 VALUE SPACES.
65 02 ERRMSG        PIC X 24 .
66 02 FILLER        PIC X 6 VALUE SPACES.
67 PROCEDURE DIVISION.
68 BEGIN.
69 OPEN INPUT CARDIN, OUTPUT HOLDORD5, PRINTOUT.
70 WRITE PRINT FROM HEADER BEFORE 2.
71 READER.
72 READ CARDIN, AT END GO TO EOF.
74 IF AREC GO TO A-REC.
76 IF HREC GO TO H-REC.
78 MOVE ERR2 TO ERRMSG.
79 ERROR-RTN.
80 MOVE CARD TO CARDOUT.
81 WRITE PRINT FROM ERRORS BEFORE 3.
82 EOF.
83 CLOSE CARDIN, HOLDORD5, PRINTOUT.
84 STOP RUN.
*****
* CHECK FOR VALID CARD TYPE AND ALLOCATION UNITS. *
*****
85 A-REC.
86 IF NOT RRECT,
87 MOVE ERR1 TO ERRMSG, GO TO ERROR-RTN.
88 MOVE RRECT IN CARD TO RRECTS.
89 MOVE #A# TO IND.
90 SET I TO 1.
91 GO TO READER.
*****
* MAKE SURE THE H CARDS ARE PRECEDED BY AN A CARD. *
*****
93 H-REC.
94 IF NOT AOK, GO TO ERROR-RTN.
96 MOVE RRECDH TO RREC.
97 ADD 1, RRECDH GIVING RRECDD,
98 MOVE RRECDD TO RRECDP.
*****
* MAKE SURE NO MORE THAN 6 TERMINALS ARE LISTED. *

```

```

*****
99  LOOP.
100  IF I EQUAL 7 MOVE ERR4 TO ERRMSG, GO TO ERROR-RTN.
103  MOVE TRMNM TO TERM I .
104  ADD RRECTS TO RRECDH.
105  IF N NOT EQUAL 0 SUBTRACT 1 FROM N, ELSE GO TO LOOP1.
109  IF N IS POSITIVE SET I UP BY 1, GO TO LOOP.
*****
*   WRITE THE H RECORD GIVING THE HEADER INFORMATION FOR THIS   *
*   TERMINAL AND READ THE NEXT H CARD.                           *
*****
112  LOOP1.
113  MOVE RREC TO RREC10.
114  MOVE TERM I TO TERMI IN RRECD, TERMI IN DETAIL.
115  WRITE PRINT FROM DETAIL BEFORE 1.
116  MOVE □□□ TO CODE.
117  WRITE HOLDORD, INVALID KEY MOVE ERR3 TO ERRMSG,
119  GO TO ERROR-RTN.
120  SET I UP BY 1.
121  GO TO READER.
*****
*   WRITE OUT THE COMPLETED A RECORD GIVING THE ALLOCATION     *
*   UNITS AND TERMINAL ID□S.                                     *
*****
122  EOF.
123  MOVE 1 TO RREC.
124  MOVE □□□ TO CODE.
125  MOVE RRECTS TO RRECT IN TERMS.
126  MOVE TERM-TABLE TO TERM-TBL.
127  WRITE HOLDORD, INVALID KEY GO TO EOF.
129  GO TO EOF.

```

STNO -A...B... COBOL SOURCE STATEMENTS .....IDENTFCN SEQ/NO S

```

PROCESS MAP,LIST,LINK P,R1
1 IDENTIFICATION DIVISION.
2 PROGRAM-ID. ORDERP.
3 REMARKS. THIS PROGRAM PRINTS THE ORDERS WHICH WERE PLACED
  IN THE HOLD ORDER FILE BY ORDER3 AND ORDER4.
4 ENVIRONMENT DIVISION.
5 CONFIGURATION SECTION.
6 SOURCE-COMPUTER. IBM-S3.
7 OBJECT-COMPUTER. IBM-S3.
8 INPUT-OUTPUT SECTION.
9 FILE-CONTROL.
10 SELECT HOLDORD5 ASSIGN TO DA-5445-R-HOLDORD5-U
    ACCESS IS RANDOM
    ACTUAL KEY IS RREC.
11 SELECT REPORT ASSIGN TO UR-1403-3-68.
12 DATA DIVISION.
13 FILE SECTION.
14 FD HOLDORD5
    LABEL RECORDS ARE STANDARD
    DATA RECORDS ARE HOLDORD, HOLDORD-OUT.
15 01 HOLDORD.
16     02 CODE1.
17         03 CODE PIC X.
18             88 AREC VALUE 0A0.
19             88 HREC VALUE 0H0.
20             88 DREC VALUE 0D0.
21             88 TREC VALUE 0T0.
22         03 FILLER PIC X.
23     02 CODE2 REDEFINES CODE1.
24         03 CODES PIC XX.
25             88 N1REC VALUE 0N10.
26             88 N2REC VALUE 0N20.
27             88 N3REC VALUE 0N30.
28             88 N4REC VALUE 0N40.
29     02 FILLER PIC X 62.
30 01 HOLDORD-OUT.
31     02 FILLER PIC X 7.
32     02 HRREC PIC 9 6.
33     02 FILLER PIC X 51.
34 FD REPORT
    LABEL RECORDS ARE OMITTED
    DATA RECORD IS REPORT-OUT.
35 01 REPORT-OUT.
36     02 FILLER PIC X 132.
37 WORKING-STORAGE SECTION.
38 77 RREC PIC S9 7 VALUE 0.
39 77 RREC PIC S9 7 VALUE 0.
40 77 TRMM PIC X 6 VALUE SPACES.
41 77 RRECDM PIC 9 6.
42 77 N PIC 9.
43 01 DATE-AREA.
44     02 YR PIC 99.
45     02 MO PIC 99.
46     02 DA PIC 99.
47 01 HEADER-1.
    
```

```

48      02  FILLER          PIC X 49  VALUE SPACES.
49      02  FILLER          PIC X 13  VALUE ORDER NUMBER .
50      02  ORDNUM          PIC 9 8 .
51      02  FILLER          PIC X 6   VALUE DATE .
52      02  UDATE.
53          03  MO          PIC ZZ.
54          03  FILLER      PIC X VALUE /.
55          03  DA          PIC 99.
56          03  FILLER      PIC X VALUE /.
57          03  YR          PIC 99.
58      02  FILLER          PIC X 48  VALUE SPACES.
59  01  HEADER-2.
60      02  FILLER          PIC X 30  VALUE SPACES.
61      02  FILLER          PIC X 8   VALUE CUSTOMER.
62      02  FILLER          PIC X 57  VALUE SPACES.
63      02  FILLER          PIC X 7   VALUE SHIP TO.
64      02  FILLER          PIC X 30  VALUE SPACES.
65  01  NAME-ADDR.
66      02  FILLER          PIC X 22  VALUE SPACES.
67      02  CUSNM          PIC X 22  VALUE SPACES.
68      02  CUSA1          REDEFINES CUSNM PIC X 22 .
69      02  CUSA2          REDEFINES CUSA1 PIC X 22 .
70      02  CUSA3          REDEFINES CUSA2 PIC X 22 .
71      02  FILLER          PIC X 44  VALUE SPACES.
72      02  SHPNM          PIC X 22 .
73      02  SHPA1          REDEFINES SHPNM PIC X 22 .
74      02  SHPA2          REDEFINES SHPA1 PIC X 22 .
75      02  SHPA3          REDEFINES SHPA2 PIC X 22 .
76      02  FILLER          PIC X VALUE SPACES.
77      02  7IPCD          PIC X 5   VALUE SPACES.
78      02  FILLER          PIC X 16  VALUE SPACES.
79  01  HEADER-3.
80      02  FILLER          PIC X 12  VALUE SPACES.
81      02  FILLER          PIC X 8   VALUE QUANTITY.
82      02  FILLER          PIC X 27  VALUE SPACES.
83      02  FILLER          PIC X 4   VALUE ITEM.
84      02  FILLER          PIC X 26  VALUE SPACES.
85      02  FILLER          PIC X 11  VALUE DESCRIPTION.
86      02  FILLER          PIC X 25  VALUE SPACES.
87      02  FILLER          PIC X 4   VALUE COST.
88      02  FILLER          PIC X 15  VALUE SPACES.
89  01  DETAILS.
90      02  FILLER          PIC X 13  VALUE SPACES.
91      02  QTY            PIC 9 6 .
92      02  FILLER          PIC X 26  VALUE SPACES.
93      02  ITEM          PIC 9 8 .
94      02  FILLER          PIC X 17  VALUE SPACES.
95      02  ITMDS          PIC X 24 .
96      02  FILLER          PIC X 13  VALUE SPACES.
97      02  PRICE          PIC Z77,ZZ7,ZZZ.ZZ.
98      02  FILLER          PIC X 11  VALUE SPACES.
99  01  HEADER-4.
100     02  FILLER          PIC X 110  VALUE SPACES.
101     02  FILLER          PIC X 11  VALUE -----.
102     02  FILLER          PIC X 11  VALUE SPACES.
103  01  TOTAL-1.
104     02  FILLER          PIC X 102  VALUE SPACES.
105     02  FILLER          PIC X 9   VALUE TOTAL .

```

106	02	PRT	PIC Z77,ZZZ.ZZ.
107	02	FILLER	PIC X 11 VALUE SPACES.
108	01	TOTAL-2.	
109	02	FILLER	PIC X 98 VALUE SPACES.
110	02	FILLER	PIC X 14 VALUE STATE TAX    □.
111	02	TAX1	PIC Z7,ZZZ.ZZ.
112	02	FILLER	PIC X 11 VALUE SPACES.
113	01	TOTAL-3.	
114	02	FILLER	PIC X 96 VALUE SPACES.
115	02	FILLER	PIC X 16 VALUE FEDERAL TAX    □.
116	02	TAX2	PIC Z7,ZZZ.ZZ.
117	02	FILLER	PIC X 11 VALUE SPACES.
118	01	TOTAL-4.	
119	02	FILLER	PIC X 91 VALUE SPACES.
120	02	FILLER	PIC X 20 VALUE TOTAL ORDER COST    □.
121	02	CONST	PIC Z77,ZZZ.ZZ.
122	02	FILLER	PIC X 11 VALUE SPACES.
123	01	HOLDORDA.	
124	02	FILLER	PIC X.
125	02	RECT	PIC 9 6 .
126	02	TERM	PIC X 6 OCCURS 6 INDEXED BY I.
127	02	FILLER	PIC X 21 .
128	01	HOLDORDH.	
129	02	FILLER	PIC X.
130	02	RECDD	PIC 9 6 .
131	02	RECDP	PIC 9 6 .
132	02	FILLER	PIC X 51 .
133	01	HOLDORDN1.	
134	02	FILLER	PIC XX.
135	02	ORDNUM	PIC 9 8 .
136	02	FILLER	PIC X 6 .
137	02	CUSNM	PIC X 22 .
138	02	CUSA1	PIC X 22 .
139	02	FILLER	PIC X 4 .
140	01	HOLDORDN2.	
141	02	FILLER	PIC XX.
142	02	CUSA2	PIC X 22 .
143	02	CUSA3	PIC X 22 .
144	02	FILLER	PIC X 18 .
145	01	HOLDORDN3.	
146	02	FILLER	PIC XX.
147	02	SHPNM	PIC X 22 .
148	02	SHPA1	PIC X 22 .
149	02	FILLER	PIC X 18 .
150	01	HOLDORDN4.	
151	02	FILLER	PIC XX.
152	02	SHPA2	PIC X 22 .
153	02	SHPA3	PIC X 22 .
154	02	ZIPCD	PIC 9 5 .
155	02	FILLER	PIC X 13 .
156	01	HOLDORDD.	
157	02	FILLER	PIC X 3 .
158	02	QTY	PIC 9 6 .
159	02	ITEM	PIC 9 8 .
160	02	ITMDS	PIC X 24 .
161	02	PRICE	PIC 9 13 V99.
162	02	FILLER	PIC X 8 .
163	01	HOLDORDT.	

```

164      02 FILLER      PIC X.
165      02 PRT        PIC 9 9 V99.
166      02 TAX1       PIC 9 8 V99.
167      02 TAX2       PIC 9 8 V99.
168      02 COST       PIC 9 9 V99.
169      02 FILLER     PIC X 21 .
170  PRCCEDURE DIVISION.
171  BEGIN.
172      OPEN I-O HOLDORD5, OUTPUT REPORT.
173      ACCEPT DATE-AREA FROM DATE.
174      MOVE MO IN DATE-AREA TO MO IN UDATE.
175      MOVE DA IN DATE-AREA TO DA IN UDATE.
176      MOVE YR IN DATE-AREA TO YR IN UDATE.
177      MOVE 1 TO RRECP.
178      PERFORM READF THRU RETURN.
179      IF NOT AREC GO TO EOJ.
181      SET I TO 1.
*****
*      FIND THE RECORDS TO BE PRINTED FOR THIS TERMINAL.      *
*****
182  TAG4.
183      IF TERM I EQUAL SPACES, GO TO EOJ.
185      IF TERM I EQUAL TRNMN, GO TO TAG6.
187      MOVE TERM I TO TRNMN.
188      MOVE 2 TO RRECP.
189      SET N TO I.
190  TAG1.
191      SUBTRACT 1 FROM N.
192      IF N NOT EQUAL TO 0
193          ADD RRECT TO RRECP, GO TO TAG1.
195      MOVE RRECP TO RRECDH.
196      PERFORM READF THRU RETURN.
197      IF NOT HREC, GO TO EOJ.
199      MOVE RRECDP TO RRECP, GO TO TAG5.
*****
*      PRINT NAME/ADDRESS RECORDS.      *
*****
201  TAG2.
202      PERFORM READF THRU RETURN.
203      IF NOT N4REC GO TO TAG2.
205      MOVE ORDNUM IN HOLDORDN1 TO ORDNUM IN HEADER-1.
206      MOVE CUSNM IN HOLDORDN1 TO CUSNM IN NAME-ADDR.
207      MOVE SHPNM IN HOLDORDN3 TO SHPNM IN NAME-ADDR.
208      WRITE REPORT-OUT FROM HEADER-1 BEFORE 3.
209      WRITE REPORT-OUT FROM HEADER-2 BEFORE 2.
210      WRITE REPORT-OUT FROM NAME-ADDR BEFORE 1.
211      MOVE CUSA1 IN HOLDORDN1 TO CUSA1 IN NAME-ADDR.
212      MOVE SHPA1 IN HOLDORDN3 TO SHPA1 IN NAME-ADDR.
213      WRITE REPORT-OUT FROM NAME-ADDR BEFORE 1.
214      MOVE CUSA2 IN HOLDORDN2 TO CUSA2 IN NAME-ADDR.
215      MOVE SHPA2 IN HOLDORDN4 TO SHPA2 IN NAME-ADDR.
216      WRITE REPORT-OUT FROM NAME-ADDR BEFORE 1.
217      MOVE CUSA3 IN HOLDORDN2 TO CUSA3 IN NAME-ADDR.
218      MOVE SHPA3 IN HOLDORDN4 TO SHPA3 IN NAME-ADDR.
219      MOVE ZIPCD IN HOLDORDN4 TO ZIPCD IN NAME-ADDR.
220      WRITE REPORT-OUT FROM NAME-ADDR BEFORE 2.
221      WRITE REPORT-OUT FROM HEADER-3 BEFORE 1.
*****

```

```

* PRINT DETAIL RECORDS. *
*****
222 TAG3.
223 PERFORM READF THRU RETURN.
224 IF DREC
225 MOVE QTY IN HOLDORDD TO QTY IN DETAILS,
226 MOVE ITEM IN HOLDORDD TO ITEM IN DETAILS,
227 MOVE ITMOSC IN HOLDORDD TO ITMOSC IN DETAILS,
228 MOVE PRICE IN HOLDORDD TO PRICE IN DETAILS,
229 WRITE REPORT-OUT FROM DETAILS AFTER 1, GO TO TAG3.
*****
* PRINT TOTAL RECORD. *
*****
231 MOVE PRT IN HOLDORDT TO PRT IN TOTAL-1.
232 MOVE TAX1 IN HOLDORDT TO TAX1 IN TOTAL-2.
233 MOVE TAX2 IN HOLDORDT TO TAX2 IN TOTAL-3.
234 MOVE COST IN HOLDORDT TO COST IN TOTAL-4.
235 WRITE REPORT-OUT FROM HEADER-4 AFTER 2.
236 WRITE REPORT-OUT FROM TOTAL-1 AFTER 1.
237 WRITE REPORT-OUT FROM TOTAL-2 AFTER 1.
238 WRITE REPORT-OUT FROM TOTAL-3 AFTER 1.
239 WRITE REPORT-OUT FROM TOTAL-4 AFTER 1.
240 MOVE RRECDH TO RREC.
241 READ HOLDORD5, INVALID KEY GO TO EOJ.
242 MOVE RRECP TO HRRECP.
243 REWRITE HOLDORD-OUT, INVALID KEY GO TO EOJ.
244 TAG5.
245 IF RRECP NOT EQUAL RRECDD GO TO TAG2.
*****
* CHECK FOR NEXT ORDER. *
*****
249 TAG6.
250 SFT I UP BY 1.
251 GO TO TAG4.
252 EOJ.
253 CLOSE HOLDORD5, REPORT.
254 STOP RUN.
*****
* READ HOLDORDER FILE AND INCREMENT THE KEY. *
*****
255 READF.
256 MOVE RRECP TO RREC.
257 READ HOLDORD5, INVALID KEY GO TO EOJ.
259 ADD 1 TO RRECP.
260 IF DREC MOVE HOLDORD TO HOLDORDD, GO TO RETURN.
263 IF HREC MOVE HOLDORD TO HOLDORDH, GO TO RETURN.
266 IF TREC MOVE HOLDORD TO HOLDORDT, GO TO RETURN.
269 IF N1REC MOVE HOLDORD TO HOLDORDN1, GO TO RETURN.
272 IF N2REC MOVE HOLDORD TO HOLDORDN2, GO TO RETURN.
275 IF N3REC MOVE HOLDORD TO HOLDORDN3, GO TO RETURN.
278 IF N4REC MOVE HOLDORD TO HOLDORDN4, GO TO RETURN.
281 IF AREC MOVE HOLDORD TO HOLDORDA, GO TO RETURN.
284 RETURN.
285 EXIT.

```



STNO -A...B... COBOL SOURCE STATEMENTS .....IDENTFCN SEQ/NO S

```

PROCESS MAP,LIST,LINK P,R1
1 IDENTIFICATION DIVISION.
2 PROGRAM-IC. ORDER1.
3 REMARKS. THIS PROGRAM USES THE CUSTOMER NUMBER AND ORDER NUMBER
  ENTERED WITH THE PROGRAM REQUEST TO READ THE CUSTOMER
  NAME AND CUSTOMER SHIP-TO FILES. IF THE RECORDS ARE FOUND
  A PRUF SCREEN IS BUILT AND SENT TO THE TERMINAL OPERATOR
  TO BE VERIFIED. THE PRUF SCREEN IS SET UP TO CALL
  PROGRAM ORDER2. INPUT IS CUSTNO/ORDNO. 6 AND 8 DIGITS.
4 ENVIRONMENT DIVISION.
5 CONFIGURATION SECTION.
6 SOURCE-COMPUTER. IBM-S3.
7 OBJECT-COMPUTER. IBM-S3.
8 INPUT-OUTPUT SECTION.
9 FILE-CONTROL.
10 SELECT CUSTNAME ASSIGN TO DA-5444-R-CUSTNAME
  ACCESS IS RANDOM
  ACTUAL KEY IS CUSNO.
11 SELECT CUSTSHIP ASSIGN TO DA-5444-R-CUSTSHIP
  ACCESS IS RANDOM
  ACTUAL KEY IS CUSNO.
12 DATA DIVISION.
13 FILE SECTION.
14 FD CUSTNAME
  LABEL RECORDS ARE STANDARD
  DATA RECORD IS CUST-NAME.
15 01 CUST-NAME.
16 02 CODE-1 PIC XX.
17 88 VALID-CUST VALUE MA.
18 02 FILLER PIC X 6 .
19 02 CUSNM PIC X 22 .
20 02 FILLER PIC X 3 .
21 02 CUSA1 PIC X 22 .
22 02 FILLER PIC X 3 .
23 02 CUSA2 PIC X 22 .
24 02 FILLER PIC X 3 .
25 02 CUSA3 PIC X 22 .
26 02 FILLER PIC X 195 .
27 FD CUSTSHIP
  LABEL RECORDS ARE STANDARD
  DATA RECORD IS CUST-SHIP.
28 01 CUST-SHIP.
29 02 CODE-2 PIC XX.
30 88 VALID-SHIP VALUE MB.
31 02 FILLER PIC X 6 .
32 02 SHPNM PIC X 22 .
33 02 FILLER PIC X 3 .
34 02 SHPA1 PIC X 22 .
35 02 FILLER PIC X 3 .
36 02 SHPA2 PIC X 22 .
37 02 FILLER PIC X 3 .
38 02 SHPA3 PIC X 22 .
39 02 FILLER PIC X 3 .
40 02 ZIPCD PIC 9 5 .
41 02 FILLER PIC X 15 .

```

```

42 WORKING-STORAGE SECTION.
43 77 CUSNO          PIC S9 7 VALUE 0.
44 77 PUT            PIC 9 4 VALUE 50 COMP-4.
45 77 PRUF-PUT      PIC 9 4 VALUE 114 COMP-4.
46 77 ERR1          PIC X 20 VALUE 0INVALID CUSSHIP FILE0.
47 77 ERR2          PIC X 20 VALUE 0INVALID CUSTMAS FILE0.
48 77 ERR3          PIC X 20 VALUE 0NO CUSSHIP RECORD0.
49 77 ERR4          PIC X 20 VALUE 0NO CUSTMAS RECORD0.
50 77 ERR5          PIC X 20 VALUE 0INVALID ORDER NUMBER0.
51 77 ERR6          PIC X 20 VALUE 0INVALID CUST. NUMBER0.
52 77 ERR7          PIC X 20 VALUE 0ERROR RETURN CODE0.
53 01 RETNCDS.
54 02 FILLER        PIC 99.
55 02 RETN-CD       PIC 99.
*****
* CCP-COBOL INTERFACE PARAMETER LIST. *
*****
56 01 PARAMETER-LIST.
57 02 RETURN-CD     PIC S9 4 COMP-4.
58 88 GOOD-RTN     VALUE 0.
59 02 OPN-CD        PIC S9 4 COMP-4 VALUE 4.
60 02 OUT-LEN       PIC S9 4 COMP-4.
61 02 EFF-LEN       REDEFINES OUT-LEN PIC S9 4 COMP-4.
62 02 ATTRIB        REDEFINES EFF-LEN PIC S9 4 COMP-4.
63 02 IN-LEN        PIC S9 4 COMP-4 VALUE 15.
64 02 FILLER        PIC X 8 .
*****
* TERMINAL INPUT RECORD AREA. *
*****
65 01 IN-REC.
66 02 TERM-IN       PIC X 6 .
67 02 CUSNOIN       PIC X 6 .
68 88 NO-CUST       VALUE SPACES.
69 02 SLASH          PIC X.
70 88 OK            VALUE 0/0.
71 02 ORDERN        PIC X 8 .
72 88 NO-ORD        VALUE SPACES.
*****
* TERMINAL OUTPUT RECORD AREA. *
*****
73 01 ERROR-REC.
74 02 TERM-OUT      PIC X 6 .
75 02 FILLER        PIC X 6 VALUE 0$ZORD50.
76 02 ERRMSG        PIC X 20 .
77 02 RTNCD         PIC X 2 VALUE SPACES.
78 01 OUT-REC.
79 02 TERM-OUT      PIC X 6 .
80 02 FILLER        PIC X 6 VALUE 0$ZORD10.
81 02 ORDERN        PIC 9 8 .
82 02 CUSNOOT       PIC 9 6 .
83 02 CUSNM         PIC X 22 .
84 02 SHPNM         PIC X 22 .
85 02 CUSA1         PIC X 22 .
86 02 SHPA1         PIC X 22 .
87 02 CUSA2         PIC X 22 .
88 02 SHPA2         PIC X 22 .
89 02 CUSA3         PIC X 22 .
90 02 SHPA3         PIC X 22 .

```

```

91     02 ZIPCD      PIC 9 5 .
92     PROCEDURE DIVISION.
93     BEGIN.
94     OPEN INPUT CUSTNAME, CUSTSHIP.
*****
*     ACCEPT INPUT. THE CUSTOMER NUMBER AND ORDER NUMBER
*     ACCOMPANY THE PROGRAM REQUEST FOR ORDER1.
*****
95     CALL DCCPCIO USING PARAMETER-LIST, IN-REC.
96     MOVE TERM-IN TO TERM-OUT IN ERROR-REC, TERM-OUT IN OUT-REC.
97     MOVE RETURN-CD TO RETNCD.
*****
*     CHECK FOR ZERO RETURN CODE. SEND ERROR MSG IF NOT.
*****
98     IF NOT GOOD-RTN,
99         MOVE ERR7 TO ERRMSG,
100        MOVE RETN-CD TO RTNCD, GO TO ERRORS.
*****
*     CHECK FOR BLANK FIELDS.
*****
102    IF NO-CUST,
103        MOVE ERR6 TO ERRMSG, GO TO ERRORS.
105    IF NO-ORD OR NOT OK,
106        MOVE ERR5 TO ERRMSG, GO TO ERRORS.
108    MOVE CUSNO IN TO CUSNO, CUSNOOT.
109    MOVE ORDERN IN IN-REC TO ORDERN IN OUT-REC.
*****
*     READ NAME AND SHIP TO RECORDS.
*****
110    READ CUSTNAME, INVALID KEY
111        MOVE ERR4 TO ERRMSG, GO TO ERRORS.
113    READ CUSTSHIP, INVALID KEY
114        MOVE ERR3 TO ERRMSG, GO TO ERRORS.
116    IF NOT VALID-CUST MOVE ERR2 TO ERRMSG, GO TO ERRORS.
119    IF NOT VALID-SHIP MOVE ERR1 TO ERRMSG, GO TO ERRORS.
*****
*     SET UP THE PRUF SCREEN WITH NAME/ADDRESS INFORMATION.
*****
122    MOVE CUSNM IN CUST-NAME TO CUSNM IN OUT-REC.
123    MOVE CUSA1 IN CUST-NAME TO CUSA1 IN OUT-REC.
124    MOVE CUSA2 IN CUST-NAME TO CUSA2 IN OUT-REC.
125    MOVE CUSA3 IN CUST-NAME TO CUSA3 IN OUT-REC.
126    MOVE SHPNM IN CUST-SHIP TO SHPNM IN OUT-REC.
127    MOVE SHPA1 IN CUST-SHIP TO SHPA1 IN OUT-REC.
128    MOVE SHPA2 IN CUST-SHIP TO SHPA2 IN OUT-REC.
129    MOVE SHPA3 IN CUST-SHIP TO SHPA3 IN OUT-REC.
130    MOVE ZIPCD IN CUST-SHIP TO ZIPCD IN OUT-REC.
*****
*     PUT THE PRUF SCREEN ON THE TERMINAL AND END. THE SCREEN
*     INCLUDES THE PROGRAM REQUEST FOR ORDER2.
*****
131    MOVE PRUF-PUT TO OPN-CD.
132    MOVE 201 TO OUT-LEN.
133    CALL DCCPCIO USING PARAMETER-LIST, OUT-REC.
134    GO TO EDJ.
135    ERRORS.
136    MOVE PUT TO OPN-CD.
137    MOVE 28 TO OUT-LEN.
138    CALL DCCPCIO USING PARAMETER-LIST, ERROR-REC.
139    EDJ.
140    CLOSE CUSTNAME, CUSTSHIP.
141    STOP RUN.

```

STNO -A...B... COBOL SOURCE STATEMENTS .....IDENTFCN SEQ/NO S

```

PROCESS MAP,LINK P,R1
1 IDENTIFICATION DIVISION.
2 PROGRAM-IC. ORDER2.
3 REMARKS. THIS PROGRAM READS THE PRUF SCREEN SENT TO THE TERMINAL BY
  ORDER1 AND UPDATED BY THE TERMINAL OPERATOR.
  THE NAME/ADDRESS RECORDS ARE WRITTEN TO THE HOLDORDER FILE.
  A PRUF SCREEN TO INPUT THE QTY AND ITMNO FIELDS IS
  SENT TO THE TERMINAL. THE SCREEN IS SET UP TO CALL ORDER3.
4 ENVIRONMENT DIVISION.
5 CONFIGURATION SECTION.
6 SOURCE-COMPUTER. IBM-S3.
7 OBJECT-COMPUTER. IBM-S3.
8 INPUT-OUTPUT SECTION.
9 FILE-CONTROL.
10 SELECT HOLDORD5 ASSIGN TO DA-5445-R-HOLDORD5-U
    ACCESS IS RANDOM
    ACTUAL KEY IS RREC.
11 DATA DIVISION.
12 FILE SECTION.
13 FD HOLDORD5
    LABEL RECORDS ARE STANDARD
    DATA RECORDS ARE HOLDORD, HOLDORDN1, HOLDORDN2, HOLDORDN3,
    HOLDORDN4.
14 01 HOLDORD.
15 02 CODE PIC X.
16 88 A-REC VALUE AAA.
17 88 H-REC VALUE HHH.
18 02 RRECD.
19 03 RRECD PIC 9 6 .
20 88 NO-DD VALUE 0.
21 03 RRECD PIC 9 6 .
22 88 NO-DP VALUE 0.
23 03 FILLER PIC X 51 .
24 02 TERMS REDEFINES RRECD.
25 03 RRECT PIC 9 6 .
26 88 NO-RRECT VALUE 0.
27 03 TERM-TBL PIC X 6 OCCURS 6 INDEXED BY I.
28 03 FILLER PIC X 21 .
29 01 HOLDORDN1.
30 02 CODE PIC XX.
31 02 ORDNUM PIC 9 8 .
32 02 CUSNO PIC 9 6 .
33 02 CUSNM PIC X 22 .
34 02 CUSA1 PIC X 22 .
35 02 FILLER PIC X 4 .
36 01 HOLDORDN2.
37 02 CODE PIC XX.
38 02 CUSA2 PIC X 22 .
39 02 CUSA3 PIC X 22 .
40 02 FILLER PIC X 18 .
41 01 HOLDORDN3.
42 02 CODE PIC XX.
43 02 SHPNM PIC X 22 .
44 02 SHPA1 PIC X 22 .
45 02 FILLER PIC X 18 .

```

```

46 01 HOLDORDN4.
47 02 CODE PIC XX.
48 02 SHPA2 PIC X 22 .
49 02 SHPA3 PIC X 22 .
50 02 ZIPCD PIC 9 5 .
51 02 FILLER PIC X 13 .
52 WORKING-STORAGE SECTION.
53 77 RREC PIC S9 7 VALUE 0.
54 77 RRECDSH PIC 9 6 VALUE 0.
55 77 PUT PIC 9 4 VALUE 50 COMP-4.
56 77 PRUF-PUT PIC 9 4 VALUE 114 COMP-4.
57 77 ERR1 PIC X 20 VALUE  INVALID ORDER FILE.
58 77 ERR2 PIC X 20 VALUE  TERMINAL NOT IN FILE.
59 77 ERR3 PIC X 20 VALUE  END OF ORDER FILE.
60 77 ERR4 PIC X 20 VALUE  ERROR RETURN CODE.
61 01 RETNCDS.
62 02 FILLER PIC 99.
63 02 RETN-CD PIC 99.
*****
* CCP-COBOL INTERFACE PARAMETER LIST. *
*****
64 01 PARAMETER-LIST.
65 02 RETURN-CD PIC S9 4 COMP-4.
66 88 GOOD-RTN VALUE 0.
67 02 OPN-CD PIC S9 4 COMP-4 VALUE 4.
68 02 OUT-LEN PIC S9 4 COMP-4.
69 02 EFF-LEN REDEFINES OUT-LEN PIC S9 4 COMP-4.
70 02 ATTRIB REDEFINES EFF-LEN PIC S9 4 COMP-4.
71 02 IN-LEN PIC S9 4 COMP-4 VALUE 203.
72 02 FILLER PIC X 8 .
*****
* TERMINAL INPUT RECORD AREA. *
*****
73 01 IN-REC.
74 02 TERM-IN PIC X 6 .
75 02 FILLER PIC X 8 .
76 02 ORDNUM PIC 9 8 .
77 02 CUSNO PIC 9 6 .
78 02 CUSNM PIC X 22 .
79 02 SHPNM PIC X 22 .
80 02 CUSA1 PIC X 22 .
81 02 SHPA1 PIC X 22 .
82 02 CUSA2 PIC X 22 .
83 02 SHPA2 PIC X 22 .
84 02 CUSA3 PIC X 22 .
85 02 SHPA3 PIC X 22 .
86 02 ZIPCD PIC 9 5 .
*****
* TERMINAL OUTPUT RECORD AREA. *
*****
87 01 ERROR-REC.
88 02 TERM-OUT PIC X 6 .
89 02 FILLER PIC X 6 VALUE  ZORD5.
90 02 ERRMSG PIC X 20 .
91 02 RTNCD PIC X 2 VALUE SPACES.
92 01 OUT-REC.
93 02 TERM-OUT PIC X 6 .
94 02 FILLER PIC X 6 VALUE  ZORD2.

```

