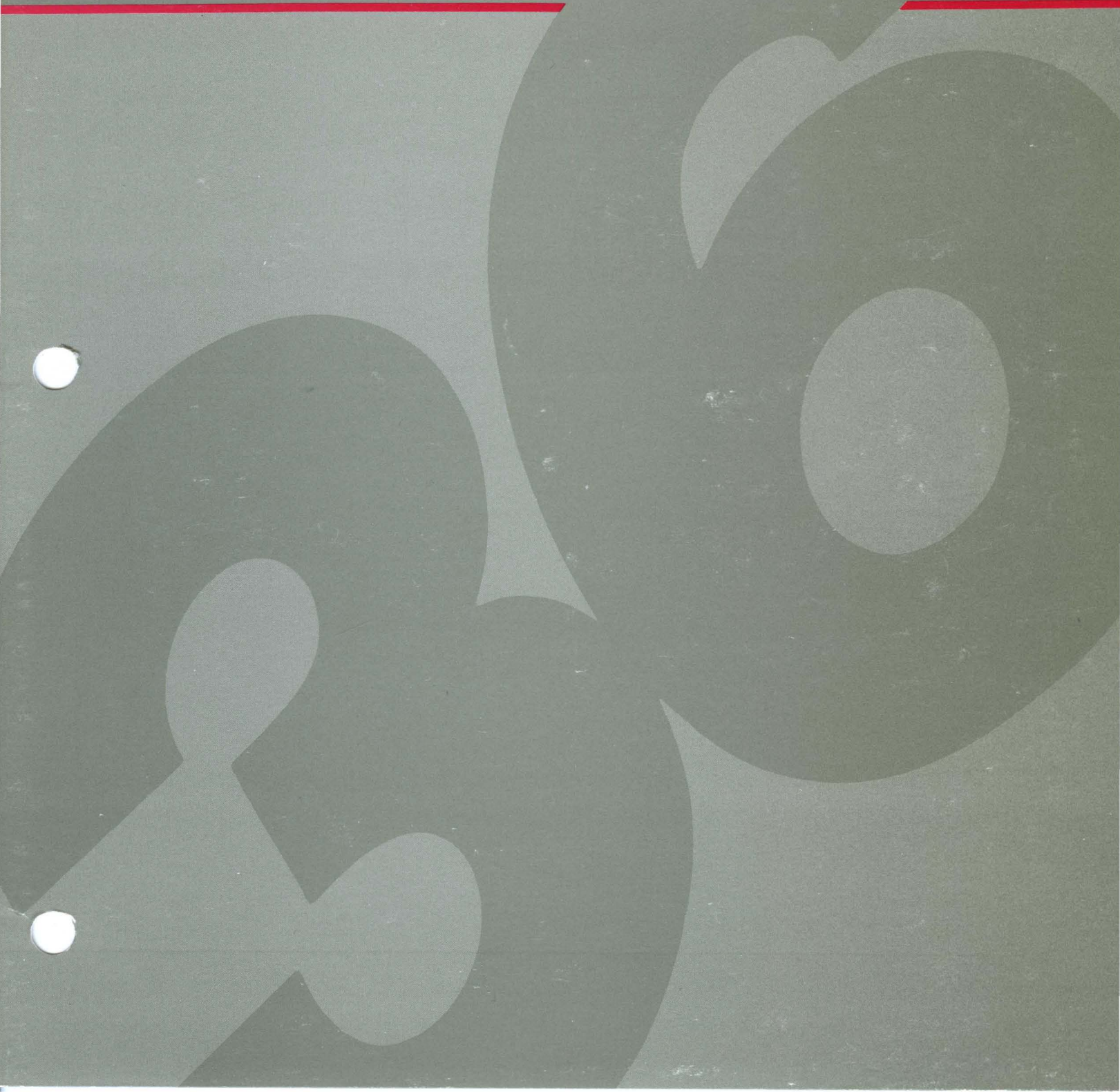
 System/36

Programming for Subsystems
and Intra Subsystem
Reference

**Interactive
Communications
Feature:**



IBM System/36

**Interactive Communications Feature:
Programming for Subsystems and
Intra Subsystem Reference**

File Number
S36-30

Order Number
SC21-9533-0

First Edition (October 1986)

This is a new manual that replaces, in part, SC21-7910. Changes are periodically made to the information herein; any such changes will be reported in subsequent revisions or Technical Newsletters.

This edition applies to Release 5, Modification Level 0, of IBM System/36 System Support Program Products (Program 5727-SS1 for the 5360 and 5362 System Units, and Program 5727-SS6 for the 5364 System Unit), and to all subsequent releases and modifications until otherwise indicated.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent program may be used instead.

The numbers at the bottom right of illustrations are publishing control numbers and are not part of the technical content of this manual.

Publications are not stocked at the address given below. Requests for IBM publications should be made to your IBM representative or to your IBM-approved remarketer.

This publication could contain technical inaccuracies or typographical errors. A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Information Development, Department 245, Rochester, Minnesota, U.S.A. 55901. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. •

Contents

About This Manual	vii
What you should know . . .	viii
If you need information about other SSP-ICF subsystems . . .	ix
If you need more information . . .	ix
Chapter 1. Introduction to the Interactive Communications Feature	1-1
Elements Used in SSP-ICF Sessions	1-3
Acquired Sessions	1-4
Remotely Started Sessions	1-6
SSP-ICF Subsystems	1-8
Types of System/36 Subsystems	1-9
Combinations of Subsystems	1-10
System/36 Communications Line Support	1-10
Communications Features Supported by Subsystems	1-11
Sharing a Communications Line	1-12
System/36 Storage and Session Considerations	1-14
Storage Requirements	1-14
Active Session Limits	1-14
Enabling and Disabling Subsystems	1-15
Enabling a Subsystem	1-15
Disabling a Subsystem	1-19
Chapter 2. Programming SSP-ICF with Assembler	2-1
Assembler Macroinstructions	2-3
\$DTFW Macro	2-4
\$DTFO Macro	2-8
\$ALOC, \$OPEN, and \$CLOS Macros	2-10
\$WSIO Macro	2-12
\$EVOK Macro	2-18
Assembler Operations Summary Chart	2-21
Return Codes	2-23
Interactive Communications Assembler Subroutines	2-23
Assembler Coding Examples	2-24
Chapter 3. Programming SSP-ICF with BASIC	3-1
BASIC Statements Used for Communications	3-3
OPEN Statement (Acquiring Sessions)	3-4
OPEN Statement Examples	3-6
READ Statement (Receiving Data)	3-8
READ Statement Examples	3-9
WAITIO Statement (Waiting for Input)	3-11
WAITIO Statement Example	3-11
WRITE Statement (Performing Operations within a Session)	3-12

WRITE Statement Operations	3-13
Starting Remote Programs (Evoke Operations)	3-14
Sending Data (Put Operations)	3-18
Ending Communications Transactions (End of Transaction Operations)	3-19
Ending Sessions (End of Session Operation)	3-20
Additional WRITE Statement Operations	3-20
CLOSE Statement (Closing Files for Sessions)	3-24
ATTRIBUTE\$ Intrinsic Function (Getting Session Attributes)	3-25
BASIC Operations Summary Chart	3-27
Checking Return Codes in BASIC	3-28
ERR Code Values	3-29
RETCODE\$ Values	3-30
Notes About Writing BASIC Programs for SSP-ICF	3-31
BASIC Coding Examples	3-31
Chapter 4. Programming SSP-ICF with COBOL	4-1
COBOL Statements Used for Communications	4-3
SELECT Statement (Defining the Transaction File)	4-4
ACQUIRE Statement (Acquiring Sessions)	4-6
ACCEPT Statement (Checking Session Status)	4-7
Session Status Information	4-8
READ Statement (Receiving Data)	4-10
WRITE Statement (Performing Operations within a Session)	4-12
WRITE Statement Operations	4-13
Starting Remote Programs (Evoke Operations)	4-14
Sending Data (Put Operations)	4-15
Ending Communications Transactions (End of Transaction Operations)	4-16
Ending Sessions (End of Session Operation)	4-17
Additional WRITE Statement Operations	4-17
DROP Statement (Releasing a Session)	4-21
COBOL Operations Summary Chart	4-22
Return Code Processing in COBOL	4-23
COBOL Coding Examples	4-23
Chapter 5. Programming SSP-ICF with RPG II	5-1
File Description Specifications	5-2
RPG II Communications Operations	5-3
Starting Remote Programs (Evoke Operations)	5-3
Sending Data (Put Operations)	5-5
Request to Change Direction Operation	5-6
Set Timer Operation	5-7
Negative Response Operations	5-8
Cancel Operations	5-9
Fail Operation	5-9
End of Session Operation	5-9
WORKSTN Operations	5-10
ACQ (Acquire) Operation	5-10
REL (Release) Operation	5-11
NEXT Operation	5-12
READ Operation	5-13
RPG Cycle Input	5-14
RPG II Operations Summary Chart	5-14

Return Code Processing in RPG II	5-15
INFSR Subroutine Coding Considerations	5-18
RPG II Status Values	5-19
RPG II Programming Considerations	5-20
Using Continuation Options on the File Description Specifications	5-20
SRT and MRT Program Considerations	5-22
End-of-File Considerations	5-23
Release Considerations	5-23
Restrictions for WORKSTN Files	5-24
Input and Output Considerations	5-24
RPG II Coding Examples	5-25

Chapter 6. The Intra Subsystem	6-1
Overview of the Intra Subsystem	6-3
Setting Up an Intra Subsystem	6-5
CNFIGICF Procedure	6-5
Subsystem Member Definition	6-7
Modifying a Subsystem Configuration	6-10
Enabling and Disabling the Intra Subsystem	6-10
Starting Communications Sessions That Use the Intra Subsystem	6-11
SESSION Statement	6-12
Procedure Start Requests	6-13
Communications Operations for the Intra Subsystem	6-14
Accept Input Operation	6-17
Acquire Operation	6-18
Cancel Operations	6-20
End of Session Operation	6-22
Evoke Operations	6-23
Fail Operation	6-33
Get Operations	6-35
Invite Operation	6-37
Negative Response Operations	6-38
Put Operations	6-39
Release Operation	6-41
Request to Change Direction Operations	6-42
Set Timer Operation	6-44
Intra Subsystem Return Codes	6-45

Glossary	G-1
-----------------	-----

Index	X-1
--------------	-----

About This Manual

The information in this manual supersedes and replaces its respective part of the *System/36 Interactive Communications Feature: Reference* manual, SC21-7910. This manual and the other associated manuals made from the *SSP-ICF Reference* resulted from the SSP Release 5.0 repackaging of the SSP-ICF 6001 and 6002. This manual supports the base communications feature Program Number 6001-SS1.

This reference manual is intended primarily for application programmers who write communications programs that use the SSP-ICF. It contains programming information for both System/36 programmers and remote system programmers. This manual contains two major units of information:

- A description (in Chapters 2 through 5) of each macroinstruction or language statement used to perform communications operations. Each macroinstruction used in assembler language for communications is described in Chapter 2. Each language statement used in BASIC, COBOL, or RPG II is described in Chapters 3 through 5, respectively.
- A description (in Chapter 6) of the Intra subsystem, which includes:
 - A description of the configuration displays used to configure the subsystem member definitions.
 - A description of the remote system generation or configuration requirements and the startup requirements needed for remote programs to communicate with System/36.
 - A description of the SESSION OCL statement used by a System/36 program to start a communications session.
 - A description of all the subsystem input and output operations used to communicate in a session. Included are summary charts of all the operation codes (by language), examples of each operation in each language, and language-dependent information.
 - A description of the programming considerations for System/36 and for the remote system.
 - A complete description of every return code that a subsystem can send to a program.

This manual contains appendixes describing subsystem operation codes, return codes, conversion considerations, and character sets. It also contains a glossary that defines the terms introduced and used in this manual.

Notes:

1. *Throughout this manual, the term **remote system** refers to the system or device with which System/36 is communicating. For the Intra subsystem, the term refers to the same System/36 because Intra supports communications only between programs on the same system.*
2. *This manual follows the convention that **he** means **he** or **she**.*

What you should know . . .

Before you use this manual, you should know or have the following information:

- You should be familiar with System/36 programming terminology, particularly work station programming, and you should be able to program in whatever language you intend to use. In some instances, you must also be familiar with the terminology of the remote system.
- You should know the concepts of data communications as described in the *Data Communications Concepts* manual, GC21-5169.
- You should understand the information and examples presented in Chapters 1 through 5 and in the appropriate subsystem chapter in the *System/36 Interactive Communications Feature: Guide and Examples* manual, SC21-7911. The *SSP-ICF Guide and Examples* manual (the shortened title used in this reference manual) introduces SSP-ICF concepts, and it provides coding examples of programs written in assembler, BASIC, COBOL, and RPG II—programs that use one of the SSP-ICF subsystems to communicate with programs on remote systems.
- You should have the *Planning for Data Communications* workbook, SA21-9441, which is part of the packet *What to Do Before Your Computer Arrives*, SBOF-4773.

Note: This manual may refer to products that are announced, but are not yet available. Such information is for planning purposes only and is subject to change before general availability.

If you need information about other SSP-ICF subsystems

...

The following SSP-ICF manuals contain detailed information about other communications subsystems:

- *Interactive Communications Feature: Base Subsystems Reference*, SC21-9530, contains information about the APPC, BSCCL, CCP, and Peer communications subsystems. The shortened title used in this manual is “SSP-ICF Base Subsystems Reference.”
- *Interactive Communications Feature: Upline Subsystems Reference*, SC21-9532, contains information about the CICS, IMS, and SNUF communications subsystems. The shortened title used in this manual is “SSP-ICF Upline Subsystems Reference.”
- *Interactive Communications Feature: Finance Subsystem Reference*, SC21-9531, contains information about the Finance communications subsystem. The shortened title used in this manual is “SSP-ICF Finance Subsystem Reference.”

If you need more information . . .

The following System/36 manuals contain additional information you may need when you use the Interactive Communications Feature:

- *Guide to Publications*, GC21-9015 lists the manuals in the System/36 library, lists the tasks that are described in the System/36 manuals, and provides a master glossary of System/36 terms.
- *Changing Your System Configuration*, SC21-9052 contains instructions for installing Interactive Communications Feature support.
- *System Security Guide*, SC21-9042 describes how to implement various levels of security on System/36.
- *Using System/36 Communications*, SC21-9082 describes in detail using System/36 for communications.
- *System Problem Determination*, SC21-7919 for the 5360 System Unit, SC21-9063 for the 5362 System Unit, or SC21-9375 for the 5364 System Unit provides procedures to help you find the cause of communications problems.
- *System Messages*, SC21-7938 describes the system messages that are displayed when you operate the Interactive Communications Feature.
- *System Reference*, SC21-9020 describes the OCL statements, system utilities, and system procedures you need when you use System/36 and the Interactive Communications Feature.

- *Performing the First System Configuration for Your System*, SC21-9067 contains instructions for performing the first system configuration for your system.
- *Getting Started with Interactive Data Definition Utility*, SC21-8003 introduces the interactive data definition utility (IDDU) and describes how to create data definitions for use with the Intra and APPC subsystems.
- *Advanced Peer-to-Peer Networking (APPN) Guide*, SC21-9471, describes how to use APPN to configure, use, and maintain the extended networking capabilities for the System/36 family.

You may need to refer to one or more of the following System/36 language reference manuals while using this manual.

- *Programming with Assembler*, SC21-7908
- *Programming with BASIC*, SC21-9003
- *Programming with COBOL*, SC21-9007
- *Programming with RPG II*, SC21-9006

Depending upon the type of SSP-ICF subsystem that you use for communications, you may need to use non-System/36 manuals that describe the remote system or operating system with which your System/36 will be communicating. These manuals are listed in the remote programming considerations section of each applicable subsystem chapter.

A few references are made in this manual to System/36 communications subsystems that are not included in the Interactive Communications Feature. Information about those subsystems is contained in the following System/36 manuals that describe those features:

- *Using the Asynchronous Communications Support*, SC21-9143 describes the asynchronous communications support, which is part of the base Communications feature. This support includes the Asynchronous subsystem, the file transfer subroutines, and the Interactive Terminal Facility.
- *3270 Device Emulation Guide*, SC21-7912 describes the BSC 3270 and SNA 3270 subsystems, which are part of the 3270 Device Emulation feature. SNA 3270 can share a communications line with the SNUF subsystem, SNA MSRJE, and the APPC subsystem.
- *Multiple Session Remote Job Entry Guide*, SC21-7909 describes the Multiple Session Remote Job Entry feature. SNA MSRJE can share a communications line with the SNUF subsystem, SNA 3270, and the APPC subsystem.

- *Distributed Disk File Facility Reference Manual*, SC21-7869 contains information about installing, setting up, and operating the Distributed Disk File Facility. The Peer subsystem must be used with this facility.
- *Communications and Systems Management Guide*, SC21-8010 contains information about the Communications and Systems Management feature. This feature includes change management (DSNX) support and problem management (alert) support. The SNUF and APPC subsystems are used with this feature.
- *Distributed Data Management Guide*, SC21-8011 contains information about the Distributed Data Management feature. The APPC subsystem is used with this feature.

Chapter 1. Introduction to the Interactive Communications Feature

Elements Used in SSP-ICF Sessions	1-3
Acquired Sessions	1-4
Remotely Started Sessions	1-6
SSP-ICF Subsystems	1-8
Types of System/36 Subsystems	1-9
Combinations of Subsystems	1-10
System/36 Communications Line Support	1-10
Communications Features Supported by Subsystems	1-11
Sharing a Communications Line	1-12
System/36 Storage and Session Considerations	1-14
Storage Requirements	1-14
Active Session Limits	1-14
Enabling and Disabling Subsystems	1-15
Enabling a Subsystem	1-15
Enabling Multiple Remote Locations (SNA Subsystems Only)	1-16
ENABLE Procedure Command	1-18
Disabling a Subsystem	1-19
Disabling Multiple Remote Locations (SNA Subsystems Only)	1-20
DISABLE Procedure Command	1-20

The System/36 Interactive Communications Feature (SSP-ICF) allows program-to-program communications between System/36 and other systems. SSP-ICF is provided as a feature of the System/36 System Support Program Product (SSP). The information needed to use SSP-ICF is contained in the manual *Using System/36 Communications* and in this reference manual.

SSP-ICF includes support for program-to-program communications between systems using BSC or SNA as well as communications between programs within the same system. SSP-ICF also allows programs on other systems to initiate System/36 procedures, and it allows System/36 programs to initiate programs or procedures on other systems without remote system operator intervention.

This chapter contains information that applies to all SSP-ICF subsystems. This chapter:

- Summarizes briefly the main elements used in an SSP-ICF session
- Introduces all the subsystem types
- Identifies the communications line features that are used by the subsystems
- Describes the ENABLE and DISABLE procedure commands

SSP-ICF provides problem determination and link verification by means of the SSP-ICF debug program. The program allows you to save information on disk about each SSP-ICF operation while your program(s) is running. You can then display or print the information to help you find the cause of an SSP-ICF problem. The procedure for running the debug program is ICFDEBUG. This procedure is described in the manual *Using System/36 Communications*.

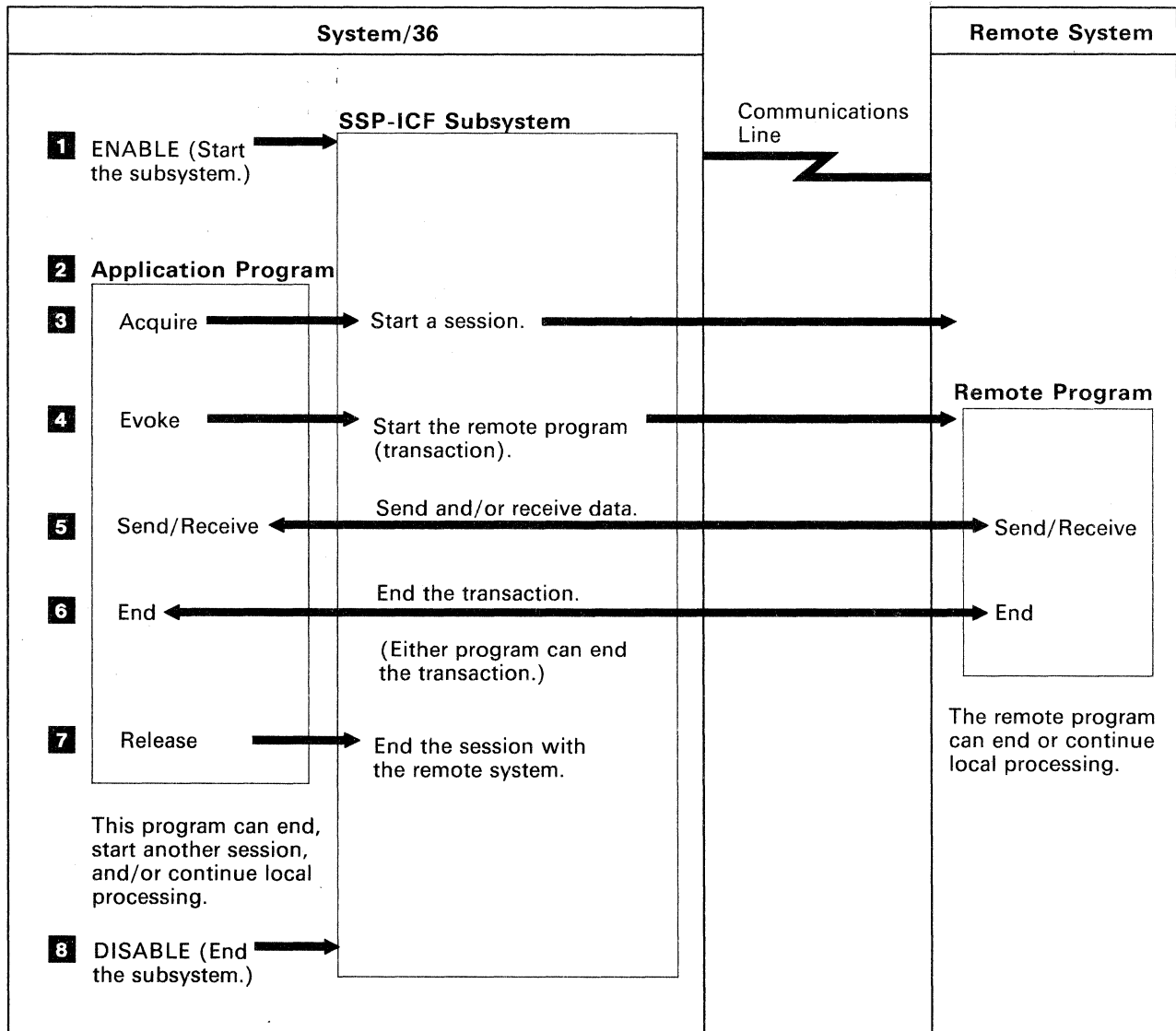
Elements Used in SSP-ICF Sessions

The following two sections summarize the main communications elements that exist while System/36 programs are using SSP-ICF to communicate with other programs. A detailed description of these elements (subsystems, programs, sessions, transactions, and data) is presented in the *SSP-ICF Guide and Examples* manual.

A session can be started either by a program on System/36 or by a program on a remote system. When a program on System/36 starts the session with an acquire operation, the session is called an **acquired session**. When a remote program starts the session by sending a procedure start request to System/36, the session is called a **remotely started session**.

Acquired Sessions

The following figure shows the order in which events occur and the elements involved when a System/36 application program (your program) starts a session with the remote system:



S7910002-0

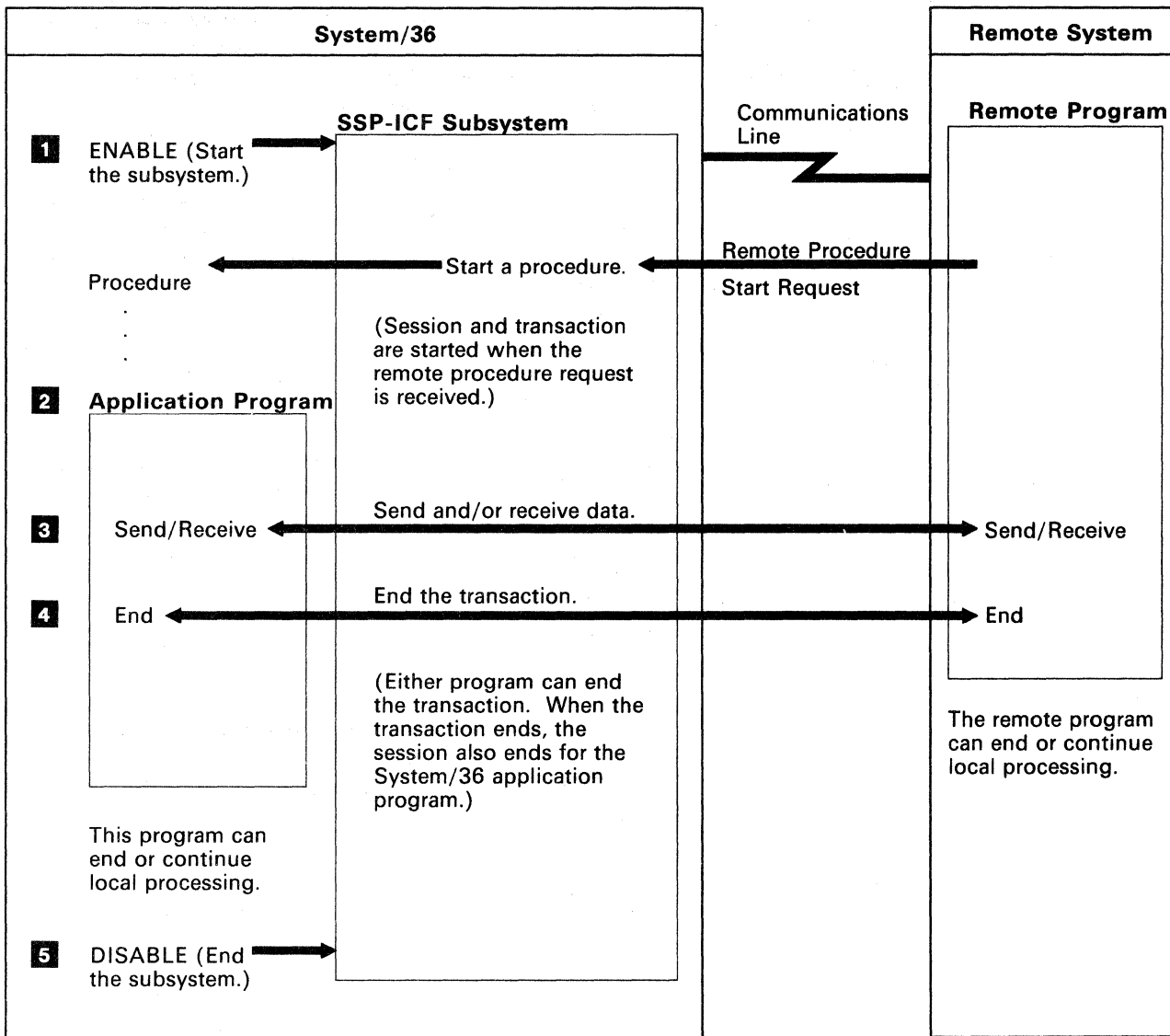
- 1** A **subsystem** must be **enabled** (started) before programs can use it to communicate with a remote system. The **ENABLE** procedure command is used to start the subsystem.
- 2** The System/36 **application program** that will communicate with the program at the remote system must be started, usually via a user-written procedure.
- 3** The System/36 program must start a **session** with the remote system before communications can begin. Your program starts a session when it issues an **acquire** operation.

When your program starts (acquires) the session, a **SESSION OCL** statement (associated with your program) is used to specify the session ID and the location name (used to identify the remote system) to be associated with the session. For some subsystems, the **SESSION** statement also defines some of the subsystem-dependent parameters for the session. These parameters remain in effect until the program terminates.

- 4** Within each session, **transactions** can be started (evoked) to allow your program to communicate with remote programs. A transaction is started when your program issues an **evoke** operation to start a specified remote program.
- 5** Within each transaction, **data** can be sent and received between your program and the program on the remote system.
- 6** When all data has been sent and/or received, either your program or the remote program can end the transaction. Your program ends the transaction using one of the end of transaction operations (evoke end of transaction or put end of transaction). When the remote system ends the transaction, the subsystem indicates this by the return code it sends to your program.
- 7** When all transactions have ended, your program should release the session. Your program can do this by using either the **release** operation or the **end of session** operation.
- 8** When the subsystem is no longer needed, it can be **disabled** using the **DISABLE** procedure command to free System/36 resources used by the subsystem.

Remotely Started Sessions

The following figure shows the order in which events occur and the elements involved when the remote system starts the session by sending a remote procedure start request:



S7910001-1

- 1** A **subsystem** must be **enabled** (started) before a remote system can use it to communicate with a System/36 program. The **ENABLE** procedure command is used on System/36 to start the subsystem.
- 2** A System/36 procedure (the procedure that starts your program) is started by the subsystem when it receives a **procedure start request** from the remote system. The procedure then starts the application program that will communicate with the program on the remote system. The **session** and the **transaction** are also started when the procedure start request is received.

Because the remote system started the session and the transaction, no acquire or evoke operation is issued by the application program. Your program can, however, acquire other sessions with the remote system once your program is running (depending upon the type of subsystem you are using).

- 3** Either one of two types of information can be sent with the procedure start request: **parameters** for the procedure or **data** for your program. If data is sent, your program must use an input operation to receive this data. If no data is expected with the procedure start request, your program can issue either an input or output operation depending upon the procedures previously set up with the remote system.
- 4** When all data has been sent or received, either program can end the transaction. When the transaction ends, the session for your program also ends.

Note: If an APPC subsystem is being used, all session groups that were started should be stopped. Before the APPC subsystem is disabled, the STOPGRP procedure command is used to stop a session group(s).

- 5** When the subsystem is no longer needed, it can be **disabled** using the **DISABLE** procedure command to free System/36 resources used by the subsystem.

For both acquired sessions and remotely started sessions, each level of events associated with an element can occur repeatedly within the next higher level. For example, multiple sessions can be acquired and released within the same program, and multiple programs can be run without disabling and enabling the subsystem configuration. For more information, see the *SSP-ICF Guide and Examples* manual.

The connection between the levels is maintained by the following parameters:

- **subsystem configuration name:** Specifies the particular subsystem to be enabled, using the ENABLE procedure command.
- **location name:** Specified during subsystem configuration. The location name is included on the SESSION statement to identify the remote location being referenced.
- **session ID:** Specified on the SESSION statement and used in your program when it acquires the session.

Because the SESSION statement associates the remote location with a session identifier, the remote location can be changed in the subsystem configuration without requiring a change in your program.

- **session group name:** Specified during subsystem configuration only for the APPC subsystem. The session group name is included on the SESSION statement to identify the session group name associated with the session.

SSP-ICF Subsystems

Interactive communications between application programs are accomplished using SSP-ICF and a subsystem. Several subsystem types are provided so that System/36 can communicate with various remote systems that have different communications methods (such as BSC or SNA). A **subsystem**, designed for a specific remote system, makes it unnecessary to handle most system-dependent considerations when coding System/36 application programs.

A System/36 program issues *communications operations* to communicate with a remote system via one of the SSP-ICF subsystems. The subsystem informs the program of the success or failure of each operation by sending the proper *return code* to the program. Several of the communications operations and return codes can be used with any of the SSP-ICF subsystem types; some operations and return codes are used with only one or two subsystem types. A program written to be used with one type of subsystem may, with little or no change, be used to communicate with a different type of subsystem. How much change is needed in the program depends on which two subsystems are involved, which communications operations are used, and which return codes are being checked for.

Types of System/36 Subsystems

When two programs on the *same* System/36 are to communicate with each other, the Intra subsystem is used. When two programs are on *different* System/36s, the BSCEL, Peer, or APPC subsystem is used. (The BSCEL subsystem is used for BSC. The Peer or APPC subsystem is used for SNA.) When two programs are on a System/36 and *another type* of remote system, the APPC subsystem or other subsystems may be used.

All of the SSP-ICF subsystems are shown in the following table. The order in which they are shown is the order in which they are described in this manual and in the *SSP-ICF Guide and Examples* manual. (System/36 can also have other communications subsystems that are not part of SSP-ICF.)

System/36 SSP-ICF Subsystem	Communicates With
Intra	Other programs in the same System/36
BSC Equivalence Link (BSCEL)	System/36, System/34, Series/1, and others
BSC CCP	System/3 Model 15 CCP
BSC CICS	CICS/VS (BTAM)
BSC IMS/IRSS	IMS/VS via IRSS (BTAM)
Finance	3601 and 4701 Finance Controllers, and 3694 Document Processor
SNA Peer	System/36 and System/34
SNA Upline Facility (SNUF)	CICS/VS and/or IMS/VS with ACF/VTAM
Advanced Program-to-Program Communications (APPC)	System/38, System/36, and CICS/VS

Other subsystems that can be used for communications include the following: BSC 3270, SNA 3270, BSC MSRJE, and SNA MSRJE. These communications subsystems are not part of SSP-ICF and are documented in other manuals. Those manuals are identified at the end of the list of manuals given under “If you need more information...” at the beginning of this manual.

Combinations of Subsystems

Several subsystem configuration members can be stored in System/36, and several subsystems can be enabled at the same time. All the subsystems that are enabled (using the configuration members to define the attributes of the enabled subsystems) do not have to be of the same type. The number of subsystems that can be enabled is determined by the number of communications lines available and whether any lines are being shared by SNA-type subsystems.

Only *one* BSC 3270 subsystem can be active on System/36 at any time; it can be active with all other combinations of subsystems, but it must be on a line by itself.

Depending on the number and types of subsystems that are active at one time, it is possible that the response time on System/36 may increase. If a particular combination of subsystems produces undesirable system performance, you should try changing a subsystem's attributes (such as the length of its data records being sent). For other information about subsystem performance considerations, refer to the individual subsystem reference manual.