```

95     02  RRCDH      PIC 9 6 .
96     02  RREC      PIC 9 6 .
97     02  ORDNUM    PIC 9 8 .
98     02  CUSNO     PIC 9 6 .
99     02  CUSNM     PIC X 22 .
100    02  SHPNM     PIC X 22 .
101    02  CUSA1     PIC X 22 .
102    02  SHPA1     PIC X 22 .
103    02  CUSA2     PIC X 22 .
104    02  SHPA2     PIC X 22 .
105    02  CUSA3     PIC X 22 .
106    02  SHPA3     PIC X 22 .
107    02  ZIPCD     PIC 9 5 .
108    02  FILLER    PIC X 55 VALUE SPACES.
109    PROCEDURE DIVISION.
110    BEGIN.
111    OPEN I-O HOLDORDS.
*****
*   ACCEPT INPUT. THE SCREEN SET UP BY ORDER1 IS INPUT.   *
*****
112    CALL MCCPCIO USING PARAMETER-LIST, IN-REC.
113    MOVE TERM-IN TO TERM-OUT IN ERROR-REC, TERM-OUT IN OUT-REC.
114    MOVE RETURN-CD TO RETNCD.
*****
*   CHECK FOR ZERO RETURN CODE. SEND ERROR MSG IF NOT.   *
*****
115    IF NOT GOOD-RTN,
116        MOVE ERR4 TO ERRMSG,
117        MOVE RETN-CD TO RTNCD, GO TO ERRORS.
*****
*   READ MASTER RECORD - A RECORD.   *
*****
119    MOVE 1 TO RREC.
120    READ HOLDORDS, INVALID KEY
121        MOVE FRR3 TO ERRMSG, GO TO ERRORS.
123    IF NO-RREC OR NOT A-REC MOVE FRR1 TO ERRMSG, GO TO ERRORS.
*****
*   LOCATE THIS TERMINAL IN THE MASTER RECORD.   *
*****
126    PERFORM LOOK-UP VARYING I FROM 1 BY 1 UNTIL I EQUAL 7.
127    MOVE ERR2 TO ERRMSG, GO TO ERRORS.
*****
*   LOCATE HEADER RECORD FOR THIS TERMINAL.   *
*****
129    TAG1.
130    SET I DOWN BY 1.
131    MOVE 2 TO RREC.
132    IF I NOT EQUAL 0 ADD RRECT TO RREC, GO TO TAG1.
135    MOVE RREC TO RRECDS.
136    READ HOLDORDS, INVALID KEY
137        MOVE ERR3 TO ERRMSG, GO TO ERRORS.
139    IF NO-DD OR NO-DP OR NOT H-REC
140        MOVE FRR1 TO ERRMSG, GO TO ERRORS.
*****
*   IF ALL ORDERS FOR THIS TERMINAL HAVE BEEN PRINTED   *
*   RESET RECORD NUMBER TO FIRST RECORD AND UPDATE THE HEADER.   *
*****
142    IF RRECDD NOT EQUAL RREC DP NEXT SENTENCE ELSE

```

```

143      ADD 1, RRECDHS GIVING RRECDD,
144      MOVE RRECDD TO RRECDP.
145      REWRITE HOLDORD, INVALID KEY GO TO EOJ.
146      MOVE RRECDD TO RREC.
147      PERFORM READF.
148      *****
149      * WRITE 1ST N RECORD. *
150      *****
151      MOVE DN1 TO CODE IN HOLDORDN1.
152      MOVE ORDNUM IN IN-REC TO ORDNUM IN HOLDORDN1,
153      ORDNUM IN OUT-REC.
154      MOVE CUSNO IN IN-REC TO CUSNO IN HOLDORDN1,
155      CUSNO IN OUT-REC.
156      MOVE CUSNM IN IN-REC TO CUSNM IN HOLDORDN1,
157      CUSNM IN OUT-REC.
158      MOVE CUSA1 IN IN-REC TO CUSA1 IN HOLDORDN1,
159      CUSA1 IN OUT-REC.
160      PERFORM CHECK.
161      *****
162      * WRITE 2ND N RECORD. *
163      *****
164      MOVE DN2 TO CODE IN HOLDORDN2.
165      MOVE CUSA2 IN IN-REC TO CUSA2 IN HOLDORDN2,
166      CUSA2 IN OUT-REC.
167      MOVE CUSA3 IN IN-REC TO CUSA3 IN HOLDORDN2,
168      CUSA3 IN OUT-REC.
169      PERFORM CHECK.
170      *****
171      * WRITE 3RD N RECORD. *
172      *****
173      MOVE DN3 TO CODE IN HOLDORDN3.
174      MOVE SHPNM IN IN-REC TO SHPNM IN HOLDORDN3,
175      SHPNM IN OUT-REC.
176      MOVE SHPA1 IN IN-REC TO SHPA1 IN HOLDORDN3,
177      SHPA1 IN OUT-REC.
178      PERFORM CHECK.
179      *****
180      * WRITE 4TH N RECORD. *
181      *****
182      MOVE DN4 TO CODE IN HOLDORDN4.
183      MOVE SHPA2 IN IN-REC TO SHPA2 IN HOLDORDN4,
184      SHPA2 IN OUT-REC.
185      MOVE SHPA3 IN IN-REC TO SHPA3 IN HOLDORDN4,
186      SHPA3 IN OUT-REC.
187      MOVE ZIPCD IN IN-REC TO ZIPCD IN HOLDORDN4,
188      ZIPCD IN OUT-REC.
189      PERFORM CHECK.
190      *****
191      * PUT THE PRUF SCREEN ON THE TERMINAL AND END. THE SCREEN *
192      * INCLUDES THE PROGRAM REQUEST FOR ORDER3. *
193      *****
194      MOVE RRECDHS TO RRECDH.
195      MOVE RREC TO RRECP.
196      MOVE PRUF-PUT TO OPN-CD.
197      MOVE 268 TO OUT-LEN.
198      CALL CCCPCID USING PARAMETER-LIST, OUT-REC.
199      GO TO EOJ.
200      ERRORS.

```

```
176     MOVE PUT TO OPN-CD.
177     MOVE 28 TO OUT-LEN.
178     CALL CCCPCIDP USING PARAMETER-LIST, ERROR-REC.
179     ENJ.
180     CLOSE HOLDORDS.
181     STOP RUN.
182     LOCK-UP.
183     IF TERM-TBL I EQUAL TERM-IN, GO TO TAG1.
185     READF.
186     READ HOLDORDS, INVALID KEY
187     MOVE ERR3 TO ERRMSG, GO TO ERRORS.
189     CHECK.
190     REWRITE HOLDORD, INVALID KEY GO TO ENJ.
192     MOVE SPACES TO HOLDORD.
193     ADD 1 TO RREC.
194     PERFORM READF.
```



STNO -A...B... C O B O L S O U R C E S T A T E M E N T S .....IDENTFCN SEQ/NO S

```

PROCESS MAP,LIST,LINK P,R1
1 IDENTIFICATION DIVISION.
2 PROGRAM-IC. ORDER3.
3 REMARKS. THIS PROGRAM READS THE PRUF SCREEN SENT TO THE
  TERMINAL BY ORDER2 AND UPDATED BY THE OPERATOR WITH ITEM
  AND QUANTITY INFORMATION. THE QUANTITY AND ITEM NUMBERS
  ARE WRITTEN TO THE HOLDORDER FILE AS DD RECORDS
  AFTER LOCATING THE MATCHING INVENTORY RECORD. THE SCREEN
  IS SET UP TO RECALL ORDER3 UNTIL THE PFI KEY IS PRESSED
  TO END THE ORDER. THEN A PRUF SCREEN IS SENT TO THE TERMINAL
  TO CALL ORDER4. ORDER3 IS A MRT PROGRAM.
4 ENVIRONMENT DIVISION.
5 CONFIGURATION SECTION.
6 SOURCE-COMPUTER. IBM-S3.
7 OBJECT-COMPUTER. IBM-S3.
8 INPUT-OUTPUT SECTION.
9 FILE-CONTROL.
10 SELECT HOLDORD5 ASSIGN TO DA-5445-R-HOLDORD5-U
  ACCESS IS RANDOM
  ACTUAL KEY IS RREC.
11 SELECT INVENTORY ASSIGN TO DA-5444-R-INVENTORY
  ACCESS IS RANDOM
  ACTUAL KEY IS ITEMS.
12 DATA DIVISION.
13 FILE SECTION.
14 FD HOLDORD5
  LABEL RECORDS ARE STANDARD
  DATA RECORDS ARE HOLDORD, HOLDORD-OUT.
15 01 HOLDORD.
16 02 CODE PIC X.
17 88 H-REC VALUE DD.
18 02 FILLER PIC X 63 .
19 01 HOLDORD-OUT.
20 02 D-CODE PIC X.
21 02 LIN2 PIC 99.
22 02 QTY2 PIC 9 6 .
23 02 ITM2 PIC 9 8 .
24 02 DESC PIC X 24 .
25 02 PRICE PIC 9 13 V99.
26 02 FILLER PIC X 8 .
27 FD INVENTORY
  LABEL RECORDS ARE STANDARD
  DATA RECORD IS INVENTORY.
28 01 INVENTORY.
29 02 CODES PIC XX.
30 88 MI VALUE DMI.
31 02 FILLER PIC X 10 .
32 02 ITMDSC PIC X 24 .
33 02 FILLER PIC X 5 .
34 02 PRICE PIC 9 4 V99.
35 02 FILLER PIC X 119 .
36 02 QTYOH PIC 9 6 .
37 02 FILLER PIC X 128 .
38 WORKING-STORAGE SECTION.
39 77 RREC PIC S9 7 VALUE 0.

```

```

40 77 ITEMS          PIC S9 7 VALUE 0.
41 77 PRICES        PIC 9 13 V99.
42 77 ACCEPT-CD     PIC 9 4 VALUE 4 COMP-4.
43 77 INVITE        PIC 9 4 VALUE 5 COMP-4.
44 77 RELEASE       PIC 9 4 VALUE 10 COMP-4.
45 77 PUT           PIC 9 4 VALUE 50 COMP-4.
46 77 PRUF-PUT      PIC 9 4 VALUE 114 COMP-4.
47 77 ERR1          PIC X 20 VALUE 0ERROR RETURN CODE0.
48 77 ERR2          PIC X 20 VALUE 0NO ITEMS ENTERED0.
49 77 ERR3          PIC X 20 VALUE 0END OF ORDER FILE0.
50 77 ERR4          PIC X 20 VALUE 0ITEM ENTRIES INVALID0.
51 77 ERR5          PIC X 20 VALUE 0BACK ORDER ITEM0.
52 77 ERR6          PIC X 20 VALUE 0ITEM NOT IN INVENTORY0.
53 77 ERR7          PIC X 20 VALUE 0INVALID INVENTORY REC0.
54 01 RETNCDS.
55 02 FILLER        PIC 99.
56 02 RETN-CD       PIC 99.

```

\*\*\*\*\*  
\* CCP-COBOL INTERFACE PARAMETER LIST. \*  
\*\*\*\*\*

```

57 01 PARAMETER-LIST.
58 02 RETURN-CD    PIC S9 4 COMP-4.
59 88 GOOD-RTN    VALUE 0.
60 02 OPN-CD       PIC S9 4 COMP-4 VALUE 4.
61 02 OUT-LEN      PIC S9 4 COMP-4.
62 02 EFF-LEN REDEFINES OUT-LEN PIC S9 4 COMP-4.
63 88 NO-INVITES  VALUE 0.
64 02 ATTRIB REDEFINES EFF-LEN PIC S9 4 COMP-4.
65 02 IN-LEN       PIC S9 4 COMP-4 VALUE 91.
66 02 FILLER       PIC X 8 .

```

\*\*\*\*\*  
\* TERMINAL INPUT RECORD AREA. \*  
\*\*\*\*\*

```

67 01 IN-REC.
68 02 TERM-IN     PIC X 6 .
69 02 AID         PIC X.
70 88 PA1        VALUE 0%.
71 88 PF1        VALUE 01.
72 02 FILLER     PIC X 7 .
73 02 RRECDH     PIC 9 6 .
74 02 RRECOD     PIC 9 6 .
75 02 LIN1       PIC 99.
76 02 LIN1R REDEFINES LIN1 PIC XX.
77 88 NO-LIN1   VALUE SPACES.
78 02 QTY1       PIC 9 6 .
79 02 QTY1R REDEFINES QTY1 PIC X 6 .
80 88 NO-QTY    VALUE SPACES.
81 02 ITM1       PIC 9 8 .
82 02 ITM1R REDEFINES ITM1 PIC X 8 .
83 88 NO-ITM    VALUE SPACES.
84 02 LIN2       PIC 99.
85 02 LIN2R REDEFINES LIN2 PIC XX.
86 88 NO-LIN2   VALUE SPACES.
87 02 QTY2       PIC 9 6 .
88 02 ITM2       PIC 9 8 .
89 02 DESC       PIC X 24 .
90 02 PR         PIC 9 13 V99.

```

\*\*\*\*\*

```

*   TERMINAL OUTPUT RECORD AREA.   *
*****
91  01  ERROR-REC.
92      02  TERM-OUT      PIC X 6 .
93      02  FILLER        PIC X 6  VALUE  □$ZORD5□.
94      02  ERRMSG        PIC X 20 .
95      02  RTNCD         PIC X 2  VALUE  SPACES.
96  01  OUT-REC.
97      02  TERM-OUT      PIC X 6 .
98      02  FILLER        PIC X 6  VALUE  □$ZORD4□.
99      02  RRECDH        PIC 9 6 .
100     02  RRECD         PIC 9 6 .
101     02  LIN1          PIC 99.
102     02  FILLER        PIC X 14 .
103     02  FRR-MSGS      PIC X 22 .
104     02  LIN2          PIC 99.
105     02  QTY2          PIC 9 6 .
106     02  ITM2          PIC 9 8 .
107     02  DESC          PIC X 24 .
108     02  PR            PIC ZZZZ,ZZZ,ZZZ.ZZ.
109  01  END-REC.
110     02  TERM-OUT      PIC X 6 .
111     02  FILLER        PIC X 6  VALUE  □$ZORD3□.
112     02  RRECDH        PIC 9 6 .
113     02  RRECD         PIC 9 6 .
114  01  TABLES.
115     02  TERM          PIC X 6  OCCURS 6 INDEXED BY IT.
116     02  PRI           PIC 9 13 V99 OCCURS 6 INDEXED BY IP.
117  PROCEDURE DIVISION.
118  BEGIN.
119     OPEN INPUT INVENTORY, I-O HOLDORD5.
120     MOVE SPACES TO TABLES.
*****
*   ACCEPT INPUT. THE SCREEN SET UP BY ORDER2 IS INPUT.   *
*****
121  ACCEPT-INPUT.
122     CALL OCCPCIO□ USING PARAMETER-LIST, IN-REC.
123     MOVE TERM-IN TO TERM-OUT IN ERROR-REC, TERM-OUT IN OUT-REC,
        TERM-OUT IN END-REC.
124     MOVE RETURN-CD TO RETNCD.
125     MOVE RRECDH IN IN-REC TO RRECDH IN OUT-REC, RRECDH IN END-REC.
*****
*   SAVE TERMINAL ID IN CASE ANOTHER TERMINAL CALLS ORDER3.   *
*****
126     PERFORM TERM-SEARCH VARYING IT FROM 1 BY 1
        UNTIL IT GREATER THAN 6.
127     PERFORM NO-TERM VARYING IT FROM 1 BY 1
        UNTIL IT GREATER THAN 6.
*****
*   CHECK FOR ZERO RETURN CODE. SEND ERROR MSG IF NOT.   *
*****
128  CHECK-RETURN.
129     IF NOT GOOD-RTN,
130         MOVE ERR1 TO ERRMSG,
131         MOVE RETN-CD TO RTNCD, GO TO ERRORS.
*****
*   CHECK FOR PA1 KEY WHICH MEANS CANCEL ORDER.
*   THE CANCELLED MESSAGE IS SENT TO THE TERMINAL, WHICH   *

```

```

* IS THEN RELEASED. *
*****
133 IF PA1 GO TO ERRORS.
135 IF NO-LIN2 AND PF1,
136 MOVE ERR2 TO ERRMSG, GO TO ERRORS.
138 SET IP TO IT.
139 IF PF1 AND NOT NO-LIN2, GO TO TAG3.
141 IF NO-LIN1 OR NO-QTY OR NO-ITM,
142 MOVE ERR4 TO ERR-MSGS, GO TO SEND-SCREEN.
144 MOVE ITM1 IN IN-REC TO ITEMS.
145 READ INVENTORY, INVALID KEY
146 MOVE ERR6 TO ERR-MSGS, GO TO SEND-SCREEN.
148 IF NOT MI,
149 MOVE ERR7 TO ERR-MSGS, GO TO SEND-SCREEN.
151 IF QTYOH LESS THAN QTY1,
152 MOVE ERR5 TO ERR-MSGS.
*****
* CHECK FOR LINE UPDATE. OPERATOR WANTS TO CHANGE THIS LINE. *
*****
153 IF LIN1 IN IN-REC EQUAL LIN2 IN IN-REC,
154 OR NO-LIN2 GO TO TAG2.
155 TAG3.
156 MOVE RRECDD IN IN-REC TO RREC.
157 READ HOLDORD5, INVALID KEY GO TO BAD-HOLDORD.
159 IF H-REC GO TO BAD-HOLDORD.
161 MOVE H00 TO D-CODE.
162 MOVE LIN2 IN IN-REC TO LIN2 IN HOLDORD-OUT.
163 MOVE QTY2 IN IN-REC TO QTY2 IN HOLDORD-OUT.
164 MOVE ITM2 IN IN-REC TO ITM2 IN HOLDORD-OUT.
165 MOVE DESC IN IN-REC TO DESC IN HOLDORD-OUT.
166 MOVE PRI IP TO PRICE IN HOLDORD-OUT.
167 REWRITE HOLDORD-OUT, INVALID KEY GO TO EOJ.
169 ADD 1 TO RRECDD IN IN-REC.
*****
* IF THE OPERATOR PRESSES THE PF1 KEY TO END THE ORDER, *
* PUT THE PRUF SCREEN ON THE TERMINAL AND END. THE SCREEN *
* INCLUDES THE PROGRAM REQUEST FOR ORDER4. *
*****
170 IF PF1 AND NOT NO-LIN2,
171 MOVE RRECDD IN IN-REC TO RRECDD IN END-REC,
172 MOVE PRUF-PUT TO OPN-CD,
173 MOVE 18 TO OUT-LEN,
174 CALL HCCPCION USING PARAMETER-LIST, END-REC,
175 GO TO RELEASE-TERM.
176 TAG2.
177 MOVE LIN1 IN IN-REC TO LIN1 IN OUT-REC, LIN2 IN OUT-REC.
178 ADD 1 TO LIN1 IN OUT-REC.
179 MOVE QTY1 IN IN-REC TO QTY2 IN OUT-REC.
180 MOVE ITM1 IN IN-REC TO ITM2 IN OUT-REC.
181 MOVE ITMDESC TO DESC IN OUT-REC.
182 COMPUTE PRICES EQUAL
PRICE IN INVENTORY * QTY1 IN IN-REC.
183 MOVE PRICES TO PRI IP, PR IN OUT-REC.
*****
* SEND THE DETAIL SCREEN. THE SCREEN IS SET UP TO CALL ORDER3 *
* EACH TIME THE OPERATOR ENTERS A LINE OR USES THE PF1 KEY. *
*****
184 SEND-SCREEN.

```

```

185     MOVE RRECDD IN IN-REC TO RRECDD IN OUT-REC.
186     MOVE PUT TO OPN-CD.
187     MOVE 111 TO OUT-LEN.
188     CALL □CCPCIO□ USING PARAMETER-LIST, OUT-REC.
*****
*     REINVOKE THE TERMINAL. *
*****
189     MOVE INVITE TO OPN-CD.
190     MOVE 8 TO IN-LEN.
191     CALL □CCPCIO□ USING PARAMETER-LIST, END-REC.
*****
*     GO READ THE NEXT SCREEN. *
*****
192     MORE-INPUT.
193     MOVE ACCEPT-CD TO OPN-CD.
194     MOVE 91 TO IN-LEN.
195     GO TO ACCEPT-INPUT.
196     BAC-HOLDORD.
197     MOVE ERR3 TO ERRMSG.
198     ERRORS.
199     MOVE PUT TO OPN-CD.
200     MOVE 28 TO OUT-LEN.
201     CALL □CCPCIO□ USING PARAMETER-LIST, ERROR-REC.
202     RELEASE-TERM.
203     MOVE RELEASE TO OPN-CD.
204     CALL □CCPCIO□ USING PARAMETER-LIST, ERROR-REC.
205     MOVE SPACES TO TERM IT.
206     IF NOT NO-INVITES, GO TO MORE-INPUT.
208     EQJ.
209     CLOSE INVENTORY, HOLDORD5.
210     STOP RUN.
211     TERM-SEARCH.
212     IF TERM IT EQUAL TERM-IN GO TO CHECK-RETURN.
214     NO-TERM.
215     IF TERM IT EQUAL SPACES,
216     MOVE TERM-IN TO TERM IT, GO TO CHECK-RETURN.

```

STNO -A...B... C O B O L S O U R C E S T A T E M F N T S .....IDENTFCN SEQ/NO S

```

PROCESS MAP,LIST,LINK P,R1
1 IDENTIFICATION DIVISION.
2 PROGRAM-ID. ORDER4.
3 REMARKS. THIS PROGRAM READS THE PRUF SCREEN SENT TO THE
  TERMINAL BY ORDER3.
  THE ITEM AND QUANTITY DETAIL RECORDS ARE NOW ON THE
  HOLDORD FILE. THE DETAIL RECORDS ARE READ FROM THE HOLDORD
  FILE AND THE INVENTORY FILE IS UPDATED. THE ORDER TOTAL
  IS WRITTEN TO THE HOLDORD FILE.
4 ENVIRONMENT DIVISION.
5 CONFIGURATION SECTION.
6 SOURCE-COMPUTER. IBM-S3.
7 OBJECT-COMPUTER. IBM-S3.
8 INPUT-OUTPUT SECTION.
9 FILE-CONTROL.
10 SELECT HOLDORD5 ASSIGN TO DA-5445-R-HOLDORD5-U
  ACCESS IS RANDOM
  ACTUAL KEY IS RREC.
11 SELECT INVENTORY ASSIGN TO DA-5444-R-INVENTORY-U
  ACCESS IS RANDOM
  ACTUAL KEY IS ITEMS.
12 DATA DIVISION.
13 FILE SECTION.
14 FD HOLDORD5
  LABEL RECORDS ARE STANDARD
  DATA RECORDS ARE HOLDORDH, HOLDORDD, HOLDORDT.
15 01 HOLDORDH.
16 02 CODE PIC X.
17 88 H-REC VALUE 000.
18 02 RRECDP PIC 9 6 .
19 02 RREC1 REDEFINES RRECDP PIC 9 6 .
20 02 FILLER PIC X 6 .
21 02 TRNMN PIC X 6 .
22 02 FILLER PIC X 45 .
23 01 HOLDORDD.
24 02 CODE PIC X.
25 88 D-REC VALUE 000.
26 02 FILLER PIC XX.
27 02 QTYP PIC 9 6 .
28 02 ITMP PIC 9 8 .
29 02 FILLER PIC X 24 .
30 02 PR PIC 9 13 V99.
31 02 FILLER PIC X 8 .
32 01 HOLDORDT.
33 02 CODE PIC X.
34 02 PRICE PIC 9 9 V99.
35 02 TAX1 PIC 9 8 V99.
36 02 TAX2 PIC 9 8 V99.
37 02 PRT PIC 9 9 V99.
38 02 FILLER PIC X 21 .
39 FD INVENTORY
  LABEL RECORDS ARE STANDARD
  DATA RECORD IS INVENTORY.
40 01 INVENTORY.
41 02 FILLER PIC X 160 .

```

```

42 02 QTYND PIC 9 6 .
43 02 QTYOH PIC 9 6 .
44 02 FILLER PIC X 128 .
45 WORKING-STORAGE SECTION.
46 77 RREC PIC S9 7 VALUE 0.
47 77 ITEMS PIC S9 7 VALUE 0.
48 77 RRECP PIC 9 6 .
49 77 PRTC PIC 9 9 V99.
50 77 TAX1C PIC 9 8 V99.
51 77 TAX2C PIC 9 8 V99.
52 77 PRICES PIC 9 9 V99 VALUE 0.
53 77 PUT PIC 9 4 VALUE 50 COMP-4.
54 77 PRUF-PUT PIC 9 4 VALUE 114 COMP-4.
55 77 ERR1 PIC X 20 VALUE □ERROR RETURN CODE□.
56 77 ERR2 PIC X 20 VALUE □END OF ORDER FILE□.
57 77 ERR3 PIC X 20 VALUE □ORDER FILE OVERLAID□.
58 77 QTYT PIC 9 6 .
59 01 RETNCDS.
60 02 FILLER PIC 99.
61 02 RETN-CD PIC 99.

```

\*\*\*\*\*  
\* CCP-COBOL INTERFACE PARAMETER LIST. \*  
\*\*\*\*\*

```

62 01 PARAMETER-LIST.
63 02 RETURN-CD PIC S9 4 COMP-4.
64 88 GOOD-RTN VALUE 0.
65 02 OPN-CD PIC S9 4 COMP-4 VALUE 4.
66 02 OUT-LEN PIC S9 4 COMP-4.
67 02 EFF-LEN REDEFINES OUT-LEN PIC S9 4 COMP-4.
68 02 ATTRIB REDEFINES EFF-LEN PIC S9 4 COMP-4.
69 02 IN-LEN PIC S9 4 COMP-4 VALUE 20.
70 02 FILLER PIC X 8 .

```

\*\*\*\*\*  
\* TERMINAL INPUT RECORD AREA. \*  
\*\*\*\*\*

```

71 01 IN-REC.
72 02 TERM-IN PIC X 6 .
73 02 FILLER PIC X 8 .
74 02 RRECDH PIC 9 6 .
75 02 RRECDD PIC 9 6 .
76 02 FILLER PIC X 69 .

```

\*\*\*\*\*  
\* TERMINAL OUTPUT RECORD AREA. \*  
\*\*\*\*\*

```

77 01 ERROR-REC.
78 02 TERM-OUT PIC X 6 .
79 02 FILLER PIC X 6 VALUE □$ZORD5□.
80 02 ERRMSG PIC X 20 .
81 02 RTNCD PIC X 2 VALUE SPACES.
82 01 OUT-REC.
83 02 TERM-OUT PIC X 6 .
84 02 FILLER PIC X 6 VALUE □$ZORD6□.
85 02 PRICE PIC ZZZ,ZZZ,ZZZ.ZZ.
86 02 TAX1 PIC ZZZ,ZZZ,ZZZ.ZZ.
87 02 TAX2 PIC ZZZ,ZZZ,ZZZ.ZZ.
88 02 PRT PIC ZZZ,ZZZ,ZZZ.ZZ.
89 PROCEDURE DIVISION.
90 BEGIN.

```

```

91 OPEN I-O INVENTORY, HOLDORD5.
*****
* ACCEPT INPUT. THE SCREEN SET UP BY ORDER3 IS INPUT. *
*****
92 CALL CCCPCID0 USING PARAMETER-LIST, IN-REC.
93 MOVE TRM-IN TO TERM-OUT IN ERROR-REC, TERM-OUT IN OUT-REC.
94 MOVE RETURN-CD TO RETNCD5.
*****
* CHECK FOR ZERO RETURN CODE. SEND ERROR MSG IF NOT. *
*****
95 IF NOT GOOD-RTN,
96 MOVE ERR1 TO ERRMSG,
97 MOVE RETN-CD TO RTNCD.
*****
* READ THE HOLDORDER FILE AND CHECK FOR VALID H-RECORDS. *
* UPDATE THE H-RECORDS WITH NEW RECORD NUMBERS. *
*****
99 MOVE RRECDD TO RREC.
100 READ HOLDORD5, INVALID KEY GO TO BAD-FILE.
102 IF NOT H-REC GO TO READER.
104 IF TRMNM NOT EQUAL TERM-IN GO TO BAD-FILE.
106 READER.
107 MOVE RRFCDH TO RREC.
108 READ HOLDORD5, INVALID KEY GO TO OVERLAID.
110 IF NOT H-REC GO TO OVERLAID.
112 ADD 4, RRECDP GIVING RRECP.
113 ADD 1, RRECDD GIVING RREC1.
114 REWRITE HOLDORDH, INVALID KEY GO TO EOJ.
*****
* READ THE DETAIL RECORDS AND TOTAL THEM UP. *
*****
116 TAG1.
117 MOVE RRECP TO RREC.
118 READ HOLDORD5, INVALID KEY GO TO EOJ.
120 IF NOT D-REC
121 MOVE ERR3 TO ERRMSG, GO TO ERRORS.
123 ADD PR TO PRICES.
124 ADD 1 TO RRECP.
*****
* UPDATE THE INVENTORY RECORD TO REFLECT THIS ORDER. *
*****
125 MOVE ITMP TO ITEMS.
126 READ INVENTORY, INVALID KEY GO TO EOJ.
128 IF QTYOH NOT LESS THAN QTYP,
129 SUBTRACT QTYP FROM QTYOH,
130 GO TO QTY-OK.
131 SUBTRACT QTYOH FROM QTYP GIVING QTYT.
132 ADD QTYT TO QTYOH.
133 MOVE 0 TO QTYOH.
134 QTY-OK.
135 REWRITE INVENTORY, INVALID KEY GO TO EOJ.
137 IF RRECP NOT EQUAL RRECDD GO TO TAG1.
139 COMPUTE TAX1C EQUAL PRICES * .04.
140 COMPUTE TAX2C EQUAL PRICES * .05.
141 COMPUTE PRTC EQUAL PRICES + TAX1C.
142 ADD TAX2C TO PRTC.
*****
* WRITE THE TOTAL RECORD INTO THE HOLDORDER FILE. *

```