System/36 Communications Line Support

System/36 can have up to eight telecommunications lines. Each telecommunication line can be one of the following types (all the lines do not have to be the same type):

- Point-to-point switched (manual answer, automatic answer, manual call, or automatic call)
- Point-to-point nonswitched
- Multipoint tributary

In addition, if your system has the LAN Attachment feature, you can have up to two Token-Ring Network lines (lines 9 and 10).

Each SSP-ICF subsystem (except Intra) requires at least one communications line to communicate with a remote system. An Intra subsystem can be enabled regardless of the number of line-dependent subsystems enabled on the System/36.

The maximum number of lines available is controlled by the communications features installed on your system. Refer to the manual *Using System/36 Communications* for information about communications features.

Communications Features Supported by Subsystems

The following chart shows all the subsystems supported on System/36, and it shows the communications line features that each subsystem can use. The chart shows the SSP-ICF subsystems, and it includes, for your information, the other System/36 communications subsystems that are not part of SSP-ICF.

Subsystem	Line Types Supported			Features Supported			
	Point -to- Point	Multi- point	Token- Ring Network	Auto- call	X.25	X.21	LAN Attach Feature
<i>SSP-ICF Subsystems:</i>							
Intra ¹	—	—	—	—	—	—	—
BSCCL	Yes	Yes	No	Yes	No	Yes	No
CCP	Yes	Yes	No	Yes	No	Yes	No
CICS	Yes	Yes	No	Yes	No	Yes	No
IMS	No	Yes	No	No	No	Yes ²	No
Finance	Yes	Yes	No	No	Yes ⁵	Yes ³	No
Peer	Yes	Yes	No	Yes	Yes	Yes	No
SNUF	Yes	Yes	Yes	Yes	Yes	Yes	Yes
APPC	Yes	Yes	Yes	Yes	Yes	Yes	Yes
<i>Other Subsystems:⁴</i>							
Asynchronous	Yes	No	No	No	Yes	No	No
BSC 3270	No	Yes	No	No	No	Yes	No
SNA 3270	Yes	Yes	Yes	Yes	Yes	Yes	Yes
BSC MSRJE	Yes	Yes	No	Yes	No	Yes	No
SNA MSRJE	Yes	Yes	Yes	Yes	Yes	Yes	Yes
PC Support/36	No	No	Yes	No	No	No	Yes
¹ The Intra subsystem does not use any communications lines; it handles communications only between two programs in the same System/36. ² The IMS subsystem supports X.21 on nonswitched lines only. ³ The Finance subsystem supports X.21 on nonswitched lines and on switched lines in autoanswer mode only. ⁴ These communications subsystems are not part of SSP-ICF; they are included in other communications features. ⁵ The Finance subsystem supports X.25 on nonswitched lines and on only permanent virtual circuits over that connection.							

Sharing a Communications Line

BSC subsystems (BSC, CCP, CICS, IMS/IRSS, BSC 3270, and BSC MSRJE) cannot share a communications line with another subsystem.

If you are using SNA, subsystems can share the same line, with the following restrictions:

- For an SNA/SDLC line:
 - The Peer primary, APPC primary, and Finance subsystems can share a line if they are configured for a nonswitched line. The APPC subsystem, however, cannot share the same line member with the Finance subsystem. A line member that the APPC and Peer subsystems are sharing can run concurrently with a line member used by the Finance subsystem. In addition, remote work station support (RWS) can share this same line.
 - A Peer secondary subsystem cannot share a line with any other subsystem.
 - The SNA 3270, SNA MSRJE, APPC secondary, and SNUF subsystems can share a line provided all the subsystems use the same line member. In addition, a SNUF subsystem used for Communications and Systems Management change management and an APPC subsystem used for Communications and Systems Management alert support can also share the line, provided they use the same line member. SNA 3270 is described in the *3270 Device Emulation Guide*, MSRJE is described in the *Multiple Session Remote Job Entry Guide*, and change management and alert support are described in the *Communications and Systems Management Guide*.
- For an SNA/X.25 line:
 - All SNA subsystems specifying **primary** for the line member data link protocol, along with RWS, can share a line, but only one configuration of each may be enabled at one time for the line. Therefore, you cannot enable Peer primary and Peer secondary, or APPC primary and APPC secondary, on the same line.
 - The SNA 3270, SNA MSRJE, APPC secondary, and SNUF subsystems, specifying **secondary** for the line member data link protocol, can share a line, provided all the subsystems use the same line member. In addition, a SNUF subsystem used for Communications and Systems Management change management and an APPC subsystem used for Communications and Systems Management alert support can also share the line, provided they use the same line member. SNA 3270 is described in the *3270 Device Emulation Guide*, MSRJE is described in the *Multiple Session Remote Job Entry Guide*, and change management and alert support are described in the *Communications and Systems Management Guide*.

- The SNA 3270, SNA MSRJE, APPC, and SNUF subsystems specifying **negotiable** for the line member data link protocol, may run simultaneously on the same line. The System/36 may communicate with multiple remote systems at the same time and on the same line. At the remote end of the line, the remote systems can be configured as primary, secondary, or negotiable. At the local end of the line, the line member must be configured as negotiable.
- For an SNA/LAN communications line:
 - The SNA 3270, SNA MSRJE, APPC, and SNUF can run simultaneously on the same line. The System/36 can communicate with multiple remote systems at the same time and on the same line.
 - The RWS, PC Support/36, and APPC, 3270, SNUF, and MSRJE subsystems can all share a line, but only one configuration of each may be enabled at one time on the line. APPC, 3270, SNUF, and MSRJE can share a line only if they use the same line member. On Token-Ring Network, all the these subsystems use negotiable line member protocol.

System/36 Storage and Session Considerations

Storage Requirements

If multiple subsystems of different types are enabled at the same time, the response time on System/36 may increase, especially on a system that has the minimum size for main storage. You may need to increase the size of main storage, reduce the number of jobs running in the system, or have fewer subsystems enabled at the same time.

Active Session Limits

A maximum of 360 sessions can be active concurrently for all the subsystems enabled on System/36. This maximum includes two groups of sessions that also have limits:

- A maximum of 260 **user-acquired** sessions (acquired by user programs with SESSION statements) can be concurrently active.
- A maximum of 100 other sessions (remotely started and/or specially acquired) can be concurrently active.
 - **Remotely started sessions** are started by procedure start requests sent by remote programs (or by 3741-type devices).
 - **Specially acquired sessions** are started by BASIC programs without using a SESSION statement. Sessions can be specially acquired if the program is written in BASIC and if the session is to be an *interactive* session.

Enabling and Disabling Subsystems

Enabling a Subsystem

To run a System/36 program that uses SSP-ICF for communications, you must enable (start) the particular subsystem configuration that you want to use. (The subsystem configuration must have already been defined by using the CNFIGICF procedure to specify the attributes of the subsystem, the remote system, and the communications line to be used.) The name of the subsystem configuration which consists of a line member and a subsystem member, must be specified on the ENABLE procedure command, along with the line number of the communications line to be used by the subsystem. (An Intra subsystem does not use a communications line.)

You can enable a subsystem by having the ENABLE procedure command automatically run after IPL (initial program load). See the *System Reference* manual for a description of how to specify a procedure (named #STRTUP2) to be run automatically after IPL.

When the ENABLE procedure command is used to start a subsystem, it performs the following functions:

- Ensures compatibility between the subsystem configuration and the communications hardware.
- Determines whether the requested communications line is available.
- Loads the subsystem support for that type of subsystem (such as BSCEL or Peer) if it is not already active.
- Loads any other required tasks (BSC or SNA) if they are not already active.
- Loads the subsystem configuration that contains the attributes of the subsystem that is being enabled.
- Determines, for SNA subsystems, the remote locations with which communications are to be established.
- Assigns storage for required data areas and buffers.

The ENABLE procedure command only *prepares* the local end of the line to communicate with the remote location; the remote location must also be prepared for communication. When both ends are prepared and a physical connection is established, communication can begin. However, for an APPC subsystem, a session group must be started before a session can be established. This session may be started by the operator command STRTGRP, or STRTGRP will run automatically provided a STOPGRP was not previously issued.

A program that uses SSP-ICF for communications can be loaded before the subsystem is enabled, but no sessions for that subsystem can be started until it is enabled. After the subsystem has established communications, programs can begin acquiring sessions using that subsystem. The subsystem waits for an acquire operation to be issued by a System/36 program or for a procedure start request to be issued by a remote program.

If the line type set by the configuration record does not correspond to the line type (identified by line number on the ENABLE procedure command) to be used by the subsystem, a message is issued and the ENABLE procedure command is terminated unless message option 0 is provided, which allows you to continue and automatically use the line type specified in the configuration record. You can use the SETCOMM or ALTERCOM procedure to change the line type. These two procedures are described in the manual *Using System/36 Communications*.

For Finance, Peer, SNUF, and APPC subsystems, when the ENABLE procedure command is used to activate communications with a particular location and the subsystem configuration is already active, the procedure ensures that the subsystem configuration is on the specified line before enabling the location. If the first ENABLE procedure command specifies a location name, the functions required to enable the subsystem are performed before communications is established with that remote location.

Enabling Multiple Remote Locations (SNA Subsystems Only)

For Finance, Peer, SNUF, and APPC subsystems only, if a location name is specified on the `ENABLE` procedure command, communication can be established with that remote location. If there are multiple remote locations defined in the configuration record for a subsystem, these remote locations can be automatically enabled when the subsystem is enabled, without having to specify the names of the remote locations.

If multiple remote locations were defined during configuration, the `ENABLE` procedure, when it enables the subsystem, also establishes communications with one, several, or all the remote locations specified in the subsystem's configuration. The number of remote locations with which communications is activated when the subsystem is enabled depends on how each remote location was defined during configuration, and whether a remote location name was specified on the `ENABLE` procedure command.

- If only the subsystem configuration member name is specified on the `ENABLE` command, the subsystem becomes active, and communications with all the locations that were so indicated during configuration also become active. That is, communications is activated with each location for which a value of Y (yes) was specified on prompt 1 (active location at enable) of display 30.0 during the `CNFIGICF` procedure.
- If a remote location name is also specified on the `ENABLE` command that enables the subsystem, communication with only that remote location is activated when the subsystem is enabled.

After the subsystem and some locations are active, other locations can be activated individually each time that the `ENABLE` command is used to specify a different remote location name. In this case, communications with the specified remote location is all that is activated; the locations that are already active are not affected.

The `ENABLE` procedure command also ensures that all remote location names associated with a subsystem configuration are unique in the system. If a subsystem is active and one of the location names matches a remote location name in the configuration of the subsystem being enabled, a message is issued indicating that the location you specified is already active. The operator is given the option of continuing the `ENABLE` procedure command and skipping that location or of canceling the entire `ENABLE` procedure command.

After the subsystem is enabled, it is ready to handle sessions that are started by System/36 programs or by procedure start requests that are received from remote systems. It does not, however, accept procedure start requests if prompt 3 (switch type at enable) on display 12.0 was specified as *inactive* during the `CNFIGICF` procedure.

ENABLE Procedure Command

The syntax of the ENABLE procedure command is:

```
ENABLE    subsystem configuration name, [ library name  
                                         current library ], [ line number ],  
  
        [ NOSHOW  
          SHOW ], [ location name ], [ line member name ]
```

S7910044-0

subsystem configuration name: Specifies the subsystem member name of the subsystem configuration to be enabled. This is the name that was specified when the CNFIGICF procedure was used to configure the subsystem. (For all subsystems except Intra, the subsystem configuration consists of two members, a line member and a subsystem member. The subsystem member contains the name of the line member to be used when the subsystem configuration is enabled.) This parameter is required.

library name: Specifies the name of the library that contains the specified subsystem configuration. (The line member, if any, and subsystem member must be in the same library.) If no library name is specified, the current library is assumed, and only that library is searched.

line number: Specifies the number of the communications line for which this subsystem is to be enabled. Depending on the number of lines available on your system, you can specify 1 through 10. Omit this parameter when enabling an Intra subsystem. This parameter is required for all other subsystem types.

SHOW or NOSHOW: Specifies whether subsystem configuration parameters are to be displayed before the subsystem is enabled. If SHOW is specified, the subsystem configuration parameters are displayed (not the line member parameters); however, no changes can be made to the values displayed while the ENABLE procedure command is being performed. If no parameter is specified, NOSHOW is assumed.

location name: Specifies the name of the remote location with which the enabled subsystem is to communicate. The location name is optional, and can be specified only if the subsystem being enabled is a Finance, Peer, SNUF, or APPC subsystem. This name must have been specified as a remote location name during subsystem configuration. If the location name is omitted when a Finance, Peer, SNUF, or APPC subsystem is enabled, a Y (yes) must have been specified for prompt 1 on display 30.0 to activate communications.

line member name: Specifies the name of the line member to be enabled. This parameter is valid only for APPN.

Disabling a Subsystem

To disable a subsystem, the `DISABLE` procedure command must be run. When a subsystem is disabled, it no longer exists; only the definition of the subsystem configuration still exists on the system. Finance, Peer, SNUF, and APPC subsystems also allow communications with a specific remote location to be terminated without the subsystem itself being disabled.

When the `DISABLE` procedure command is used to disable a subsystem, it performs the following functions:

- If no sessions are active for the subsystem being disabled, the subsystem is disabled, and the main storage being used is freed. Also, if no other subsystem of this type (such as BSCCL or Peer) is active, the subsystem support for this type of subsystem is terminated.
- If no sessions are active between the subsystem and the remote location that is being disabled, communications with that remote location only is terminated.
- If sessions are active for the subsystem or location specified on the `DISABLE` procedure command, a message is issued to the operator who issued the `DISABLE` procedure command. The operator can respond with one of the following options:
 - 0 Hold (pend) the disable request. New sessions cannot be started for this subsystem or location and, when all sessions have been completed, a normal disable occurs (see note).
 - 1 Retry the disable request. Check again for any active sessions for this subsystem or location.
 - 2 Cancel active sessions and disable the subsystem or location. Active sessions for this subsystem or location are immediately terminated, and the `DISABLE` procedure command is performed.
 - 3 Ignore the disable request. The `DISABLE` procedure command is canceled and must be run again when the subsystem or location is to be disabled.
- If a disable request is pending (waiting to be performed) or is in progress, a message is issued to the operator. The message indicates that the operator can specify either that the subsystem or location be immediately disabled (option 2) or that this `DISABLE` procedure command be canceled and the pending disable request be allowed to complete normally (option 3).

Note: When a disable request is pending, each program performing a successful input operation to the location(s) affected by the `DISABLE` procedure command receives a major return code indicating that a disable operation is pending.

For an APPC subsystem only, all session groups should be stopped before disabling the subsystem. The `STOPGRP` procedure command is used to stop a session group(s). See “`STOPGRP` Procedure” in Chapter 1 of the *SSP-ICF Base Subsystems Reference* manual for more information.

Disabling Multiple Remote Locations (SNA Subsystems Only)

If an SNA subsystem is communicating with multiple locations, the `DISABLE` procedure command can terminate communications with one location or all the locations defined in the subsystem. When multiple locations are active, the number of remote locations that are disabled depends on whether a remote location name is specified on the `DISABLE` command.

- If only the subsystem configuration member name is specified on the `DISABLE` command, communications between the subsystem and all its locations are terminated, and the subsystem is disabled. (However, the communications line remains active if it is also being used by SNA MSRJE or SNA 3270 device emulation.)
- If a location name is specified with the subsystem configuration member name, communications between the subsystem and that location *only* is terminated; all other locations that are active for that subsystem remain active.
- For APPN only, if a line number is specified with the subsystem configuration member name, communication between the subsystem and all locations active on the line is terminated.

DISABLE Procedure Command

The syntax of the `DISABLE` procedure command is:

```
DISABLE subsystem configuration name, [location name], [line number]
```

S7910045-0

subsystem configuration name: Specifies the subsystem member name of the subsystem configuration to be disabled.

location name: Specifies the name of the remote location to be disabled. Location name is optional, and it can be specified only if the subsystem being disabled is a Finance, Peer, SNUF, or APPC subsystem. This name must have been specified as a remote location name during subsystem configuration. If the location name is omitted, all the remote locations that have communications activated are disabled.

line number: Specifies the number of the line to be disabled. This parameter is valid only for APPN.

Chapter 2. Programming SSP-ICF with Assembler

Assembler Macroinstructions	2-3
\$DTFW Macro	2-4
\$DTFW Example	2-7
\$DTFO Macro	2-8
\$DTFO Example	2-10
\$ALOC, \$OPEN, and \$CLOS Macros	2-10
\$ALOC Macro	2-10
\$OPEN Macro	2-11
\$CLOS Macro	2-11
\$WSIO Macro	2-12
\$WSIO Examples	2-16
\$WSIO Macro Parameters Summary Chart	2-16
\$EVOK Macro	2-18
\$EVOK Examples	2-20
Sending Data with an Evoke Operation	2-20
Assembler Operations Summary Chart	2-21
Return Codes	2-23
Interactive Communications Assembler Subroutines	2-23
Assembler Coding Examples	2-24

The communications portion of an assembler program consists of preparing data for transmission, using macroinstructions to define control blocks and to perform operations, processing data that was received, and checking and handling the return codes. This chapter briefly describes:

- The macroinstructions needed in assembler to execute the various communications operations allowed in each subsystem. Only the parameters needed for communications are described in this chapter.
- All the assembler communications operations and the subsystems for which each operation is valid (shown in a summary chart).
- Return code considerations for assembler.
- Communications subroutine considerations for assembler.

The parameters you need to specify for the \$DTFO, \$ALOC, \$OPEN, and \$CLOS macros are *introduced* in this chapter. Complete descriptions are provided in the manual *Programming with Assembler*.

A *complete* description of all the parameters on the \$EVOK macro and a description of *only* the parameters that are used for interactive communications on the \$DTFW and \$WSIO macros are also given in this chapter. (Other parameters that are used for display station input and output are described in the *Programming with Assembler* manual.) The \$DTFW and \$WSIO macros define and modify fields in work station DTFs. The complete format of the DTF, including field labels, is in the *Program Problem Diagnosis and Diagnostic Aids* manual, SY21-0593.

If you are using the Intra or APPC subsystem, externally described field, format, and file definitions (also called data definitions) can be used to send data records. Data definitions, which describe data records and communications functions, are defined separately from the application program. The interactive data definition utility (IDDU) is used to create data definitions. Refer to the manual *Getting Started with Interactive Data Definition Utility* for more information.

The details about using IDDU with the Intra or APPC subsystem are described in the appropriate subsystem reference manual.

Assembler Macroinstructions

To perform communications operations in assembler language, use the following macroinstructions:

Macro	Function
\$DTFW	Defines an interactive communications DTF (define the file)
\$DTFO	Defines the address offsets in a DTF
\$ALOC, \$OPEN, and \$CLOS	Allocate, open, and close the file used by the program
\$WSIO	Performs a communications operation
\$EVOK	Defines a parameter list to be used during an evoke operation to start a remote program or procedure

All these macros except \$EVOK can be used in both the communications and noncommunications portions of assembler programs. The \$EVOK macro can only be used in programs that use an SSP-ICF or BSC 3270 subsystem to perform communications.

\$DTFW Macro

The \$DTFW macro is used to generate an interactive communications DTF. It defines the fields in the DTF. The syntax of the \$DTFW macro is:

```
[label] $DTFW [UPSI- $\left\{ \begin{array}{l} 00000000 \\ \text{8-bit UPSI} \end{array} \right\}$ ] [ ,CHAIN- $\left\{ \begin{array}{l} \text{X'FFFF'} \\ \text{DTF address} \end{array} \right\}$  ]  
[ ,RCAD- $\left\{ \begin{array}{l} \text{X'0000'} \\ \text{address} \end{array} \right\}$  ] [ ,INLEN- $\left\{ \begin{array}{l} 0000 \\ \text{input length} \end{array} \right\}$  ]  
[ ,OUTLEN- $\left\{ \begin{array}{l} 0000 \\ \text{output length} \end{array} \right\}$  ] [ ,TERMID- $\left\{ \begin{array}{l} \text{bb} \\ \text{session id} \end{array} \right\}$  ]  
[ ,TIDTAB- $\left\{ \begin{array}{l} 0000 \\ \text{session id table address} \end{array} \right\}$  ]  
[ ,ENTLEN- $\left\{ \begin{array}{l} 00 \\ \text{length} \end{array} \right\}$  ] [ ,TNUM- $\left\{ \begin{array}{l} 1 \\ \text{number of entries} \end{array} \right\}$  ]  
[ ,HALTS- $\left\{ \begin{array}{l} \text{N} \\ \text{Y} \end{array} \right\}$  ] [ ,IDUCM-file name ]  
[ ,DICTCM-dictionary name ] [ ,EXTEND- $\left\{ \begin{array}{l} \text{N} \\ \text{Y} \end{array} \right\}$  ]
```

S7910012-1

UPSI Parameter: Specifies a string of eight binary digits used to condition the opening of this DTF. When the corresponding bits are on in the switch (as specified in the SWITCH OCL statement), the DTF is opened. For example, to test bits 0, 3, 5, and 7, you would code UPSI-10010101. If this parameter is omitted, zeros are assumed, and the file is opened unconditionally.

CHAIN Parameter: Specifies the address of the next DTF in the chain. If this parameter is omitted, hex FFFF is assumed, and the chain is ended.

RCAD Parameter: Specifies the address of the leftmost byte of the logical record buffer in the user program. If the buffer is also to be used for display station input, the specified *address must be on an 8-byte boundary*. The default is hex 0000.

INLEN Parameter: Specifies, in decimal, the maximum amount of input data that the user program is prepared to receive. For communications operations, the maximum is 4075 bytes for all subsystems except Intra, IMS, and APPC, for which the maximum is 4096 bytes. If the INLEN parameter is omitted, zeros are assumed, and no data can be received unless this field is modified by the \$WSIO macro.

OUTLEN Parameter: Specifies, in decimal, the length of the data in the buffer pointed to by the RCAD parameter. For communications operations, the maximum is 4075 bytes for all subsystems except Intra, IMS, and APPC, for which the maximum is 4096 bytes. This parameter is used only for output operations; however, the DTF field it modifies is also used for input operations. This parameter should be specified for all output operations, especially when combined input/output operations are being performed. If this parameter is omitted, zeros are assumed, and no data can be sent unless this field is modified by the \$WSIO macro.

TERMID Parameter: Specifies the identifier of the session in which this program is to communicate.

- If the session is to be started by *your program* with the acquire operation, the value specified in the TERMID parameter must be the same as the value specified in the SYMID parameter of the SESSION OCL statement.
- If the session is to be started by the *remote system* with a procedure start request, the TERMID parameter can be omitted. If it is omitted, blanks are assumed unless the identifier is specified in the \$WSIO macro. If this field (\$WSNAME) is blank when a remote program starts the session and evokes this program, the system automatically assigns a session identifier and puts it in this field.

TIDTAB Parameter: Specifies the address of the session and work station ID table. Programs that communicate with multiple display stations and sessions should maintain a list of identifiers and associated status indicators. By specifying the TIDTAB, TNUM, and ENTLEN parameters, an area is reserved for this list. During an open operation, the identifier of the session or display station that requested the program is placed in the first 2 bytes of the first entry in the list. In addition, the first 2 bits of the third byte are set.

For each WORKSTN and SESSION OCL statement, an entry is created that has the SYMID parameter value in the first 2 bytes. The first bit of the third byte is set on if REQD-YES was specified on the WORKSTN statement; the second bit is set off. The table must be large enough to contain each of these identifier entries and any additional entries up to the MRTMAX parameter value specified on the ATTR OCL statement. After the open operation has been completed, the user program must maintain the list.

If an end of session operation is issued or if a return code of 80xx or 81xx is received, zeros are placed in the first 2 bytes and the first 2 bits of the third byte in the appropriate entry. The first 2 bytes and the first 2 bits of the third byte must be set to zeros before the DTF is opened. If the TIDTAB parameter is omitted, zeros are assumed, and no table is built.

ENTLEN Parameter: Specifies, in bytes, the decimal length (maximum of 255 bytes) of each entry in the session and work station ID table. If the TIDTAB parameter was specified, the ENTLEN parameter must be specified and should be 3 or greater (2 bytes for the session identifier and 1 byte for status indicators).

TNUM Parameter: Specifies the number of entries (maximum of 255) in the session and work station ID table. The TNUM parameter value should be greater than or equal to the maximum number of concurrent active sessions and attached display stations. If the TIDTAB parameter was specified, the TNUM parameter must also be specified. If the TNUM parameter is omitted, 01 is assumed.

HALTS Parameter: Specifies whether a halt should be issued for communications errors that result in return codes greater than 3401. If Y (yes) is specified, a system message is issued that allows the operator the option of ending the job or of returning control to the user program with an error return code. If N (no) is specified, an informational message is displayed at the system console, and the user program receives control with the error return code. If the HALTS parameter is omitted, N is assumed.

IDDU CM Parameter: Specifies the IDDU file definition name. For more information about file definitions, refer to the manual *Getting Started with Interactive Data Definition Utility*.

DICTCM Parameter: Specifies the IDDU data dictionary name that contains the IDDU file definition name specified above. For more information about data dictionaries, refer to the manual *Getting Started with Interactive Data Definition Utility*.

EXTEND Parameter: Extends the DTF.

See the manual *Programming with Assembler* for a description of the other \$DTFW macro parameters.

\$DTFW Example

This example shows a DTF named ICDTF1 that is to be used for receiving input.

```
ICDTF1  $DTFW  CHAIN-PRTDTF,INLEN-256,HALTS-Y,RCAD-BUF1
```

This DTF is part of a DTF chain; the next DTF is a printer DTF (PRTDTF). Any communications errors result in a system message that requires operator intervention. The program's logical record buffer is located at the address labeled BUF1. This DTF can be used in multiple sessions; the TERMID parameter, which defaults to blanks in this example, can be specified with different session identifiers in the \$WSIO macro expansions used to issue operations in different sessions.

The examples shown later in this chapter under “\$WSIO Macro” use this DTF.

\$DTFO Macro

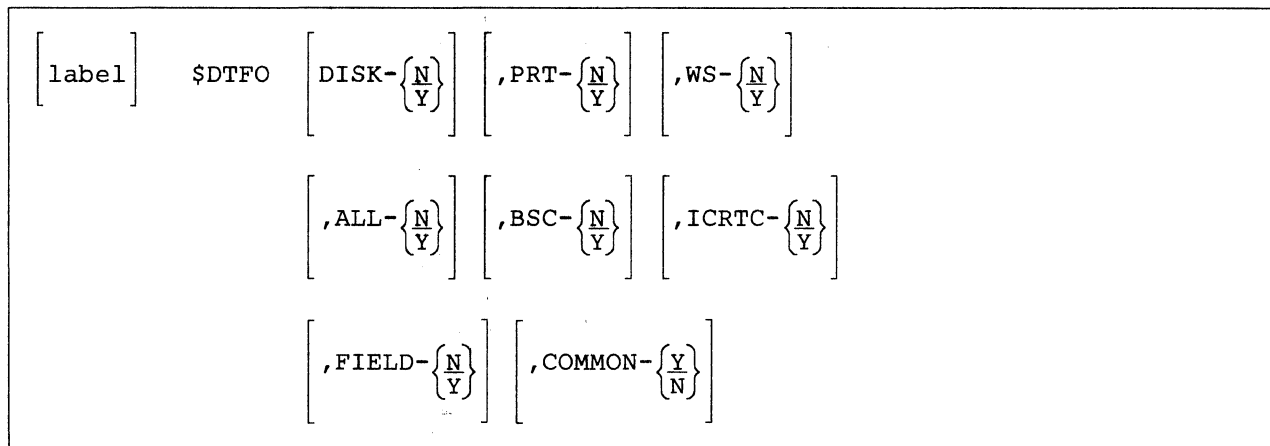
The \$DTFO macro is used to generate the DTF address offsets. It defines the DTF labels, offsets, field contents, and field lengths for all devices and access methods supported by System/36. Labels are generated only for the items for which Y (yes) is specified in the \$DTFO parameters.

To avoid duplicate labels, the \$DTFO macro should be used only once in each program. For a list of the fields that the \$DTFO macro defines, see the DTFs in the *Program Problem Diagnosis and Diagnostic Aids* manual, SY21-0593.

Notes:

1. *For communications operations, the WS and FIELD parameters **must** be specified with Y (yes).*
2. *To generate the labels for the SSP-ICF return codes, ICRTC-Y must also be specified.*
3. *The default value for all of the following parameters, except COMMON, is N (no).*

The syntax of the \$DTFO macro is:



S7910013-0

DISK Parameter: Specifies whether labels are to be generated for the disk devices.

PRT Parameter: Specifies whether labels are to be generated for the printer.

WS Parameter: Specifies whether labels are to be generated for work station devices and for SSP-ICF.

ALL Parameter: Specifies whether labels are to be generated for all devices supported on System/36 and for SSP-ICF.

BSC Parameter: Specifies whether labels are to be generated for batch BSC. Note that this parameter does not apply to SSP-ICF BSC subsystems.

ICRTC Parameter: Specifies whether labels are to be generated for SSP-ICF return codes.

FIELD Parameter: Specifies whether to generate the labels that define the contents of the DTF fields.

COMMON Parameter: Specifies whether to generate the labels defining the field contents of the common portion of the DTF (that is, from the beginning of the DTF to the end of the name field). If this parameter is omitted, Y (yes) is assumed.

\$DTFO Example

```
DTFO1  $DTFO  WS-Y,ICRTC-Y,FIELD-Y
```

This example defines the DTF labels for work station devices and SSP-ICF communications, SSP-ICF return codes, and the DTF fields.

\$ALOC, \$OPEN, and \$CLOS Macros

The \$ALOC, \$OPEN, and \$CLOS macros are needed to identify and control a file that is to be used by a communications program. These three macros, which perform the same functions as they do when used with noncommunications programs, are described only briefly here. For a complete description of each one, see the manual *Programming with Assembler*. The DTF parameter is used on each of the macros and has the following meaning:

DTF Parameter: Specifies the address of the leftmost byte of the first DTF being allocated, opened, or closed. (When chaining is used, multiple DTFs can be allocated, opened, or closed at the same time.) If no address is specified, the DTF address is assumed to be in index register 2.

\$ALOC Macro

The \$ALOC macro allocates the communications file (identified in the DTF) to be used with the program. The \$ALOC macro is supported for compatibility only; it is not required for SSP-ICF communications. The syntax of the \$ALOC macro is:

```
[label] $ALOC [DTF-address]
```

S7910014-0

\$OPEN Macro

The \$OPEN macro opens the communications file to be used for data input and output. It formats the DTF for the file and prepares the program data buffers to be used for data transfer. The syntax of the \$OPEN macro is:

```
[label] $OPEN [DTF-address]
```

S7910015-0

\$CLOS Macro

The \$CLOS macro closes the communications file and updates the file's status after the program has completed communications. The syntax of the \$CLOS macro is:

```
[label] $CLOS [DTF-address]
```

S7910016-0

\$WSIO Macro

The \$WSIO macro is used to perform communications operations. The macro specifies which operation is to be performed and, for certain operations, it can pass data between the communicating programs. It can also be used to modify certain fields in the DTF (specified by the \$DTFW macro) that are used when the specified operation is performed. The syntax of the \$WSIO macro is:

```
[ label ] $WSIO [ DTF-address ] [ ,INLEN=length ] [ ,OUTLEN=length ]  
[ ,RCAD-address ] [ ,TERMID=session id ] [ ,OPC-code ]  
[ ,OPMa-modifier ] [ ,PL@-address ]
```

^a Either OPM or OPMOD is valid as the keyword for this parameter.

S7910017-0

DTF Parameter: Specifies the address of the leftmost byte of the DTF. This address is used as the label on a \$DTFW macro. If this parameter is omitted, the address of the DTF is assumed to be in index register 2.

INLEN Parameter: Specifies, in decimal, the maximum amount of input data that the user program is prepared to receive. For programs using SSP-ICF, the maximum is 4075 bytes for all subsystems except Intra, IMS, and APPC, for which the maximum is 4096 bytes. If the INLEN parameter is omitted, the DTF remains unchanged.

OUTLEN Parameter: Specifies, in decimal, the length of the data in the buffer pointed to by the RCAD parameter. For programs using SSP-ICF, the maximum is 4075 bytes for all subsystems except Intra, IMS, and APPC, for which the maximum is 4096 bytes. If the OUTLEN parameter is omitted, the field in the DTF remains unchanged. This parameter is used only for output operations; however, the DTF field it modifies is also used for input operations. This parameter should be specified for all output operations, especially when combined input/output operations are being performed. After a successful input operation, the actual length of the data returned is put in this field.

RCAD Parameter: Specifies the address of the leftmost byte of the logical record buffer in the user program; the buffer can be used for input, output, or both input and output operations. If this parameter is omitted, the DTF remains unchanged. If the buffer is also to be used for display station input, the specified *address must be on an 8-byte boundary*. If this parameter was not specified on the \$DTFW macro, it should be specified in the first \$WSIO macro issued in the program to establish the record address.

TERMID Parameter: Specifies the 2-character identifier of the session for which this operation is intended. For an acquired session, this ID should be the same as the SYMID parameter value on the corresponding SESSION OCL statement. For a remotely started session, you should not specify the TERMID parameter because you do not know the ID of the remotely started session at the time you code this parameter.

The ID should be specified in a program that has multiple sessions and/or display stations to assure that the operation is issued to the correct location. If this parameter is omitted, the DTF remains unchanged. Following each accept operation, SSP-ICF returns the identifier of the session from which data was received in this field (\$WSNAME).

OPC Parameter: Specifies the code of the communications input/output operation desired. If this parameter is omitted, the DTF remains unchanged. Refer to either the “Assembler Operations Summary Chart” or the “\$WSIO Macro Parameters Summary Chart” later in this chapter for a complete list of the operation codes that can be specified in this parameter.