```

*****
143       MOVE RREC TO RREC.
144       READ HOLDORD5, INVALID KEY GO TO E0J.
146       MOVE DT# TO CODE IN HOLDORDT.
147       MOVE PRICES TO PRICE IN HOLDORDT, PRICE IN OUT-REC.
148       MOVE TAX1C TO TAX1 IN HOLDORDT, TAX1 IN OUT-REC.
149       MOVE TAX2C TO TAX2 IN HOLDORDT, TAX2 IN OUT-REC.
150       MOVE PRTC TO PRT IN HOLDORDT, PRT IN OUT-REC.
151       REWRITE HOLDORDT, INVALID KEY GO TO E0J.
*****
*       SEND THE TOTAL SCREEN AND END.       *
*****
153       MOVE PUT TO OPN-CD.
154       MOVE 66 TO OUT-LEN.
155       CALL @CCPCID# USING PARAMETER-LIST, OUT-REC.
156       GO TO E0J.
157       BAC-FILE.
158       MOVE ERR2 TO ERRMSG, GO TO ERRORS.
160       OVERLAD.
161       MOVE ERR3 TO ERRMSG.
162       ERRORS.
163       MOVE PUT TO OPN-CD.
164       MOVE 28 TO OUT-LEN.
165       CALL @CCPCID# USING PARAMETER-LIST, ERROR-REC.
166       E0J.
167       CLOSE INVENTORY, HOLDORD5.
168       STOP RUN.

```



After you have written a program which is to be run under CCP, you must take steps to make it usable under CCP:

1. Compile or assemble the program to create an object module.
2. Link-edit the object module.
3. You may need to copy the program module to your production CCP disk pack or to the system pack that will be online by using the Library Maintenance program (\$MAINT). When using 5704-SC2, the program can reside on any of the simulation areas.
4. Prepare assignment specifications to include the program and the resources required by the program (such as disk files, terminals, and unit record devices) in one or more assignment sets in the assignment file, \$CCPFILE, and execute the Assignment Build program (see *CCP System Reference Manual*).

Before taking the preceding steps, be sure to read *Disk File Considerations* and *Unit Record File Considerations*, later in this chapter.

Source programs cannot be compiled or assembled under CCP control. Programs may be compiled or assembled in the non-CCP program level of a DPF (dual partition) system; however, if the program is a recompilation of a program that is available to CCP in the current assignment set, the new object module should be placed on a disk pack that is not being used by CCP. New programs may be placed on a CCP program pack, but will not be available to CCP until they are included in an assignment set.

**Note:** 5704-SC2 supports an application development feature that allows new or modified programs to be catalogued online to an active CCP library on the system or program pack, or any 3340/3344 simulation area. By specifying EXECFIND-YES on the PROGRAM assignment statement, the program is located in the library each time the program is requested. If this function is used extensively, system performance can be degraded because the program must be located each time it is requested rather than once at Startup.

**Note:** If EXECFIND-YES is specified on the program assignment statement, CCP searches the program, system, and 3340/3344 simulation area libraries. For more information see the *IBM System/3 Model 15 Communications Control Program System Reference Manual*, GC21-7620.

## COMPILING AND LINK-EDITING THE PROGRAM – MODEL 15 CCP

Procedures for compiling or assembling a source program are given in the appropriate programming language reference manual (see *Appendix C: Bibliography*). See also the *IBM System/3 Model 15 System Control Programming Reference Manual*, GC21-5077 (5704-SC1), or the *IBM System/3 Model 15 System Control Programming Concepts and Reference Manual*, GC21-5162 (5704-SC2). If the program is written in Basic Assembler language and uses CCP-provided macros (\$Nxxx), the source program must be processed by both the Macro Processor and the Basic Assembler.

The compilation (or assembly) output is a relocatable object module that must be link-edited to form a load module with a start address of X'8000'. CCP Startup accepts only programs that are link-edited to start at X'8000'. If compilation and link-editing are to be performed as a single step (as they can be, in RPG II, SORT, COBOL, and FORTRAN IV), the start address is specified by using the LINKADD parameter on the // COMPILE OCL statement (LINKADD=8000). If compilation and link-editing are to be separate steps, the // PHASE Overlay Linkage Editor control statement must specify LINKADD=X'8000'. Assembling and link-editing Basic Assembler programs must always be separate steps. RPG II, COBOL, and FORTRAN IV programs can be compiled and link-edited either as a single step or as separate steps, as determined by:

- |                   |  |
|-------------------|--|
| RPG II            | – The entry in column 10 (Object Output) of the RPG II Control Card Specifications |
| COBOL and FORTRAN | – The LINK operand of the PROCESS statement  |

The maximum size of a user program under Model 15 CCP is 32K, including any program-appended storage area for DFF, but not including any additional storage for execution of memory resident overlay programs. Under 5704-SC2, external buffers can be defined and supported. The storage required for the external buffers is not included in the maximum 32K program size. See the *IBM System/3 Model 15 System Control Programming Concepts and Reference Manual*, GC21-5162, for additional information on external buffers.

Programs which use 5445/3340 disk files must be compiled/link-edited on a system of that disk-file type. This will enable the proper data management support to be included in the program.

*Notes:*

1. Programs running directly under Model 15 disk system management (not running under CCP) must be link-edited to start at X'4000'. A halt occurs if these programs are link-edited at X'8000'.
2. Programs that use indexed random add disk file access (IRA or IRUA — see index entry *access value*) for files that can be shared must be link-edited on a system that has been generated to support CCP.

### COMPILING THE PROGRAM—MODEL 10 AND MODEL 12 CCP

Procedures for compiling or assembling a source program are given in the appropriate programming language reference manual or in the *IBM System/3 Models 8 and 10 Disk System Control Programming Reference Manual*, GC21-7512, or the *IBM System/3 Model 12 System Control Programming Reference Manual*, GC21-5130.

#### *RPG II—Model 10 and Model 12 CCP*

During generation, CCP copies the RPG II communications service subroutines (SUBR90, SUBR91, SUBR92, and SUBR93) to the disk pack specified by the PPUNIT-xx parameter of the \$EPLG generation statement. This pack must be used to compile RPG II programs to run under CCP.

If unit record devices—MFCU, 1442 Card Read/Punch, 3741 (data mode support only), 5203 or 1403 printers—were specified as being supported by CCP, CCP also copies special intermediary routines for unit record data management modules to the RPG II pack specified by the PPUNIT-xx parameter. These intermediary routines are named the same as the RPG II unit record data management modules and the RPG II modules are renamed. If unit record devices were specified, this pack cannot be used to compile RPG II programs to run under disk system management (DSM) that use unit record devices. See *Link Editing the Program*, later in this section, for additional information.

#### *COBOL and FORTRAN IV—Model 10 and Model 12 CCP*

During generation, CCP copies the COBOL and/or FORTRAN IV communications service subroutines (CCPCIO, CCPFIO) to the pack specified by the PPUNIT-xx parameter of the \$EPLG generation statement for COBOL and/or the \$EPLG statement for FORTRAN.

If unit record devices—MFCU, 1442 Card Read/Punch, 3741 (data mode support only), 5203 or 1403 printers—were specified as being supported by CCP, CCP generation also copies special intermediary routines for the unit record data management modules to the COBOL or FORTRAN pack specified by the PPUNIT-xx parameter. These routines interface with the standard DSM modules; they are not replacements for the DSM modules.

The COBOL and FORTRAN IV compilers permit you to choose, by means of a PROCESS statement:

1. To produce a relocatable module that can be processed in a separate link-editing run (if the program uses unit record devices).
2. To invoke the Overlay Linkage Editor automatically to produce a load module (if not using unit record devices). (See *Link Editing the Program*.)

#### *Basic Assembler—Model 10 and Model 12 CCP*

During generation, CCP copies CCP-provided macros (\$Nxxx) to the pack specified in the PPUNIT-xx parameter of the \$EPLG statement (LANG-ASSEM).

If unit record devices—MFCU, 1442 Card Read/Punch, 3741 (data mode support only), 5203 or 1403 printers—were specified as being supported by CCP, CCP generation also copies special intermediary routines for the unit record data management modules to the pack specified by the PPUNIT-xx parameter. These routines interface with the standard DSM modules; they are not replacements for the DSM modules. See *Link-Editing the Program* for additional information.

If your Basic Assembler program uses CCP-provided macros, both the Macro Processor and the Basic Assembler program are required to process the source program prior to link editing. The \$Nxxx macros must either be on the same pack as the Macro Processor or on the system pack. If your program does not use CCP macros, you need only assemble and link edit the program.

## LINK-EDITING THE PROGRAM—MODEL 10 AND MODEL 12 CCP

Link-editing is not a separate step when preparing an RPG II program to be run under CCP, since RPG II compilation includes link-editing. Link-editing is always a separate step in preparing a Basic Assembler program.

When compiling COBOL and FORTRAN IV programs, you can specify that compilation and link editing be a single step by means of the LINK operand on the PROCESS statement. However, you *must not* choose to invoke the Overlay Linkage Editor automatically for programs that access one or more unit record devices (MFCU, 1442 Card Read Punch, 5203 or 1403 printers).

If the program being prepared was written in COBOL, FORTRAN IV, or Basic Assembler and accesses one or more unit record devices, you must enter // EQUATE Overlay Linkage Editor statements to cause the special CCP-provided subroutines to be inserted in the load module between your program and the DSM unit record data management modules your program uses. These statements must be in the following form:

```
// EQUATE OLDNAME-$$xxxx,NEWNAME-$Nxxxx
```

Figure 9-1 shows the OLDNAME- and NEWNAME- parameters that must be used for various unit record files. Use Figure 9-3 and 9-4 to determine which DSM unit record data management modules will be used by your COBOL or FORTRAN program.

You must not enter // EQUATE statements if the disk pack you are using for link-editing also supports RPG II, since the DSM modules have already been renamed. Figure 9-2 shows the normal RPG II unit record data management module names and CCP intermediary module names. Since the intermediary modules have the standard DSM module names in this case, no renaming is required.

## Overlay Linkage Editor Control Statements—Model 10 and Model 12 CCP

The following Overlay Linkage Editor control statements are used for COBOL, FORTRAN IV, and Basic Assembler programs that use unit record devices (see *System/3 Overlay Linkage Editor Reference Manual*, GC21-7561, for complete descriptions of these statements):

1. Disk pack used for link-editing does not support RPG II for CCP (see Figure 9-1):

```
// PHASE . . .
// OPTIONS . . .
// EQUATE OLDNAME-$$xxxx,NEWNAME-
  $Nxxxx,UNIT-xx
// EQUATE OLDNAME-@@xxxx,NEWNAME-
  $$xxxx,UNIT-xx
// INCLUDE NAME-user program name,... (or relo-
  catable object deck)
* // INCLUDE NAME-$$xxxx,. . .
* // INCLUDE NAME-$Nxxxx,. . .
.
:
// END
```

- \* The INCLUDE statements can be eliminated if:
- a. The data management modules reside on the same pack as the Overlay Linkage Editor.
  - b. The UPACK- keyword is specified on the // OPTIONS statement.

*Example:* Assume the following:

- a. Program SAMPLE resides in the relocatable library on R1.
- b. Program SAMPLE uses the printer under CCP.
- c. The Overlay Linkage Editor and the data management modules reside on the same pack (R1).
- d. R2 is the pack to be used during a CCP run.

The required Overlay Linkage Statements are:

```
// PHASE NAME-SAMPLE,UNIT-R2,RETAIN-P
// OPTIONS MAP-XREF
// INCLUDE NAME-SAMPLE,UNIT-R1
// EQUATE OLDNAME-$$LPRT,NEWNAME-$NLPRT
// EQUATE OLDNAME-@@LPRT,NEWNAME-$$LPRT
// END
```

MODEL 10 AND MODEL 12 UNIT RECORD DATA MANAGEMENT FUNCTION	FIRST // EQUATE		SECOND // EQUATE	
	OLDNAME-	NEWNAME-	OLDNAME-	NEWNAME-
1442 card read/punch	\$\$ARFF	\$NARFF	@@ARFF	\$\$ARFF
5203/1403 printer	\$\$LPRT	\$NLPRT	@@LPRT	\$\$LPRT
5424 MFCU read/punch	\$\$MFRU	\$NMFRU	@@MFRU	\$\$MFRU
Read/print	\$\$MFRP	\$NMFRP	@@MFRP	\$\$MFRP
Read only	\$\$MFRD	\$NMFRD	@@MFRD	\$\$MFRD
Punch only	\$\$MFPU	\$NMFPU	@@MFPU	\$\$MFPU
Print only	\$\$MFPR	\$NMFPR	@@MFPR	\$\$MFPR
Print/punch	\$\$MFPP	\$NMFPP	@@MFPP	\$\$MFPP
Full function	\$\$MFFF	\$NMFFF	@@MFFF	\$\$MFFF
3741 Read	\$\$CPIP	\$NCPIP	@@CPIP	\$\$CPIP
Punch	\$\$CPOP	\$NCPop	@@CPOP	\$\$CPOP

Figure 9-1. Unit Record Data Management Names to be used in // EQUATE Overlay Linkage Editor Statements for Model 10 and Model 12 COBOL, FORTRAN IV, and Basic Assembler Programs Using Unit Record Devices (non-RPG II Disk Pack)

2. Disk pack used for link-editing supports RPG II for CCP (see Figure 9-2):

```
// PHASE . . .
// OPTIONS . . .
// INCLUDE NAME-user program name . . . (or
relocatable object deck)
* // INCLUDE NAME-$$xxx,...
* // INCLUDE NAME-$$Uxxx,...
:
:
// END
```

\* See note under 1.

*Example:* Assume the following:

- Program SAMPLE is a relocatable object deck.
- Program SAMPLE uses the printer under CCP.
- The Overlay Linkage Editor resides on a pack separate from the data management modules.
- The pack to be used during the CCP run resides on R2.

The required Overlay Linkage Editor statements are:

```
// PHASE NAME-SAMPLE,UNIT-R2,RETAIN-P
// OPTIONS MAP-XREF
Object deck
// INCLUDE NAME-$$LPRT,UNIT-R1
// INCLUDE NAME-$$UPRT,UNIT-R1
// END
```

#### Link-Editing a Program to Run Under DSM—Models 10 and 12

Special considerations apply to link-editing programs to run directly under Model 10 and Model 12 DSM if unit record devices are used in the programs and are also supported by the CCP.

#### RPG II—Model 10 and Model 12 CCP

If unit record devices are being used by both the CCP RPG II programs and the non-CCP RPG II programs, you must use separate disk packs for compiling the two different types of programs. If unit record devices are not being used in both the CCP environment and the non-CCP environment, the same pack can be used for compiling both types.

Model 10 and Model 12 Unit Record Data Management Function	Unit Record Data Management Module	CCP Intermediary Module
1442 card read/punch	\$\$URFF	\$\$ARFF
5203/1403 printer	\$\$UPRT	\$\$LPRT
5424 MFCU read/punch	\$\$UFRU	\$\$MFRU
Read/print	\$\$UFRP	\$\$MFRP
Read only	\$\$UFRD	\$\$MFRD
Punch only	\$\$UFPU	\$\$MFPU
Print only	\$\$UFPR	\$\$MFPR
Print/punch	\$\$UFPP	\$\$MFPP
Full function	\$\$UFFF	\$\$MFFF
3741		
Read	\$\$UPIP	\$\$CPIP
Punch	\$\$UPOP	\$\$CPOP

Figure 9-2. Unit Record Data Management Names for All Languages, if the Pack Also Supports RPG II for Model 10 and Model 12 CCP. // INCLUDE Overlay Linkage Editor statements are required if the Overlay Linkage Editor is on a different pack from these modules.

Model 10 and Model 12 Data Management Module Name	Language Statements as they Appear in a COBOL Program
\$\$ARFF	UR-1442-RD or UR-1442-PU
\$\$LPRT	UR-1403-n- <i>P</i> nnn or UR-5203-n- <i>P</i> nnn
\$\$MFRU	* { UR-5424 } <i>P</i> -RD-n
	<i>S</i> -PU-n
\$\$MFRP	* { UR-5424 } <i>P</i> -RD-n
	<i>S</i> -PR-n
\$\$MFFF	* { UR-5424 } <i>P</i> -RD-n
	<i>S</i> -PU-n
	<i>S</i> -PR-n
\$\$MFRD	UR-5424 <i>P</i> -RD
\$\$MFPU	UR-5424 <i>P</i> -PU
\$\$MFPR	UR-5424 <i>P</i> -PR
\$\$MFPP	UR-5424 <i>P</i> -PI

\* Hopper (P or S) and association value (n) are the same in each statement.

Figure 9-3. Unit Record Data Management Used by Model 10 and Model 12 COBOL Programs

Model 10 and Model 12 Data Management Module Name	Device Option Statements and Language Statements Used in a FORTRAN Program
\$\$ARFF	// READ DEVICE-1442 READ (9, ...) ... // PUNCH DEVICE-1442 WRITE (9, ...) ...
\$\$LPRT	// PRINT DEVICE- { 1403 } WRITE (3, ...) ... CALL P1403 (...) } 1403 CALL S1403 (...) } CALL SP1403 (...) } CALL PRINT (...) } 5203 CALL SKIP (...) } CALL SPACE (...) }
\$\$MFRU	// READ DEVICE-MFCU2 // PUNCH DEVICE-MFCU2 CALL READ (...) CALL PUNCH (...)
\$\$MFRP	// READ DEVICE-MFCU2 // PRINT DEVICE-MFCU2 READ (2, ...) ... WRITE (2, ...) ... CALL READ (...)
\$\$MFFF	// READ DEVICE-MFCU2 // PUNCH DEVICE-MFCU2 // PRINT DEVICE-MFCU2 READ (2, ...) ... WRITE (2, ...) ... CALL READ (...) CALL PUNCH (...)
\$\$MFRD	// READ DEVICE-MFCU1 READ (1, ...) ... // READ DEVICE-MFCU2 READ (2, ...) ... CALL READ (...) CALL STACK (...)
\$\$MFPU	// PUNCH DEVICE-MFCU2 WRITE (2, ...) ... CALL PUNCH (...) CALL STACK (...)
\$\$MFPR	// PRINT DEVICE-MFCU2 WRITE (2, ...) ...
\$\$MFPP	// PRINT DEVICE-MFCU2 // PUNCH DEVICE-MFCU2 WRITE (2, ...) ... CALL PUNCH (...)

Figure 9-4. Unit Record Data Management Used by Model 10 and Model 12 FORTRAN Programs

Different considerations apply to link editing COBOL, FORTRAN, and Basic Assembler programs to run directly under Model 10 and Model 12 DSM, depending on whether or not the disk pack used for link-editing contains the CCP intermediary routines for RPG II unit record data management.

*Pack Not Containing RPG II CCP Intermediary Modules:* If unit record devices are used, no special INCLUDE or EQUATE statements are required for unit record data management modules. Therefore, automatic link editing can be done when compiling COBOL and FORTRAN programs.

*Pack Containing RPG II CCP Intermediary Modules:* If unit record devices are to be used, a separate link edit step is required. The required Overlay Linkage Editor control statements are:

```
// PHASE ...
// OPTIONS ..
// EQUATE OLDNAME-$$xxxx,NEWNAME-$$Uxxx
// INCLUDE NAME-user program name,... (or relocatable
  object deck)
* // INCLUDE NAME-$$Uxxx,...
.
:
// END
```

\* See note under 1.

*Example:* Assume the following:

- a. Program SAMPLE resides in a relocatable library on R1.
- b. Program SAMPLE uses the printer.
- c. The Overlay Linkage Editor resides on a pack different from the data management modules.
- d. The pack to be used for CCP execution resides on R1.

The required Overlay Linkage Editor statements are:

```
// PHASE NAME-SAMPLE,UNIT-R1,RETAIN-P
// OPTIONS MAP-XREF
// EQUATE OLDNAME-$$LPRT,NEWNAME-$$UPRT
// INCLUDE NAME-SAMPLE,UNIT-R1
// INCLUDE NAME-$$UPRT,UNIT-R1
// END
```

## COPYING THE LOAD MODULE

During CCP execution, the programs to be executed under the CCP must be on either the pack from which the CCP was loaded (// LOAD \$CCP,xx) or the pack from which DSM was initially loaded (IPL'ed). For 5704-SC2, they can reside on any of the simulation areas. If the compile and link edit processes did not cause the program module to be located in the object library of one of these packs, you must execute the Library Maintenance program to copy the module to the appropriate pack.

## MAKING ASSIGNMENTS

To permit a new program to be executed under the CCP, the program must be defined in an assignment set. A // PROGRAM statement and possibly other statements must be added to at least one of the assignment sets, and an Assignment Build run must be made (see *CCP System Reference Manual*).

You must provide the following information about your program to the person responsible for maintaining the CCP assignment sets:

- If the program uses the 3270 Display Format Facility:
  - Number of terminals in the program that use DFF
  - Number of display formats used by the program
  - Size of the largest Field Descriptor Table for display formats used by the program, rounded up to the next multiple of 256 (the size of this table is given in the printed output of the Display Format Generator)

*Note:* See *Assignment* in *CCP System Reference Manual* for an example of calculating storage sizes.

- Disk file usage—disk file organizations and processing methods used (see *Disk File Considerations*); are disk files sharable (see Index entry *disk file sharing*)?
- Input and output communications record area sizes.
- Terminals used by the program and whether they are required by the program in order to run or acquired by the program as it is running.
- Terminal attribute sets required for each terminal to be used in your program.
- Program type—SRT, MRT, never-ending, SORT, dedicated (runalone), reusable—if MRT, how many terminals can the program service at one time? (Dedicated and reusable types are defined for Models 10 and 12.)



- Program name.
- Programming language used.
- The unit record devices used:
  - 5203 Printer (Models 10 and 12)
  - 1403 Printer
  - MFCU
  - MFCM (Model 15 only)
  - 1442 Card Read Punch
  - 2501 Card Reader (Model 15 only)
  - 3741 Data Station Model 1 or 2 or Programmable Work Station Model 3 or 4
- Is data allowed with the program request?
- Whether or not this is a CCP/Sort program.
- Whether or not you want a message sent to a requesting terminal when either you release the terminal or your program goes to end of job.
- The priority of this program when executing (5704-SC2 only).
- How much additional storage is desired for execution of memory resident overlay programs (Model 15).
- Whether or not this program is in the object library at CCP startup (5704-SC2 only).
- If your program is written in either RPG II or FORTRAN and disk file buffer sharing is allowed (Models 10 and 12).

#### UNIT RECORD FILE CONSIDERATIONS—MODEL 10 AND MODEL 12 CCP

The following considerations apply to use of unit record devices by programs running under the CCP and between the CCP and non-CCP program levels of a DPF system:

- A unit record device (1442, MFCU, 3741, 1403, and 5203) cannot be shared by programs running under the CCP and cannot be shared by programs running in opposite levels of a DPF system. If the printer is spooled in the CCP program level on a Model 12, the printer can be shared by other programs in the CCP program level. If your CCP application program uses unit record devices, the CCP allocates the devices to your program before it is allowed to run. Your program keeps control of the devices until it terminates. If no program running under the CCP (or in the process of being loaded) requires a unit record device, the system operator can allocate the use of unit record devices to the non-CCP program level in a DPF system.

- You must not use the printer/keyboard (console) under the CCP except as a communications device, using CCP operations.
- A program using unit record devices compiled to run under the CCP will not run on a non-CCP system.

See *Link-Editing the Program*, earlier in this chapter, for additional considerations for using unit record devices in COBOL, FORTRAN, and Basic Assembler programs.

#### UNIT RECORD FILE CONSIDERATIONS—MODEL 15 CCP

The following considerations apply to use of unit record devices (1403 Printer; MFCU; MFCM; 2501 Card Reader; 3741 Model 1 and 2 Data Station and Model 3 and 4 Programmable Work Station; 1442 Card Read Punch) by programs running under CCP and among the CCP partition and non-CCP partitions:

- If your application program uses unit record devices, CCP allocates the devices to your program before your program is allowed to run. Your program retains control of the devices until it terminates.
- In general, unit record devices cannot be shared by programs running under CCP; they can never be shared among programs running in the CCP partition and programs running in a non-CCP partition. If, however, the unit record device is being spooled in the CCP partition, the device is always available to CCP, but only to one application program at a time. Also, the printer may be shared, even when it is not spooled, by CCP application programs for which PRINTER – SHR is specified in the PROGRAM assignment statement.
- The CRT/Keyboard (console) can be used only as a communications device, that is, CCP operations must be used to perform I/O to the console.
- If you request a program that uses a unit record device and that device is not immediately available to the program, your request is rejected, regardless of whether or not a queue command is in effect. An exception is the Model 15 CCP; it will queue for a printer if the printer is spooled or if it is permanently allocated to CCP and is not in use by an NEP or MRT program.

## DISK FILE CONSIDERATIONS

### Models 10 and 12 Considerations

The following considerations apply to the use of disk files by programs running under Model 10 and 12 CCP:

**Multivolume Files:** Multivolume disk files are not supported under CCP.

**Online Files:** All files available to programs on the current run of CCP must be online (mounted on a 5444, 5445, 5448 or 3340) and described by FILE OCL statements at CCP startup time.

**Disk File Names:** If two or more programs that are meant to execute in the same CCP run reference the same disk file, the file name used in each of those programs must be the same (except if symbolic files are used — see *DISKFILE* and *SYMFILE* assignment statements in the *CCP System Reference Manual*, GC21-7588, for additional information).

**Opening and Closing of Files:** All disk files described by FILE and matching DISKFILE statements are opened during CCP startup according to the file organization and access methods specified at assignment time (see *Determining the Disk File Access Value* in this chapter). All files are closed during CCP shutdown.

**Index Sort:** If required, the index sort for index files is done during CCP shutdown.

**Accessing Records Added to an Indexed File:** There are special considerations when using certain file organizations and processing methods during a CCP run. Figure 9-5 illustrates special considerations when records are added to an indexed file during a CCP run.

**Abnormal Termination:** If CCP terminates abnormally (for example, because of a power failure, processor check, or U-halt), CCP does not automatically close files. When an abnormal termination of CCP occurs, the CCP disk file recovery programs (\$CCPRB for a Model 10, \$RINDEX for a Model 12) can be used to close all disk files, and optionally can be used to update the necessary file pointers so that records added during the previous CCP run can be accessed

in subsequent runs. See the *CCP System Operator's Guide*, GC21-7581, for a description of \$CCPRB; the *Model 12 SCP Reference Manual*, GC21-5130, for a description of \$RINDEX.

**Accessing Indexed Sequential Add and Ordered Indexed Files:** Once an indexed sequential add or an ordered indexed file load is done by a program running under CCP, the file cannot be accessed by other programs during the CCP run until the program that first accesses the file closes the file.

**Sharing Files Between Program Levels:** The rules for sharing disk files between program levels of a DPF system are:

CCP Program Level	Other Program Level
Input file	Can retrieve and update.
Update file	Can retrieve.
Add file	Cannot process file.
Load file	Cannot process file.

**Direct Files:** A program running under CCP cannot create a direct file. When using a direct file, at least a *dummy* file (no records) must exist prior to the CCP run.

**Consecutive Updates to the Same Record:** When file sharing is allowed (specified by the FSHARE parameter on the \$EFAC generation statement), do not attempt to update the same record in succession without an intervening read of that same record within the program.

**Indexed Load Files:** If a disk file, defined as indexed-load access (access value IO or IOU), is not loaded during a CCP run, the file exists after CCP shutdown but contains no records.

**Master Index:** A master index for 5444 disk files is built by CCP based on the MSTRINDEX keyword of the DISKFILE assignment statement (see the *CCP System Reference Manual*, GC21-7588). Therefore, you should not specify a master index in your application programs.

**Adding to a file by Two Programs:** Two programs adding to the same file cannot run concurrently.

**Program 1 (P1)**  
 Accesses old records in indexed file, FILEX, and adds new records to the file (access value IRA or IRUA).

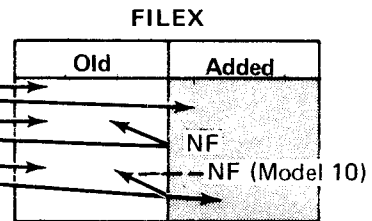
**Program 2 (P2)**  
 Accesses old records (access value IR or IRU).

**Program 3 (P3)**  
 Accesses old records and records added during this CCP run, but does not add records itself (access value IRANA or IRUANA for Model 10 and Model 12 CCP, IRA or IRUA for Model 15 CCP).

(Note: Old records are records in the file prior to CCP startup.)

**Example 1:**

P1 loaded and executing (adding records)  
 P2 loaded while P1 is executing  
 P3 loaded while P1 is executing



NF = not found

**Example 2:**

P1 loaded, adds records, goes to EJ \_ \_ \_  
 P3 loaded, goes to EJ \_ \_ \_  
 P1 reloaded, adds more records, goes to EJ  
 P3 reloaded \_ \_ \_ \_ \_  
 P2 loaded \_ \_ \_ \_ \_  
 Note (Models 10 and 12): Another program of the same type as P1 cannot be loaded while P1 is executing. Two programs of this type cannot be executing concurrently.

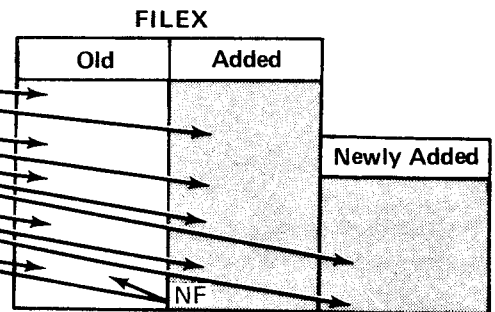


Figure 9-5. Accessing Added Records in Indexed Files Under the CCP.

*Consecutive Output Files:* If a file is defined with an access value CO (consecutive output), the program loading that file will overlay records loaded previously during the same CCP run. For example, suppose program 1 loads file 1 and goes to end of job. If program 1 is loaded again, any records added during the second execution will overlay records loaded during the first execution.

*Update Files:* If one program in an assignment set processes a file in update mode, that file is treated as an update file at CCP startup. Therefore, the following could occur:

- Program 1 processes file 1 as a direct update (DU).
- Program 2 processes file 2 as direct update.
- At CCP startup, file 1 and file 2 are made identical by a FILE statement (// FILE NAME-FILE1, . . . , LABEL-FILE2).

In this case, CCP startup cannot open file 2. Therefore, if different programs are to access the same file in update mode, they must both reference that file by the same name.

#### **Model 15 Considerations**

The following considerations apply to the use of disk files by programs running under Model 15 CCP:

*Multivolume Files:* Multivolume disk files under CCP are not supported by Program Number 5704-SC1. Only random access to indexed or direct multivolume disk files are supported by Program Number 5704-SC2.

*Online Files:* All files available to programs on the current run of CCP must be online (mounted on a 5444, 5445, 3340, or 3344) and described by FILE OCL statements at CCP startup time.

*Disk File Names:* Different names can be used for the same file if the names are related to the file label by using the NAME and LABEL parameters of the FILE OCL statement (see *IBM System/3 Model 15 System Control Programming Reference Manual*, GC21-5077, if operating under 5704-SC1; or *IBM System/3 Model 15 System Control Programming Concepts and Reference Manual*, GC21-5162, if operating under 5704-SC2, for additional information).

*Opening and Closing of Files:* All disk files described by FILE and matching DISKFILE statements are opened during CCP startup according to the file organization and access method specified at assignment time (see *Determining the Disk File Access Value* in this chapter). All files are closed during CCP shutdown.

*Index Sort:* The index sort for index files is done by using \$CCPCL while CCP is running. CCP programs can access the file again during the current CCP run if the system operator executes the \$CCPOP facility. For 5704-SC2 the index sort for index files can also be done by using the \$CCPCO utility to close the file from a batch partition while CCP is running. The file can then be reopened with \$CCPCO to continue processing under CCP. See the *IBM System/3 Model 15 CCP System Reference Manual*, GC21-7620, and the *IBM System/3 Model 15 CCP System Operator's Guide*, GC21-7619, for more information on \$CCPCL and \$CCPOP.

For 5704-SC2, the index sort is done if the KSORT option has been specified in the FILES parameter on the PROGRAM statement. The index sort is done only if there are no other users of the file. (NOSHR is assumed).

**Accessing Records Added to an Indexed File:** Programs using certain nonadd disk access methods can access records added to an indexed file during the current CCP run even though the index sort has not been performed. However, the programs using the add access method must either CLOSE (for COBOL and Assembler) the file, or terminate (for RPG II) before the program using the nonadd access method can open and use the file. These nonadd access methods are:

- Consecutive input (CG)
- Consecutive update (CU)
- Direct input (DG)
- Direct input and update (DU)

Unless an index sort is performed, the programs using nonadd access methods (other than those in the preceding list) cannot access records added to an indexed file during the current CCP run. Index sorting can be accomplished in one of the following ways:

- The system operator \$CCPOP/\$CCPCL facility can be used to open/close the file.
- The \$CCPCO program (5704-SC2 only) can be used to open/close the file from a batch partition.
- The KSORT parameter on the PROGRAM statement can be used to cause the index to be sorted when the using program terminates (5704-SC2 only).

Figure 9-5 illustrates special considerations when records are added to an indexed file during a CCP run.

**Abnormal Termination:** If CCP terminates abnormally (for example, because of a power failure, processor check, or U-halt), CCP does not automatically close files. When an abnormal termination of CCP occurs, CCP disk file recovery program \$RINDEX, can be used to close all disk files and, optionally, update the necessary file pointers so that the records added during the CCP run that abnormally terminated can be accessed in subsequent runs. See the *IBM System/3 Model 15 System Control Programming Reference Manual* (5704-SC1), GC21-5077, or the *IBM System/3 Model 15 System Control Programming Concepts and Reference Manual* (5704-SC2), GC21-5162, for a description of the use of the file recovery programs.

**Sharing Files Among Partitions:** When operating under control of 5704-SC1, the rules for sharing disk files between partitions are:

CCP Partition	Other Partition
Load file	Cannot process file.
Input file	Can retrieve and update.
Update file	Can retrieve; can update if CCP update program processing this file is not currently executing.
Add file	Can retrieve or update <i>nonadd</i> records if CCP program adding records is not currently executing. Can retrieve or update <i>added</i> records <ul style="list-style-type: none"> <li>● if CCP program adding records has gone to end of job and the system operator runs the \$CCPCL/\$CCPOP facility; or</li> <li>● if the file is closed and reopened from a batch partition using the \$CCPCO program (5704-SC2 only); or</li> <li>● if the index has been sorted as a result of using the KSORT option on the FILES parameter of the PROGRAM statement (5704-SC2 only).</li> </ul>

When operating under control of 5704-SC2, the rules for sharing disk files using compatible access methods among partitions are determined by the SHARE parameter of the FILE OCL statement and the FILES parameter of the PROGRAM assignment statement. The compatible access methods allowed with file sharing across partitions are defined in the *IBM System/3 Model 15 System Control Programming Concepts and Reference Manual*, GC21-5162. When operating under CCP, the compatible access methods allowed with file sharing across partitions are defined in *Model 15 CCP File Sharing Considerations* in this chapter.

Basically, the SHARE value of the FILE OCL statement has effect only across partitions, while the FILES parameter of the PROGRAM assignment statement has effect within and across partitions. The relation between these parameters is shown in Figure 9-6; the symbol ● indicates that the program will be allowed to start. For example, if a batch program in a partition is using a file with SHARE-NO, then a program (including CCP startup) in another partition using that same file cannot be initiated. If CCP is started with SHARE-NO specified on the FILE OCL statement, that file is still considered shareable within CCP; the SHR/NOSHR value of the FILES parameter in the PROGRAM assignment statement determines whether the file can be shared and a second program initiated.

*Direct Files:* A CCP program can create a direct file if a direct output access file exists in the assignment set.

*Consecutive Updates to the Same Record:* When file sharing is allowed, do not attempt to update the same record in succession without an intervening read of that same record within the program.

*Updating or Adding to Disk Files:* When updating or adding to disk files, do not issue an Accept Input operation between the reading and writing of a disk record if you intend to update or add the record.

*Adding to a File by Two Programs:* Two programs adding to the same file can run concurrently (5704-SC2 only).

*Load Files:* Files defined as a load access (access values IO, IOU, CO, or DO) by CCP cannot be accessed by other partitions until CCP is shut down or, for 5704-SC2 only, the file is closed and reopened from a batch partition by the utility \$CCPCO. The system operator \$CCPCL/\$CCPOP facility does not allow access to load files by other partitions.

*Indexed Load Files:* If a disk file defined as indexed-load access (access value IO or IOU) is not loaded during the CCP run, the file exists in the pack VTOC after CCP shut-down but contains no records.

*Master Index:* If operating under 5704-SC1, a master index for 5444 disk files is built by CCP based on the MSTRINDEX keyword of the DISKFILE assignment statement (see the *CCP System Reference Manual*, GC21-7620). Therefore, you should not specify a master index in your application programs.

**Consecutive Output Files (5704-SC1 Only):** If a file in a FORTRAN IV or CCP/disk sort program is defined with an access value CO (consecutive output), the program loading that file will overlay records loaded previously during the same CCP run. For example, suppose program 1 loads file 1 and goes to end of job. If program 1 is loaded again, any records loaded during the second execution will overlay records loaded during the first execution. More than one CO access can be done to a file and no records are overlaid in RPG II, COBOL, and Assembler programs. Actually, CO is treated as CA access.

**Multivolume Files (5704-SC2 Only):** CCP will support a maximum of 4 volumes. All volumes must be online and located in main data areas. Access to a multivolume file must be random. The valid accesses are DG, DU, IR, IRA, IRU, and IRUA. File sharing a multivolume file remains as currently defined for the specific access used. RPG file specifications required are identical to those required for non-CCP multivolume file descriptions. Any one volume of a multivolume file may be opened as a separate file. A multivolume subset of a multivolume file is restricted to include the first volume; this subset requires the used volumes to be sequential.

**Consecutive Output Files (5704-SC2 Only):** If a file is defined with an access value of CO (consecutive output), the program loading that file will overlay records loaded previously during the same CCP run. For example, suppose program 1 loads file 1 and goes to end of job. If program 1 is loaded again, any records added during the second execution will overlay records loaded during the first execution.

Program Trying to Start With Same File	OCL or Assignment Statement Value		Current File Status								
			Batch (using file)		CCP <sup>2</sup>				CCP <sup>3</sup>		
	SHARE	FILES	YES	NO	YES		NO		YES	NO	
Batch <sup>1</sup>	YES	—	•		•					•	
	NO	—									
CCP	—	SHR	•		•		•			•	•
	—	NOSHR								•	•

<sup>1</sup> During startup, CCP is considered a batch program.

<sup>2</sup> The file is assigned to a user task that is currently executing.

<sup>3</sup> The file is not assigned to any user task that is currently executing.

**Figure 9-6. File Sharing Across Partitions (5704-SC2 only)**

## Model 10 and Model 12 CCP File Sharing Considerations

The following considerations apply when file sharing is allowed under Model 10 and Model 12 CCP (FSHARE parameter on the \$EFAC generation statement):

- Programs running concurrently under CCP can update the same file. CCP protects the block of disk sectors containing the record to be updated until the program releases the block of sectors by writing the block to disk or reading another block. When programs are concurrently updating a disk file, a potential “lockout” condition could occur if a program does not release a block of sectors.

For example, programs that read and update two or more files they share can be locked out of files if the following sequence of events occurs:

1. Program A reads from file 1.
2. Program B reads from file 2.
3. Program A attempts to read from the same part of file 2, but cannot, because program B has not released the sectors. Likewise, if program B attempts to read from the same sectors of file 1 that were read by program A, it cannot, because program A has not released the sectors.

Because of this possibility, you should either code your program so that it does not do consecutive reads from *different* shared files without doing an intervening write to the first file, or if consecutive reads cannot be avoided, you should inform those personnel responsible for CCP assignments that the files should not be shared.

- When updating files, do not attempt to update (write) the same record in succession without doing an intervening read of that same record within the program. If you do, the task will be terminated.
- Programs defined at assignment time with access values IRUANA or IRANA can run concurrently (see *Determining the Disk File Access Value* for the meaning of IRUANA and IRANA).
- Two IRUANA or IRANA programs can access the same file concurrently.
- If a program defined as IRUANA or IRANA attempts to add records, it will be canceled by CCP.

Figure 9-7 shows how two programs attempting to use the same disk file will interact.

## Model 15 CCP File Sharing Considerations

Model 15 CCP allows concurrent access to a file by two or more programs whenever sharing is logically possible. If a program requires exclusive use of a file, NOSHR must be specified for the FILES parameter of the // PROGRAM assignment statement.

CCP allows programs running concurrently to update or do indexed adds to the same disk file. CCP protects the block of sectors containing the record to be updated until the program releases the block of sectors by writing the block to disk or reading another block. CCP protects the add area of an indexed file when an add is requested. The file is not protected if a record-not-found condition occurs. This could cause the following sequence of conditions to occur: A record-not-found condition, followed by an add of that record, which might give a duplicate record condition if the record had been added by another CCP task. When programs are concurrently sharing a disk file, a potential lock-out condition could occur if a program does not release a block of sectors.

Figures 9-8 and 9-9 show how two programs attempting to use the same disk file will interact. The following considerations and exceptions apply to sharing of nonexclusive disk files:

- An ordered indexed load must precede all other access to that file and must be done only once during a CCP run.
- Double buffering is never done when files are being shared. CCP treats any double buffer accesses as single buffer accesses.
- A file processed as unordered indexed load can be shared serially, in time, with other unordered load accesses; however, no other types of file access are allowed during a CCP run.
- Consecutive load and consecutive add accesses cannot be done concurrently to the same disk file and neither access can be done concurrently with any other access of the same file (5704-SC1 only).
- Indexed sequential addition to a file (with or without update) cannot be done concurrently with another access to that file. Indexed sequential addition cannot be preceded by any other indexed add access that actually adds records to the file unless:
  - A CCP program adding records has gone to end of job and the system operator runs the \$CCPCL/\$CCPOP facility; or
  - The file is closed and reopened from a batch partition using the \$CCPCO program (5704-SC2 only); or
  - The index has been sorted as the result of using the KSORT option on the FILES parameter of the PROGRAM statement (5704-SC2 only).



- (5704-SC1 only) Consecutive output (load) to an existing file is treated by CCP as consecutive add, except in FORTRAN IV, or CCP/SORT. Therefore, consecutive output can be done more than once to the same file during a CCP run, with results the same as consecutive add. In FORTRAN IV, consecutive output can be done only once to a file during a CCP run.
- An indexed file that has been added to, but has not yet gone through a key sort, can contain records that cannot be accessed by non-add indexed data managements (see *Disk File Considerations*).
- Unlike indexed random add data managements (IRA and IRUA), indexed sequential add data managements (ISA and ISUA) do not search the area of added records. Therefore, a file opened ISA or ISUA can later be opened IRA or IRUA, but not vice versa. To open a file ISA or ISUA after a file has been opened IRA or IRUA the system operator (1) must run the \$CCPCL/\$CCPOP facility, or (2) the index must have been sorted as the result of using the KSORT option on the FILES parameter of the PROGRAM statement (5704-SC2 only), or (3) the file must have been closed and reopened from a batch partition using the \$CCPCO program (5704-SC2 only).
- Programs that read and update (and/or add) to two or more files they share can be locked out of a file if the following sequence of events occurs:
  1. Program A reads from file 1.
  2. Program B reads from file 2.
  3. Program A attempts to read from the same part of file 2, but cannot, because program B has not released the sectors. Likewise, if program B attempts to read from the same sectors of file 1 that were read by program A, it cannot, because program A has not released the sectors.
- If an MRT-NEP program (5704-SC1 only) uses file sharing and issues an accept input with no outstanding invite inputs, the disk file sectors protected from file sharing after the disk file was read are no longer protected. Under these conditions, if you try to add or update files, you get a disk error (invalid address). Before attempting to update files accessed by an MRT-NEP following an accept input, you must first retrieve the record to be updated.
- If an MRT-NEP program (5704-SC2 only) uses file sharing and issues an accept input with no outstanding invite inputs, all the disk sectors protected will remain protected. To release any protected sectors, a flush of the buffers should be done before the accept input operation.
- When updating files, do not attempt to update (write) the same record in succession without doing an intervening read of that same record within the program. If you do, the user task terminate with a 0H halt. The program logic should be reviewed and corrected.

Because of this possibility, you should either code your program so that it does not do consecutive reads from *different* shared files without doing an intervening write to the first file, or if consecutive reads cannot be avoided, you should inform those personnel responsible for CCP assignments that the files should not be shared.

File Already in Use (Open) As: (Opened at Startup) As: Attempting to use (Open) File as:	CA		IS			ISL		IA		ISA		DU	DG
	CO	CU	CG	IO	IOU	IR	ISU	IRA	ISUA	IRUA	ISUA		
CO, CA	S	S	S	N	N	N	N	N	N	N	N	S	S
CU	S	Y	Y	N	N	N	N	N	N	N	N	Y	Y
CG	S	Y	Y	S	N	Y	Y	Y	S	Y	Y	Y	Y
IO	N	N	N	N#	N	N	N	N	N	N	N	N	N
IOU	N	N	N	N	S	N	N	N	N	N	N	N	N
IS, ISL, IR, IRU	N	N	Y	S	N	Y	Y	Y	S	Y	Y	Y	Y
ISU, ISUL	N	N	Y	S	N	Y	Y	Y	S	Y	Y	Y	Y
IA, IRA, IRUA	N	N	Y	S	N	Y	Y	Y#	S	Y	Y	Y	Y
ISA, ISUA	N	N	S	S	N	S	S	N#	N#	S	S	S	S
DU	S	Y	Y	S	N	Y	Y	Y	S	Y	Y	Y	Y
DG	S	Y	Y	S	N	Y	Y	Y	S	Y	Y	Y	Y

**KEY**

Y – Access permitted.

Y# – If any combination of IRA or IRUA is specified in two or more // PROGRAM statements, they will not run concurrently.

N – Access not permitted.

N# – A second access is not permitted because of DSM termination key sort first. An ordered index load must precede all other access done to that file and must be done only once during a CCP run.

S – Serialized access. First access must CLOSE within program before second may OPEN.

Figure 9-7. Sharing Access to Disk Files (Between CCP Tasks)—Model 10 and Model 12 CCP

File Already in Use (Open) As:  Attempting to use (Open) File as:												
	CA	CU	CG	IO	IOU	IS ISL IR	ISU ISUL IRU	IA IRA IRUA	ISA ISUA	DO (New File)	DU	DG
CO, CA	S	S	S	N	N	N	N	N	N	N	S	S
CU	S	Y	Y	N	N	N	N	N	N	Y	Y	Y
CG	S	Y	Y	S	N	Y	Y	Y	S	Y	Y	Y
IO	N	N	N@	N#	N	N@	N@	N@	N@	N	N@	N@
IOU	N	N	N	N	S	N	N	N	N	N	N	N
IS, ISL, IR, IRU	N	N	Y	S	N	Y	Y	Y	S	N	Y	Y
ISU, ISUL	N	N	Y	S	N	Y	Y	Y	S	N	Y	Y
IA, IRA, IRUA	N	N	Y	S	N	Y	Y	Y	S	N	Y	Y
ISA, ISUA	N	N	S	S	N	S	S	N#	N#	N	S	S
DO	N	Y*	Y*	N	N	N	N	N	N	Y*	Y*	Y*
DU	S	Y	Y	S	N	Y	Y	Y	S	Y	Y	Y
DG	S	Y	Y	S	N	Y	Y	Y	S	Y	Y	Y

- Y — Access permitted.
- Y\* — Direct output to an old or previously used file is treated as direct update.
- N — Access not permitted.
- N# — A second access is not permitted unless the file has been closed and reopened by the system operator.  
An ordered index load must precede all other access done to that file and must be done only once during a CCP run.
- N@ — IO must be first access of file if used.
- S — Serialized access. First access must CLOSE within program before second may OPEN.

Figure 9-8. Sharing Access to Disk Files (Between CCP Tasks)—Model 15 CCP (5704-SC1)

Attempting to Use (Open) File as:	File Already in Use (Open) As:										
	CO	CA	CU	CG	IO	IOU	IS ISL IR IRU ISU ISUL	IA IRA IRUA	ISA ISUA	DO	DU DG
CO	N	S	S	S	N	N	N	N	N	N	S
CA	S	S	Y	Y	N	N	N	N	N	N	Y
CU	S	Y	Y	Y	N	N	Y	Y	Y	Y <sup>3</sup>	Y
CG	S	Y	Y	Y	S	N	Y	Y	Y	Y <sup>3</sup>	Y
IO	N	N	N	N <sup>5</sup>	N <sup>4</sup>	N	N <sup>5</sup>	N <sup>5</sup>	N <sup>5</sup>	N	N <sup>5</sup>
IOU	N	N	N	N	N	S	N	N	N	N	N
IS	N	N	Y	Y	S	N	Y	Y	Y <sup>1</sup>	N	Y
ISL	N	N	Y	Y	S	N	Y	Y	Y <sup>1</sup>	N	Y
IR	N	N	Y	Y	S	N	Y	Y	Y <sup>1</sup>	N	Y
IRU	N	N	Y	Y	S	N	Y	Y	Y <sup>1</sup>	N	Y
ISU	N	N	Y	Y	S	N	Y	Y <sup>2</sup>	Y	N	Y
ISUL	N	N	Y	Y	S	N	Y	Y <sup>2</sup>	Y	N	Y
IA	N	N	Y	Y	S	N	Y	Y	S	N	Y
IRA	N	N	Y	Y	S	N	Y	Y	S	N	Y
IRUA	N	N	Y	Y	S	N	Y	Y	S	N	Y
ISA	N	N	Y	Y	S	N	Y <sup>1</sup>	N <sup>4</sup>	N <sup>4</sup>	N	Y <sup>1</sup>
ISUA	N	N	Y	Y	S	N	Y <sup>1</sup>	N <sup>4</sup>	N <sup>4</sup>	N	Y <sup>1</sup>
DO	N	N	Y <sup>3</sup>	Y <sup>3</sup>	N	N	N	N	N	Y <sup>3</sup>	Y <sup>3</sup>
DU	S	Y	Y	Y	S	N	Y	Y	Y	Y <sup>3</sup>	Y
DG	S	Y	Y	Y	S	N	Y	Y	Y	Y <sup>3</sup>	Y

KEY

Y – Access permitted.  
N – Access not permitted.  
S – Serialized access. First access must CLOSE within program before second may OPEN.

<sup>1</sup>Files accessed using the index sequential add access method under which adds have been made cannot be reopened using ISA unless the key sort has occurred. If any adds have been made and the key sort has not occurred, only random access methods can be used to add to the file.  
<sup>2</sup>The added records are not accessed if random add has occurred.  
<sup>3</sup>Direct output to an old or previously used file is treated as direct update.  
<sup>4</sup>A second access is not permitted unless the file has been CLOSED and OPENED by the system operator. Files defined as load access by CCP cannot be accessed by other partitions until CCP is shutdown. The system operator facility \$CCPCL/\$CCPOP does not allow access to the load files by other partitions.  
<sup>5</sup>If ordered indexed load access method is used, it must be the first access method used to access the file.

Figure 9-9. Sharing Access to Disk Files (Between CCP Tasks and Batch)—Model 15 CCP (5704-SC2)

## Determining the Disk File Access Value

When specifying the FILES keyword for the PROGRAM assignment statement (see *CCP System Reference Manual*), an access value must be given for each disk file. The access value indicates the mode of access of a disk file used by an application program. You should inform the person responsible for completing the CCP assignments of the type of organization and the access value for each disk file you use in your program. You can determine the access value for RPG II, COBOL, and FORTRAN disk file processing from Figures 9-10, 9-11, and 9-12. The access value symbols and their meanings are:

- CO — Consecutive Output
  - CG — Consecutive Input
  - CU — Consecutive Update
  - CA — Consecutive Add
  - DG — Direct Input
  - \*DGA — Direct Input Binary Relative Record Numbers
  - DU — Direct Input and Update
  - \*DUA — Direct Input and Update, Binary Relative Record Numbers
  - DO — Direct Output (valid only for Model 15 CCP)
  - 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 Sequential 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 — Records which were added to the file by another program that has gone to end of job in the current CCP run will be read using Indexed Random Input and Add (IRA), but no records will be added.
  - \*\*IRUANA — Records which were added to the file by another program that has gone to end of job in the current CCP run will be read and possibly updated using Indexed Random Input, Update, and Add (IRUA), but no records will be added.
- \*Valid for Model 10 and Model 12 CCP. For Model 15 CCP, use DG or DU.
- \*\*Valid for Model 10 and Model 12 CCP. For Model 15 CCP, use IRA or IRUA.

The valid combinations of disk file organization (specified in the // DISKFILE assignment statement) and method of access (specified in the // PROGRAM assignment statement) are shown in the following chart.

Method of Access (// PROGRAM)	Organization (// DISKFILE)		
	Sequential	Indexed	Direct
CO & CA	YES	NO	NO
CU	YES	NO	YES
CG	YES	YES	YES
Indexed Access Methods	NO	YES	NO
DO	NO	NO	YES
DU and DUA	YES	YES	YES
DG and DGA	YES	YES	YES

**INDEXED FILES**

Type of Processing

Sequential\*  
by Key, no ADD  
by Key, with ADD  
by Key, with ADD  
by Key, with ADD  
by Key, no ADD  
by Key, with ADD  
by Key, with ADD  
by Limits (RAF)  
by Limits (RAF)  
by Limits (RAF)  
by Limits (RAF)  
by Key, no ADD  
by Key, with ADD  
by Key, no ADD  
by Key, with ADD  
by Key, with ADD  
by Key, with ADD  
by Limits (RAF)  
by Limits (RAF)

Sequential\*

Random  
by ADDRUT  
by ADDRUT  
by ADDRUT  
by ADDRUT

Load  
Unordered  
Ordered  
Add records only  
{ ADD only

Figure 9-10 (Part 1 of 4). Disk Files Access Values for RPG II

**File Description Specification**

Line	Filename			File Type				Mode of Processing				Device	Symbolic Device	File Addition/Unordered			Access Value	
	Form Type	1-8	9-14	File Designation	End of File	Sequence	File Format	Record Length	Length of Key Field or Record Address Field	Type of File	Organization			Overflow Indicator	Key Field Starting Location	Number of Tracks for Cylinder Overflow		Type Rewind
01	F			R/U/C/D														IS
02	F			P														IS
03	F			S														ISA
04	F			P														ISA
05	F			S														ISA
06	F			U/P														ISU
07	F			S														ISU
08	F			U/P														ISUA
09	F			S														ISUA
10	F			P														ISL
02	F			S														ISL
03	F			U/P														ISUL
04	F			S														ISUL
05	F			U/P														IS
06	F			S														ISA
07	F			U/P														ISA
08	F			S														ISUA
09	F			U/P														ISL
10	F			S														ISL
02	F			U/P														ISUL
03	F			S														IS
04	F			U/P														ISA
05	F			S														ISA
06	F			U/P														ISUA
07	F			S														ISL
08	F			U/P														ISL
09	F			S														ISUL
10	F			U/P														IS
02	F			U/C														IR
03	F			S														IRA
04	F			U/P														IRU
05	F			S														IRUA
06	F			U/P														Models 10 and 12
07	F			S														Model 15
08	F			U/P														DGA
09	F			S														DGA
10	F			U/P														DUA
02	F			S														DUA
03	F			U/P														DUA
04	F			S														IOU
05	F			U/P														IO
06	F			S														IA

\* Sequential processing by key or limits must use the file index, which is always arranged in ascending sequence. When an indexed file is processed record-by-record from beginning to end, the file is processed through the index using the sequential-by-key method.

**Notes:**

1. DISK, DISK40, or DISK45 can be specified as the device (columns 40-46).
2. If your system has 3340s attached, DISK is not a valid entry in columns 40-46 since index files are not supported on the simulation areas.

### File Description Specification

Line	Form Type	Filename										File Type										Mode of Processing										Symbolic Device										Name of Label Exit										File Addition/Unordered Number of Tracks for Cylinder Overflow										Access Value																																					
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60		61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97
01	F																																																																																																		
02	F											IP										R										DISK																														CG																																					
03	F											IS										R										DISK																														CG																																					
04	F											IT										R										DISK																														CG																																					
05	F											UP										R										DISK																														CU																																					
06	F											US										R										DISK																														CU																																					
07	F											ID										R										DISK																														CG																																					
08	F											UD										R										DISK																														CU																																					
09	F											IC										R										DISK																														DG																																					
10	F											UC										R										DISK																														DU																																					
11	F											IP										R										DISK																														Models 10 and 12																																					
12	F											IS										R										DISK																															DGA																																				
13	F											UP										R										DISK																															DGA																																				
14	F											US										R										DISK																															DUA																																				
15	F											O										R										DISK																															DUA																																				
16	F											O										R										DISK																														CO																																					
17	F																																																																							Model 15																											
18	F																																																																								DG																										
19	F																																																																								DG																										
20	F																																																																																	DU																	
21	F																																																																																	DU																	
22	F																																																																																	CA																	

Note: DISK, DISK40, or DISK45 can be specified as the device (columns 40-46).

Figure 9-10 (Part 2 of 4). Disk Files Access Values for RPG II





**RECORD ADDRESS FILES**

**Figure 9-10 (Part 4 of 4). Disk Files Access Values for RPG II**

Line	File Name		File Type		Mode of Processing		Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Exit M	File Addition/Unordered			Access Values	
	End of File		Length of Key Field or Record Address Field		Record Address Type	Type of File Organization or Additional Area						Extension Code E/L	Number of Tracks for Cylinder Overflow	Tape Rewind		File Condition U1-UG
	F/V/S/M/D	A/D	F	R												
0 2	F	I/R	F	3	I/T	EDISK									CG	
0 3	F	I/R	F	3	I/T	EDISK									CG	
0 4	F	I/R	F	3	I/T	EDISK									CG	
0 5	F	I/R	F	3	I/T	EDISK									CG	
0 6	F	I/R	F	3	I/T	EDISK									CG	
0 7	F	I/R	F	3	I/T	EDISK									CG	
0 8	F	I/R	F	3	I/T	EDISK									CG	
0 9	F	I/R	F	3	I/T	EDISK									CG	
1 0	F	I/R	F	3	I/T	EDISK									CG	

\* ADDRUT files may be associated with indexed, sequential, or direct disk files.

\* Record address files containing record key limits may only be associated with indexed disk files, but may be a disk or MFCU file.

Note: DISK, DISK40, or DISK45 can be specified as the device (columns 40-46).

COBOL LANGUAGE STATEMENTS					Access Value
ASSIGN	ACCESS	KEY	OPEN	I/O Verbs	
DA-544x-S-name	SEQUENTIAL		OUTPUT		CO
UT-544x-S-name	SEQUENTIAL		OUTPUT		
DA-544x-S-name	SEQUENTIAL		INPUT		CG
UT-544x-S-name	SEQUENTIAL		INPUT		
DA-544x-R-name	SEQUENTIAL		INPUT		
DA-544x-S-name-U	SEQUENTIAL		I-O		CU
DA-544x-R-name	RANDOM	ACTUAL	INPUT		DG
DA-544x-R-name-U	RANDOM	ACTUAL	I-O		DU
DA-544x-I-name	SEQUENTIAL	RECORD	INPUT		IS
DA-544x-I-name	SEQUENTIAL	RECORD	INPUT	START ...	ISL
		NOMINAL			
DA-544x-I-name-U	SEQUENTIAL	RECORD	I-O	REWRITE ...	ISU
DA-544x-I-name-U	SEQUENTIAL	RECORD	I-O	START ...	ISUL
		NOMINAL		REWRITE ...	
DA-544x-I-name	RANDOM	RECORD	INPUT		IR
		NOMINAL			
DA-544x-I-name-N	RANDOM	RECORD	I-O	REWRITE	IRU
		NOMINAL			
DA-544-x-I-name-U	RANDOM	RECORD	I-O	WRITE	IRUA
		NOMINAL			
DA-544-x-I-name	SEQUENTIAL	RECORD	OUTPUT		IO

*Note:* On systems with 3344 disk files, DA-544x can be replaced with DA-3340 for consecutive or direct files; DA-5445 can be replaced with DA-3340 for indexed files.

Figure 9-11. Disk File Access Values for COBOL

FORTRAN IV LANGUAGE STATEMENTS		Access Value
Device Option Statement	Input/Output Statements	
// SEQ40 UNITNO-n, ... // SEQ44 UNITNO-n, ... // SEQ45 UNITNO-n, ...	READ (...) ... WRITE (...) ...	CO*
// DAD40 UNITNO-n, ... // DAD44 UNITNO-n, ... // DAD45 UNITNO-n, ...	READ (...) ... WRITE (...) ... DEFINE FILE ... FIND (...)	DUA

\*If the program only reads from the file, access value CG may be used. The file could then be sharable.

Figure 9-12. Disk File Access Values for FORTRAN IV



After you have written a communications program to run under the CCP, you will usually want to test the program before using it in a production environment. If the terminal devices to be used by the program are available, you can test the program according to the following general procedure:

1. Perform the procedures for program preparation described in *Chapter 9. Program Preparation*.
2. Perform the CCP Startup procedures given in the *CCP System Operator's Guide*. Include // FILE statements among the OCL statements at Startup for any test files that were described to the CCP at assignment time.
3. Enter a request for the program from a terminal or from the system operator's console. (See *Program Request* in either the *CCP System Operator's Guide* or the *CCP Terminal Operator's Guide*.)
4. Enter test input data as required by the program.
5. If test files are to be printed using the Disk Copy/Dump utility, the CCP must be shut down prior to using the utility, since the utility can neither be executed under the CCP nor in the opposite level from the CCP in a DPF (dual partition) system, as all test output data may not be available to it. (The Disk Copy/Dump utility program is described in *IBM System/3 Model 10 Disk System Control Programming Reference Manual*, GC21-7512; in *IBM System/3 Model 12 System Control Programming Reference Manual*, GC21-5130; and in *IBM System/3 Model 15 System Control Programming Reference Manual*, GC21-5077, if operating under 5704-SC1, or *IBM System/3 Model 15 System Control Programming Concepts and Reference Manual*, GC21-5162, if operating under 5704-SC2.
6. Evaluate the program by inspecting test files and other output from the program.

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



The PORTLINE function under CCP allows communications between two System/3 Model 15Ds via a BSCA or a BSCC line. This function is similar to the support for channel-connected systems via SIOC (see *IBM System/3 Model 15D Channel Connected Systems Program Reference and Logic Manual*, GC21-5199).

The CCP program support establishes each system as a command mode terminal to the other system. This means that each system can load programs in the other system or pass transactions to already loaded programs in the other system. User programs (RPG II, COBOL, FORTRAN, or Assembler) communicate with the other system in much the same way that they communicate with 3270 terminals.

CCP divides the line into logical ports. Each port is treated as a terminal by both systems. Thus, each port is used by an application program as though it were a terminal. The application program need not be aware that other applications may also be using the line.

After CCP startup, each port is automatically defined as signed-on and in queue mode. A port is further defined as being acquirable or nonacquirable. Each port that is capable of being acquired by one system must be paired with a nonacquirable port by the other system. An application program establishes a communication link with the other system by either acquiring or requiring an acquirable port. An acquirable port may be acquired either by specifying a specific port name or by requesting any general port using a new acquire operation code. This new operation code using a generic key allows an application program to acquire from the command processor any available port that is not being used by another task. A required port is assigned by specifying the port name on the TERMS parameter in the PROGRAM statement.

After an application program has acquired a port, some CCP commands may be sent, via the port, to the other system, provided that the other side of the port is owned by the command processor. The other side of the port is owned by the command processor unless an application program is running and using the port. Allowable commands are program requests, /FILE, and /MSG commands. If the paired port is owned by an application program at each system, the two programs may communicate via the port.

Features of the CCP program support:

- Two System/3 Model 15Ds communicate via the BSCA or BSCC line.
  - CCP divides the line into logical ports.
  - Each port is treated as a terminal.
  - An application program need not be aware that the other applications may also be using the line.
  - Ports are defined as being acquirable or nonacquirable.
  - Each port that is capable of being acquired by one system must be paired with a nonacquirable port by the other system.
- Note:* It is the user's responsibility to ensure that the ports are paired. CCP will not diagnose this error.
- Prior to execution a program may require a port as defined on the TERMS parameter on the PROGRAM statement at assignment.
  - During execution a program may request a port with the new acquire operation code (either generic or specific).
  - After acquiring a port, a program may issue some CCP commands to the command processor of the other system.
  - Allowable commands are program requests and /FILE and /MSG commands.
  - After a program request is issued, the application may communicate with the requested program.
  - The programs must be compatible. For example, for every PUT instruction executed by one application, the other must execute a GET instruction or invite operation.

A user application might perform the following steps:

1. Acquire a port.
2. Issue a program request.
3. Communicate with the requested program.
4. Release the port or terminate (done by EOJ on termination).

Possible types of applications:

- File transmission (may be better under batch)
- Inquiry or conversational
  - Request a record from a file.
  - Update or add to a file when the file exists in the other system.
  - Pass a key with the program request and read the record found via the channel.
- Concentrator—a program to handle only the terminal input/output operations and passing the data to and from an application in the second system.

#### Required Ports

The user has the option to require a specific terminal via the PROGRAM statement. He may do the same and require a port by specifying a specific acquirable port by name. But if that port is being used, the application must wait until the port is available before the application may execute even though there may have been other ports available. The user may, however, require any available port for a specific line by specifying the port prefix defined for that line instead of the specific port name.

Example:

```
// PORTLINE MODE-PRIMARY,BLKL-280,  
      MAXMSG-280,AQPORT-3,NAQPORT-3,  
      PORTPRFX-SIOC  
  
// PROGRAM NAME-APGM1, TERMS-SIOC (note 1)  
  