When the **get attributes** (GTA) operation is specified for the OPC parameter, it returns status information about a specific session. If the session is active or a SESSION OCL statement exists for the identifier (TERMID parameter) specified, the first 10 bytes of the record area (RCAD parameter) are as follows:

Position	Value	Meaning
1	A C R	Session not yet acquired by the program. Session is an acquired session. Session is a remotely started session.
2	N I O	Input not invited for this session. Input invited for this session, but no input is available. Invited input is available for this session.
3 through 10	Name	Location name (specified during configuration and on the SESSION OCL statement).

*Note: If the identifier for the operation is not that of a session, the format of the attribute information is different. See the manual **Programming with Assembler** for the format of attribute data for display stations.*

For the Intra and APPC subsystems only, when the **get status** (GST) operation is specified for the OPC parameter, additional status information is returned. The fields are as follows (starting with byte 11):

Position	Value	Meaning
11	I A	Intra subsystem is being used. APPC subsystem is being used.
12	0 1	Synchronization level is NONE. Synchronization level is CONFIRM.
13	M B	Mapped conversation. Basic conversation.
14 through 16	Blanks	Reserved.
17 through 33	Name	Own fully qualified LU name.
34 through 41	Name	Partner LU name.
42 through 58	Name	Partner fully qualified LU name.
59 through 66	Name	Session group name.
67 through 74	Name	User ID.
75 through 128	Blanks	Reserved.

When the **set timer** (STM) operation is specified for the OPC parameter, it specifies an interval of time to wait before issuing a timer-expired return code. The first 6 bytes of the user record area specify the time interval in the format **hhmmss**, where **hh** is hours, **mm** is minutes, and **ss** is seconds. A timer-expired return code is returned on the first accept input operation following expiration of the timer. The value in the TERMID parameter returned with the timer-expired return code has no meaning. If the timer was set by a previous set timer operation and it has not yet expired when another set timer operation is issued, the old value is discarded and the new interval is set.

OPM Parameter: Specifies the operation modifier to be associated with this operation. If this parameter is omitted, the DTF remains unchanged. The following list shows the valid modifiers for sessions and their descriptions:

Modifier	Description
CONFIRM	Indicates that a confirm indication is to be sent with the data associated with the evoke, put, get (in send state only), and invite (in send state only) operations. This modifier is valid only for the Intra and APPC subsystems.
FMH	Indicates that a function management header is sent with the data associated with the evoke operation. This modifier is valid only for the SNUF, Finance, or Intra subsystems.
ZERO	Resets the operation modifier to zeros.

PL@ Parameter: Specifies, only when an evoke operation is specified by the OPC parameter, the address of an associated evoke parameter list. The value for this parameter is used as the label on the \$EVOK macro. This parameter must be specified on the first evoke operation, and remains unchanged if not specified thereafter.

\$WSIO Examples

The following are three typical examples of the \$WSIO macro. All three examples use the DTF named ICDTF1 as it was defined in the \$DTFW example shown previously.

```
BEGIN  $WSIO  DTF-ICDTF1,TERMIN-1S,OPC-ACQ,OPM-ZERO
```

This example issues an acquire operation to acquire the session 1S.

```
EVOK  $WSIO  DTF-ICDTF1,RCAD-INBUFF,OPC-EVG,PL@-EVKLST
```

This example evokes a transaction in the acquired session and then waits for input. The evoke parameter list begins at label EVKLST associated with the \$EVOK macro used in the program. The INLEN parameter is not specified because it was specified on the \$DTFW macro.

```
OTPT  $WSIO  DTF-ICDTF1,OUTLEN-256,RCAD-OTBUFF,OPC-PUT
```

This example shows a put operation being issued in the session and transaction that have been started. The length of the data is 256 bytes, and it is stored at the label OTBUFF.

\$WSIO Macro Parameters Summary Chart

The following chart shows all the assembler communications operation codes and all the parameters on the \$WSIO macro. This chart also shows, for each operation, whether the parameters are required, optional, or ignored. The meanings of the letters as used in the parameter columns are:

- R** Parameter is required. (However, a required parameter does not have to be specified if it was previously specified in the DTF and the value does not have to be changed.)
- O** Parameter is optional.
- I** Parameter is ignored.
- X** ZERO must be specified in the OPM parameter.

Operation Code	\$WSIO Macro Parameters							
	DTF	OUTLEN	INLEN	RCAD	TERMID	OPM	OPC	PL@
ACI	O	I	R	R	I	I	R	I
ACQ	O	I	I	I	R	I	R	I
CAN	O	I	I	I	R ⁷	X	R	I
CANG	O	I	R	R	R ⁷	X	R	I
CANI	O	I	I	I	R ⁷	X	R	I
CNI	O	I	I	I	R ⁷	X	R	I
EOS	O	I	I	I	R ⁷	X	R	I
EVE	O	R ¹	I	R ²	R	O ^{3,8}	R	R
EVG	O	R	R	R	R	O ^{3,8}	R	R
EVI	O	R ¹	I	R ²	R	O ^{3,8}	R	R
EVK	O	R ¹	I	R ²	R	O ^{3,8}	R	R
FAIL	O	O	I	O	R ⁷	X	R	I
GET	O	I	R	R	R ⁷	O ⁸	R	I
GST	O	I	I	R ⁹	R ⁷	I	R	I
GTA	O	I	I	R ⁴	R ⁷	I	R	I
INV	O	I	I	I	R ⁷	O ⁸	R	I
NRP	O	O ⁵	I	O ⁵	R ⁷	X	R	I
NRPG	O	O ⁵	R	R	R ⁷	X	R	I
NRPI	O	O ⁵	I	O ⁵	R ⁷	X	R	I
PEC	O	R ¹	I	R ²	R ⁷	X	R	I
PEF	O	R ¹	I	R ²	R ⁷	X	R	I
PEX	O	R ¹	I	R ²	R ⁷	O ⁸	R	I
PEM	O	R	I	R	R ⁷	X	R	I
PFMG	O	R	R	R	R ⁷	X	R	I
PFMI	O	R	I	R	R ⁷	X	R	I
PTG	O	R	R	R	R ⁷	O ⁸	R	I
PTI	O	R ¹	I	R ²	R ⁷	O ⁸	R	I
PUT	O	R ¹	I	R ²	R ⁷	O ⁸	R	I
PCDG	O	I	R	R	R ⁷	X	R	I
RCDI	O	I	I	I	R ⁷	X	R	I
REL	O	I	I	I	R ⁷	X	R	I
STM	O	I	I	R ⁶	I	I	R	I

1If zero, no data accompanies the request, and the RCAD parameter value is ignored.

2Required only if the OUTLEN parameter value is not zero.

3For the Intra and SNUF subsystems only, OPM-FMH can be specified on all evoke operations.

FMH indicates that a function management header is in the record area pointed to by the RCAD parameter. If OPM is not FMH, it must be ZERO.

4The record area must be at least 10 bytes long.

5Up to 8 bytes of negative response information can be sent. Therefore, the OUTLEN parameter gives the length and, if it is not zero, the RCAD parameter gives the address of the leftmost byte of the information to be sent.

6The RCAD parameter points to a 6-byte zoned decimal field that specifies the timer value being set in the format *hhmmss*.

7If this operation is issued in a remotely started session, the session identifier must be in the field \$WSNAME before the \$WSIO macro is executed. The TERMID parameter should not be specified because you do not know the identifier of a remotely started session when you code your program.

8For the Intra and APPC subsystems only, CONFIRM can be specified on evoke, put, get (in send state only), and invite (in send state only) operations.

9The record area must be at least 128 bytes long.

\$EVOK Macro

The \$EVOK macro builds a parameter list to be associated with an evoke operation. The label on this macro should be the label specified on the PL@ parameter of the \$WSIO macro that evokes a program or procedure. The syntax of the \$EVOK macro is:

[label]	\$EVOK	V- $\left\{ \begin{array}{l} \text{DC} \\ \text{ALL} \\ \text{EQU} \end{array} \right\}$	[,PNAME-address]	[,PWORD-address]
			[,UID-address]	[,LNAME-address]
			[,SYNCL- $\left\{ \begin{array}{l} \text{NONE} \\ \text{CONFIRM} \end{array} \right\}$]	[,CONVT- $\left\{ \begin{array}{l} \text{MAPPED} \\ \text{BASIC} \end{array} \right\}$]

S7910018-0

V Parameter: Specifies the type of expansion for the parameter list. If EQU (equate) is specified, only the displacement labels are generated, and all other parameters are ignored. If DC (define constant) is specified, only the parameter list is generated. If ALL is specified, both the labels and the parameter list are generated. If this parameter is omitted, DC is assumed. Within a program, only one \$EVOK macro can be used that includes equates; that is, only one V parameter can specify ALL or EQU.

PNAME Parameter: Specifies the address of the first character of the name of the remote program or procedure to be evoked. The name must be followed by blanks up to the 8-character length of the field (that is, the PNAME value must be left-adjusted). For the APPC subsystem only, the name can be up to 64 characters. (The first byte contains, in hexadecimal, the length of the name minus one. The name immediately follows, beginning in the second byte.) If the PNAME parameter is omitted, an address of hex FFFE is assumed, and no program or procedure name is passed on the evoke operation.

PWORD Parameter: Specifies the address of the first character of the password. The password must be followed by blanks up to the 8-character length of the field. If this parameter is omitted, an address of hex FFFE is assumed, and no password is passed on the evoke operation.

UID Parameter: Specifies the address of the first character of the user identifier. The identifier must be followed by blanks up to the 8-character length of the field. If this parameter is omitted, an address of hex FFFE is assumed, and no user identifier is passed on the evoke operation.

LNAME Parameter: Specifies the address of the first character of the library name associated with the program or procedure. The library name must be followed by blanks up to the 8-character length of the field. If this parameter is omitted, an address of hex FFFE is assumed, and no library name is passed on the evoke operation.

SYNCL Parameter: Specifies, for the Intra and APPC subsystems only, the synchronization level. If NONE is specified, a confirm is not allowed. If CONFIRM is specified, a confirm is allowed. If this parameter is omitted, a synchronization level of NONE is assumed.

CONVT Parameter: Specifies, for the APPC subsystem only, the conversation type. The conversation type can be mapped or basic. If this parameter is omitted, the system assumes that conversations are mapped.

\$EVOK Examples

```
EVKLST  $EVOK  V-ALL,PNAME-ICPROC,LNAME-ICLIB,  
          UID-USERID,PWORD-PASS  
      •  
      •  
ICPROC  EQU    *  
        DC  CL8'ICFPROC '  
ICLIB   EQU    *  
        DC  CL8'COMMLIB '  
USERID  EQU    *  
        DC  CL8'JJOHNSON'  
PASS    EQU    *  
        DC  CL4'J4AG'
```

This example shows an evoke parameter list that could be used by a \$WSIO macro that specifies the EVKLST label (PL@-EVKLST), such as the second example shown earlier under “\$WSIO Examples.” The name of the procedure to be evoked (ICFPROC) is at the address labeled ICPROC, and the library name (COMMLIB) is at the address labeled ICLIB. The user identifier (JJOHNSON) is at the address labeled USERID, and the user’s password (J4AG) is at the address labeled PASS.

```
EVKL2   $EVOK  V-DC,PNAME-R3PROC,LNAME-ICLIB  
      •  
      •  
R3PROC  EQU    *  
        DC  CL8'COMMPROC '  
ICLIB   EQU    *  
        DC  CL8'COMMLIBR'
```

This \$EVOK example shows an evoke parameter list that is labeled EVKL2 and is used by a \$WSIO macro that specifies PL@-EVKL2. The procedure name (COMMPROC) is at the address labeled R3PROC, and the library name (COMMLIBR) is at the address labeled ICLIB. Because the remote system does not require security, the UID and PWORD parameters were not specified.

Sending Data with an Evoke Operation

Data sent with an evoke operation can be **parameters** to be used by the evoked **procedure**, or it can be **data** to be used by the evoked **program**. When you are creating a procedure that is to have parameters sent to it by an evoke operation, answer *no* to the prompt PROGRAM DATA IN THE INCLUDE STATEMENTS on the end of job menu (second display) of the SEU procedure, or specify PDATA-NO on the COPY control statement for \$MAINT. If you want data to be sent to a program, answer *yes* to the SEU prompt for program data, or specify PDATA-YES on the COPY control statement. Refer to Chapters 2 and 3 of the *SSP-ICF Guide and Examples* manual for more information.

Assembler Operations Summary Chart

The following chart shows the valid assembler communications operations for each subsystem. An x in a subsystem column indicates that the subsystem supports the operation. A – indicates that the subsystem does not support the operation.

Coding information (including any assembler-related dependencies) about each of these operations is described in each of the subsystem reference manual for which the operation is valid. For several of these operations, the coding information varies by subsystem because of the different characteristics of each subsystem. For a general description of how each operation is performed, see the *SSP-ICF Guide and Examples* manual.

Assembler SSP-ICF Operation	Assembler Operation	Communications Subsystems									
		Intra	BSC ¹	CCP	CICS	IMS	3270 ¹	Finance	Peer	SNUF	APPC
Accept input	ACI	x	x	x	x	x	x	x	x	x	x
Acquire	ACQ	x	x	x	x	x	x	x	x	x	x
Cancel	CAN	x	-	-	-	-	-	-	-	x	-
Cancel invite	CNI	x	x	-	-	-	-	-	-	-	-
Cancel then get	CANG	x	-	-	-	-	-	-	-	x	-
Cancel then invite	CANI	x	-	-	-	-	-	-	-	x	-
End of session	EOS	x	x	x	x	x	x	x	x	x	x
Evoke	EVK	x	x	x	x	x	x ²	-	x	x	x
Evoke end of transaction	EVE	x	x	-	x	x	-	-	x	x	x
Evoke then get	EVG	x	x	x	x	x	x ²	-	x	x	x
Evoke then invite	EVI	x	x	x	x	x	x ²	-	x	x	x
Fail	FAIL	x	-	-	-	-	-	-	x	-	x
Get	GET	x	x	x	x	x	x	x	x	x	x
Get attributes	GTA	x	x	x	x	x	x	x	x	x	x
Get status ³	GST	x	-	-	-	-	-	-	-	-	x
Invite	INV	x	x	x	x	x	x	x	x	x	x
Negative response	NRP	x	-	-	-	-	-	-	-	x	-
Negative response then get	NRPG	x	-	-	-	-	-	-	-	x	-
Negative response then invite	NRPI	x	-	-	-	-	-	-	-	x	-
Put	PUT	x	x	x	x	x	-	x	x	x	x
Put end of file/chain	PEF/PEC	x	x	x	x	-	x	x	x	x	-
Put end of transaction	PEX	x	x	-	x	x	-	-	x	x	x
Put then get	PTG	x	x	x	x	x	x	x	x	x	x
Put then invite	PTI	x	x	x	x	x	x	x	x	x	x
Put FMH	PFM	x	-	-	-	-	-	x	-	x	-
Put FMH then get	PFMG	x	-	-	-	-	-	x	-	x	-
Put FMH then invite	PFMI	x	-	-	-	-	-	x	-	x	-
Release	REL	x	x	x	x	x	x	x	x	x	x
Request to change direction then get	RCDG	x	x	x	-	-	-	-	x	x	x
Request to change direction then invite	RCDI	x	x	x	-	-	-	-	x	x	x
Set timer	STM	x	x	x	x	x	x	x	x	x	x

¹Although the BSC 3270 subsystem is not part of SSP-ICF, its operations are listed here to show its similarities to other subsystems.

²Valid only when the DATAID and FLDLTH parameters are specified on the SESSION OCL statement, and when the HOSTNAME parameter on the SESSION statement is CICS or IMS.

³The record area must be at least 128 bytes long.

Return Codes

Whenever an interactive communications operation is issued (using the \$WSIO macro), the next instruction should check the return code. The return code indicates the result of the operation and/or the status of the session or transaction.

All the return codes that apply to a subsystem are described in detail in each subsystem reference manual to which the codes apply. A *brief* description of all return codes for all subsystems is contained in Appendix B. (General information about handling return codes is contained in the *SSP-ICF Guide and Examples* manual.)

Each return code contains two parts (1 byte each): a major code and a minor code. The major code is located at offset \$WSRTC in the DTF, and the minor code is at offset \$WSMINOR in the DTF. Usually, the communications program can determine what action to take by checking only the major code. (The major code identifies the overall condition of the session.) The program might check a few minor codes for specific conditions that require special recovery action.

Interactive Communications Assembler Subroutines

Because of the additional capability and flexibility available in the assembler interactive communications support, you might want to write assembler subroutines for high-level language programs. The considerations and restrictions for writing interactive communications subroutines must be carefully observed to make this approach feasible. The recommended approach is to write a complete program in assembler, and then use the Intra subsystem to communicate with the high-level language program. If, however, you use an assembler subroutine, keep the following considerations in mind:

- All input operations should be done in the same place, that is, either in the subroutine or in the main program. If there is a work station file in the main program, input should be done in the main program. Any input that is done in the subroutine should include thorough error recovery; the subroutine must also handle the effects of errors and exceptions on the main program.
- The subroutine cannot issue a release or end of session operation, unless the DTF is in the subroutine instead of in the main program (meaning that the main program has no work station file).
- The DTF must reside in a portion of the program that is not overlaid while the program is running.
- If the subroutine and the main program both have a work station file, the format member name (\$WSFMBR) in the subroutine DTF must be set to blanks before the DTF is opened.

Assembler Coding Examples

For a complete example of an assembler communications program, see “Writing an Assembler Program to Use Intra” in Chapter 6 of the *SSP-ICF Guide and Examples* manual. The assembler example described in the Intra chapter can also be used by the BSCCL, Peer, and APPC subsystems.

Chapter 3. Programming SSP-ICF with BASIC

BASIC Statements Used for Communications	3-3
OPEN Statement (Acquiring Sessions)	3-4
OPEN Statement Examples	3-6
Example of an Acquired Session	3-6
Example of a Remotely Started Session	3-6
Example of Acquired Sessions with No SESSION Statements	3-6
READ Statement (Receiving Data)	3-8
READ Statement Examples	3-9
Notes about Receiving Data	3-10
WAITIO Statement (Waiting for Input)	3-11
WAITIO Statement Example	3-11
WRITE Statement (Performing Operations within a Session)	3-12
WRITE Statement Operations	3-13
Starting Remote Programs (Evoke Operations)	3-14
IDDU Evoke Operation Considerations	3-14
Sending Program Data with an Evoke Operation	3-16
Procedure for Sending Data with an Evoke Operation	3-16
Sending Data (Put Operations)	3-18
Ending Communications Transactions (End of Transaction Operations)	3-19
Ending Sessions (End of Session Operation)	3-20
Additional WRITE Statement Operations	3-20
Request to Change Direction Operation	3-20
SSP-ICF and Work Station Timer Operations	3-21
\$\$TIMER Operation	3-21
\$\$TIMER Operation Example	3-21
TIMER Intrinsic Function	3-21
TIMER Intrinsic Function Example	3-21
Negative Response Operations	3-22
Cancel Operations	3-23
Fail Operation	3-23
CLOSE Statement (Closing Files for Sessions)	3-24
ATTRIBUTE\$ Intrinsic Function (Getting Session Attributes)	3-25
BASIC Operations Summary Chart	3-27
Checking Return Codes in BASIC	3-28
ERR Code Values	3-29
RETCODE\$ Values	3-30
Notes About Writing BASIC Programs for SSP-ICF	3-31
BASIC Coding Examples	3-31

This chapter briefly describes the BASIC language statements and operations that you use when you write BASIC programs that are to communicate with remote programs via the Interactive Communications Feature (SSP-ICF). To use a BASIC program to communicate with SSP-ICF, do the following:

- Configure and enable the subsystem. (These procedures are described in the appropriate subsystem reference manual.)
- Begin a communications session by opening an SSP-ICF file.
- Begin a program or procedure at the remote system and start a communications transaction. (If you are using the Intra subsystem, the program or procedure being started is in the same System/36.)
- Send or receive data.
- Check return codes.
- End the communications transaction.
- End the communications session.
- Disable the subsystem.

The BASIC operations you need for interactive communications are *introduced* in this chapter. The details about each operation – its function, syntax, programming considerations, and coding example (for some operations) – are described in each subsystem reference manual for which the operation is valid.

General (conceptual) information about these operations and diagrams showing how these operations work are given in Chapter 3 of the *SSP-ICF Guide and Examples* manual.

The operations you use in the communications portion of your program are similar to work station operations. In the noncommunications portion of your program, you can use all of the noncommunications operations (such as LET, USE, and PRINT) that you normally use to process the data that is sent or received between your program and the remote program. Therefore, the noncommunications operations are not described in this manual.

If you are using the Intra or APPC subsystem, externally described fields, format, and file definitions (also called data definitions) can be used to send data records. Data definitions, which describe data records and communications functions, are defined separately from the application program. The interactive data definition utility (IDDU) is used to create data definitions. Refer to the manual *Getting Started with Interactive Data Definition Utility* for more information.

The details about using IDDU with the Intra or APPC subsystem are described in the appropriate subsystem chapter.

BASIC Statements Used for Communications

In BASIC, the communications operations are performed primarily by the statements shown in the following list. Only these statements are described in this chapter, and only the communications-related information about each one is given. The manual *Programming with BASIC* contains additional information about these statements and about all the other statements that can be used in a communications program.

Statement	Function
OPEN	Acquires (starts) a session and opens a communications file.
READ	Receives data from a remote program (a get operation is performed).
WAITIO	Waits for data from a remote program (an accept input operation is performed). This statement is used with the READ statement.
WRITE	Performs many of the SSP-ICF communications operations within a session.
CLOSE	Closes the file used in the session and releases the session if it is still active.

Also included in this chapter is a description of the `ATTRIBUTE$` intrinsic function, which returns status information about the session.

OPEN Statement (Acquiring Sessions)

To start (acquire) a session, use the OPEN statement to open the SSP-ICF file you are using for this session. Each OPEN statement starts one session between your program and the remote system. If your program is evoked by a (remote) procedure start request, no session identifier or location name is needed on the OPEN statement.

The syntax of the OPEN statement is:

```
OPEN #file-ref: " {SESSION} [ ,ID=session ID  
                  {WS}   [ ,ID=current WSID$  
                        [ ,LOC=location name  
                        [ ,RECL=record length [ ,GROUP=session group name  
                        [ ,DESCR=IDDU format file name ] "  
                        [ ,OUTIN ] [ ,INTERNAL ] [ ,SEQUENTIAL ] [ EXIT line reference  
                                                                IOERR line reference ]
```

S7910019-1

SESSION or WS Parameter: SESSION specifies that this OPEN statement should be used for an SSP-ICF session only. WS specifies that this OPEN statement can be used for either a work station or an SSP-ICF session.

- If your program is to acquire the session, you *must* specify either the ID parameter (for work station or SSP-ICF sessions) or the LOC parameter (for SSP-ICF sessions only). These parameters determine whether the file being opened is for a work station session or an SSP-ICF session.
- If the session is to be started remotely, *do not* specify the ID or LOC parameter; the identifier of a remotely started session is returned in the WSID\$ intrinsic function when the open operation has been completed.

ID Parameter: If your program is to acquire the session and you need to use a SESSION OCL statement, enter the 2-character identifier for the session. The first character must be numeric (0 through 9), and the second must be alphabetic (A through Z, \$, #, or @). The identifier specified in the ID parameter must be the same as the identifier specified in the SYMID parameter of the SESSION statement.

Note: You need to specify a SESSION statement for a BASIC program only when: (1) for some subsystems, you want to specify any parameters on the SESSION statement other than the LOCATION or SYMID parameters, or (2) for the Finance subsystem, you want to acquire a session with a 3601 or 4701 controller.

LOC Parameter: If your program is to acquire the session and you do not need to specify a SESSION statement, enter the name of the remote location that is to communicate with your program. The name must be the same as the location name that was specified during configuration of the subsystem being used for this session.

RECL Parameter: Enter the length of the longest record (or system message) you expect to send or receive. A system message for the Intra subsystem, for example, is 75 bytes long.

GROUP Parameter: Specifies, for the APPC subsystem only, the session group name. This parameter is valid only if the LOC parameter is also specified. Enter the name of a session group. If a blank session group name is desired, enter *BLANK. If the group parameter is not specified and the LOC parameter is specified, the default session group name specified in the session group configuration is used.

DESCR Parameter: Specifies, if you are using externally described data definitions, the name of the file definition (also called data definition), which describes data records and communications functions. File definitions must be defined in the current data dictionary.

See the manual *Programming with BASIC* for a description of the other OPEN statement parameters.

Note: If the OPEN statement is being used for an SSP-ICF session and you enter the NAME, LIBR, or KEYS parameters, they are ignored.

OPEN Statement Examples

Example of an Acquired Session

```
020 OPEN #1: "SESSION, ID=1S, RECL=255" IOERR ICFERR
```

An example SESSION statement for this OPEN statement is:

```
// SESSION LOCATION=INTRA, SYMID=1S, BATCH=YES
```

Example of a Remotely Started Session

For a remotely started session, no SESSION statement is required because the session is started by a procedure start request. Once a System/36 program has been evoked in the remotely started session, however, it can start other sessions using the acquire operation. In this case, a SESSION statement may be required for each additional session that the evoked program starts.

```
020 OPEN #1: "SESSION, RECL=255" IOERR ICFERR
```

Example of Acquired Sessions with No SESSION Statements

In this example, two OPEN statements are used to acquire two sessions with the remote location named INTRA. No SESSION statement is required.

```
OPEN #1: "SESSION, LOC=INTRA, RECL=255" IOERR ICFERR
```

```
OPEN #2: "SESSION, LOC=INTRA, RECL=255" IOERR ICFERR
```

This page is intentionally left blank.

READ Statement (Receiving Data)

To receive a data record, use the READ statement to get the data record from an SSP-ICF session. The type of operation performed depends on whether a WAITIO statement is also used. If a WAITIO statement precedes the READ statement, an **accept input** operation is performed; the program waits until data is available from *any* work station or SSP-ICF session. (If your program is communicating with only one session, you do not need to use the WAITIO statement.) If no WAITIO statement precedes the READ statement, a **get** operation is performed; the program waits until data is available from a *specific* work station or SSP-ICF session—the one that has the same file reference number entered in the OPEN statement. The WAITIO statement also sets the intrinsic function FILENUM to the file reference number of the communications session from which data is to be read.

The syntax of the READ statement is:

```
READ #file-ref [ ,USING {line reference  
                  {character-expression} ] :  
  
  {MAT array name} [ , {MAT array name} ... ]  
  {variable}       [ , {variable}       ]  
  
  [ EXIT line reference  
    or  
    [ CONV line reference ] [ ,EOF line reference ]  
    [ ,IOERR line reference ] [ ,SOFLOW line reference ] ]
```

S7910020-1

READ Statement Examples

The following READ and WAITIO statements, for example, read one data record into the variable DATA\$.

```
030 WAITIO IOERR ICFERR
040 READ #FILENUM, USING 50: DATA$ IOERR ICFERR
050 FORM V 255
```

- The WAITIO statement at line 30 causes the program to wait for data to be received from any work station or SSP-ICF session. When data is received, the WAITIO statement sets the intrinsic function FILENUM to the file reference number of the file from which the data was received. The READ statement then gets the data received by the WAITIO statement. Without the WAITIO statement at line 30, the READ statement would cause the program to wait until data was available from the work station or SSP-ICF session assigned to the file reference number (#FILENUM) specified in the READ statement, then the program would read the data into DATA\$.
- The intrinsic function FILENUM contains the file reference number of the file (for this session) from which the data is to be read and the V parameter is used on the FORM statement if you do not know the length of the data record received. Up to 255 characters are read into the variable DATA\$.

The following statements read a system message, which can be up to 80 characters, into the variable MESSAGE\$:

```
040 IF ERR=70 THEN&
    &READ #1, USING "FORM V 80": MESSAGE$ IOERR ICFERR
```

S7910058-0

- A** If a system message is received (ERR = 70), the message is read into MESSAGE\$.
- B** Up to 80 characters of the system message are read.

Notes about Receiving Data

1. For SSP-ICF input operations, the maximum amount of data that can be received by a program is 4075 bytes for all subsystems except Intra, IMS, and APPC, for which the maximum is 4096 bytes.
2. You should use the EOF clause with the READ statement to determine when an end of transaction indication is received from the remote system. However, if data is also received with the end of transaction indication, BASIC does not branch to the EOF reference until the next operation for the file is performed.

If the next operation is to be an evoke operation, BASIC must branch to the evoke operation using the EOF clause. For example:

```
100  READ #3, USING "FORM C 255": DATA$ IOERR ICFERR, &  
    &EOF EVOKE  
    .  
    .  
200  GOTO 100  
210  EVOKE: ...
```

When statement 100 causes the last record to be read, the data is placed in DATA\$ and the program continues. When the program returns to statement 100 and performs another read operation, it detects the end of file condition and branches to statement 210, which performs the evoke operation.

(In the situation just described, note that although BASIC, on the first read operation, sets the RETCODE\$ intrinsic function to indicate the end of the transaction, the ERR intrinsic function value is not changed until the second read operation is performed.)

3. You can use the STOP\$ intrinsic function to test for a major return code of 02 (stop system or disable subsystem request pending). If STOP\$ equals Y, a 02 major return code has been returned to your program indicating that a system shutdown has been requested; if not, STOP\$ equals N.
4. The REREAD statement can be used to get more data from the last record read from the file. An error occurs for a REREAD statement (and the program ends) if the last input/output operation to the file (that is, the session) was not a successful READ or REREAD operation.
5. The data passed with the evoke end of transaction operation can be read by the first read operation in the evoked BASIC program. The record length entered in the OPEN statement must be at least 1 larger than the length of the procedure parameters plus the largest size of of the data sent or received. Using a WAITIO statement before a READ or WRITE statement is acceptable but not necessary.

WAITIO Statement (Waiting for Input)

If your program is to interact with multiple sessions concurrently, the WAITIO statement should be used. This statement causes your program to wait until one of the sessions sends input to your program or until it receives an error indication. When the WAITIO statement has completed its operation, the intrinsic function FILENUM is set to the reference number of the file that completed the wait operation. Your program can then use a READ statement to read from that file and get a data record.

The WAITIO statement can also be used to wait for a timer operation to be completed and to determine the action to be taken next. For examples of how the WAITIO statement can be used with either the \$\$TIMER operation or the TIMER intrinsic function, see “SSP-ICF and Work Station Timer Operations” later in this chapter.

The syntax of the WAITIO statement is:

WAITIO	[EXIT line reference IOERR line reference]
--------	---

S7910021-0

WAITIO Statement Example

The following is an example of a WAITIO statement:

```
060 WAITIO IOERR ICFERR
```

Note: If an error condition occurs, the intrinsic function FILENUM is set to the reference number of the file that caused the error, and the program branches to the statement labeled ICFERR (not shown). The RETCODE\$, STOP\$, WSID\$, ERR, and FILENUM intrinsic function values can be set by the WAITIO statement.

WRITE Statement (Performing Operations within a Session)

Use the WRITE statement to perform many of the communications operations between two programs once a session has been started. The type of operation is determined by the value specified for the FORMAT parameter (which is described below).

The syntax of the WRITE statement is:

```

WRITE #file-ref [ ,USING {line reference}
                 {character-expr} ]
                [ ,FORMAT {character-expr}
                 {$$SENDNI} ]
                [ ,INDIC character-expr ] :
                {MAT array name} [ , {MAT array name} ... ]
                [ {EXIT line reference
                  or
                  { CONV line reference } [ ,EOF line reference ]
                  [ ,IOERR line reference ] [ ,SOFLOW line reference ]
                } ] ]

```

S7910022-1

FORMAT Parameter: Identifies the SSP-ICF operation that is to be performed. See “BASIC Operations Summary Chart” later in this chapter for a complete list of the operations you can specify in the FORMAT parameter. Any of the SSP-ICF operations beginning with \$\$ can be used. If the FORMAT parameter is not specified, a put with no invite operation is performed by the WRITE statement.

If you are using IDDU, this parameter identifies the IDDU format definition that externally describes the SSP-ICF operation that is to be performed. For more information about using IDDU with BASIC, refer to the manual *Programming with BASIC*.

INDIC Parameter: If you are using IDDU, selects the SSP-ICF operation that is to be performed. For more information about using IDDU with BASIC, refer to the manual *Programming with BASIC*.

See the manual *Programming with BASIC* for a description of other WRITE statement parameters.

WRITE Statement Operations

The following are the primary communications operations that you can specify on the WRITE statement.

- Evoke operations: To start a remote program
- Put operations: To send data to the remote program
- End of transaction operations: To end a communications transaction
- End of session operation: To end the session in which the remote program was started

These operations are described in the following pages; other types of operations that can also be specified by the WRITE statement are described under “Additional WRITE Statement Operations.” An example of each type of operation that can be specified in the FORMAT parameter is also provided.

Starting Remote Programs (Evoke Operations)

To start a remote program or procedure and to start a communications transaction, specify an **evoke** operation in the FORMAT parameter of the WRITE statement (\$\$EVOK, \$\$EVOKNI, or \$\$EVOKET). See “WRITE Statement (Performing Operations within a Session)” earlier in this chapter for the syntax of the WRITE statement.

With an evoke operation, you must send an **evoke parameter list**. If you specify the evoke operation in the FORMAT parameter of the WRITE statement, the parameters (fields) in that list must be specified in the following order:

Positions	Field Description
1 through 8	The name of the program or procedure to be evoked (left-adjusted).
9 through 16	The password you use to sign on the remote system (left-adjusted).
17 through 24	The user identifier you use to sign on the remote system (left-adjusted).
25 through 32	The name of the remote system library that contains the program or procedure to be evoked (left-adjusted).
33 through xxxx	User data, positional parameters, or keyword parameters. (Leading blanks are ignored.)

If a field is not used, enter the correct number of blanks for the unused field.

IDDU Evoke Operation Considerations:

If you are using IDDU, you can select one of the following IDDU functions to perform an evoke operation: evoke process (EVOKE keyword), evoke process and send detach (EVOKE and DETACH keywords), or evoke process and invite (EVOKE and INVITE keywords).