// PROGRAM NAME-APGM2, TERMS-SIOC01 (note 2)
```

Notes:

1. Require any available and acquirable port.
2. Specifically require port SIOC01.

Program APM1, when given control, will have any one of the three ports. The port that is allocated to this program has its TUB address stored in the user's TCB.

When the user does an I/O operation via the port he uses the name given on the PORTPRFX parameter (SIOC, in the above example) to refer to the actual port given in the program. CCP recognizes the name as a generic port and uses the TUB given in the TCB.

This is similar to the use of a blank terminal name for SRT (single requesting terminal) programs. A blank terminal name means the program is communicating with the requestor of the SRT.

Program APM2, when given control, can only have port SIOC01 and must wait if it's not available.

*Note:* After a SHUTDOWN command has been entered and accepted, the allocation of required ports to user programs will not be permitted.

#### OPERATION CODES

The following user operation codes are available for communication over logical ports:

ACC	Accept input
ACQ, CMD	Acquire a command-mode, non-PRUF terminal
ACQ, GEN	Acquire a command-mode, non-PRUF terminal by generic name
GET	Normal get operation
GTA	Get terminal attributes
INV	Invite input
PCR	Port command request
PTG, MSG	Put (message)-then-get
PUT, MSG	Put message
REL	Release the terminal and the communication line
SPI	Stop invite input

Refer to *operations* in the index for a further explanation of these codes.



### Portline Exception to Get Terminal Attributes

The purpose of this operation is to determine the attributes of a specified terminal.

#### Operation Code

Hex	Dec	RPG II	Meaning
0008	0008	bbbH	Get Terminal Attributes

The only consideration for use with PORTLINE is in position 2 (Terminal class) of the information returned in the record as shown in the following chart. See *get terminal attributes* under *operations* in the index for additional information.

**Position 2—Terminal Class:** Position 2 contains a single character representing the class of terminal as follows:

EBCDIC Character	Terminal Class
0	CONSOL 5741 Printer/Keyboard (Models 10 and 12), CRT/Keyboard (Model 15)
1	MLTA terminal other than 1050
2	1050
3	3277M1 (480-character), 3284M1, or 3286M1
4	3277M2 (1920-character), 3284M2, or 3286M2
5	3275M1 (480-character), with or without 3284M3
6	3275M2 (1920-character), with or without 3284M3
7	3735
8	Another computer system (CPU) on a BSCA or BSCC line
9	3741 Data Station, 5231 Controller Model 2
A	Acquirable port
B	Nonacquirable port



## Appendix A: CPU to CPU Considerations

The CCP is adaptable to the *concentrator* and *subhost* functions in a teleprocessing environment. A combination of the two functions can be attained through application programming under control of the CCP.

In a concentrator role, the CCP can gather data from a variety of terminals and forward the data to a host system (System/360 or System/370) for further processing. In the reverse direction, the CCP can collect data from the host system and distribute the data to a variety of terminals. In the concentrator role, the CCP supports user-written application programs to handle the message traffic in a *store and forward* manner. (*Store and forward* is defined as the interruption of data flow from the originating terminal to the designated receiver by storing the information enroute and forwarding it at a later time.)

In a subhost role, the CCP can gather data from a variety of terminals, act intelligently on the data, and provide responses in most cases. In those cases where additional data is required, the request can be forwarded to the host system for further processing.

The desirability of the concentrator function lies in reducing line and network costs and in retaining a consistent, common host interface in a changing terminal environment. The subhost function has the same advantages as the concentrator function plus the advantages of distributed processing capability, fast response time, and back-up capability (possibly degraded) in those occasions when the host system is unavailable. In the concentrator and subhost roles, the CCP would most likely be attached to the host as a point-to-point or multipoint tributary station.

The previous paragraphs have described the CCP in an online (nonswitched) connection to a host CPU. The CCP can also function in a switched environment where the CCP acts as a host during certain periods and, when necessary, functions as a subhost, connecting to a host CPU for special transmissions or receptions such as transmission of batched data. The connection can be via a switched line defined as auto or manual call and auto or manual answer. If the host transmissions can be scheduled properly with the normal online terminal operations, an RPQ is available which provides a switching capability between a dial network for remote communications and multipoint control through the EIA local feature (IBM World Trade Corporation EIA/CCITT). If two adapters are used, each can have this RPQ. Use of this RPQ requires separate CCP startups and separate assignment sets for dial and multipoint operation. (See index entry *switched lines* for further information.)

## ATTACHMENT CONFIGURATIONS

CCP can be attached to the following IBM systems that have the indicated hardware and programming support:

System	Communications Adapter	Programming Support	Remote Processor Configurations
System/3 Model 6	BSCA	RPG II Telecommunications Feature (data mode support only)	Point-to-point Switched Multipoint tributary
System/3 Models 4, 10, 12, or 15	BSCA	RPG II Telecommunications Feature (data mode support only)	Point-to-point Switched Multipoint tributary
	BSCA	Multiline/ Multipoint <sup>2</sup> (Features 6030-6031)	Point-to-point Switched Multipoint tributary Control station
	BSCA	CCP	Point-to-point Switched Multipoint tributary Control station
	BSCC <sup>3</sup>	CCP	Point-to-point Control station
System 7 <sup>1</sup>	BSCA (RPO#S40076)	Programming support for RPO#S40076 and applications programming support	Point-to-point Switched Multipoint tributary
	MLTA (RPG#S40065)	Programming support for RPO#S40065 and applications programming support	Point-to-point Switched Multipoint tributary (see Note)
System/360 and System/370	BSCA	BSCA Programming support (BTAM)	Point-to-point Switched Control station

<sup>1</sup>System/7 with MLTA functions as a 2740 Model 1 with the following features:  
 Checking—leased line.  
 Dial with checking—switched line.  
 Station control with checking—multipoint tributary.

<sup>2</sup>The Model 4 does not support MLMP.

<sup>3</sup>5704-SC2 only.

## PROGRAMMING CONSIDERATIONS

This section provides an application-to-application understanding of programming between CCP and an attached CPU.

The programming of an attached system for linkage to CCP can differ greatly, depending upon the type of telecommunications support provided by the attached system. On the other hand, a particular application program running under CCP communicating with a CPU can be easily used with another CPU, with little or no change.

For BSC attachments, the programmer of the attached CPU should be familiar with the Multiline/Multipoint (MLMP) support for an understanding of the base CCP data management and IOCS. (See the publication *IBM System/3 Multiline/Multipoint Binary Synchronous Communications Reference Manual*, GC21-7573.)

For MLTA attachments, the programmer of the attached CPU should be familiar with the MLTA Input/Output Control System for an understanding of the base CCP data management and IOCS. (See the publication *IBM System/3 Multiple Line Terminal Adapter RPQ Program Reference and Component Description Manual*, GC21-7560.)

The CCP application programmer must be aware that CCP always considers the attached CPU to be a terminal. As such, the attached CPU must be identified as either a command mode or a data mode terminal. The attached CPU should be designated a command mode terminal if it is to initiate CCP application programs without CCP system operator intervention. If program initiation by the attached CPU is not required, it should be designated a data mode terminal. In this case, the CCP system operator or the operator of a CCP command mode terminal initiates the program.

#### Command Mode

As a command mode terminal, the attached CPU is expected to provide certain commands in order to initiate a CCP application program and can optionally provide other commands to request special CCP control functions. The attaching CPU programming support must support variable length messages in order to handle CCP commands and responses. Messages must be non-transparent, non-ITB, consisting of a single block followed by EOT.

The following commands can be issued by the attached CPU to the CCP:

/ON [password]	(required)
/Q	(optional)
/NOQ	(optional)
/FILE	(dictated by requirements of the CCP application program)
Program Request	(required)
/OFF	(optional)
/MSG	(optional)
/NAME	(optional)
/RUN	(optional)
Data Mode Escape	(optional)
/RELEASE	(optional)

Command	Good	Error
1. /ON	A01, E03	E01, E02, E04, (R01, R02), E27
2. /Q	A03	E01, E02, E05, (R01, R02), E27, E29
3. /NOQ	A04	E01, E02, E05, (R01, R02), E27, E29
4. /FILE	A06	E01, E02, E17, E18, E19, E20, E21, E26 (R01, R02), E27, E29
5. /OFF	A10	E01, E02, E08, E09, E10, (R01, R02), E27
6. /NAME	A05	E01, E02, E22, E23, E24, E25, (R01, R02), E27
7. /MSG	A02	E01, E02, E06, (R01, R02), E27
8. /RELEASE	A08	E01, E02, E07, (R01, R02), E27
9. /RUN	A09	E01, E02, E16, (R01, R02), E27
10. Program request	User defined	E01, E02, E11, E12, E13, E14, E15, (R01, R02), R03, R04, R05, R06, R07, R08, R09, R10, R11, R12, R13, R14, R15, R16, R17, R18, R19, R20, R21, R23, R24, R25, R26, R27, R28, R29, R30, R31, E27, E28, E29,
11. Data mode escape sequence	A07	E30, E31, E32, E33, E34, E35, (R01, R02)

#### Notes:

1. R02 cannot be received unless R01 has been received.
2. If message or response prefix is in italics, it can be received only under Model 10 and Model 12 CCP. If a prefix is in bold type, it can be received only under Model 15 CCP.
3. Refer to the *IBM System/3 Communications Control Program Messages Manual*, GC21-5170, for a complete description of the commands and messages.

Figure A-1. Command Mode Message and Response Prefixes

Attached CPU Application Program

CCP

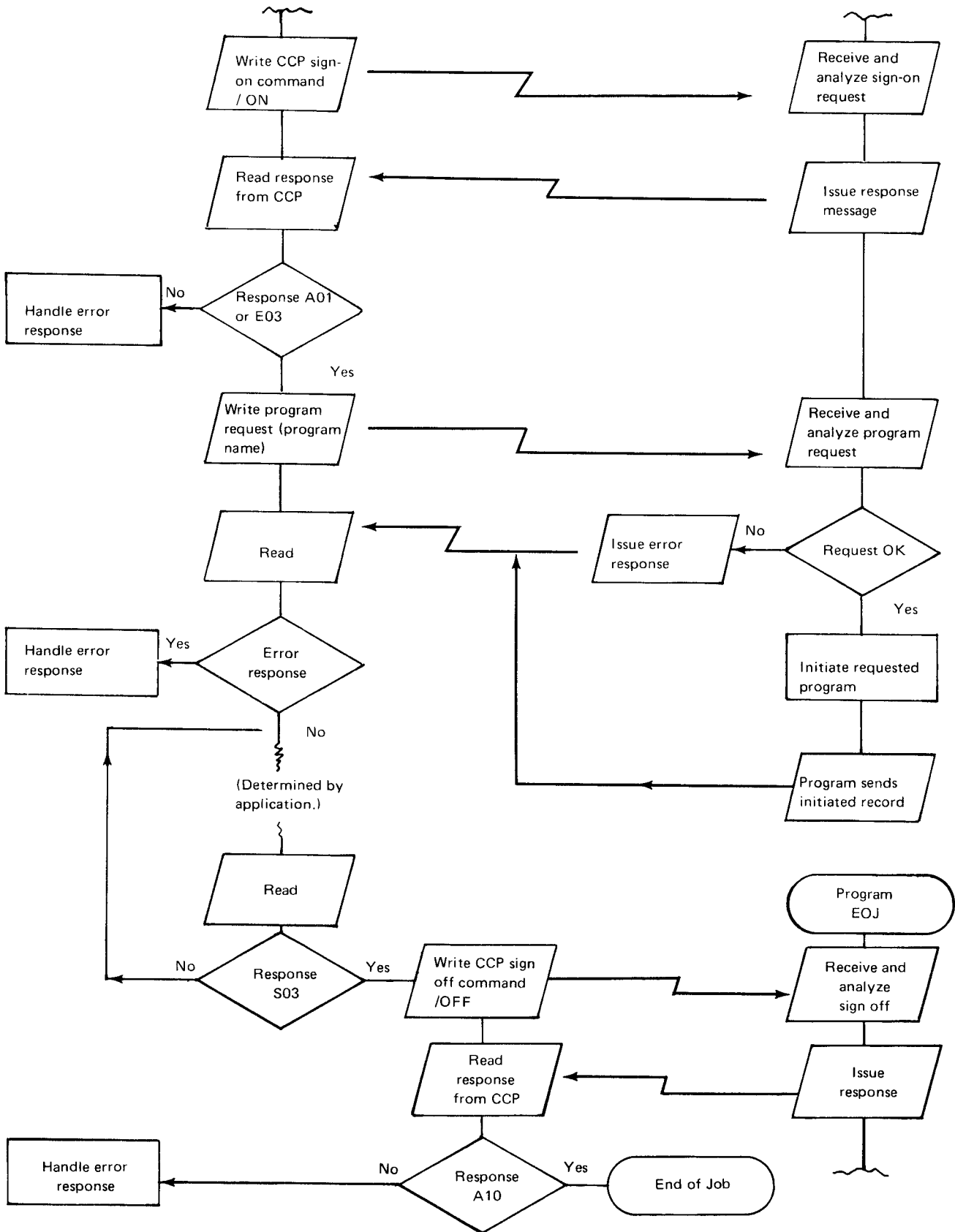


Figure A-2. Procedure for Issuing a Program Request from an Attached Command Mode CPU

Descriptions of the functions of these commands are given in the *CCP Terminal Operator's Guide*, GC21-7580. The *CCP Messages Manual*, GC21-5170, also describes the responses given by CCP to these commands. The attached CPU application program must analyze these responses and take appropriate action. All messages with an S prefix are suppressed by CCP when communicating with an attached CPU.

Figure A-1 shows the CCP commands and the prefixes of the possible responses to each command. Figure A-2 illustrates the general procedure that should be followed by the attached CPU application program in signing on to CCP, issuing a program request, and signing off from CCP.

If a program running in a command mode CPU on a switched line has sent a /ON command, that program must issue a /OFF command to cause the line to disconnect normally. The /OFF command causes CCP to issue a response message to the program and disconnect the CPU before the line itself is disconnected. The program must be prepared to receive the response. If /OFF is not sent and a disconnection of the line occurs (caused, for example, by end of job on the command mode CPU), CCP attempts to issue a receive to the signed-on CPU on the disconnected line, and the CPU terminal is put into error recovery.

#### Data Mode

When an attached CPU is defined as a data mode terminal at CCP assignment time, programming of the attached CPU is greatly simplified, since it is not necessary to program for CCP application program requests and the possible CCP responses. In this case, however, the CCP application program must be initiated by the CCP system operator or another CCP command mode terminal. The initiated program can either specify the attached CPU as a required terminal (see PROGRAM assignment statement in *CCP System Reference Manual*) or acquire the CPU during execution.

### GENERATION CONSIDERATIONS

The following generation statements must be considered when generating a CCP system that will support an attached CPU:

- \$EMLA
- \$EMLD
- \$EBSC
- \$EBSD
- \$ECSC (5704-SC2 only)
- \$ECSD (5704-SC2 only)

All CPU-to-CPU support must be specified in the generation stage; a subset of the generated support can be selected in the assignment stage. See *CCP System Reference Manual* for descriptions of the generation and assignment statements.

#### \$EMLA and \$EMLD Statements

CCP to System/7 MLTA support must be described in these generation statements. When specifying the \$EMLA parameters, you must decide whether or not translation is to be performed on the CPU-to-CPU messages.

When specifying the parameters for the \$EMLD statement, you must select one of the following System/7 types:

- SYS7C
- SYS7SC
- SYS7DC

The XMCODE parameter must be specified as PTTCEBCD.

#### \$EBSC and \$EBSD Statements

These statements are used to specify CCP to System/3, System/360, System/370, System/7 BSCA, and 3600 support. When specifying the parameters of the \$EBSC statement, consider what the physical attachment of the CPU will be: switched, nonswitched point-to-point, multipoint, or CCP as host (control station).

The parameter for the ITB keyword should be YES if the CPU-to-CPU application program data blocks and data records are of fixed length and transmission checking is desired on each record.

The RECSEP keyword should specify the record separator byte if the CPU-to-CPU application program transmissions are to be of variable block and record length. (*Note:* Block lengths can only be variable up to the maximum block length.)

Either ASCII or EBCDIC code transmission must be specified for the CPU-to-CCP link.

If CS is specified YES (control station) on the \$EBSC statement, use of the RESPOL-YES parameter offers considerable performance improvement by making the BSCA polling routines resident.

If the MP-YES parameter is specified (multipoint tributary), the AUTORS-YES parameter offers performance improvement.

The XPRNCY-YES operand should be specified if the full 256-character EBCDIC set is to be used in transmission between the CPU and the CCP application program.

The TYPE operand of the \$EBSD statement must specify CPU.

*Note:* Record formatting for transmission using ITB, RECSEP, and XPRNCY applies only when the CPU is in data mode, communicating with the CCP application program.

### **\$ECSC and \$ECSD Statements (5704-SC2 Only)**

These statements are used to specify CCP to System/3, System/360, System/370, and System/7 BSC support. When specifying the parameters of the \$ECSC statement, consider what the physical attachment of the CPU will be: nonswitched point-to-point, or CCP as host (control station).

The parameter for the ITB keyword should be YES if the CPU-to-CPU application program data blocks and data records are of fixed length and transmission checking is desired on each record.

The RECSEP keyword should specify the record separator byte if the CPU-to-CPU application program transmissions are to be of variable block and record length. (*Note:* Block lengths can be variable only up to the maximum block length).

Either ASCII or EBCDIC code transmission must be specified for the CPU-to-CCP link.

The XPRNCY-YES operand should be specified if the full 256 character EBCDIC set is to be used in transmission between the CPU and the CCP application program.

The TYPE operand of the \$ECSD statement must specify CPU.

*Note:* Record formatting for transmission using ITB, RECSEP, and XPRNCY applies only when the CPU is in data mode, communicating with the CCP application program.

### **ASSIGNMENT CONSIDERATIONS**

The following assignment statements must be considered when making assignment selections for a CCP system that will support an attached CPU:

```
// TERMATTR
// BSCALINE
// BSCATERM
// MLTALINE
// MLTATERM
// PROGRAM
```

The CCP assignment stage allows the user to select a subset of the generated options. The TERMATTR statement must be used to specify the attributes of the attaching CPU. The BSCALINE or MLTALINE statements must be used to specify the characteristics of the line connection to the attaching CPU. The BSCATERM or MLTATERM statement must be used to specify the terminal characteristics of the attaching CPU.

The data transmitted by the CCP application program is formatted for line transmission (block length, record length, etc.) based on information specified in the TERMATTR statement. Data transmitted to the CCP application program must be formatted in the same manner.

### **RECOMMENDATIONS AND EXAMPLES**

It is generally recommended that, in CPU-to-CPU attachments, the remote CPU be defined as a data mode terminal. This greatly simplifies the attachment interface. In data mode, message synchronization is established by the application programs.

In command mode, it is recommended that the CCP program be designed to send data first. The message length for the remote CPU read operations must be 82 characters minimum to handle command mode message responses. Under Program Number 5704-SC2, the maximum amount of data that can be entered with a program request is 253.

The examples on the following pages depict System/360/370 BTAM-to-CCP data mode and command mode sequences. The BTAM operations are as follows:

```
WRITE TI — Write Initial
WRITE TQ — Write Inquiry
WRITE TR — Write Reset
WRITE TT — Write Continue
READ TI  — Read Initial
READ TT  — Read Continue
```





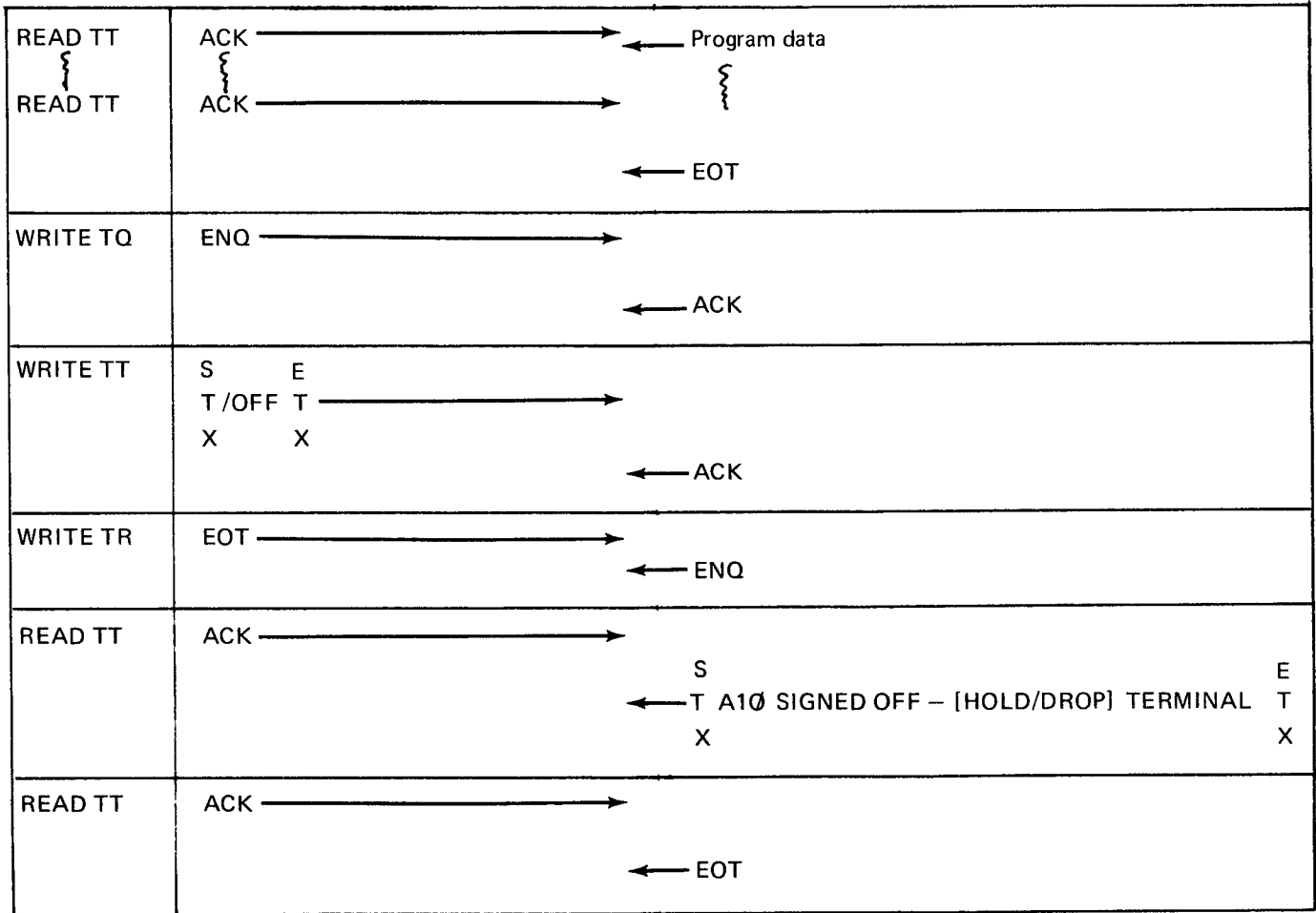
Example 2: Point-to-Point Command Mode

BTAM Operation	BTAM	CCP
WRITE TI	ENQ → S T /ON[password] T → X	← ACK ← ACK
WRITE TR	EOT →	
READ TI	ACK →	← ENQ ← S T A01 SIGNED ON – PROCEED T X
READ TT	ACK →	← EOT
WRITE TI	ENQ → S T (program name) T → X	← ACK ← ACK
WRITE TR	EOT →	
READ TI	ACK →	← ENQ ← Program data or CCP error message
READ TT ⋮ READ TT	ACK → ⋮ ACK →	← More program data or EOT ← EOT
WRITE TI	ENQ → S T /OFF T → X	← ACK ← ACK
WRITE TR	EOT →	
READ TI	ACK →	← ENQ ← S T A10 SIGNED OFF – [HOLD/DROP] X TERMINAL E T X
READ TT	ACK →	← EOT

Example 3: Point-To-Point Switched Command Mode

BTAM Operation	BTAM	CCP
WRITE TI	<p>E ID sequence N →</p> <p>Q ←</p> <p>S E T/ON[password]T →</p> <p>X X</p>	<p>A ← ID sequence C</p> <p>K ← ACK</p>
WRITE TR	EOT →	← ENQ
READ TT	ACK →	<p>S E ← T A01 SIGNED ON -PROCEED T</p> <p>X X</p>
READ TT	ACK →	← EOT
WRITE TQ	ENQ →	← ACK
WRITE TT	<p>S E T (program name) T →</p> <p>X X</p>	← ACK
WRITE TR	EOT →	← ENQ
READ TT	ACK →	← Program data or CCP error message
READ TT { READ TT	ACK → { ACK →	← More program data or EOT { ← EOT
WRITE TQ	ENQ →	← ACK
WRITE TT {	Data to CCP application program → {	← ACK {
WRITE TR	EOT →	← ENQ

Example 3 (continued)



Example 4: Point-to-Point Data Mode

BTAM Operation	BTAM	CCP	CCP Application Program Operation
WRITE TI	ENQ →	← ACK	GET
	(Data) →	← ACK	GET
WRITE TT { WRITE TT	(Data) → { (Data) →	{ ← ACK { ← ACK	{ GET { GET
WRITE TR	EOT →		
READ TI	ACK →	← ENQ ← (Data)	PUT
READ TT { READ TT	ACK → { ACK →	← (Data) { ← (Data)	PUT { PUT MESSAGE
READ TT	ACK →	← EOT	
End of Job			End of Job



For definitions of communications and data processing terms that are not included in this glossary, see *IBM Data Processing Glossary*, GC20-1699, or publications listed in *Appendix C: Bibliography*.

**\$CCPFILE:** A CCP control file on a 5444 disk or the 3340 simulation area in which, during CCP assignment stage, the user defines one or more specific operating environments for CCP. Each operating environment consists of a *set* of terminals, file, and programs that can be used during a particular run of CCP.

**AID character:** *Attention Identification character.*

**assignment stage:** The special preparatory CCP run during which the user defines one or more *sets* of specific operating environments in which the CCP can run.

**Attention Identification (AID) character:** A code that is set in a 3270 display station when the operator takes an action that produces an I/O pending condition. The character identifies the action or key that caused the condition to be generated. The AID is set when the display station operator presses a program access key, ENTER key, TEST REQ key, or program function key; when a Selector Pen attention occurs; or when a successful operator identification card read-in occurs. It also identifies device addresses assigned to printers.

**attribute (3270):** A characteristic of a display field. The attributes of a display field include: protected or unprotected; numeric-only or alphameric input control; displayed, nondisplayed, display intensified; selector-pen-detectable or nondetectable; and modified or not modified.

**attribute character (3270):** A code that defines the attributes of the display field that follows. An attribute character is the first character in a display field, but it is not a displayable character.

**block mode operations:** BSCA operations that result in all data from an operation in the program up to ETB being moved into or from the user program's record area.

**BSCA:** Binary Synchronous Communications Adapter.

**BSCC:** Binary Synchronous Communications Controller.

**CCC:** *Copy Control Character.*

**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 re-enters data mode) or until the program is cancelled by a RELEASE command (terminal enters command mode).

**command mode:** The operating mode of a command 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 as either command terminals or data terminals at assignment time.

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

**communications service subroutine:** A relocatable subroutine provided by 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. Separate subroutines are provided for COBOL, FORTRAN IV, and RPG II; a macro is provided for Basic Assembler.

**control station:** The primary or controlling computer in a multipoint telecommunications configuration.

**Copy Control Character (CCC):** A character used in conjunction with the 3270 Copy command to specify that a particular operation, or combination of operations, is to be performed at a display station or printer in the data that is to be copied.

**Copy operation:** A 3270 DFF operation that copies the contents of the buffer from one display station or printer to another display station or printer attached to the same control unit.

*cursor:* A unique symbol (an underscore) that identifies a character position in a 3270 screen display, usually the character position at which the next character to be entered from the keyboard will be displayed.

*data entry application:* A communications-based system application in which terminals are in relatively prolonged communication with an application program (as opposed to the typical inquiry application), for example, entering data for document preparation (such as invoice preparation), or entering data directly into data files from terminal.

*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 requesting terminal while the terminal is in data mode. The data mode escape command temporarily suspends a terminal's communication with a program and places the terminal in command interrupt mode.

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

*dedicated program:* A program running under CCP that requires sole use of the CCP user program area. (Applies to Model 10 and Model 12 CCP.)

*designator character:* A character that immediately follows the attribute character in a 3270 selector-pen-detectable field. The designator character controls whether a detect on the field will or will not cause an attention. For a non-attention-producing field, the designator character also determines whether the modified data tag for the field is to be set or reset as the result of a selector-pen detect.

*DFP:* 3270 Display Format Facility of the CCP.

*disk system management:* The group of system programs which control the operation of the IBM System/3 Model 10 Disk System or the IBM System/3 Model 12 and Model 15. Disk system management performs scheduling, input/output control, storage assignment, data management, and related services.

*DSM:* Disk System Management.

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

*format find:* A program (CCPFMT) that will find a newly created or modified format while running under CCP and additionally provides the capability to update the value of the DFFSFD parameter of the PROGRAM assignment statement.

*generation stage:* The initial stage of creating the CCP, during which the user specifies the size and range of function he requires in his version of the CCP, and creates that version on his disk pack.

*implied Invite Input:* An Invite Input that is not actually issued by the user program, but exists because data is allowed with the program request. Implied Invite Inputs are included in the count of outstanding Invite Inputs in the communications parameter list for certain operations.

*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, typically, a single transaction or 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*).

*interface:* In application programming under the CCP, the data areas (parameter list and record area), communications service subroutines, and defined operations by which user programs and the CCP communicate with each other.

*line buffer:* The internal main storage area associated with a communication line from which data is transmitted to a terminal or into which data is received from a terminal. Data in this area includes device and line control characters inserted or removed by the CCP.

*master terminal name:* In multicomponent terminals (1050), the symbolic terminal name specified during CCP assignment as referring to the principle input and output component.

*MDT:* Modified Data Tag.

*message mode operations:* BSCA operations that result in all blocks of data including the EOT signal being sent or received in a single operation.

*MLMP:* Multiline/Multipoint BSCA IOCS, the base data management and IOCS included in the CCP for binary synchronous communications.



**MLTA:** Multiple Line Terminal Adapter RPO. MLTA IOCS, the System/3 programming support for the MLTA RPO device, is included in the CCP for asynchronous (start-stop) communications.

**Modified Data Tag (MDT):** A bit in the attribute character of a 3270 display field which, when set on, causes the field to be read on an input operation. The modified data tag may be set by (1) a keyboard input to the field, (2) a selector-pen detection in the field, (3) a card read-in operation, or (4) program control. The modified data tag may be reset by (1) a selector-pen detection in the field, (2) program control, or (3) ERASE INPUT key.

**null character:** A hex 00 character on a 3270 that occupies a position in the storage buffer and is displayed as a blank.

**MRT program:** *multiple requesting terminal program.*

**multicomponent terminal:** A class of terminals that can have more than one input and/or output devices attached. The 1050 system is the only terminal supported by the CCP that is considered to be a multicomponent terminal.

**multiple requesting terminal (MRT) program:** A type of application program under the CCP that can process additional requests for it even though it is still processing an earlier request.

**NEP:** *never-ending program.*

**never-ending program:** A user application program which, after it has been initiated, normally remains in main storage and does not go to end of job until the CCP is shut down.

**operation stage:** The stage of the CCP during which the CCP is started, supports an online network of terminals, and is shut down.

**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.

**output/input field:** One of four classes of fields defined under the 3270 Display Format Facility. Output/input fields contain data that has been supplied either during format generation or during execution of the application program; this data can be changed by the terminal operator using the keyboard.

**password security option:** An optional CCP feature, selected during generation, which requires a terminal operator to enter a predetermined password before the CCP will allow the terminal to enter commands.

**physical file:** See *symbolic file.*

**program management:** The major function of the CCP that fetches programs, allocates system resources to programs, manages the concurrent execution of two or more 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 request under format (PRUF):** A method of requesting a program from a display format on a 3277 or 3275. The entire screen can be used to pass data with the program request. The name of the program to be requested appears as the first input field from the 3270 terminal.

**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).

**program termination code:** A two character code provided by the CCP when an application program has been cancelled by the CCP because of certain coding errors or program logic errors, or because the system operator requested cancellation of the program. This code identifies the reason for the cancellation (refer to the *CCP System Operator's Guide*).

**protected field:** A 3270 display field for which the display operator cannot use the keyboard or operator identification card reader to enter, modify, or erase data.

**PRUF terminal:** A 3275 or 3277 terminal whose last Put operation was a PRUF (program request under format) Put operation. The terminal remains a PRUF terminal until a non-PRUF Put is issued to that terminal or until the CLEAR key is pressed.

**record mode operations:** Application program input and output operations that result in a single record (for BSCA, a single record of a data block) being moved into or out of the program's record area.

**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.

**RVI:** A signal from a receiving device to a device that is transmitting to interrupt its transmission as soon as possible (see Get operation).

**selector pen detectable (SPD) field:** One of four classes of fields defined under the 3270 Display Format Facility. SPD fields allow the terminal operator to enter data by using the selector pen.

**Shutdown:** The final stage of the CCP operation, during which the CCP allows programs currently executing or scheduled to finish processing, then closes files, adapters, and communication lines.

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

**single requesting terminal (SRT) program:** A type of application program under the CCP that can process a request from only one requesting terminal during its execution.

**SPD field:** *Selector Pen Detectable field.*

**SRT program:** *single requesting terminal program.*

**standby mode:** The mode of a data (non-command) terminal when it is not under control of a user program.

**startup:** The initial phase of the CCP operational stage, during which all necessary initialization occurs, including opening of disk files, adapters, and communication lines, and the completion of various tables and control blocks.

**subhost:** A telecommunications system which, while directly controlling a group of terminals, is itself a tributary station to another central processor.

**symbolic file:** A file reference (symbolic name) which allows, on separate executions of a program, reference 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.

**symbolic subterminal name:** Symbolic names assigned to individual components of a multicomponent terminal (1050). See *master terminal name*.

**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:** See *system task* and *user task*.

**task chaining:** The process of requesting initiation of a CCP task from within a currently executing CCP task, without requiring system or operator action.

**task identification:** An identifying character associated with a task which differentiates between that task and other tasks running concurrently under the CCP.

**terminal attributes:** Characteristics of a terminal from the point of view of the CCP and CCP application programs, including block length, record length, data format, and other information.

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

**translation:** Under the CCP, conversion of the transmission line data code (if not EBCDIC) into EBCDIC or conversion from EBCDIC into transmission line data code.

**tributary station:** A secondary or non-controlling device in a multipoint telecommunications configuration.

**truncation:** Loss of excess data when the length of data received from a terminal is greater than the maximum input length specified in the parameter list or when more data is provided in an output operation than the line buffer for the terminal can hold (in BSCA record mode output operations, if the output length exceeds the record length specified in the terminal attributes set).

**unit record device:** Under the CCP, the MFCU, 1442 Card Read Punch, 5203 and 1403 printers, MFCM, 2501 Card Reader, and 3741 Data Station directly attached.

**unprotected field:** A 3270 display field for which the terminal operator can manually enter, modify, or erase data.

**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.

**WCC:** *Write Control Character.*

**Write Control Character (WCC):** A character used in conjunction with 3270 write operations to specify that a particular operation, or combination of operations, is to be performed at a display station or printer.

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

### CCP

- *IBM System/3 Communications Control Program System Design Guide*, GC21-5165
- *IBM System/3 Communications Control Program General Information Manual*, GC21-7578
- *IBM System/3 Communications Control Program Terminal Operator's Guide*, GC21-7580
- *IBM System/3 Models 8, 10, and 12 Communications Control Program System Operator's Guide*, GC21-7581
- *IBM System/3 Model 15 Communications Control Program System Operator's Guide*, GC21-7619
- *IBM System/3 Models 8, 10, and 12 Communications Control Program System Reference Manual*, GC21-7588
- *IBM System/3 Model 15 Communications Control Program System Reference Manual*, GC21-7620
- *IBM System/3 Communications Control Program Messages Manual*, GC21-5170

### General System/3

- *IBM System/3 Models 8, 10, 12, and 15 Components Reference Manual*, GA21-9236
- *IBM System/3 Overlay Linkage Editor Reference Manual*, GC21-7561
- *IBM System/3 Models 10 and 12 Control Programming Macros Reference Manual*, GC21-7562
- *IBM System/3 Model 15 System Control Programming Macros Reference Manual*, GC21-7608

### MLTA and MLTA Terminals

- *IBM System/3 Multiple Line Terminal Adapter RPO Program Reference and Component Description Manual*, GC21-7560
- *IBM 2740 Communications Terminal Models 1 and 2 Component Description*, GA24-2403
- *IBM 2741 Communication Terminal*, GA24-3415
- *IBM 1050 Data Communication System Principles of Operation*, GA24-3474
- *IBM 3767 Models 1 and 2 Communications Terminal Component Description Manual*, GA27-3096

### BSC and BSCA Terminals/Systems

- *General Information: Binary Synchronous Communications*, GA27-3004
- *IBM 3270 Information Display System Component Description*, GA27-2749
- *IBM System/3 3735 Support Program Coding Manual*, GC21-5096
- *IBM 3735 Programmer's Guide*, GC30-3001
- *IBM 3740 BTAM/TCAM Programmers Guide*, GC21-5071
- *IBM 3741 Data Station Reference Manual*, GA21-9183
- *IBM System/7 RPO Binary Synchronous Communications Module Programming Guide and Reference Manual*, SC34-1510
- *IBM System/7 System Summary*, GA34-0002
- *IBM System/3 Multiline/Multipoint Binary Synchronous Communications Reference Manual*, GC21-7573

### **Programming Language Manuals**

- *IBM System/3 RPG II Reference Manual, SC21-7504*
- *IBM System/3 Subset American National Standard COBOL, GC28-6452*
- *IBM System/3 Disk Sort Reference Manual, SC21-7522*
- *IBM System/3 FORTRAN IV Reference Manual, SC28-6874*
- *IBM System/3 Basic Assembler Reference Manual, SC21-7509*

### **General Telecommunications**

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

### **SIOC Channel Connected Systems**

- *IBM System/3 Model 15D Channel Connected Systems Program Reference and Logic Manual, GC21-5199*

Table D-1 shows the decimal and hexadecimal values and RPG II codes that represent CCP communications operations. The following symbols are used in the table to represent operations:

**Operation Symbols**

ACC	Accept Input
ACQ	Acquire Terminal
ANW	Accept No-Wait Input (5704-SC2)
CPY	Copy (3270 DDF only)
EAU	Erase All Unprotected (3270 DFF only)
EOF	Force End-of-File (RPG II only)
GET	Get
GTA	Get Terminal Attributes
INV	Invite Input
PCR <sup>1</sup>	Port Command Request <sup>2</sup>
PNW	Put-No-Wait
PTG	Put-Then-Get
PUT	Put
REL	Release Terminal
RTC	Release and Task Chain <sup>2</sup>
SHQ	Shutdown Inquiry
SPI	Stop Invite Input
TCH	Chain Task Request (5704-SC2 only)
WAT	Wait Operation (Model 15 only)

**Operation Modifier Symbols**

CMD	Command-mode Terminal
BLK	Block – send end-of-block (EOB)
DLY	Shutdown Delay Inquiry Modifier
GEN <sup>1</sup>	Generic Modifier to Acquire <sup>2</sup>
KPL	Keep line
MSG	Message – end end-of-transmission (EOT)
NEL	Not End New Line
NNL	Not Start New Line
OVR	Override (3270 DFF only)
PRF	Program Request under Format (PRUF)
RVI	Send reverse interrupt (RVI)
STA	Set terminal attributes

<sup>1</sup> See *IBM System/3 Model 15D Channel Connected Systems Program Reference and Logic Manual*, GC21-5199.

<sup>2</sup> 5704-SC2 only

Operation	Value		RPG II Code
	Dec	Hex	
ACC	4	0004	ⓂⓂⓂD
ANW	68	0044	ⓂⓂDD
ACQ	9	0009	ⓂⓂⓂI
ACQ,CMD	41	0029	ⓂⓂBI
ACQ,STA	25	0019	ⓂⓂAI
ACQ, CMD, STA	57	0039	ⓂⓂCI
ACQ,GEN <sup>1</sup>	105	0069	ⓂⓂFI
ACQ, GEN <sup>1</sup> ,STA	121	0079	ⓂⓂGI
CPY	66	0042	ⓂⓂDB
EAU	82	0052	ⓂⓂEB
EOF	–	–	ⓂⓂGA
GET	1	0001	ⓂⓂⓂA
GET,RVI	17	0011	ⓂⓂAA
GTA	8	0008	ⓂⓂⓂH
INV	5	0005	ⓂⓂⓂE
PCR <sup>1</sup>	58	003A	ⓂⓂCK
PNW	6	0006	ⓂⓂⓂF
PNW,NEL	262	0106	ⓂAⓂF
PNW,NNL	518	0206	ⓂBⓂF
PNW,NNL,NEL	774	0306	ⓂCⓂF
PNW,BLK	38	0026	ⓂⓂⓂF
PNW,BLK,NEL	294	0126	ⓂAⓂF

Table D-1 (Part 1 of 3). CCP Operation Codes

Operation	Value		RPG II Code
	Dec	Hex	
PNW,BLK,NNL	550	0226	ØBBF
PNW,BLK,NNL,NEL	806	0326	ØCBF
PNW,MSG	54	0036	ØØCF
PNW,MSG,INV	–	–	ØØØW
PNW,MSG,INV,OVR	–	–	ØHØW
PNW,MSG,NEL	310	0136	ØACF
PNW,MSG,NNL	566	0236	ØBCF
PNW,MSG,NNL,NEL	822	0336	ØCCF
PNW,MSG,OVR	2102	0836	ØHCF
PNW,MSG,OVR,PRF	2166	0876	ØHGF
PNW,MSG,PRF	118	0076	ØØGF
PTG	3	0003	ØØØC
PTG,NEL	259	0103	ØAØC
PTG,NNL	515	0203	ØBØC
PTG,NNL,NEL	771	0303	ØCØC
PTG,MSG	51	0033	'ØØCC'
PUT	2	0002	ØØØB
PUT,NEL	258	0102	ØAØB
PUT,NNL	514	0202	ØBØB
PUT,NNL,NEL	770	0302	ØCØB

Table D-1 (Part 2 of 3). CCP Operation Codes

Operation	Value		RPG II Code
	Dec	Hex	
PUT,BLK	34	0022	ØØBB
PUT,BLK,PRF	98	0062	ØØFB
PUT,BLK,NEL	290	0122	ØABB
PUT,BLK,NNL	546	0222	ØBBB
PUT,BLK,NNL,NEL	802	0322	ØCBB
PUT,MSG	50	0032	ØØCB
PUT,MSG,INV	–	–	ØØCS
PUT,MSG,INV,OVR	–	–	ØHCS
PUT,MSG,NEL	306	0132	ØACB
PUT,MSG,NNL	562	0232	ØBCB
PUT,MSG,NNL,NEL	818	0332	ØCCB
PUT,MSG,OVR	2098	0832	ØHCB
PUT,MSG,PRF	114	0072	ØØGB
PUT,MSG,PRF,OVR	2162	0872	ØHGB
REL	10	000A	ØØØK
REL,KPL	26	001A	ØØAK
RTC	74	004A	ØØDK
SHQ	0	0000	ØØ00
SHQDLY	256	0100	ØA00
SPI	1025	0401	ØDØA
TCH	42	002A	ØØBK
WAT	20	0014	ØØAD

Table D-1 (Part 3 of 3). CCP Operation Codes

This appendix contains summary tables of return codes issued to application programs by the CCP following tele-communications operations. (See index entry *return codes* for references to additional information.) The following summary tables are provided:

- Table E-1 Success/Exception Conditions Return Codes
- Table E-2 Common I/O Error Return Codes
- Tables E-3, E-4, and E-5 Unique 3270, 3735, and 3741 BSCA I/O Error Return Codes
- Table E-6 Success/Exception Return Codes Per Operation Type
- Table E-7 Common I/O Error Return Codes Per Operation Type
- Tables E-8, E-9, and E-10 3270, 3735, and 3741 I/O Error Return Codes Per Operation Type
- Table E-11 Contents of Effective Input Length Field and Record Area Per Operation Type and Return Code

Tables 1-3 include a suggested program action for each CCP return code. The recommended actions are as follows:

Code	Recommended Program Actions
A1	Continue normal processing
A2	Attempt to determine the cause of the error and retry the operation if appropriate; otherwise go to the next logical operation <i>Note:</i> If the failure that caused the error condition is not corrected before the retry, the same error may occur on the retry.
A3	Do not reissue the operation to this terminal
A4	Go to End-of-Job as soon as logically possible
A5	Process according to the requirements of the application
A6	Process data until EOT on the terminal for which data is pending
A7	Issue a Get operation or go to End-of-Job

On data transfer operations, input and/or output data accompanies the following return codes, and no others:

0, 1, 2, 3, 6

**Negative Return Codes**

In most cases, when the CCP encounters an error condition, it stops processing the operation and returns a negative return code. Therefore, if concurrent error conditions occur, the CCP returns a code for only the first error condition detected.

The results of the data transfer for any operation yielding a negative return code are unpredictable, except that a translation error on an output operation always indicates the data has not been transmitted.

For BSCA, a negative return code always terminates the current transmission with the terminal, except for a translation error on a Put operation. If a translation error occurs when data is being sent to a terminal, the line is still connected to the terminal. Except for the translation error, a Put Message should not be issued if you are transmitting when a negative return code occurs and another Get should not be issued unless you wish to re-establish communication with the terminal.

See *Put-Then-Get* operation under *Operations* in Chapter 2 for information about the meaning of negative return codes for that operation. See also index entry *return codes, negative (DFF)*.

**Use of Data Truncated Return Code in 3270 DFF**

The data truncated (hex 01) return code has a special meaning in programs using the 3270 Display Format Facility. If this return code is received from a Get or Accept Input operation, it indicates that the maximum input length given in the parameter list for the operation was less than the total length of all the fields the program should have expected, as defined by the last Put Message or Put Overrides (with select input fields) operation.

The data truncated return code does not necessarily mean that more data was entered from the terminal than was received by the program. A situation could exist where a second format was Put to the terminal without clearing a format that was already there. The terminal operator could key only fields in the second format, the sum of which was less than the length the program expected. The data truncated return code may be returned from the operation.

CCP RETURN CODES			Description	Program Action
Dec	Hex	RPG II		
0	0000	00	Successful operation	A1
1	0001	01	DATA TRUNCATED: Data was truncated; for input operations, the length of the data was greater than the input length specified in the parameter list; for output operations, the length of the data was greater than the length of the teleprocessing line buffer.	A2
2	0002	02	EOT: Input operation was successful (EOT was received), or non-PRUF program request data was returned to a PRUF program at program request time.	A1
3	0003	03	DATA TRUNCATION AND EOT: Data truncation occurred on this operation and EOT was received.	A2
4	0004	04	SHUTDOWN REQUESTED: The system operator has requested CCP shutdown. The requested operation has not been performed.	A4
5	0005	05	DATA PENDING: The operation was issued to a terminal on a BSCA line which is currently controlled by another terminal in use by your program (line awaiting EOT). An Invite Input was issued to a terminal which is currently awaiting an EOT. A Put or Get of a record or block was issued to another terminal on the same line. Put or Get operations with the previous terminal must be continued until EOT is transmitted on the line.	A6
6	0006	06	TERMINAL INTERRUPT/RVI: A terminal interrupt (MLTA) or RVI (BSCA) has been received from the remote station. The operation was successful. For BSCA terminals, this return code indicates that the RVI was received after the initial positive response to addressing. An RVI to addressing will result in -14 for CPU, -40 for 3735, or -20 to -28 for 3270.	A5
7	0007	07	3270 CLEAR: The terminal operator at 3270 pressed the CLEAR key (see <i>3270 Component Description</i> , GA27-2749).	A5
8	0008	08	TERMINAL NO LONGER AVAILABLE: Command interrupt mode was entered and the terminal operator released (/RELEASE command) the terminal. The terminal is no longer available to this program.	A3

Table E-1 (Part 1 of 2). Success/Exception Condition Return Codes



CCP RETURN CODES			Description	Program Action
Dec	Hex	RPG II		
09	0009	09	TERMINAL OFFLINE: The requested terminal has been varied offline and is not available to this program.	A3
10	000A	10	STOP INVITE INPUT SUCCESSFUL: The request to stop Invite Input was successful; the Invite input has been cancelled.	A1
11	000B	11	ACQUIRE TERMINAL FAILED: The attempt to acquire a terminal for this program has failed.	A1
12	000C	12	CHAIN TASK QUEUE FULL: The maximum number of chain task requests are already queued; this chain task request cannot be accepted.	A2
13	000D	13	INSUFFICIENT TP BUFFER: Insufficient TP is available to queue this request.	A2
14	000E	14	CHAIN REQUEST DATA ACCEPTED SUCCESSFULLY: An operation to accept the data from a chain task request was successful.	A1
15	000F	15	CHAIN REQUEST DATA TRUNCATED: The data associated with a chain task request was truncated.	A2
16	0010	16	NO DATA RETURNED ANW: There is no data available from invited terminals, program requests, or task chain operations when accept no-wait input was used.	
17	0011	17	SHUTDOWN PENDING: Shutdown delay has been entered, and the delay time has not expired.	A1

Table E-1 (Part 2 of 2). Success/Exception Condition Return Codes

CCP Return Codes			Description	Program Action
Dec	Hex	RPG II		
-1	FFFF	0J	DATA CHECK: Data was received incorrectly, checking error condition detected.	A2
-2	FFFE	0K	INVALID CHARACTER: (1) During translation of data, an invalid character was found. (2) An invalid ASCII character has been detected by BSCA.	A2
-3	FFFD	0L	LOST DATA: Data received was lost because it exceeded the size of the input buffer.	A2
-4	FFFC	0M	PERMANENT BSCA ERROR: Operation failed because a permanent error condition was detected.	A2
-5	FFFB	0N	ABNORMAL RESPONSE: An invalid response was received from the remote station.	A2
-6	FFFA	0O	TRANSMIT/RECEIVE ABORT: Data transfer failed and the teleprocessing line has been closed. CCP has varied offline all terminals on this line.	A3
-7	FFF9	0P	NO RESPONSE TO POLLING/ADDRESSING: The selected terminal does not respond to polling or addressing.	A2
-8	FFF8	0Q	TEXT TIME OUT: The terminal does not respond to attempted data transfer.	A2
-9	FFF7	0R	WAIT TIME EXCEEDED: Data not sent or received before EOT within a specified time frame. <i>Note:</i> If using 3270, retry (program action A2) is not allowed. Go to End-of-Job as soon as logically possible.	A2
-10	FFF6	1 }	NO CONNECTION: Unable to establish a connection with the remote station.	A2
-11	FFF5	1J	INVALID ID'S: The ID exchange with the remote station failed.	A2
-12	FFF4	1K	ABORT, DISCONNECT: The switched line connection to the remote station has been lost.	A2
-13	FFF3	1L	ADAPTER CHECK: A hardware check occurred on the teleprocessing line adapter.	A2
-14	FFF2	1M	NEGATIVE RESPONSE TO ADDRESSING: The remote terminal has an error condition which prevents it from successfully receiving data. For the 3284, 3286 and 3288 printer components of the 3270 system, this return code indicates the printer is busy.	A2
-91	FFA3	9J	OPERATION NOT PERFORMED (RPG II SPECIAL): The operation was not performed because indicator 91 was set on by a previous operation and was not set off.	A2

Table E-2. Common I/O Error Return Codes  
E-4

CCP RETURN CODES			Description	Program Action
Dec	Hex	RPG II		
-19	FFED	1R	A poll for status was performed by the system to get the status from a terminal after a put operation failed. The terminal did not respond with the status.	
-20	FFEC	2 }	Device unavailable or not ready.	A2
-21	FFEB	2J	Not used. This return code is not returned to the user. If CCP detects this condition, an operations message is displayed followed by a 2F termination code.	
-22	FFEA	2K	Equipment check; device end.	A2
-23	FFE9	2L	3270 detected a BSCA error. <sup>1</sup>	A2
-24	FFE8	2M	Control check; data check.	A2
-25	FFE7	2N	Data check on copy command.	A2
-26	FFE6	2O	Operation check—copy command.	A2
-27	FFE5	2P	Device busy with copy command.	A2
-28	FFE4	2Q	Control check, operation check, data check during copy command.	A2
-29	FFE3	2R	Invalid data from 3270 using DFF. This return code is caused by the terminal operator pressing the TEST REQ key on the 3270 keyboard.	A2

<sup>1</sup> 3271 answers line errors with NAK, and CCP retransmits a number of times. 3275 answers line errors with EOT, and CCP places the terminal in ERP without doing any retransmissions.

**Table E-3. Unique 3270 BSCA I/O Error Return Codes**

CCP RETURN CODES			Description	Program Action
Dec	Hex	RPG II		
-40	FFD8	4 }	Attempted send before receive.	A7
-41	FFD7	4J	Invalid character.	A2
-42	FFD6	4K	Buffer overflow.	A2
-43	FFD5	4L	Disk full.	A5
-44	FFD4	4M	Directory full.	A5
-45	FFD3	4N	Undefined header.	A2
-46	FFD2	4O	3735 disk error.	A2

**Table E-4. Unique 3735 BSCA I/O Error Return Codes**

CCP RETURN CODES			Description	Program Action
Dec	Hex	RPG II		
-50	FFCE	5}	Transparency error occurred.	A2
-51	FFCD	5J	No activity in 20 seconds.	A2
-52	FFCC	5K	Data check.	A2
-53	FFCB	5L	Received line bid error.	A2
-54	FFCA	5M	Wrong length error.	A2
-55	FFC9	5N	Reset was pressed on 3741.	A5
-56	FFC8	5O	Security check.	A5
-57	FFC7	5P	Disk overflow.	A5
-58	FFC6	5Q	Bad extent error.	A2
-59	FFC5	5R	Both stations transmit.	A7
-60	FFC4	6}	Length error.	A2
-61	FFC3	6J	No record found on disk.	A2
-62	FFC2	6K	Seek error.	A2
-63	FFC1	6L	Read error.	A2
-64	FFC0	6M	Write error.	A2
-65	FFBF	6N	Not ready.	A2
-66	FFBE	6O	Diskette write protected.	A2

**Table E-5. Unique 3741 BSCA I/O Error Return Codes**

Table E-6. Success/Exception Return Codes Per Operation Type

Return Code Value			Description	Operation (Abbreviations are explained under Operation Codes in Appendix D)																
Dec	Hex	RPG II		ACC ANW	ACQ	CPY	EAU	EOF	GET	GTA	INV	PNW	PTG	PUT	REL	SHO	SPI	TCH	WAT	RTC
0	0000	00		X	X	X		X	X	X	X	X	X	X	X <sup>2</sup>	X <sup>1</sup>	X	X	X <sup>2</sup>	
1	0001	01		X				X	X				X	X		X				
2	0002	02		X <sup>4</sup>				X					X	X		X				
3	0003	03		X				X					X	X		X				
4	0004	04		X											X					
5	0005	05						X			X	X	X	X						
6	0006	06										X	X	X						
7	0007	07		X				X				X				X				
8	0008	08		X <sup>2</sup>				X <sup>2</sup>				X <sup>2</sup>				X <sup>2</sup>				
9	0009	09		X				X			X	X	X	X						
10	000A	10														X				
11	000B	11											X							
12	000C	12															X			
13	000D	13																X		
14	000E	14		X																
15	000F	15		X																
16	0010	16		X <sup>2,5</sup>																
17	0011	17														X <sup>6</sup>				

1 This specifies that the Get operation was successful while the Stop Invite input failed.  
 2 The count of Invite Input operations outstanding for this user program is returned in the effective input length field of the parameter list.  
 3 Indicates shutdown not requested.  
 4 If a PRUF program, indicates that non-PRUF data was received.  
 5 Returned only on Accept No-Wait INPUT operation.  
 6 Returned only for Shutdown Inquiry with Delay Modifier.

Return Code Value			Description	Operation (Abbreviations are explained under Operation Codes Appendix D)															
Dec	Hex	RPG II		ACC/ANW	ACQ	CPY	EAU	EOF	GET	GTA	INV	PNW	PTG	PUT	REL	SHO	SPI	TCH	WAT
-1	FFFF	0J	Data check	X		X		X				X <sup>1</sup>	X <sup>1</sup>				X		
-2	FFFE	0K	Invalid character	X		X		X				X	X				X		
-3	FFFD	0L	Lost data	X		X		X				X					X		
-4	FFFC	0M	Permanent BSCA error	X		X		X				X	X				X		
-5	FFFB	0N	Abnormal response	X		X		X				X	X				X		
-6	FFFA	0O	Transmit/receive abort	X		X		X				X	X				X		