Because IDDU does not reserve positions for unused parameters or duplicated parameters, one or more of the parameters in the evoke parameter list may not be specified or may be specified in a different order. Therefore, after defining an IDDU format to be used with the evoke operation, you should do the following:

1. Use IDDU to print a format definition listing for the format you defined.
2. Use the listing to determine the order and starting position of each of the parameters in the evoke parameter list.

The following example starts a procedure at a remote system.

```

030  WRITE #1, USING 40, FORMAT "$$EVOK": "BASICR", PASS$, USERID$, &
      &"#LIBRARY", "ICFPROG, USERLIB" IOERR ICFERR
040  FORM 4*C 8, C 15

```

S7910046-0

- 1** Write data to interactive communications file #1 using the FORM statement at line 40.
- 2** Use an evoke operation (\$\$EVOK) to start the procedure, which is identified in the evoke parameter list. In this example, the evoke parameter list to be sent to the remote system contains:
 - 3** Four positional procedure parameters to be used by the remote system to start a procedure.
 - BASICR: The name of the procedure to be evoked (sent in positions 1 through 8)
 - PASS\$: The name of the variable containing the password (positions 9 through 16)
 - USERID\$: The intrinsic function containing the user identifier (positions 17 through 24)
 - #LIBRARY: The name of the library on the remote system in which the BASICR procedure is located (positions 25 through 32)
 - 4** Two positional parameters to be passed to the BASICR procedure (sent in positions 33 through 47). The BASICR procedure is to *call* the program ICFPROG, which is in the user library USERLIB.
- 5** Send four fields of 8 characters each (the evoke parameters in positions 1 through 32).
- 6** Send 15 bytes of positional parameters (those in positions 33 through 47).
- 7** If an error occurs during the WRITE operation, the program goes to the statement labeled ICFERR (not shown).

Sending Program Data with an Evoke Operation

Data sent with an evoke operation can be **parameters** to be used by the evoked **procedure**, as shown in the previous example, or it can be **data** to be used by the evoked **program**. When you are creating a procedure that is to have parameters sent to it by an evoke operation, answer *no* to the prompt **PROGRAM-DATA IN THE INCLUDE STATEMENTS** on the end of job menu (second display) of the SEU procedure, or specify **PDATA-NO** on the **COPY** control statement for **\$MAINT**. If you want data to be sent to a program, answer *yes* to the SEU prompt for program data, or specify **PDATA-YES** on the **COPY** control statement. Refer to Chapters 2 and 3 of the *SSP-ICF Guide and Examples* manual for more information.

Note: You cannot use the BASICR or BASICP procedure as is if you send program data with the evoke operation, because the BASICR or BASICP procedure expects procedural parameters. The following is a procedure that you can use (with the BASICR or BASICP procedure) to send program data with an evoke operation.

Procedure for Sending Data with an Evoke Operation

The following procedure uses the BASICR procedure to send data to a *program*.

```
// MEMBER PROGRAM1-#BL#M1, PROGRAM2-#BL#M2, LIBRARY-#BLLIB
// LIBRARY NAME-user library _____ 1
// REGION SIZE-BASIC region size _____ 2
// LOCAL AREA-SYSTEM, OFFSET-1, BLANK-40
// LOCAL AREA-SYSTEM, OFFSET-1, DATA-'BASICR'
// LOCAL AREA-SYSTEM, OFFSET-9, DATA-'member name' _____ 3
// LOCAL AREA-SYSTEM, OFFSET-17, 'user library name' _____ 4
// LOCAL AREA-SYSTEM, OFFSET-25, DATA-'status' _____ 5
// AREA-SYSTEM, OFFSET-26, DATA-'data dictionary name' _____ 6
// LOCAL-AREA-USER
// LOAD #BLSIC, #BLLIB
// INCLUDE procedure name _____ 7
// RUN
```

S7910038-1

You must supply the following information:

- 1** Enter the name of the current user library.
- 2** Enter the BASIC region size (28K to 64K bytes).
- 3** Enter the name of your BASIC program.
- 4** Enter the name of the library that contains your BASIC program.
- 5** Enter Y (yes) if you want status information printed. Enter N (no) if you do not want status information printed.
- 6** If you are using IDDU, enter the name of your IDDU data dictionary.
- 7** Enter the procedure name to be included. If there is no procedure name to be included, omit this statement.

This procedure example can also be used to send data to a *procedure* written in BASIC instead of to a program. Make the following changes to the example so that it uses the *BASICP* procedure:

- In the second LOCAL statement, change BASICR to BASICP.
- In the third LOCAL statement, specify a procedure member name instead of a program member name.
- In the LOAD statement, change #BLSIC to #BLPIC.

Sending Data (Put Operations)

To send a data record to a remote system or program, specify a **put** operation in the **FORMAT** parameter of the **WRITE** statement (**\$\$SENDNI**, **\$\$SENDE**, **\$\$SENDET**, **\$\$SENDNF**, or **\$\$SENDFM**). If you are using **IDDU** functions, specify a put operation in a user-defined field, select the send detach, or specify a put operation in a user-defined field and select the invite. See “**WRITE Statement (Performing Operations within a Session)**” earlier in this chapter for the syntax of the **WRITE** statement.

A maximum of 4075 bytes can be sent by a put operation for all subsystems except Intra, IMS, and APPC, for which the maximum is 4096 bytes.

The following example sends one data record:

```
030 WRITE #1, USING 40, FORMAT "$$SEND": DATA$ IOERR ICFERR
040 FORM C 255
```

Ending Communications Transactions (End of Transaction Operations)

You can end the transaction by specifying the **put end of transaction** operation or the **evoke end of transaction** operation in the **FORMAT** parameter of the **WRITE** statement (**\$\$\$SENDET** for the put end of transaction or **\$\$EVOKET** for the evoke end of transaction). If you are using **IDDU** functions, select the **send detach** for the put end of transaction or the **evoke process and send detach** for the evoke end of transaction.

- If your program has finished sending data, a put end of transaction in the **FORMAT** parameter tells the remote system that you have no more data to send and that you do not expect to receive any data.
- If your program is receiving data, check for an end of transaction return code received from the subsystem to determine when the remote system has finished sending. (See “**READ Statement (Receiving Data)**” earlier in this chapter.)
- If you want to start a program or procedure at the remote system and immediately end the transaction, an evoke put end of transaction in the **FORMAT** parameter indicates that your program does not expect to receive any data. For example, you can send data to a remote program and then start a different program on the remote system to use that data:
 - Your program starts program A at the remote system and sends data to program A.
 - Program A stores the data on disk.
 - When your program has finished sending data to program A, your program uses the put end of transaction operation to end program A.
 - Your program then uses the evoke end of transaction operation to start program B at the remote system.
 - Program B processes the data that program A stored on disk previously.

Once the transaction has ended, you can end the session or start another transaction with this or another session.

The following statement sends a put end of transaction operation and, therefore, tells the remote system that this is the end of this communications transaction:

```
030 WRITE #1, FORMAT "$$$SENDET": IOERR ICFERR
```


Ending Sessions (End of Session Operation)

To end a session with a remote system, either use the CLOSE statement, or use the WRITE statement (to specify the \$\$EOS operation) followed by the CLOSE statement. (The CLOSE statement is described later in this chapter.) If you use the WRITE statement, specify \$\$EOS in the FORMAT parameter. For example:

```
090 WRITE #1, FORMAT "$$EOS": IOERR ICFERR
```

Additional WRITE Statement Operations

The following are additional interactive communications operations you can specify on the WRITE statement:

- Request to change direction operation
- Set timer operation
- Negative response operations (used only with the Intra and SNUF subsystems)
- Cancel operations, for canceling a group (chain) of data records (used only with the Intra and SNUF subsystems), or for canceling any valid invite operation for which no input has yet been received (used only with the Intra and BSCSEL subsystems)
- Fail operation (used only with the Intra, Peer, and APPC subsystems)

Request to Change Direction Operation

To request a change in the direction of transmission, specify a **request to change direction** operation in the FORMAT parameter of the WRITE statement (\$\$RCD). If you are using IDDU functions, select the send request to write and invite. After you issue the \$\$RCD operation, your program must continue to receive data until it receives a return code indicating that the remote program is ready to begin receiving data. No additional parameters or data is associated with the \$\$RCD operation.

The following WRITE statement shows how to request that the remote system stop sending so that your program can send data:

```
030 WRITE #1,FORMAT "$$RCD": IOERR ICFERR
```

SSP-ICF and Work Station Timer Operations

To use the SSP-ICF and work station timer, use either the **\$\$TIMER** operation on the WRITE statement or the TIMER intrinsic function to set the timer. With either type of operation, use the WAITIO statement to determine when the time has ended and to determine the action to take next based on the return code received.

The FORMAT parameter in the WRITE statement is used to specify the \$\$TIMER operation. Return code 0310 (RETCODE\$) or BASIC error code 73 (ERR) is returned when the time has ended.

Note: If you use the \$\$TIMER operation, a work station or session must be attached to your program before you can set the time. If you use the TIMER intrinsic function, no work station or session need be attached.

\$\$TIMER Operation: To set the timer, use the \$\$TIMER operation, in the format:

hhmmss

where **hh** is hours, **mm** is minutes, and **ss** is seconds.

\$\$TIMER Operation Example:

```
030 A$="013000"
040 WRITE #1,USING 50,FORMAT "$$TIMER": A$ IOERR ICFERR
050 FORM C 6
060 WAITIO IOERR TIME
  •
  •
  •
910 TIME: IF ERR<>73 THEN GOTO ICFERR
```

TIMER Intrinsic Function: To set the timer, use the TIMER intrinsic function in the format:

TIMER(time\$)

where time\$ is the time in the format **hhmmss**; **hh** is hours, **mm** is minutes, and **ss** is seconds. If the timer function is successful, TIMER returns a 0; if the timer function is not successful, TIMER returns a 1.

TIMER Intrinsic Function Example:

```
030 TIME=TIMER("013000")
040 IF TIME=1 THEN PRINT "TIMER CANNOT BE SET"&
  &ELSE WAITIO IOERR TIME1
  •
  •
  •
910 TIME1: IF ERR<>73 THEN GOTO ICFERR
```

Negative Response Operations

To tell the remote system or program that your program found something wrong with the data it received (that is, to send a negative response), use the **FORMAT** parameter in the **WRITE** statement to specify one of the **negative response** operations (**\$\$NRSP** or **\$\$NRSPNI**). These operations can only be used with the **Intra** and **SNUF** subsystems.

Sense data can also be sent with the negative response. The following is the format of the sense data:

Positions	Description
1 through 8	The sense data sent with the negative response. The sense data must begin with 10xx, 08xx, or 0000. For a description of the first 4 characters, see the <i>Systems Network Architecture Reference Summary</i> . The last four positions are user-defined.

For example, the following statements send a negative response operation that includes the sense data 08008000:

```
020 SENSE$="08008000"  
030 WRITE #1,USING 40,FORMAT "$$NRSPNI": SENSE$ IOERR ICFERR  
040 FORM C 8
```

Cancel Operations

There are two types of cancel operations: those that cancel a *group* of records that has just been sent (used by the SNUF and Intra subsystems only), and the *cancel invite* operation, which cancels an invite operation for which no input has yet been received (used by the BSCCEL and Intra subsystems only).

- For the SNUF and Intra subsystems only, to cancel a group of records that has just been sent, use the `FORMAT` parameter in the `WRITE` statement to specify one of the **cancel** operations (`$$CANL` or `$$CANLNI`). The cancel operations have no additional parameters or data associated with them.

For example, the following `WRITE` statement cancels the current chain of records:

```
030  WRITE #1,FORMAT "$$CANL": IOERR ICFERR
```

- For the BSCCEL and Intra subsystems only, to cancel any valid invite operation for which no input has yet been received from any invited session, use the `FORMAT` parameter in the `WRITE` statement to specify the **cancel invite** operation (`$$CNLINV`). The cancel invite operation has no additional parameters or data associated with it. For restrictions on the use of this operation by the BSCCEL subsystem, see "Cancel Invite Operation" in Chapter 2 of the *SSP-ICF Base Subsystems Reference* manual.

For example, the following `WRITE` statement cancels an invite operation that no session has yet responded to:

```
030  WRITE #1,FORMAT "$$CNLINV": IOERR ICFERR
```

Fail Operation

To tell the remote system that your program detected an abnormal condition (for example, it received incorrect data), specify a **fail** operation in the `FORMAT` parameter of the `WRITE` statement (`$$FAIL`). If you are using `IDDU` functions, select the send fail. The fail operation has no additional parameters associated with it, and no data can be sent with the fail operation. The fail operation can be used only with the Intra, Peer, and APPC subsystems.

For example, the following `WRITE` statement sends a fail operation:

```
030  WRITE #1,FORMAT "$$FAIL": IOERR ICFERR
```

CLOSE Statement (Closing Files for Sessions)

The CLOSE statement closes the communications file used in the session. Also, if the \$\$EOS operation was not specified in your program and the session is still active, the CLOSE statement ends the session before it closes the file. All transactions with the remote program must be completed before you end the session.

The syntax of the CLOSE statement is:

```
CLOSE #file-reference: { EXIT line reference }  
                      { IOERR line reference }
```

S7910023-0

For example, this CLOSE statement closes (releases) the SSP-ICF session for file #1:

```
099 CLOSE #1: IOERR ICFERR
```

If an error occurs while executing this CLOSE statement, the program goes to the statement labeled ICFERR. BASIC then automatically issues a \$\$EOS operation to end the session so that the next time you attempt to close the file, no error will occur.

ATTRIBUTE\$ Intrinsic Function (Getting Session Attributes)

The ATTRIBUTE\$ intrinsic function returns status information about a specified session. The status is returned for the session identified by the 2-character identifier specified as the value for the intrinsic function. If no identifier is specified, the status of the session identified by the current value of the WSID\$ intrinsic function is returned.

The syntax of the ATTRIBUTE\$ intrinsic function is:

```
ATTRIBUTE$ [ { ('session ID')  
             { (character-expression) }  
             (WSID$) } ]
```

S7910024-0

For example, this statement gets the attributes of the SSP-ICF session identified as 2S.

```
050 A$=ATTRIBUTE$('2S')
```

This statement gets the attributes of the session identified by a variable named ICFSSN\$.

```
050 A$=ATTRIBUTE$(ICFSSN$)
```

For SSP-ICF sessions, a 10-character constant is returned. The first character indicates the **type** of the session, the second character indicates the **invite status** of the session, and the last eight characters give the **location name** associated with the session. The positions, the values, and the meaning of the values are as follows:

Positions	Value	Meaning
1	A C R	Session not yet acquired. Session is an acquired session. Session is a remotely started session.
2	N I O	Input not invited for this session. Input invited for this session, but no input is available. Invited input is available for this session.
3 through 10	Name	Location name (specified during subsystem configuration and on the SESSION OCL statement).

For the values used for *work station* sessions, see the manual *Programming with BASIC*.

For the Intra and APPC subsystems only, additional status information is returned. The fields are as follows (starting with byte 11):

Positions	Value	Meaning
11	I A	Intra subsystem is being used. APPC subsystem is being used.
12	0 1	Synchronization level is NONE. Synchronization level is CONFIRM.
13	M B	Mapped conversation. Basic conversation.
14 through 16	Blanks	Reserved.
17 through 33	Name	Own fully qualified LU name.
34 through 41	Name	Partner LU name.
42 through 58	Name	Partner fully qualified LU name.
59 through 66	Name	Session group name.
67 through 74	Name	User ID.
75 through 128	Blanks	Reserved.

If a field is not used, enter the correct number of blanks for the unused field.

BASIC Operations Summary Chart

The following chart shows the valid BASIC operations for each subsystem. An x in a subsystem column indicates that the subsystem supports the operation. A - indicates that the subsystem does not support the operation.

BASIC SSP-ICF Operation	BASIC Operation	Communications Subsystems									
		Intra	BSCCL	CCP	CICS	IMS	3270 ¹	Finance	Peer	SNUF	APPC
Accept input	WAITIO ²	x	x	x	x	x	x	x	x	x	x
Aquire	OPEN	x	x	x	x	x	x	x	x	x	x
Cancel	\$\$CANLNI	x	-	-	-	-	-	-	-	x	-
Cancel invite	\$\$CNLINV	x	x	-	-	-	-	-	-	-	-
Cancel then invite	\$\$CANL	x	-	-	-	-	-	-	-	x	-
End of session	\$\$EOS	x	x	x	x	x	x	x	x	x	x
Evoke	\$\$EVOKNI	x	x	x	x	x	x ³	-	x	x	x
Evoke end of transaction	\$\$EVOKET	x	x	-	x	x	-	-	x	x	x
Evoke then invite	\$\$EVOK	x	x	x	x	x	x ³	-	x	x	x
Fail	\$\$FAIL	x	-	-	-	-	-	-	x	-	x
Get	READ	x	x	x	x	x	x	x	x	x	x
Get attributes	ATTRIBUTE	x	x	x	x	x	x	x	x	x	x
Get status ⁶	ATTRIBUTE	x	-	-	-	-	-	-	-	-	x
Invite ⁴	\$\$SEND	x	x	x	x	x	x	x	x	x	x
Negative response	\$\$NRSPNI	x	-	-	-	-	-	-	-	x	-
Negative response then invite	\$\$NRSP	x	-	-	-	-	-	-	-	x	-
Put	\$\$SENDNI	x	x	x	x	x	-	x	x	x	x
Put end of file/chain	\$\$SENDE	x	x	x	x	-	x	x	x	x	-
Put end of transaction	\$\$SENDET	x	x	-	x	x	-	-	x	x	x
Put FMH	\$\$SENDNF	x	-	-	-	-	-	x	-	x	-
Put FMH then invite	\$\$SENDFM	x	-	-	-	-	-	x	-	x	-
Put then invite	\$\$SEND	x	x	x	x	x	x	x	x	x	x
Release	CLOSE	x	x	x	x	x	x	x	x	x	x
Request to change direction then invite	\$\$RCD	x	x	x	-	-	-	-	x	x	x
Set timer	\$\$TIMER ⁵	x	x	x	x	x	x	x	x	x	x

¹Although the BSC 3270 subsystem is not part of SSP-ICF, its operations are listed here to show its similarities to other subsystems.