-7	FFF9	0P	No response to polling/addressing	X		X		X				X	X				X		
-8	FFF8	0Q	Text time out	X		X		X				X	X				X		
-9	FFF7	0R	Wait time exceeded	X		X		X				X	X				X		
-10	FFF6	1 }	No connection	X		X		X				X	X				X		
-11	FFF5	1J	Invalid IDs	X		X		X				X	X				X		
-12	FFF4	1K	Abort, disconnect	X		X		X				X	X				X		
-13	FFF3	1L	Adapter check	X		X		X				X	X				X		
-14	FFF2	1M	Negative response to addressing																
-91		9J	Operation not performed <sup>2</sup>																

Table E-7. Common I/O Error Return Codes Per Operation Type

<sup>1</sup> Data check cannot occur on a Put for BSCA EBCDIC.  
<sup>2</sup> RPG II SPECIAL only.

Return Code Value		Description	Operation (Abbreviations are explained under Operation Codes in Appendix D)															
Dec	Hex		ACC ANW	ACQ	CPY	EAU	EOF	GET	GTA	INV	PNW	PTG	PUT	REL	SHQ	SPI	TCH	WAT
-19	FFED	1R			X	X						X	X					
-20	FFEC	2 }			X	X						X	X					
-21	FFEB	2J																
-22	FFEA	2K			X	X						X	X					
-23	FFE9	2L			X	X						X	X					
-24	FFE8	2M			X	X						X	X					
-25	FFE7	2N			X	X						X	X					
-26	FFE6	2O			X	X						X	X					
-27	FFE5	2P			X	X						X	X					
-28	FFE4	2Q			X	X						X	X					
-29	FFE3	2R																X

Table E-8. 3270 I/O Error Return Codes Per Operation Type



Return Code Value		Description	Operation (Abbreviations are explained under Operation Codes in Appendix D)															
Dec	Hex		ACQ	ANW	CPY	EAU	EOF	GET	GTA	INV	PNW	PTG	PUT	REL	SHQ	SPI	TCH	WAT
-40	FFD8	RPG II 4 }									X	X						
-41	FFD7	4J									X	X						
-42	FFD6	4K									X	X						
-43	FFD5	4L									X	X						
-44	FFD4	4M									X	X						
-45	FFD3	4N									X	X						
-46	FFD2	4O	X								X							X

Table E-9. 3735 I/O Error Return Codes Per Operation Type

Return Code Value			Description	Operation (Abbreviations are explained under Operation Codes in Appendix D)															
Dec	Hex	RPG II		ACC ANW	ACO	CPY	EAU	EOF	GET	GTA	INV	PNW	PTG	PUT	REL	SHO	SPI	TCH	WAT
-50	FFCE	5 }	Transparency error occurred	X				X					X						
-51	FFCD	5K	No activity in 20 seconds	X				X			X		X			X			
-52	FFCC	5K	Data check	X				X					X						
-53	FFCB	5L	Received line bid error	X				X			X		X			X			
-54	FFCA	5M	Wrong length error	X				X			X		X			X			
-55	FFC9	5N	Reset was pressed on 3741	X				X			X		X			X			
-56	FFC8	5O	Security check	X				X			X		X			X			
-57	FFC7	5P	Disk overflow	X				X			X		X			X			
-58	FFC6	5Q	Bad extent error	X				X					X						
-59	FFC5	5R	Both stations transmit	X				X					X						
-60	FFC4	6 }	Length error	X				X					X			X			
-61	FFC3	6J	No record found on disk	X				X					X			X			
-62	FFC2	6K	Seek error	X				X					X			X			
-63	FFC1	6L	Read error	X				X					X			X			
-64	FFC0	6M	Write error	X				X					X			X			
-65	FFBF	6N	Not ready										X			X			
-66	FFBE	6O	Diskette write protected										X			X			

Table E-10. 3741 I/O Error Return Codes Per Operation Type

Return Code	Operation																	
	ACC <sup>1</sup>		ACQ		ANW		CPY		EAU		EOF		GET <sup>1</sup>		GTA		INV	
	EFFL	RECA	EFFL	RECA	EFFL	RECA	EFFL	RECA	EFFL	RECA	EFFL	RECA	EFFL	RECA	EFFL	RECA	EFFL	RECA
0	LIN	DATA	ATTRID	NC	LIN	DATA	NC	NC	NC	NC			LIN	DATA	LIN	ATTRS	NC	NC
1 – Truncated	LIN* LIN**	DATA* DATA**			LIN* LIN*	DATA* DATA*							LIN* LIN**	DATA* DATA**	LIN*	DATA*		
2 – EOT/non-PRUF data	LIN	DATA			LIN	DATA							LIN	DATA				
3 – Truncated – EOT <sup>2</sup>	LIN*	DATA*			LIN*	DATA*							LIN*	DATA*				
4 – Shutdown	NC 3 ZERO	NC BLANK			#INV	NC BLANK												
5 – Data pending													NC ZERO	NC BLANK				
6 – RVI																		
7 – CLEAR key	ZERO	NC BLANK			ZERO	NC BLANK							ZERO	NC BLANK				
8 – Released	#INV	NC BLANK			#INV	NC BLANK							#INV	NC BLANK				
9 – Terminal offline	NC	NC			NC	NC	NC	NC	NC	NC			NC ZERO	NC BLANK			NC	NC
10 – Stop Invite Input													ZERO	BLANK				
11 – Acquire failed			NC	NC														
12 – Chain queue full																		
13 – Insufficient TP buffer																		
14 – Chain data accepted	LIN	DATA			LIN	DATA												
15 – Chain data truncated	LIN*	DATA*			LIN*	DATA*												
16 – No data (ANW)	#INV	NC BLANK																
-n – All negative return codes	ZERO	BLANK			ZERO	BLANK	NC	NC	NC	NC			ZERO	BLANK				

**Legend:**

- LIN = Actual length of input data received in the record area.
- LIN\* = Data was truncated – the value shown is the maximum input length.
- LIN\*\* = Data was truncated – the value shown is the actual length of data received in the record area.
- EFFL #INV = Count of outstanding Invite Inputs.
- NC = No change.
- ZERO = Set to zero by the CCP.
- ATTRID = Attributes Identifier.
- DATA = Data received from terminal.
- DATA\* = Data received from terminal up to maximum input length.
- DATA\*\* = Data received from the terminal, up to the last field that would fit in the record area.
- BLANK = Set to blanks (X'40') by the CCP.
- ATTRS = Terminal attributes information (21 positions).

<sup>1</sup>The top symbol is for non-DFF programs, the bottom symbol is for DFF programs.  
<sup>2</sup>Not applicable for DFF.  
<sup>3</sup>Returns the number of outstanding invites (S704-SC2).

Table E-11 (Part 1 of 2). Contents of Effective Input Length Field and Record Area Per Operation Type and Return Code

Return Code	Operation																	
	PNW		PTG <sup>2,1</sup>		PUT		REL		SHQ		SPI <sup>1</sup>		TCH		WAT		RTC	
	EFFL	RECA	EFFL	RECA	EFFL	RECA	EFFL	RECA	EFFL	RECA	EFFL	RECA	EFFL	RECA	EFFL	RECA	EFFL	RECA
0	NC	NC	NC LIN	NC DATA	NC	NC	#INV	NC	NC <sup>4</sup>	NC	LIN	DATA	NC	NC			#INV	NC
1 – Truncated			NC LIN	NO DATA	NC	NC					LIN	DATA						
2 – EOT/Non-PRUF data			LIN	DATA							LIN	DATA						
3 – Truncated – EOT <sup>1</sup>			LIN	DATA							LIN	DATA						
4 – Shutdown									NC <sup>4</sup>	NC								
5 – Data pending	NC	NC	NC NC	NC NC	NC	NC												
6 – RVI			NC	NC	NC	NC												
7 – CLEAR key			ZERO	NC							ZERO	NC BLANK						
8 – Released			#INV	NC							#INV	NC BLANK						
9 – Terminal offline	NC	NC	NC NC	NC NC	NC	NC					NC ZERO	NC BLANK						
10 – Stop Invite Input											NC BLANK	NC BLANK						
11 – Acquire failed																		
12 – Chain queue full													NC	NC				
13 – Insufficient TP buffer													NC	NC			NC	NC
14 – Chain data accepted																		
15 – Chain data truncated																		
17 – Shutdown pending									NC <sup>4</sup>	NC								
-n – All negative return codes			NC ZERO	NC BLANK	NC	NC					ZERO	BLANK						

**Legend:**

- LIN = Actual length of input data received in the record area.
- LIN\* = Data was truncated – the value shown is the maximum input length.
- LIN\*\* = Data was truncated – the value shown is the actual length of data received in the record area.
- EFFL #INV = Count of outstanding invite Inputs.
- NC = No change.
- ZERO = Set to zero by the CCP.
- ATRID = Attributes Identifier.
- DATA = Data received from terminal.
- DATA\* = Data received from terminal up to maximum input length.
- DATA\*\* = Data received from the terminal, up to the last field that would fit in the record area.
- RECA BLANK = Set to blanks (X'40') by the CCP.
- ATTRS = Terminal attributes information (21 positions).

<sup>1</sup>The top symbol is for non-DFF programs; the bottom symbol is for DFF programs.  
<sup>2</sup>The top symbol is for Put portion of the operation; the bottom symbol is for the Get portion of the operation.  
<sup>3</sup>Not applicable for DFF.  
<sup>4</sup>Returns the number of outstanding invites(5704-SC2 only).

**Table E-11 (Part 2 of 2). Contents of Effective Input Length Field and Record Area Per Operation Type and Return Code**

- /FILE command
  - command mode message/response
  - prefixes A-3
  - description 3-16
- /MSG command mode message/response
  - prefixes A-3
- /NAME command mode message/response
  - prefixes A-3
- /NOQ command mode message/response
  - prefixes A-3
- /OFF command
  - command mode message/response
  - prefixes A-3
  - description 3-20
- /ON command mode message/response
  - prefixes A-3
- /Q command mode message/response
  - prefixes A-3
- /RELEASE command mode message/response
  - prefixes A-3
- /RUN command mode message/response
  - prefixes A-3
- 91 return code 6-5
- \$\$\$CPFILE
  - definition B-1
  - generation/initialization 1-1
- \$\$\$CPAU 7-16
- \$\$\$CPDT 8-61
- \$\$\$C4Z9 7-16
- \$EBSC generation statement considerations,
  - CPU attachment A-5
- \$EBSD generation statement considerations,
  - CPU attachment A-5
- \$ECOM macro 7-19
- \$ECPL macro 7-19
- \$ECSC generation statement considerations,
  - CPU attachment A-6
- \$ECSD generation statement considerations,
  - CPU attachment A-6
- \$EEQU macro 7-19
- \$EMLA generation statement considerations,
  - CPU attachment A-5
- \$EMLD generation statement considerations,
  - CPU attachment A-5
- \$ETNT macro 7-19
- \$ETRC macro 7-19
- \$ETUB macro 7-19
- \$NCIO macro
  - description 7-8
  - examples of use 7-12
- \$NCOM macro 7-1
- \$NOPV macro 7-3
- \$NPL macro 7-6
- \$NPLO macro 7-2
- \$NRTV macro 7-3
- \$SIT macro 2-17
- ACCEPT (COBOL programming
  - restriction) 4-15
- accept input operation
  - count of outstanding invite inputs 2-28
  - data with program request 2-29
  - description 2-28
  - for chain task requests 2-29
  - maximum input length 2-28
  - MRT programs 3-3
  - never-ending program 3-4
  - put with invite input
    - consideration 6-12
  - relation to invite input 2-28
  - RPG II 6-13
  - when allowed 2-28
  - 3270 DFF 2-28, 8-53
- accept-no-wait operation
  - description 2-30
  - use of third parameter list field 2-3
- access value, disk files 9-19
- acquire terminal operation
  - (see also generic acquire terminals operation)
  - description 2-36
  - multicomponent terminals 2-6
  - RPG II, operation code 6-13
  - set attributes 2-36
  - use of third parameter list field 2-3
- adding records to disk files 9-8, 9-10
- address of CCP user programs (Model 15) 9-1
- address table, relocation 7-17
- address, record area 2-4
- AID (attention identification) character
  - definition B-1
  - input record format (3270 DFF) 8-21
- allocation
  - BSCA line 2-14
  - BSCA switched line 3-17
  - MLTA lines 2-12
  - MLTA switched lines 3-20
  - status (get attributes) 2-33

- alphameric data, defining (3270 DFF) 8-17
- alphameric field
  - characteristics (3270 DFF) 8-5
  - characteristics (3270 DFF), by field type 8-6
  - RPG II EXIT/RLABL interface 6-27
- answering a call on a switched line 3-17
- application program
  - examples of logic 3-8
  - standard interface to CCP 2-1
- APPLY CORE-INDEX, COBOL programming consideration 4-15
- array name (RPG II) 6-9
- array, parameter
  - extension specifications (RPG II) 6-9
  - FORTTRAN 5-2
  - loading (RPG II) 6-9
- ASCII data, translation 2-8
- assembler (see BASIC Assembler)
- assignment control statement considerations
  - CPU-CPU attachment A-6
  - 3270 DFF 8-58
- assignment stage
  - definition B-1
  - description 1-2
- asynchronous communication 2-12
- attachment configurations, CPU-CPU A-2
- attention fields (3270 DFF) 8-10
- attention identification character (see AID character)
- attribute (3270), definition B-1
- attribute character (3270 DFF)
  - defining 8-11
  - definition B-1
  - terminating 8-11
- attributes identifier
  - program 3-4
  - setting in COBOL 4-7
  - setting in FORTRAN 5-5
  - setting in RPG II 6-3
- attributes set, terminal 2-6
- attributes, terminal (see terminal attributes)
- ATTRID
  - \$NCIO macro 7-12
  - \$NPL macro 7-6
- auto answer 3-17
- auto call 3-17
- automatic skip
  - cursor positioning 8-16
  - input fields (3270 DFF) 8-26
  - output/input fields (3270 DFF) 8-26
- basic assembler programming for CCP
  - description 7-1
  - examples of using \$NCIO 7-12
  - macro support mnotes 7-14
  - programming restrictions 7-14
- batch processing 3-8
- bibliography C-1
- binary synchronous communications 2-13
- blank symbolic terminal name 2-5
- blank terminal name, use in SRT program logic 3-9
- block mode
  - input operations 2-14
  - length of record area 2-16
  - terminal attribute 2-33
- block mode operations, definition B-1
- BSCA (binary synchronous communications adapter)
  - input operations 2-14
  - line, allocating 2-14
  - output operations 2-16
  - record mode operations 2-8
- BSCA switched line
  - allocation 3-17
  - exchange identification characters 3-17
  - invite input operation 3-17
  - program-selected terminals (example) 3-18
  - requested terminals (example) 3-17
- BSCA terminals
  - attributes 2-6
  - communicating with 2-13
  - device control characters 2-13
  - record separator character 2-13
- BSCALINE assignment statement considerations
  - CPU attachment A-6
  - CPU attachments A-6
  - 3270 DFF 8-58
- BSCATERM assignment statement considerations, CPU attachment A-6
- BSYPRT (busy printer) support 2-17
- BTAM-to-CCP message sequence examples A-6
- buffer
  - line 2-7
  - TP (3270 DFF) 8-59
- busy printer (BSYPRT-YES) support 2-17

- call
  - auto 3-17
  - manual 3-17
- CALL statement
  - COBOL 4-9
  - FORTRAN 5-8
- calling on a switched line 3-17
- calling the communications service
  - subroutine
    - COBOL 4-9
    - FORTRAN 5-8
    - RPG II (see desired RPG II subroutine)
- carriage return (RPG II op code modifiers) 6-10
- carriage return and idle characters 2-11
- CCC (see copy control character)
- CCP
  - introduction 1-1
  - operation codes, RPG II 6-10
  - relation to other programs, display format facility 8-59
  - standard application program interface 2-1
- CCP operations summary D-1
- CCP stages 1-1
- CCP-to-BTAM message sequence examples A-6
- CCP work area field in parameter list 2-4
- CCPCIO 4-9
- CCPFIO 5-8
- CCPFMT 8-64
- channel-connected systems (see PORTLINE)
- character-oriented keys (3270) 8-60
- characteristics of 3270 DFF fields
  - description 8-5
  - input fields 8-8
  - output fields 8-6
  - output/input fields 8-9
  - selector pen fields 8-10
- checking return codes, recommended (see return codes)
- checkpoint/restart restriction 4-15
- classes of fields (3270 DFF) 8-4
- classes of terminals 3-1
- clear before writing 8-23
- CLEAR key (3270)
  - terminal operator actions 8-60
- CMCST CCP support 1-3
- COBOL
  - communications service subroutine (CCPCIO) 4-1, 4-9
  - compiling and link editing 9-1
  - CONSOL, use 4-15
  - count of outstanding invite inputs 4-7
  - determining the disk access value 9-25
  - examining returned information 4-11
  - parameter list 4-4
  - program preparation 9-1
  - programming considerations 4-15
  - programming examples
    - MRT program 4-24
    - SRT program 4-17
    - 3270 DFF program 8-100
  - record area 4-1
  - return code 4-3
  - setting the contents of the parameter list 4-4
- COBOL (Continued)
  - setting the contents of the parameter list, example 4-7
  - setting the record area 4-7
  - use of the standard interface 4-1
- combined files RPG II SPECIAL
  - defining 6-5, 6-7
  - put-then-get operation 6-19
- command interrupt mode, definition B-1
- command mode
  - CPU attachment considerations A-3
  - definition B-1
  - message/response prefixes A-3
  - message sequences, BTAM-to-CCP A-6
  - requesting terminal 3-2
  - switched lines 3-17
- command terminal 3-1
- command terminal, definition B-1
- comments 8-26, 8-40
- common I/O error return codes E-4, E-9
- common values, generate equates (\$NCOM macro) 7-1
- communicating magnetic card SELECTRIC typewriter (CMCST) 1-3
- communicating with MLTA terminals 2-13
- communication line buffer (see line buffer)
- communications interface
  - basic assembler 7-1
  - COBOL 4-1
  - FORTRAN 5-1
  - general description 2-1
  - RPG II EXIT/RLABL 6-24
  - RPG II SPECIAL 6-2
- communications management, definition B-1
- communications operation, performing 2-1
- communications programming topics 3-1
- communications service subroutines
  - COBOL (CCPCIO) 4-1, 4-9
  - definition B-1
  - FORTRAN (CCPFIO) 5-8
  - general description 2-2
  - RPG II
    - SUBR90 6-26
    - SUBR91 6-23
    - SUBR92 6-3
    - SUBR93 6-3
- COMPILE OCL statement 9-1
- compile time array 6-9
- compiling and link editing 9-1
- compiling chain task requesting programs 6-27
- concentrator function A-1
- configurations, CPU-CPU attachment A-2
- CONSOL
  - (see also 3277, 5471)
  - COBOL 4-15
  - FORTRAN 5-13
  - put-then-get 4-15, 5-13, 6-32
  - RPG II 6-32
  - symbolic files with 3-17
  - symbolic terminal name 2-5

- console output operations, maximum lengths
  - put 2-22
  - put-then-get 2-24
- continuation
  - display control form entry 8-25
  - field definition form entry 8-28, 8-37
  - specification (RPG II SPECIAL) 6-8
- control characters
  - device 2-21
  - line 2-7, 2-13
  - 3270, handling without 3270 DFF 4-17, 5-15, 6-32
- control station, definition B-1
- controlled termination of user program 2-49
- copy control character (CCC)
  - definition B-1
  - selecting 8-52
- copy operation (3270 DFF) 8-50
- copy operation (3270 DFF), RPG II 6-13
- copy operation, definition B-1
- copying the load module (program preparation) 9-6
- core index
  - APPLY CORE-INDEX (COBOL) 4-15
  - disk file consideration 9-8, 9-11
  - RPG II 6-32
- count of outstanding invite inputs
  - accept input operation 2-28
  - COBOL 4-7, 4-13
  - FORTTRAN 5-5, 5-13, 5-27
  - get operation 2-19
  - in parameter list 2-3
  - multiple requesting terminals 3-13
  - put-then-get operation 2-23
  - RPG II programming considerations 6-31
  - stop invite input operation 2-31
- CPU (central processing unit)
  - attached to CCP
    - command mode A-3
    - data mode A-5
    - procedure for issuing program requests A-5
  - device control characters 2-13
- CPU-to-CPU attachment
  - assignment considerations A-6
  - attachment configurations A-2
  - command mode attachment A-3
  - concentrator function A-1
  - data mode attachment A-5
  - generation considerations A-5
  - programming considerations A-2
  - recommendations and examples (BTAM) A-6
  - store-and-forward function A-1
  - subhost function A-1
- creating a direct file (restriction under Models 10 and 12) 9-8
- cursor position
  - autoskip 8-16
  - fieldname for initial positioning 8-23
  - put override 8-45
- cursor, definition B-2
- data
  - defining (3270 DFF) 8-17
  - numeric (3270 DFF) 8-19
  - user output area 8-46
- data (entry on field definition form) 8-28, 8-37
- data code translation 2-7
- data entry application, definition B-2
- data entry at terminal (3270 DFF) 8-17
- data field length (3270 DFF) 8-11
- data length (PRUF) 3-7
- data management modules (Models 10 and 12) 9-1
- data mode
  - attached CPU A-5
  - definition B-2
  - point-to-point BTAM-to-CCP example A-11
  - requesting terminal 3-2
- data mode escape
  - definition B-2
  - RPG II programming consideration 6-30
- data mode escape sequence command mode message/response prefixes A-3
- data mode message sequences, BTAM-to-CCP A-6
- data passed to the application program (3270 DFF) 8-17
- data set number 3-17
- data source (3270 DFF)
  - output/input fields 8-26, 8-37
  - SPD fields 8-27
- data terminal 3-1
- data terminal, definition B-2
- data transfer
  - input 2-6
  - output 2-8
- data translation
  - input 2-7
  - output 2-8
- data-truncated return code E-1
- data truncation
  - input 2-7
  - output 2-8
  - return code use E-1
- data with program request
  - maximum length 2-7, 2-29
  - requesting terminal 3-2
  - symbolic files 3-17
- decimal values, CCP operation codes D-1
- dedicated program
  - definition B-2
  - general description 3-5
- default TERMATTR statement 2-37
- defining attribute (3270 DFF) 8-11
- defining data (3270 DFF) 8-17
- defining elementary data items (COBOL) 4-1
- defining macro instructions, symbols used (BAL) 7-1
- defining RPG II SPECIAL files 6-5



- defining the parameter list (array)
  - COBOL 4-1
  - FORTRAN 5-1
  - RPG II 6-9
- demand file (RPG II SPECIAL)
  - defining 6-7
  - performing CCP operations 6-13
  - put-then-get operation 6-19
- designator character 8-11, 8-14
- designator character, definition B-2
- detail output (RPG II SPECIAL)
  - defining the file 6-7
  - performing CCP operations 6-16, 6-19
  - put-then-get operation 6-19
- detectable characteristic (3270 DFF fields) 8-4
- detectable field (see selector pen detectable field)
- determining terminal attributes (see get attributes operation)
- device
- device-busy, 3284/3286 printer 2-17
- device control characters
  - BSCA terminals 2-13
  - carriage return and idle characters 2-11
  - CPU to CPU 2-13
  - MLTA typewriter terminals 2-12
  - output data 2-8
  - record separators 2-10, 2-12
  - tab characters 2-11
- device, display control form entry (3270 DFF fields) 8-22
- device, file description specification (RPG II SPECIAL) 6-8
- DFF (3270 display format facility)
  - invite input with 2-27
  - put-no-wait with 2-26
  - put with 2-22
  - RPG II SRT inquiry example 8-95
- diagnostic messages
  - basic assembler macro support
  - mnotes 7-14
  - programming languages (see program language reference manual)
- direct file, restriction in creating (Models 10 and 12) 9-8
- disconnecting switched lines 3-20
- disk devices, requests for programs that use (see also disk files; disk unit)
- disk file recovery program 9-8, 9-11
- disk files
  - access value (FILES keyword of PROGRAM statement) 9-19
  - adding records 9-8, 9-10
  - considerations 9-8, 9-10
  - index sort 9-8, 9-10
  - loading as consecutive output 9-10, 9-13
  - loading direct files (Models 10 and 12 restrictions) 9-8, 9-11
- disk files (Continued)
  - master core index 9-8, 9-12
  - multivolume (not supported under CCP)
  - online requirements 9-8, 9-10
  - opening 9-8, 9-10
  - organizations and methods of
    - access 9-19
    - recovery 9-8, 9-11
  - sharing between program
    - level/partitions 9-8, 9-11
  - sharing Model 15 considerations 9-14
  - sharing Models 10 and 12
    - considerations 9-14
    - updating 9-8, 9-12
    - usage (RPG II) 6-32
    - using different name for same file
      - file 9-8, 9-11
- disk system management (DSM),
  - definition B-2
- disk unit
  - for display formats 8-24
  - for printer formats 8-35
- DISKFILE statement, symbolic files 3-16
- display concepts 8-56
- display control form 8-22
- display format control routine
  - functions 8-21
  - overview of operation 8-2
- display format facility (DFF)
  - COBOL inquiry program example 8-100
  - description 8-1
  - duplication 8-29
  - example 8-65
  - FORTRAN IV inquiry program 8-102
  - operations 8-41
  - RPG II MRT order entry example 8-72
  - storage area 8-58
- display format generation routine
  - display control form 8-22
  - display format specifications 8-22
  - display layout sheet 8-22
  - OCL considerations 8-32
  - options 8-62
  - printout examples 8-70, 8-98
  - summary diagram 8-3
- display format specification
  - examples 8-65, 8-97
- display format test routine 8-61
- display format, maximum number of fields 8-21
- display layout
  - examples 8-66, 8-97
  - planning 8-11
  - sheet 8-22
- display name (entry on display control form) 8-23
- DISPLAY operation (COBOL restriction) 4-15
- display operations, 3270 8-43
- display size (entry on display control form) 8-23

- distributed processing A-1
- DPF (dual programming feature) sharing
  - disk files 9-8
- DROP option (switched line disconnect) 3-20
- DSM (disk system management), definition B-2
- dual partitions, sharing disk files 9-8, 9-11
- dual programming feature (DPF) sharing disk files 9-8
- DUP key, terminal operator actions (3270 DFF) 8-60
- duplication 8-38
  
- EBCDIC translation 2-7
- editing with SUBR90 (RPG II) 6-27
- effective input length field (EFFL)
  - COBOL 4-7, 4-13
    - contents by operation code and return code E-13
  - examining
    - COBOL 4-13
    - FORTRAN 5-11
    - RPG II EXIT/RLABL 6-30
    - RPG II SPECIAL 6-15
  - FORTRAN 5-5, 5-9
  - in parameter list 2-3
  - RPG II 6-15, 6-30
- EFFL (effective input length field)
  - contents by return code E-13
- EM (end-of-message) orders (3270) 8-25
- end line
  - description 2-12
  - put operation 2-21
  - put-then-get operation 2-23
- end of block (EOB)
  - (see also block mode; put block)
  - put-no-wait operation 2-25
  - put operation 2-21
- end of file
  - entry on RPG II file description 6-8
  - false SPECIAL file (RPG II) 6-11
  - force end of file operation 6-11
  - with SUBR92 (RPG II SPECIAL) 6-32
- end of message (EM) orders (3270) 8-25
- end of transmission (EOT)
  - issuing after put record 2-16
  - put-no-wait operations 2-25
  - put operations 2-21
  - sending 2-16
  - sending and receiving 2-14
- ENTER key (3270 DFF) 8-59
- EOB (see end of block)
- EOT (see end of transmission)

- EQUATE statement
  - examples 9-3
  - link editing the program 9-3
- equates
  - common values (\$NCOM) 7-1
  - parameter list offsets (\$NPLO) 7-2
  - return code values (\$NRTV) 7-3
  - system (user security routine) 7-16
  - transient (user security routine) 7-16
- EQUIVALENCE statement (FORTRAN) 5-1
- erase all unprotected operation (3270 DFF) 8-53
- erase all unprotected operation (3270 DFF), RPG II op code 6-13
- ERASE EOF key (3270) 8-60
- erase indicator 8-46
- ERASE INPUT key (3270) 8-60
- error return code
  - (see also return code)
  - basic assembler 7-4
  - general description 2-2
  - per operation type E-8
  - put-then-get operation 2-24
- error return codes, I/O (see I/O error return codes)
- error, program 2-18
- examining returned information
  - COBOL 4-11
  - FORTRAN 5-9
  - RPG II EXIT/RLABL 6-30
  - RPG II SPECIAL 6-15
- examples, chain task request 2-40
- exception output
  - defining file 6-7
  - performing CCP operations 6-16
  - put-then-get operation 6-19
- exception return code
  - (see also return code)
  - basic assembler 7-4
  - per operation type (table) E-8
- exception/success condition return codes E-2
- exchange identification characters 3-17
- exchange identification characters, verification 2-35
- EXCPT (exception output) operation 6-18
- execution time, defining 3270 display data 8-17
- EXIT/RLABL communication interface (RPG II)
  - alphanumeric and numeric fields 6-27
  - editing with SUBR90 6-27
  - examining returned information 6-30
  - non-I/O operations 6-24
  - parameter array 6-9, 6-24
  - record area 6-23
  - RLABLs 6-23
  - setting the parameter 6-27
  - SUBR90 6-22
  - SUBR91 6-22
  - when used 6-1

EXIT to SUBR87 6-27  
 EXIT to SUBR88 6-27  
 EXIT to SUBR90 6-26  
 EXIT to SUBR91 6-23  
 extension specifications (RPG II) 6-9

false SPECIAL file (RPG II SUBR93)  
   defining 6-3  
   force end of file 6-11

feature support 1-3

field characteristics (3270 DFF) 8-5  
   alphanumeric field 8-5  
   input field types 8-8  
   intensity 8-6  
   modified data tag 8-5  
   numeric 8-5  
   output field types 8-7  
   output/input field types 8-9  
   protected 8-5  
   selector pen detectable field types 8-10  
   unprotected 8-5

field classes (3270 DFF) 8-4

field classes (3270 DFF), space requirements 8-11

field concepts (3270 DFF) 8-4

field definition form (3270 DFF) 8-17, 8-26, 8-36

field length  
   data passes to application program (3270 DFF) 8-17  
   entry of field definition form (3270 DFF) 8-26, 8-37, 8-38

FIELD MARK key 8-60

field name  
   entry on field definition form (3270 DFF) 8-26, 8-36, 8-38  
   initial cursor positioning, entry on field definition form 8-26, 8-36, 8-38  
   put override operation (3270 DFF) 8-45

field-oriented keys (3270 DFF) 8-60

field starting location  
   entry on field definition form (3270 DFF) 8-26, 8-36, 8-38

field type (3270 DFF)  
   field characteristics by type 8-6  
   input field types 8-8  
   output field types 8-7  
   output/input field types 8-9  
   put override 8-45  
   selector pen detectable field types 8-10

file considerations  
   disk 9-8  
   unit record 9-7

file description specifications (RPG II)  
   SPECIAL files 6-8  
     summary chart 6-7

file designation (RPG II) 6-8

file format (RPG II) 6-8

file management, definition B-2

file name (RPG II) 6-8

FILE OCL statements, symbolic files 3-16

file sharing considerations  
   Model 10 9-14  
   Model 15 9-14

file type (RPG II) 6-8

files  
   disk considerations 9-8  
   sharing 9-14  
   symbolic 3-16  
   unit record considerations 9-7

FILES keyword of PROGRAM statement, determining access values for disk 9-14

force end of file  
   description 6-12  
   when to issue 6-13, 6-20

form identifier  
   entry on display control form (3270 DFF) 8-23  
   entry on field definition form (3270 DFF) 8-26, 8-36  
   entry on printer control form (3270 DFF) 8-34

format control characters (3270)  
   format find routine 8-64  
   printable and unprintable 4-17, 5-15, 6-32

format find, definition B-2

format of input record (3270 DFF) 8-22

format of output record (3270 DFF) 8-22

formatting example (3270 DFF) 8-64

formatting messages for 3270 without DFF  
   COBOL 4-17  
   FORTRAN 5-15  
   RPG II 6-33

forms feed 8-35

FORTRAN IV  
   communications service subroutine (CCPFIO) 5-8  
   compiling the program 9-1  
   CONSOL, use 5-13  
   count of outstanding invite inputs 5-5  
   determining the disk access value 9-25  
   examining returned information 5-9  
   parameter list 5-2  
   program preparation 9-1  
   programming considerations 5-13  
   programming examples  
     DFF inquiry program 8-102  
     MRT program 5-21  
     SRT program 5-15  
   record area 5-1  
   return code 5-2, 5-9

## FORTRAN IV (Continued)

- setting the contents of the parameter list 5-3
- setting the record area 5-6
- use of the standard interface 5-1
- from filename (RPG II) 6-9
- from terminal (3270 DFF copy) 8-50

- generate equates for common value (\$NCOM) 7-1
- generate equates for parameter list offsets (\$NPLO) 7-2
- generate equates for return code values (\$NRTV) 7-3
- generate operation code/modifier values (\$NOPV) 7-3
- generate parameter list (\$NPL) 7-6
- generation considerations (CPU-to-CPU) A-5
- generation routine (3270 DFF) 8-2
- generation stage description 1-1
- generation stage, definition B-2
- generation statement considerations, CPU-to-CPU attachment A-5
- generation time, defining display data (3270 DFF) 8-17
- generic acquire terminals operation description 2-38
- get input operation (see stop invite input)
- get operation
  - description 2-19
  - RPG II 6-12
  - 3270 DFF 8-53
- get terminal attributes operation description 2-32
- get terminal attributes, portline exception 11-3
- get terminal attributes, RPG II 6-12
- GLOBAL statement (FORTRAN restriction under CCP) 5-13

- halt indicators (RPG II programming consideration) 6-32
- halt, unidentified record (avoid in RPG II) 6-20
- heading output (RPG II) 6-16, 6-19
- hexadecimal values, CCP operation codes D-1
- high intensity output field (3270 DFF) 8-7
- hold area, 3270 DFF output 8-58

## I/O error return codes

- blocking 2-13
- common E-4, E-9
- 3270 E-5, E-10
- 3735 E-6, E-11
- 3741 E-7, E-12
- identification-sequence
  - display control form (3270 DFF) 8-25
  - printer control form (3270 DFF) 8-36
- idle characters 2-11
- implied invite input (see data with program request)
- implied invite input, definition B-2
- INCLUDE statements 9-3
- index sort, disk files 9-8, 9-10
- indexed files, adding records 9-8, 9-10
- indicator
  - erase 8-46
  - modify data 8-46
- indicators
  - halt (RPG II) 6-32
  - 91 and 92 (RPG II) 6-5, 6-15
- initial mode, definition B-2
- INLEN-operand
  - \$NICO macro 7-12
  - \$NPL macro 7-6
- input class fields, length in display layout 8-13
- input data transfer 2-6
- input data translation 2-7
- input data, examining
  - COBOL 4-13
  - FORTRAN 5-13
  - RPG II EXIT/RLABL 6-30
  - RPG II SPECIAL 6-15
- input field class 8-4
- input field types 8-8
- input length, effective
  - examining in COBOL 4-13
  - examining in FORTRAN 5-11
  - examining in RPG II 6-15, 6-30
  - in parameter list 2-3
- input length, maximum
  - accept input operation 2-28
  - COBOL 4-7
  - FORTRAN 5-5
  - in parameter list 2-4
  - invite input operation 2-27
  - RPG II 6-14, 6-27
  - standard parameter list
- input operations
  - accept input 2-28
  - BSCA 2-14
  - COBOL 4-13
  - FORTRAN 5-8
  - get 2-19
  - message mode 2-14
  - modes 2-14
  - put-then-get 2-23
  - RPG II 6-12
  - stop invite input 2-31

input operations (Continued)  
 use of third parameter list field 2-3  
 3270 DFF 8-53

input/output error return codes E-4

input record area  
 processing in 3270 DFF 8-21  
 RPG II 6-2  
 RPG II, summary 6-21

input record format (3270 DFF) 8-21

input time, issuing RPG II  
 operations 6-12

input time, issuing RPG II operations,  
 summary 6-21

input type  
 entry on field description form (3270  
 DFF) 8-26

input, automatic skip, entry on field  
 definition form (3270 DFF) 8-26

inquiry program  
 COBOL example 4-17  
 COBOL example, 3270 DFF 8-100  
 example of logic (SRT) 3-3  
 FORTRAN example 5-15  
 FORTRAN example, 3270 DFF 8-102  
 RPG II example 6-32  
 RPG II example, 3270 DFF 8-95

inquiry-with-update, definition B-2

inquiry, definition B-2

inquiry, shutdown 2-48

intensity of field (3270 DFF) 8-6

interface  
 definition B-2  
 standard application program 2-1

intermediary data management modules 9-6

interrupt, reverse 2-19

interval timer used to clear

interval timer used to clear, 3284/3286  
 printer busy 2-17

invite input operation  
 BSCA switched line 3-17  
 count of  
 COBOL 4-13  
 FORTRAN 5-5  
 in parameter list 2-3  
 RPG II 6-31  
 description 2-27  
 example of logic 3-9  
 implied (see data with program request)  
 maximum input length 2-27  
 multicomponent terminals 2-6  
 relation to accept input 2-28  
 RPG II 6-13  
 3270 DFF 2-27, 8-54

INVOKE statement (FORTRAN programming  
 considerations) 5-13

IRANA disk file access value  
 consideration 9-14

IRUANA disk file access value  
 consideration 9-14

issuing RPG II operations 6-12

ITB 2-35

KEYBD statement (FORTRAN programming  
 consideration) 5-13

LABEL parameter FILE, use with  
 Model 15 9-10

length of data fields (3270 DFF) 8-11

length of entry (RPG II extension  
 specifications) 6-9

length of 3270 DFF field classes  
 input 8-13  
 output 8-13  
 output/input 8-13  
 selector pen detectable 8-14

length, program request data 3-7

line allocation (BSCA) 2-14

line buffer  
 definition B-2  
 input data transfer 2-7  
 new line and end line  
 considerations 2-12  
 space in (get terminal attributes) 2-33

line control characters  
 CPU-to-CPU (BTAM) examples A-6  
 description 2-13  
 input data transfer 2-7  
 input data translation 2-7  
 output data transfer 2-8

line length 8-34

line number (get terminal  
 attributes) 2-34

line transmission code translation 2-7

line type (get terminal attributes) 2-34

lines per page 8-34

link editing the program  
 Model 10 or 12 9-3  
 Model 15 9-1

LINK operand of PROCESS statement 9-1, 9-3

LINKADD parameter of // COMPILE  
 (Model 15) 9-1

load module, copying during program  
 preparation 9-6

loading direct files,  
 consideration 9-8, 9-11

loading files defined as consecutive output  
 disk file considerations 9-10, 9-13

loading the RPG II parameter array 6-9

lockout condition (disk file  
 caution) 9-14

logic of application programs  
 (examples) 3-8

look ahead fields, RPG II programming  
 consideration 6-32

- macro instruction (BAL), symbols used in defining 7-1
- macro mnotes (BAL) 7-14
- macro processor, correct version to use 7-1
- main file description (RPG II SPECIAL) 6-8
- making assignments (program preparation) 9-6
- manual answer 3-17
- manual call 3-17
- master index
  - COBOL 4-15
  - considerations 9-8, 9-11
  - RPG II 6-32
- master terminal name, definition B-2
- master terminal name, multicomponent terminals 2-6
- maximum input length
  - accept input operation 2-28
  - COBOL 4-7
  - FORTRAN 5-5
  - in parameter list 2-4
  - invite input operation 2-27
  - RPG II EXIT/RLABL 6-27
  - stop invite input operation 2-31
- maximum number of terminals in an MRT program 3-3
- MDT (see modified data tag)
- message formatting for 3270 without DFF
  - COBOL 4-17
  - FORTRAN 5-15
  - RPG II 6-32
- message mode operations
  - definition B-2
  - description 2-14
  - length of record area 2-16
- message prefixes, command mode (CPU-to-CPU) A-3
- message sequences, BTAM-to-CCP examples A-6
- method of access (disk files) 9-19
- MFCU file considerations 9-7
- minimum number of fields in a display (3270 DFF) 8-21
- MLMP (multiline/multipoint), definition B-2
- MLMP IOCS 2-13
- MLTA
  - definition B-3
  - switched lines 3-20
  - terminals, communicating with 2-12
  - typewriter terminals, device control characters 2-11
- MLTA terminal attributes 2-6
- MLTALINE assignment statement
  - considerations, CPU attachment A-6
- MLTATERM assignment statement
  - considerations, CPU attachment A-6
- mnotes (CCP basic assembler macros) 7-14
- mode of input operations
  - length of record area 2-14
  - with put-then-get 2-23
- mode, command (see command mode)
- modified data tag (MDT)
  - definition B-3
  - field characteristic 8-5
  - field classes affected by setting 8-5
  - ON or OFF 8-6
  - put override 8-44
- modifiers, operation code, RPG II 6-10
- modify data indicator 8-46
- MRT (see multiple requesting terminal program)
- MRTMAX keyword 3-3
- multicomponent terminals
  - definition B-3
  - symbolic terminal name
  - considerations 2-5
- multiple output lines, RPG II programming consideration 6-31
- multiple requesting terminal (MRT) program
  - accept input in 3-3
  - COBOL programming example 4-24
  - count of outstanding invite inputs 3-3
  - definition B-3
  - FORTRAN programming example 5-21
  - maximum number of terminals 3-3
  - MRTMAX keyword 3-4
  - program logic example 3-13
  - release terminal operation 3-13
  - releasing requesting terminal 3-3
  - RPG II programming example 6-38
  - shutdown return code E-13
  - shutdown return code 3-15
  - work areas for terminals 3-15
  - 3270 DFF example 8-72
- multipoint attachment (CPU-to-CPU) A-1
- multipoint command mode message sequences, BTAM-to-CCP A-7
- multivolume disk files
  - restriction against use under CCP 6-32, 9-8
- name (see program name, symbolic terminal name)
- name field in record area 2-4
- name of label exit (RPG II SPECIAL) 6-8
- NAME parameter of FILE 9-10
- negative numbers
  - RPG II EXIT/RLABL (3270 DFF) 6-27
  - 3270 DFF 8-19
- negative return codes
  - description E-1
  - summary tables E-4
- NEP (see never-ending program)

- never-ending program (NEP)
  - definition B-3
  - general description of attribute 3-4
  - SRT 3-4
- new line
  - general description 2-12
  - put operation 2-21
  - put-then-get operation 2-23
- new line (NL) orders, 3270 8-25, 8-37
- new screens 8-56
- non-display field characteristics (3270 DFF)
  - output field types 8-7
  - selector pen detectable field types 8-10
- non-I/O operations
  - RPG II EXIT/RLABL 6-24
  - RPG II SPECIAL 6-13, 6-20
- nonswitched CPU-to-CPU considerations A-1
- normal intensity output fields (3270 DFF) 8-7
- not new line 2-12
- null character 8-11
- null character, definition B-3
- number of entries per array (RPG II) 6-9
- number of entries per record (RPG II) 6-9
- number of fields in a display format 8-19
- numeric data (3270 DFF) 8-19
- numeric field
  - defining (3270 DFF) 8-19
  - field characteristic (3270 DFF) 8-5
  - RPG II EXIT/RLABL 6-27
- OCL considerations
  - display format generator (3270 DFF) 8-32
  - symbolic files 3-16
  - vs. assignment share 9-13
- offsets for parameter list generating in basic assembler 7-2
- online processing 3-8
- online requirements (disk files) 9-8, 9-10
- online, information returned from get terminal attributes 2-35
- OP-(\$NCIO macro operand) 7-10
- opening disk files
  - considerations 9-8, 9-10
  - symbolic files 3-17
- operation code
  - (see also operations)
  - and EFFL contents E-13
  - and RECA contents E-13
  - COBOL parameter list 4-3
  - FORTRAN parameter list 5-3
  - generating in basic assembler (\$NOPV macro) 7-3
  - in parameter list 2-3
- operation code (Continued)
  - PORTLINE 11-2
  - RPG II 6-10
  - RPG II parameter list 6-3
  - setting in COBOL 4-5
  - setting in FORTRAN 5-4
  - setting in RPG II 6-14, 6-27
  - summary table D-1
  - operation stage description 1-2
  - operation stage, definition B-3
  - operations
    - accept input 2-28
    - accept input, 3270 DFF 8-53
    - accept no-wait input 2-30
    - acquire terminal 2-36
    - BSCA input 2-14
    - copy (3270 DFF) 8-50
    - erase (3270 DFF) 8-53
    - forced end of file (RPG II) 6-11
    - generic acquire terminals 2-38
    - get 2-19
    - get terminal attributes 2-32
    - get, 3270 DFF 8-53
    - invite input 2-27
    - invite input, 3270 DFF 8-54
    - message mode input 2-14
    - non-I/O, performing in RPG II 6-13, 6-20, 6-24
    - performing using RPG II SPECIAL 6-13
    - port command request 2-39
    - PRUF put message 8-44
    - PRUF put override 8-45
    - put override 8-44
    - put 2-21
    - put block 2-16
    - put BSCA output 2-17
    - put message 2-16
    - put message, 3270 DFF 8-43
    - put-no-wait 2-25
    - put record 2-16
    - put RPG II SPECIAL 6-19
    - put-then-get 2-23
    - put with invite input (RPG II) 6-12, 6-12
    - put 3270 DFF 8-44
    - release and task chain 2-44
    - release terminal 2-42
    - release terminal, 3270 DFF 8-54
    - setting parameters in
      - basic assembler 7-8
      - COBOL 4-4
      - FORTRAN 5-3
      - RPG II 6-12, 6-27
    - shutdown inquiry 2-48
    - standard application program interface 2-1
    - stop invite input 2-31
    - stop invite input, 3270 DFF 8-53
    - task chain request 2-40
    - use of parameter list fields 2-3
    - wait 2-49
    - 3270 DFF operations 8-43

- operator identification card reader (3270 DFF) 8-60
- OPTIONS statement 9-3, 9-6
- order entry application (definition), RPG II example (3270 DFF) 8-72
- order entry application, definition B-3
- organization and method of access of disk files 9-20
- OUTLEN-operand
  - basic assembler \$NCIO macro 7-12
  - basic assembler \$NPL macro 7-6
- output
  - data source (3270 DFF) entry on field definition form 8-26
  - type (3270 DFF) entry on field definition form 8-26
- output area data 8-46
- output data area
  - COBOL 4-9
  - FORTTRAN 5-6
  - RPG II 6-2
- output data transfer 2-8
- output data translation 2-8
- output field class (3270 DFF) 8-4
- output field class (3270 DFF), length in display layout 8-13
- output field types (3270 DFF) 8-7
- output file, defining in RPG II 6-7
- output hold area (3270 DFF) 8-58
- output/input field class (3270 DFF)
  - description 8-4
  - entries on field definition form 8-26
  - length in display layout 8-13
- output/input field types (3270 DFF) 8-9
- output/input field, definition B-3
- output length
  - COBOL 4-6
  - FORTTRAN 5-6
  - in parameter list 2-3
  - RPG II 6-3, 6-19, 6-27
- output lines, multiple (RPG II) 6-31
- output operations
  - (see also operations)
  - BSCA 2-16
  - use of third parameter list field 2-3
- output record area (RPG II SPECIAL) 6-3, 6-21
- output record format (3270 DFF) 8-21
- output time, performing RPG II operations 6-12
- outstanding invite inputs
  - COBOL 4-13
  - FORTTRAN 5-5
  - in parameter list 2-3
  - RPG II 6-31
- overlay linkage editor control statements 9-3
- overlay screens (3270 DFF) 8-56
- overlying records (caution) 9-10, 9-13
- PA (program access) keys 8-60
- parameter array (RPG II)
  - (see also parameter list)
  - defining 6-9
  - EXIT/RLABL 6-23
  - SPECIAL 6-2
  - summary 6-21
  - when loaded 6-9
- parameter list
  - basic assembler 7-6
  - COBOL 4-1, 4-3
  - format 2-2
  - FORTTRAN 5-1
  - general discription 2-2
  - RPG II 6-2, 6-23
  - setting fields
    - basic assembler 7-7
    - COBOL 4-4
    - COBOL, example 4-7
    - FORTTRAN 5-3
    - FORTTRAN, example 5-6
    - RPG II 6-12, 6-24
    - RPG II, example 6-14
- password security option, definition B-3
- PAUSE statement (FORTTRAN restriction) 5-13
- performing CCP operations with SPECIAL (RPG II) 6-12
- PF (program function keys) 8-60
- PHASE statement 9-1, 9-3
- physical file 3-16
- physical file (see symbolic file)
- planning the display layout (3270 DFF) 8-11
- platen length 8-34
- PLIST operand (\$NCIO macro) 7-9
- point-to-point
  - CPU-to-CPU considerations A-1
- point-to-point command mode message sequences, BTAM-to-CCP A-8, A-9
- point-to-point data mode message sequences, BTAM-to-CCP A-11
- port command request operation 2-39
- PORTLINE application interface 11-1
- PORTLINE control statement 11-2
- pre-execution time array (RPG II) 6-9, 6-13
- prefixes, command mode
  - message/response A-3
- preparing application programs to run under CCP 9-1
- primary input file (RPG II SPECIAL)
  - defining 6-7
  - performing CCP operations with SPECIAL (RPG II) 6-13, 6-14
- printable format control characters 6-33
- printer
  - control (3270 DFF) 8-25, 8-38
  - name 8-34
  - size 8-34



- printer (3284/3286) device busy 2-17
- printer considerations, 3284/3286 2-17
- printer format generator routine
  - control form 8-33
  - duplication 8-38
  - field definition form 8-36
  - layout sheet 8-33
  - OCL 8-40
  - printer control 8-33
- Printer/Keyboard (see CONSOL)
- PROCESS statement 9-1, 9-3
- processing an input record area (3270 DFF) 8-21
- program access (PA) keys (3270 DFF) 8-60
- program actions for return codes E-1
- PROGRAM assignment statement
  - considerations, CPU-CPU attachment A-6
  - determining the disk file access value 9-19
  - 3270 DFF considerations 8-58
- program attention keys (3270 DFF) 8-60
- program attributes 3-4
- program function (PF) keys 8-60
- program interface to CCP 2-1
- program logic examples 3-8
- program management, definition B-3
- program name 2-4
- program name description 2-4
- program preparation 9-1
- program request
  - definition B-3
- program request command mode
  - message/response prefixes A-3
- program request count, definition B-3
- program request procedure, attached CPU A-4
- program request queueing (MRT program) 3-4
- program request under format (PRUF)
  - data with
    - maximum length 2-7, 2-29
    - single requesting terminal 3-9
    - symbolic files 3-17
  - definition B-3
  - description 3-5
  - PRUF RPG II order order entry program 8-103
- program selected terminals
  - BSCA switched lines 3-2
  - definition B-3, 3-2
  - description 3-2
  - how selected 3-2
  - restriction with certain terminals 3-2
  - with multiple requesting terminals 3-15
  - with single requesting terminal 3-11
- PROGRAM statement for portline
  - function 11-2
- program termination code, definition B-3
- program termination, controlled 2-49
- program testing 10-1
- program types 3-3
- program use of terminals 3-2
- programming a user security routine
  - Model 10 and 12 7-16
  - Model 15 7-22
- programming considerations
  - COBOL 4-15
  - CPU-CPU attachment A-2
  - FORTTRAN 5-13
  - RPG II 6-30
- programming examples
  - COBOL
    - MRT 4-24
    - SRT 4-17
    - 3270 DFF 8-100
  - FORTTRAN/MRT 5-21
  - FORTTRAN/SRT 5-15
  - FORTTRAN 3270 DFF 8-102
  - RPG II
    - MRT 6-38
    - SRT 6-32
    - 3270 DFF 8-72, 8-99
    - user security routine 7-19, 7-23
- programming restrictions
  - basic assembler 7-14
  - COBOL 4-15
  - FORTTRAN 5-13
  - RPG II 6-32
- programming user security routine
  - Model 10 and 12 7-16
  - Model 15 7-22
- prologue, transient
- prologue, transient, programming a user security routine 7-16
- protected field
  - definition B-3
  - description 8-5
- PRUF (program request under format)
  - definition B-3
  - description 3-5
  - program request length 3-7
  - put message 8-44
  - put override 8-45
- PRUF terminal, definition B-3
- PRUFLNG 3-6
- put block 2-16
  - BSCA output operation 2-16
  - operation 2-21
  - RPG II 6-13
- put message
  - BSCA output operation 2-16
  - operation 2-21
  - PRUF 8-44
  - RPG II 6-13
  - 3270 DFF 8-43
- put-no-wait operation
  - BSCA output operation 2-17
  - description 2-25
  - RPG II 6-12, 6-13
  - with invite input 6-12
  - 3270 DFF 8-44

- put operation
  - blank terminal name 3-9
  - description 2-21
  - record 2-16
  - RPG II 6-13
  - with invite input 6-12
- put override (PRUF) 8-45
- put override (3270 DFF) 8-45
- put override (3270 DFF), selecting the WCC 8-49
- put record 2-16
- put-then-get operation
  - BSCA output operation 2-17
  - COBOL 4-15
  - COBOL, output data area 4-9
  - description 2-23
  - FORTRAN 5-13
  - RPG II SPECIAL 6-19
- READ (FORTRAN restriction) 5-13
- read continue (READ TT) operation message sequences, BTAM-to-CCP A-6
- read initial (READ TI) operation message sequences, BTAM-to-CCP A-6
- READ operation (RPG II) (see demand files)
- RECA (record area) contents by return code E-13
- RECA-operand (basic assembler)
  - \$NCIO macro 7-12
  - \$NPL macro 7-6
- record area (RECA)
  - address in parameter list 2-4
  - COBOL 4-1
  - COBOL, setting contents 4-4
  - contents by operation code and return code E-13
  - description 2-4
  - format 2-4
  - FORTRAN 5-1
  - FORTRAN, setting 5-3, 5-6
  - get attributes operation 2-33
  - input and output formats (3270 DFF) 8-21
  - processing (3270 DFF) 8-21
  - RPG II EXIT/RLABL format 6-24
  - RPG II SPECIAL format 6-2
  - standard interface elements 2-3
  - user program, calculating size (3270 DFF) 8-54
- record concept 8-21
- record length
  - RPG II SPECIAL 6-8
  - zero, put block operation 2-16
- record mode operations
  - BSCA 2-8
  - definition B-3
  - description 2-14
  - length of record area 2-16
- record separators
  - device control character 2-10
  - input data transfer 2-7
- redefining the record area (COBOL) 4-2
- referencing saved information
  - COBOL 4-13
  - description 3-11
  - FORTRAN 5-11
  - RPG II 6-16
- release and task chain operation
  - description 2-44
  - use of third parameter list field 2-3
  - with sort 3-8
- release terminal operation
  - (see also release and task chain operation)
  - description 2-42
  - multicomponent terminals 2-6
  - multiple requesting terminals 3-3, 3-13
  - opening files (COBOL and Basic Assembler) 2-48
  - RPG II programming consideration 6-31
  - RPG II, when to issue 6-20
  - switched line 3-17, 3-20
  - symbolic file consideration 2-43
  - use of third parameter list field 2-3
  - 3270 DFF 8-54
- relocation address table 7-17
- repeat last character 8-37
- requesting terminal
  - BSCA 3-18
  - definition B-3
  - general description 3-2
- requests for programs (see program requests)
- reserved indicators 91 and 92 (RPG II) 6-5
- response prefixes, command mode A-3
- restrictions programming
  - basic assembler 7-14
  - COBOL 4-15
  - FORTRAN 5-13
  - RPG II 6-32
- return codes
  - 91 6-12
  - and EFL contents E-13
  - and RECA contents E-13
  - basic assembler 7-3
  - COBOL 4-3
  - FORTRAN 5-2, 5-9
  - in parameter list 2-2
  - negative, 3270 DFF 8-53
  - per operation type E-8
  - recommended checking 2-2
  - RPG II EXIT/RLABL 6-30
  - RPG II SPECIAL 6-3, 6-15
  - shutdown requested 2-29
  - success/exception E-2
  - summary tables E-1
  - 08 return code (RPG II consideration) 6-30
  - 3270 DFF 8-53

- returned information, examining
  - COBOL 4-11
  - FORTRAN 5-9
- reverse interrupt (RVI) 2-19
- RLABL for RPG II
  - SUBR90 6-26
  - SUBR91 6-23
- RPG II
  - chain task request 6-27
  - compiling program 9-1
  - core index 6-1
  - count of outstanding invite inputs 6-3
  - defining SPECIAL files 6-5
  - determining the disk access value 9-21
  - examining returned information 6-15, 6-30
  - EXIT/RLABL interface 6-22
  - EXIT/RLABL interface, when used 6-1
  - force end of file operation 6-11
  - handling 3270 format control characters 6-33
  - indicators reserved for CCP use 6-5
  - inquiry feature restriction 6-32
  - non-I/O operations 6-13, 6-20, 6-24
  - operation codes 6-10
  - parameter array for SPECIAL 6-2
  - parameter array for SPECIAL, defining 6-9
  - performing CCP operations with SPECIAL 6-12
  - program preparation 9-1
  - programming considerations 6-30
  - programming examples
    - MRT program 6-35
    - SRT program 6-32
    - 3270 DFF inquiry program 8-95
    - 3270 DFF order entry program 8-72
  - put-no-wait with invite input 6-12
  - put-then-get 6-19
  - put with invite input 6-12
  - record area
    - EXIT/RLABL 6-23
    - SPECIAL 6-2
  - SPECIAL interface 6-2
  - SPECIAL interface, when used 6-1
  - SUBR90 6-22
  - SUBR91 6-22
  - SUBR92 6-3
  - SUBR93 6-3
  - unidentified record halt, avoiding 6-20, 6-40
  - use of the standard interface 6-1
- RPG II codes for CCP operations D-1
- RVI 2-19
- RVI, definition B-4
- saved information, referencing
  - COBOL 4-13
  - description 3-11
  - FORTRAN 5-11
  - RPG II 6-16
- screen, entry on display control form (3270 DFF) 8-23
- secondary file (RPG II SPECIAL)
  - defining 6-7
  - performing CCP operations 6-13
- security routine, programming
  - Model 15 7-22
  - Models 10 and 12 7-16
- selecting input fields (3270 DFF) 8-45
- selecting the copy control character (3270 DFF) 8-52
- selecting the WCC (3270 DFF) 8-49
- selector pen detectable (SPD) field
  - definition B-4
  - field class 8-4
  - field types 8-10
  - modified data tag 8-6
  - planning the display layout 8-14
- serially reusable program 3-5
- serially reusable program, symbolic files 3-17
- Series/1 CCP support 1-4
- set attributes (acquire terminal operation) 2-36
- set control information for communications operation 7-8
- setting fields in the parameter list
  - COBOL 4-4
  - FORTRAN 5-3
  - RPG II EXIT/RLABL 6-24, 6-27
  - RPG SPECIAL 6-3, 6-13
- setting the record area
  - COBOL 4-7
  - FORTRAN 5-6
  - RPG II EXIT/RLABL 6-23, 6-24
  - RPG II SPECIAL 6-2, 6-16
- sharing access to disk files
  - between program levels/partitions 9-8, 9-11
  - Model 10 considerations 9-14
  - Model 15 considerations 9-14
- sharing unit record devices 9-7
- shutdown definition B-4
- shutdown inquiry operation
  - description 2-48
  - never-ending program 3-5
  - use of third parameter list field 2-3
  - when to issue in RPG II SPECIAL 6-20
- shutdown introduction 1-2
- shutdown requested by operator 2-30
- shutdown return code
  - accept input consideration 2-30
  - multiple requesting terminals 3-13, 3-15
  - never-ending program 3-5
  - single requesting terminal 3-11

- sign-on security routine 7-16
- sign-on, definition B-4
- single requesting terminal (SRT) program
  - COBOL 4-17
  - definition B-4
  - FORTRAN example 5-15
  - general description 3-3
  - logic 3-9, 3-11
  - multiple copies running 3-3
  - RPG II 6-32
  - 3270 DFF examples
    - COBOL 8-100
    - FORTRAN IV 8-102
    - RPG II 8-95
- SIOC (see PORTLINE)
- sort programs 3-7
- sorts and task chaining 3-8
- sorts with chain task requests 2-40
- space requirements for field classes (3270 DFF) 8-11
- spanned records 2-10
- spanned records, get terminal attributes 2-35
- SPD field (see selector pen detectable field)
- SPECIAL files (RPG II)
  - CCP communication interface 6-1, 6-2
  - communications service subroutines 6-3
  - defining 6-5
  - defining the parameter array 6-9
  - false SPECIAL file 6-11
  - force end of file operation 6-11
  - indicators reserved for CCP use 6-5
  - parameter array 6-2
  - performing CCP operations 6-12, 6-21
  - put-then-get operation 6-19
  - record area 6-2, 6-21
  - summary 6-21
  - using both SPECIAL and EXIT/RLABL 6-31
- special program attributes 3-4
- spooling unit record devices (Model 15) 9-8, 9-10
- SRT program (see single requesting terminal program)
- stages, CCP 1-1
- standard application program interface to CCP
  - CCP, FORTRAN IV use 5-1
  - CCP, RPG II use 6-1
  - COBOL use 4-1
  - description 2-1
- standby mode, definition B-4
- start address of CCP user programs (Model 15) 9-1
- starting location of field (3270 DFF) 8-11
- startup introduction 1-2
- startup, definition B-4
- stop invite input operation
  - definition B-4
  - description 2-31
  - 3270 DFF 8-53
- STOP statement (COBOL restriction) 4-15
- storage areas (3270 DFF) 8-58
- store-and-forward function A-1
- subhost definition B-4
- subhost function A-1
- subroutines, communications service 2-2
- SUBR87 6-27
- SUBR88 6-27
- SUBR90 6-22
- SUBR91 6-22
- SUBR92 6-3
- SUBR92, end of file 6-32
- SUBR93 6-3
- subterminal names 2-6
- success/exception condition return codes E-2
- success/exception return codes E-8
- support, terminal/feature 1-3
- switched lines
  - allocation
    - BSCA 3-17
    - MLTA 3-20
  - answering 3-17
  - auto answer 3-17
  - auto call 3-17
  - BSCA 3-17
  - calling 3-17
  - command terminals 3-17
  - CPU-to-CPU considerations A-1
  - data set number 3-17
  - data terminals 3-17
  - definition 3-17
  - disconnect considerations 3-20
  - exchange ID characters 3-17
  - invite input operation 3-17
  - manual answer 3-17
  - manual call 3-17
  - MLTA 3-20
  - release terminal operation 3-17
  - telephone number 3-17
- symbolic files
  - CONSOL considerations 2-5
  - definition B-4
  - description 3-16, 3-16
  - release terminal considerations 2-43
- symbolic subterminal name
  - definition B-4
  - description 2-6
- symbolic terminal name
  - description 2-4
  - examining after an operation
    - COBOL 4-13
    - FORTRAN 5-9
    - RPG II EXIT/RLABL 6-30
    - RPG II SPECIAL 6-16
  - in record area 2-4
  - multicomponent terminal considerations 2-5
  - naming rules 2-4
- SYMFILE statement
- SYMFILE statement, symbolic files 3-16
- system equates (user security routine) 7-16
- SYSTEM statement, 3270 DFF considerations 8-58

- system task, definition B-4
- System/3 CCP support 1-3
- System/3 CPU-CPU attachment considerations A-2
- System/32 CCP support 1-4
- System/34 CCP support 1-4
- System/360-370 CCP support 1-3
- System/360-370 CPU-CPU attachment considerations A-2
- System/7 CCP support 1-3
- System/7 CPU-CPU attachment considerations A-2
  
- table file (RPG II restriction) 6-32
- task (see also system task, user task)
- task chain request operation (see also release and task chain operation)
  - accept input with 2-29
  - compiling programs 6-27
  - description 2-40
  - RPG II 6-27
  - with sort 3-8
  - with sorts 2-40
- task chaining and sorts 3-8
- task chaining, definition B-4
- task identification, definition B-4
- telephone number 3-17
- teleprocessing line control characters 2-7
- TERMATTR statement
  - considerations, CPU-CPU attachment A-6
  - default 2-37
  - 3270 DFF considerations 8-58
- terminal allocation 2-1
- terminal attributes (see also get terminal attributes)
  - definition B-4
  - general description 2-6
- terminal attributes identifier in parameter list 2-3
- terminal class 3-1
- terminal class, returned 2-33
- terminal name (see symbolic terminal name)
- terminal operator actions (3270 DFF) 8-59
- terminal reference identifier, definition B-4
- terminal sign-on security routine (user-written) 7-16
- terminal support 1-3
- terminal unit block (TUB) 2-33
- terminating attribute 9-14
- termination code 2-18
- termination of user programs
  - program errors 2-18
  - shutdown inquiry operation 2-48
- testing programs 10-1
- testing return codes (see return code)
- text transparency, COBOL record area consideration 4-1
  
- TIME operation (RPG II) 2-17
- TNAME operand (\$NCIO macro) 7-12
- to terminal (3270 DFF copy operation) 8-50
- total output (RPG II SPECIAL) 6-7, 6-16, 6-19
- TP buffer 8-59
- transfer of output data 2-8
- transfer, data 2-6
- transient prologue and equates 7-16
- translation, data 2-6
- translation, definition B-4
- transmission code translation 2-7
- transparency
  - COBOL record area consideration 4-1
  - get terminal attributes 2-35
- tributary station, definition B-4
- truncated data 2-7
- truncation, definition B-4
- TUB (terminal unit block) 2-33
- TYPERS statement (FORTRAN restriction) 5-13
- types of fields (3270 DFF) 8-6
- types of fields (3270 DFF), entry of field definition form 8-26
- typewriter terminals, device control characters 2-11
  
- unidentified record halt, avoiding (RPG II SPECIAL) 6-20
- unit record data management, link editing considerations 9-3
- unit record devices
  - definition B-4
  - sharing between program levels 9-7
  - spooling (Model 15) 9-8, 9-10
- unit record file considerations
  - Model 15 9-8, 9-10
  - Models 10 and 12 9-7
- unprintable 3270 format control characters
  - COBOL 4-17
  - FORTRAN 5-15
  - RPG II 6-33
- unprotected field (3270 DFF)
  - definition B-4
  - field characteristic 8-5
- updating shared files 9-14
- user-defined symbolic terminal names 2-4
- user output area data 8-46
- user program record area, calculating size (3270 DFF) 8-54
- user program storage considerations (3270 DFF) 8-59
- user security data program (\$CCPAU) 7-16
- user security routine, programming
  - Model 10 or 12 7-16
  - Model 15 7-22

- user task, definition B-4
- using the CONSOL
  - COBOL 4-15
  - FORTRAN 5-13
  - RPG II 6-32
  
- variable length records 2-7
- variable length records, get attribute operation 2-35
- VARY OFFLINE command 3-20
- vertical forms feed 8-36
  
  
- wait operation description 2-49
- WCC (see write control character)
- work areas for terminals 3-11, 3-15
- write continue (WRITE TT) operation message sequences, BTAM-to-CCP A-6
- write control character (WCC)
  - default 8-24
  - definition B-4
  - display control form entry 8-24
  - how to determine 8-24
  - put override operation 8-45
  - selecting 8-49
- write initial (WRITE TI) operation message sequences, BTAM-to-CCP A-6
- write inquiry (WRITE TQ) operation message sequences, BTAM-to-CCP A-6
- write reset (WRITE TR) operation message sequences, BTAM-to-CCP A-6
- WRITE statement (FORTRAN restriction) 5-13
  
  
- Z in first position of record area 2-36
- zero record length, BSCA output operations 2-16
  
  
- 08 return code (RPG II programming consideration) 6-30
- 1050 data communication system
  - CCP support 1-3
  - device control characters 2-12
  - multicomponent terminal considerations 2-5
- 1403 printer considerations 9-7
- 1403 printer sharing consideration 2-1
- 1442 card read punch considerations 9-7
- 2740 communication terminal
  - CCP support 1-3
  - device control characters 2-12
- 2741 communication terminal
  - CCP support 1-3
  - device control characters 2-12
- 3270 data with program request 2-7
- 3270 display format facility (see DFF)
- 3270 display format facility (see entries under display)
- 3270 display operations 8-45
- 3270 format control characters
  - COBOL 4-17
  - FORTRAN 5-15
  - RPG II 6-33
- 3270 information display system
  - CCP support 1-3
  - device control characters 2-11
  - I/O error return codes E-5,E-10
- 3275 information display system CCP support 1-3
- 3277 display station (See CONSOL)
- 3284/3286 printer busy, clearing 2-17
- 3284/3286 printer considerations 2-17
- 3600 finance communication system CCP support 1-4
- 3735 programmable terminal
  - CCP support 1-3
  - I/O error return codes E-6,E-11
  - transmitting FDPs 2-9
- 3741 data station CCP support 1-3
- 3741 data station I/O error return codes E-8,E-12
- 3767 communication terminal CCP support 1-3
- 5100/5110 portable computer CCP support 1-3
- 5203 printer considerations 9-7
- 5230 data collection system CCP support 1-3
- 5471 printer/keyboard (see CONSOL)
- 5937 display station CCP support 1-3
- 91 indicator (RPG II) 6-5
- 92 indicator (RPG II) 6-5





**International Business Machines Corporation**

**General Systems Division  
4111 Northside Parkway N.W.  
P.O. Box 2150  
Atlanta, Georgia 30301  
(U.S.A. only)**

**General Business Group/International  
44 South Broadway  
White Plains, New York 10601  
U.S.A.  
(International)**

IBM System/3 CCP Programmer's Reference (File No. S3-36) Printed in U.S.A. GC21-7579-6

GC21-7579-6



**READER'S COMMENT FORM**

**Please use this form only to identify publication errors or request changes to publications.** Technical questions about IBM systems, changes in IBM programming support, requests for additional publications, etc., should be directed to your IBM representative or to the IBM branch office nearest your location.

**Error in publication** (typographical, illustration, and so on). **No reply.**

*Page Number*    *Error*

**Inaccurate or misleading information in this publication.** Please tell us about it by using this postage-paid form. We will correct or clarify the publication, or tell you why a change is not being made, provided you include your name and address.

*Page Number*    *Comment*

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Name \_\_\_\_\_

Address \_\_\_\_\_

● No postage necessary if mailed in the U.S.A.

Cut Along Line

Fold and tape

Please do not staple

Fold and tape

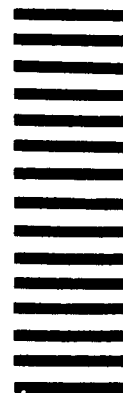


NO POSTAGE  
NECESSARY IF  
MAILED IN THE  
UNITED STATES

**BUSINESS REPLY MAIL**  
FIRST CLASS      PERMIT NO. 40      ARMONK, N. Y.

POSTAGE WILL BE PAID BY . . .

IBM CORPORATION  
General Systems Division  
Development Laboratory  
Publications, Dept. 245  
Rochester, Minnesota 55901



Fold and tape

Please do not staple

Fold and tape



**International Business Machines Corporation**

**General Systems Division  
4111 Northside Parkway N.W.  
P.O. Box 2150  
Atlanta, Georgia 30301  
(U.S.A. only)**

**General Business Group/International  
44 South Broadway  
White Plains, New York 10601  
U.S.A.  
(International)**

IBM System/3 CCP Progn. ner's Reference (File No. S3-36) Printed in U.S.A. GC21-7579-6

### READER'S COMMENT FORM

**Please use this form only to identify publication errors or request changes to publications.** Technical questions about IBM systems, changes in IBM programming support, requests for additional publications, etc, should be directed to your IBM representative or to the IBM branch office nearest your location.

**Error in publication** (typographical, illustration, and so on). **No reply.**

*Page Number*    *Error*

**Inaccurate or misleading information in this publication.** Please tell us about it by using this postage-paid form. We will correct or clarify the publication, or tell you why a change is not being made, provided you include your name and address.

*Page Number*    *Comment*

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Name \_\_\_\_\_

Address \_\_\_\_\_

Cut Along Line

Fold and tape

Please do not staple

Fold and tape

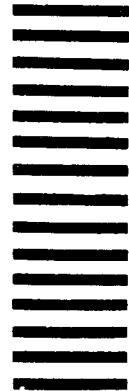


NO POSTAGE  
NECESSARY IF  
MAILED IN THE  
UNITED STATES

**BUSINESS REPLY MAIL**  
FIRST CLASS      PERMIT NO. 40      ARMONK, N. Y.

POSTAGE WILL BE PAID BY . . .

IBM CORPORATION  
General Systems Division  
Development Laboratory  
Publications, Dept. 245  
Rochester, Minnesota 55901



Fold and tape

Please do not staple

Fold and tape



**International Business Machines Corporation**

**General Systems Division  
4111 Northside Parkway N.W.  
P.O. Box 2150  
Atlanta, Georgia 30301  
(U.S.A. only)**

**General Business Group/International  
44 South Broadway  
White Plains, New York 10601  
U.S.A.  
(International)**

IBM System/3 CCP Program      's Reference (File No. S3-36)      Printed in U.S.A.      GC21-7579-6

**READER'S COMMENT FORM**

**Please use this form only to identify publication errors or request changes to publications.** Technical questions about IBM systems, changes in IBM programming support, requests for additional publications, etc, should be directed to your IBM representative or to the IBM branch office nearest your location.

**Error in publication** (typographical, illustration, and so on). **No reply.**

*Page Number    Error*

**Inaccurate or misleading information in this publication.** Please tell us about it by using this postage-paid form. We will correct or clarify the publication, or tell you why a change is not being made, provided you include your name and address.

*Page Number    Comment*

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Name \_\_\_\_\_

Address \_\_\_\_\_

● No postage necessary if mailed in the U.S.A.

Cut Along Line

Fold and tape

Please do not staple

Fold and tape

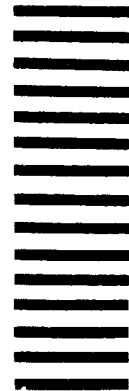


NO POSTAGE  
NECESSARY IF  
MAILED IN THE  
UNITED STATES

**BUSINESS REPLY MAIL**  
FIRST CLASS      PERMIT NO. 40      ARMONK, N. Y.

POSTAGE WILL BE PAID BY . . . .

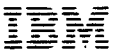
IBM CORPORATION  
General Systems Division  
Development Laboratory  
Publications, Dept. 245  
Rochester, Minnesota 55901



Fold and tape

Please do not staple

Fold and tape



**International Business Machines Corporation**

**General Systems Division  
4111 Northside Parkway N.W.  
P.O. Box 2150  
Atlanta, Georgia 30301  
(U.S.A. only)**

**General Business Group/International  
44 South Broadway  
White Plains, New York 10601  
U.S.A.  
(International)**

IBM System/3 CCP Progra... 's Reference (File No. S3-36) Printed in U.S.A. GC21-7579-6

**READER'S COMMENT FORM**

**Please use this form only to identify publication errors or request changes to publications.** Technical questions about IBM systems, changes in IBM programming support, requests for additional publications, etc, should be directed to your IBM representative or to the IBM branch office nearest your location.

**Error in publication** (typographical, illustration, and so on). **No reply.**

*Page Number    Error*

**Inaccurate or misleading information in this publication.** Please tell us about it by using this postage-paid form. We will correct or clarify the publication, or tell you why a change is not being made, provided you include your name and address.

*Page Number    Comment*

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Name \_\_\_\_\_

Address \_\_\_\_\_

● No postage necessary if mailed in the U.S.A.

Cut Along Line

Fold and tape

Please do not staple

Fold and tape

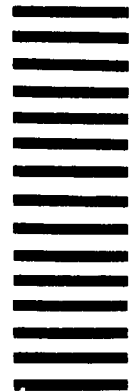


NO POSTAGE  
NECESSARY IF  
MAILED IN THE  
UNITED STATES

**BUSINESS REPLY MAIL**  
FIRST CLASS      PERMIT NO. 40      ARMONK, N. Y.

POSTAGE WILL BE PAID BY . . .

IBM CORPORATION  
General Systems Division  
Development Laboratory  
Publications, Dept. 245  
Rochester, Minnesota 55901



IBM System/3 CCP Progr

's Reference (File No. S3-36)

Printed in U.S.A. GC21-7579-6

Fold and tape

Please do not staple

Fold and tape



**International Business Machines Corporation**

**General Systems Division  
4111 Northside Parkway N.W.  
P.O. Box 2150  
Atlanta, Georgia 30301  
(U.S.A. only)**

**General Business Group/International  
44 South Broadway  
White Plains, New York 10601  
U.S.A.  
(International)**