²Valid only when it is followed by a READ operation or when it follows a timer operation.

³Valid only when the DATAID and FLDLTH parameters are specified on the SESSION OCL statement, and when the HOSTNAME parameter on the SESSION statement is CICS or IMS.

⁴Valid only when a \$\$SEND operation is issued with a record length of zero.

⁵The timer can also be set by the TIMER intrinsic function.

⁶The record area must be at least 128 bytes long.

Checking Return Codes in BASIC

You should use the IOERR parameter on all READ, REREAD, WRITE, OPEN, CLOSE, and WAITIO statements to check the status of the input or output operation. All of these statements set the value of the return code to indicate the results of that operation. You can also use the RETCODE\$, ERR, or FILE intrinsic functions to check the status of the last operation performed. The intrinsic functions contain the following:

- **ERR (error code)** contains the meaning of the error code for the last *unsuccessful* BASIC operation.
- **RETCODE\$ (return code)** contains the status of the last SSP-ICF operation or work station operation. The status tells whether the operation was *successful* or *unsuccessful* and gives you additional information about the results of the operation. If your program contains both SSP-ICF and work station operations, you may want to save the SSP-ICF return codes in a character variable.
- **FILE(x) (file status)** indicates only that the last operation was either *successful* or *unsuccessful*. If FILE is 0, the operation was successful; if FILE is not 0, the operation was unsuccessful. If FILE is 11, an end of transaction indication was received on an input operation; if FILE is 20, an error occurred on an input operation; and if FILE is 21, an error occurred on an output operation.

ERR Code Values

The ERR intrinsic function returns the error code of the last error that occurred. The value is not reset for a successful input or output operation. It stays the same until the next error occurs.

The following chart lists and briefly describes the SSP-ICF error codes that can be returned in an SSP-ICF session. Note, however, that more complete information about any error can be obtained by reading the description of the actual return code associated with the ERR value. The return codes that can be returned by your subsystem are described in that subsystem reference manual.

ERR Values	Error Code Meaning
0	No error; operation completed normally
54	End-of-file error or end of transaction indication
55	Permanent input/output error
64	No invite outstanding on SSP-ICF session
66	Cannot get session group
68	New requester for this program
69	Request to change direction was received
70	Message waiting
71	Operation failed, but session is still active
72	Error occurred in an operation that can be retried
73	Timer expired
74	Buffer too small

RETCODE\$ Values

The value in the RETCODE\$ intrinsic function is the 4-digit (major and minor) SSP-ICF return code. These return codes are described in each subsystem reference manual. A summary listing in Appendix B shows all the return codes and shows which return codes are valid for each subsystem. For general information about return codes, read “Checking Return Codes” in Chapter 3 of the *SSP-ICF Guide and Examples* manual.

The value in the ERR intrinsic function depends upon the SSP-ICF return code in the RETCODE\$ intrinsic function as shown in the following chart. Use this chart to determine the SSP-ICF return code that corresponds to the ERR value. Then see the description of the SSP-ICF return code in the appropriate subsystem reference manual.

For an example of how to check return codes, see “Checking Return Codes with BASIC” in Chapter 6 of the *SSP-ICF Guide and Examples* manual.

SSP-ICF RETCODE\$ Value	BASIC ERR Value	SSP-ICF RETCODE\$ Value	BASIC ERR Value	SSP-ICF RETCODE\$ Value	BASIC ERR Value	SSP-ICF RETCODE\$ Value	BASIC ERR Value
0000	0	8081	55	81BC	70	832D	71
0010	69	8082	55	81C2	70	8330	72
0012	0	8083	55	81C4	66	83C7	55
0020	70	8084	55	81C5	70	83C8	55
0024	0	80BD	55	81C6	70	83C9	55
0100	68	80C0	55	8213	72	83CA	55
0200	0	80C1	55	821E	55	83CB	55
0210	69	80D0	55	8285	72	83CC	55
0212	0	8136	55	8289	55	83CD	55
0220	70	8137	55	82A7	72	83CE	55
0300	0	8183	55	82A9	55	83CF	55
0302	71	8184	55	82AA	55	83D0	55
0303	54	8185	72	82AE	55	83D1	55
0310	73	8187	55	82B1	72	83D2	55
0402	71	8192	55	82BB	55	83D3	55
0411	70	8193	70	82C3	66	83D4	55
0412	71	81A3	70	82C4	66	83E0	55
0800	0	81B6	70	8319	70	83E1	55
1100	64	81B8	70	831B	72		
2800	55	81B9	70	831C	70		
3431	74	81BA	70	831D	71		

Notes About Writing BASIC Programs for SSP-ICF

1. You can use the `WSID$` intrinsic function to determine the identifier of the most recently accessed session. In this example,

```
040  WAITIO IOERR ICFERR
050  A$=WSID$
```

the value of `A$` is the identifier of the last session accessed by the `WAITIO` statement.

2. You can use the `FILENUM` intrinsic function to determine the file reference of the most recently accessed session.
3. You should use the `EXIT` clause with the `IOERR` parameter specified, or use the `IOERR` parameter alone, on all SSP-ICF I/O statements.
4. You can use the `ATTRIBUTE$` intrinsic function to determine the status of a session.
5. Do not use `PAUSE`, `BREAK`, `PRINT`, `INPUT`, `LINPUT`, or `TRACE` in an evoked program (started by a remote procedure start request) to cause information to be displayed at a display station.
6. Do not evoke the BASIC procedure. (You *can* evoke the `BASICR` or `BASICP` procedure.)

BASIC Coding Examples

For a complete example of a BASIC communications program, see “Writing a BASIC Program That Uses the Intra Subsystem” in Chapter 6 of the *SSP-ICF Guide and Examples* manual. The same programming example described in the Intra chapter is also applicable to the other subsystem chapters, but only the changed areas needed to allow communications with that type of remote system are shown.

Chapter 4. Programming SSP-ICF with COBOL

COBOL Statements Used for Communications	4-3
SELECT Statement (Defining the Transaction File)	4-4
ACQUIRE Statement (Acquiring Sessions)	4-6
Example of an Acquired Session	4-6
ACCEPT Statement (Checking Session Status)	4-7
Session Status Information	4-8
Example of an ACCEPT Statement	4-9
READ Statement (Receiving Data)	4-10
Example of a READ Statement	4-11
WRITE Statement (Performing Operations within a Session)	4-12
WRITE Statement Operations	4-13
Starting Remote Programs (Evoke Operations)	4-14
IDDU Evoke Operation Considerations	4-14
Sending Program Data with an Evoke Operation	4-15
Sending Data (Put Operations)	4-15
Ending Communications Transactions (End of Transaction Operations)	4-16
Ending Sessions (End of Session Operation)	4-17
Additional WRITE Statement Operations	4-17
Request to Change Direction Operation	4-17
SSP-ICF and Work Station Timer Operations	4-18
\$\$TIMER Operation Example	4-18
Negative Response Operations	4-19
Cancel Operations	4-20
Fail Operation	4-20
DROP Statement (Releasing a Session)	4-21
COBOL Operations Summary Chart	4-22
Return Code Processing in COBOL	4-23
COBOL Coding Examples	4-23

This chapter briefly describes the COBOL statements and operations that you use to write COBOL communications programs. The syntax of the COBOL statements is shown, and the communications operations are introduced.

To use the COBOL language with the Interactive Communications Feature (SSP-ICF), do the following:

- Configure and enable the subsystem. (These procedures are described in the appropriate subsystem reference manual.)
- Begin a communications session by issuing an ACQUIRE statement.
- Begin a program or procedure at the remote system and start a communications transaction. (If you are using the Intra subsystem, the program or procedure being started is in the same System/36.)
- Send or receive data.
- Check return codes.
- End the communications transaction.
- End the communications session.
- Disable the subsystem.

The COBOL operations you need for interactive communications are *introduced* in this chapter. The details about each operation—its function, syntax, programming considerations, and coding example (for some operations)—are described in each subsystem reference manual for which the operation is valid.

General (conceptual) information about these operations and diagrams showing how these operations work are given in Chapter 3 of the *SSP-ICF Guide and Examples* manual.

The operations you use in the communications portion of your program are similar to work station operations. In the noncommunications portion of your program, you can use all of the noncommunications operations that you normally use to process the data that is sent or received between your program and the remote program. Therefore, the noncommunications operations are not described in this manual.

If you are using the Intra or APPC subsystem, externally described field, format, and file definitions (also called data definitions) can be used to send data records. Data definitions, which describe data records and communications functions, are defined separately from the application program. The interactive data definition utility (IDDU) is used to create data definitions. Refer to the manual *Getting Started with Interactive Data Definition Utility* for more information.

The details about using IDDU with the Intra or APPC subsystem are described in the appropriate subsystem reference manual.

COBOL Statements Used for Communications

In COBOL, the communications operations are performed primarily by the statements shown in the following list. Only these statements are described in this chapter, and only the communications-related information about each one is given. The manual *Programming with COBOL* contains additional information about these statements and about all the other statements that can be used in a communications program.

Statement	Functions
SELECT	Defines the TRANSACTION file used for SSP-ICF operations
ACQUIRE	Acquires (starts) a session
ACCEPT	Gets the attributes of a session
READ	Receives data from the remote system
WRITE	Performs many of the SSP-ICF communications operations within a session
DROP	Releases the session

SELECT Statement (Defining the Transaction File)

Use the TRANSACTION file for SSP-ICF operations. Programming considerations for TRANSACTION files are described in the manual *Programming with COBOL*.

Use the SELECT statement in the FILE-CONTROL paragraph to define the TRANSACTION file. You must also open the file in the Procedure Division, and open it as *I-O*.

The syntax of the SELECT statement for a TRANSACTION file using interactive communications is:

```
SELECT file-name
```

```
ASSIGN TO WORKSTATION [ -name1 [-type] ] [ ,name2 [-type] ]
```

```
ORGANIZATION IS TRANSACTION
```

```
[ FILE STATUS IS data-name-1 [ ,data-name-4 ] ]
```

```
[ ACCESS MODE IS SEQUENTIAL ]
```

```
[ CONTROL-AREA IS data-name-5 ] .
```

S7910025-0

SELECT Clause: Specifies the name of the **TRANSACTION** file your program will use to send data to and receive data from a remote program.

ASSIGN Clause: Must be **WORKSTATION** for SSP-ICF. The **name-1** field specifies the 1- to 8-character name of the \$\$FGR-generated load member that contains the display format. If you are using IDDU, the **name-1** and **name-2** fields specify the display format and the IDDU file definition that contains the format definitions used to describe communications functions. The name field is not required if the file is to be used with SSP-ICF sessions only; however, the name field is required when using IDDU. The **type** field is used to specify whether the name is a \$\$FGR-generated load member or an IDDU file definition. Specify an S (or blank) for a \$\$FGR-generated load member or a C for an IDDU file definition.

ORGANIZATION Clause: Specifies the logical structure of a file. Must be TRANSACTION for SSP-ICF. TRANSACTION file organization allows you to control input and output operations.

FILE STATUS Clause: Allows you to check the status of input and output operations from or to the TRANSACTION file. The FILE STATUS area consists of a 2-byte COBOL return code (data-name-1) and a 4-byte IBM file status code (data-name-4) that contains the interactive communications return code. The interactive communications return code consists of two 2-byte return codes (a major and a minor return code). You must define data-name-4 in the Data Division as a 4-character alphanumeric data item.

ACCESS MODE Clause: Must always be SEQUENTIAL for TRANSACTION files.

CONTROL-AREA Clause: Specifies the 12-byte data item that receives feedback information after each TRANSACTION file input operation. The third and fourth characters of this area contain the symbolic identifier of the session or display station from which input was obtained. The symbolic identifier must be defined as a 2-byte alphanumeric data item. The remainder of the characters contain information concerning display stations only, and are described in the manual *Programming with COBOL*.

For an example of how to code the SELECT statement, see the sample programs in the *SSP-ICF Guide and Examples* manual.

ACQUIRE Statement (Acquiring Sessions)

To start (acquire) a session, use the ACQUIRE statement to specify the session you are acquiring for a specified TRANSACTION file. Each ACQUIRE statement starts one session between your program and the remote system. If your program is started by a procedure start request (remotely started program), no ACQUIRE statement is needed. However, once a remotely started program is running, it can issue acquire operations and start other sessions (depending on the type of subsystem you are using).

```
ACQUIRE {literal } FOR file-name.  
         {identifier}
```

S7910026-0

ACQUIRE Clause: Specifies a 2-character identifier for the session to be acquired. The first character of the session identifier must be numeric (0 through 9), and the second character must be alphabetic (A through Z, \$, #, or @). Use the **literal** parameter to specify the actual identifier value, or use the **identifier** parameter to specify a 2-character data item that contains the session identifier. The session identifier must be the same as the SYMID parameter specified in the SESSION statement.

FOR Clause: Specifies, in the **file-name** parameter, the name of the TRANSACTION file.

Example of an Acquired Session

```
ACQUIRE '1S' FOR TRANSACTION-FILE.
```

OR

```
ACQUIRE SSP-ICF-SESSION FOR TRANSACTION-FILE.
```

An example of a SESSION statement for this ACQUIRE statement is:

```
// SESSION LOCATION-CHICAGO, SYMID-1S, BATCH-YES
```

ACCEPT Statement (Checking Session Status)

The ACCEPT statement is used to get the attributes of a session; it is the equivalent of the SSP-ICF get attributes operation.

The syntax of the ACCEPT statement is:

```
ACCEPT identifier-1 FROM mnemonic-name
```

```
    [ FOR { identifier-2 }  
      { literal } ] .
```

S7910027-0

ACCEPT Clause: The **identifier-1** parameter must specify an area to be used for an attribute data record. The session attributes are moved into the attribute data record area when the accept operation is performed. The TRANSACTION file must be opened before the accept operation can get the attributes of the session.

FROM Option: Specifies the symbolic name associated with ATTRIBUTE-DATA in the SPECIAL-NAMES clause (coded in the Environment Division).

FOR Option: If the FOR option is specified, a get attributes operation is performed for the session specified by the **identifier-2** or **literal** parameter. If the FOR option is not specified and only one session or display station is attached to the TRANSACTION file, a get attributes operation is performed for that session or display station. If the FOR option is not specified, and multiple sessions and display stations are attached, a get attributes operation is performed for the last session or display station for which an input or output operation was performed.

For Intra and APPC subsystems only, a get status operation may be performed instead of a get attributes operation. The get status operation returns additional information about a specific session.

Session Status Information

When the get attributes (ACCEPT) operation is specified, the first 10 bytes of the record area are as follows:

Position	Value	Meaning
1	A	Session not yet acquired.
	C	Session is an acquired session.
	R	Session is a remotely started session.
2	N	Input not invited for this session.
	I	Input invited for this session, but no input is available.
	O	Invited input is available for this session.
3 through 10	Name	Location name (specified during configuration and on the SESSION OCL statement).

If the get status (ACCEPT) operation is specified, the additional fields are as follows (starting with byte 11):

Position	Value	Meaning
11	I	Intra subsystem is being used.
	A	APPC subsystem is being used.
12	0	Synchronization level is NONE.
	1	Synchronization level is CONFIRM.
13	M	Mapped conversation.
14 through 16	Blanks	Reserved.

If a field is not used, enter the correct number of blanks for the unused field.

Example of an ACCEPT Statement

SPECIAL NAMES.

ATTRIBUTE-DATA IS ATTRIBUTES.

•
•
•

DATA DIVISION.

77 SSP-ICF-SESSION PIC XX VALUE '1S'.

•
•
•

01 SESSION-ATTRIBUTES.

03 SESSION-STATUS PIC X.

03 INVITE-STATUS PIC X.

03 SESSION-NAME PIC X(8).

PROCEDURE DIVISION.

•
•
•

ACCEPT SESSION-ATTRIBUTES FROM ATTRIBUTES.

OR

ACCEPT SESSION-ATTRIBUTES FROM ATTRIBUTES FOR '1S'.

OR

ACCEPT SESSION-ATTRIBUTES FROM ATTRIBUTES FOR SSP-ICF-SESSION.

READ Statement (Receiving Data)

The READ statement is used to receive data from a remote program. The statement performs either a get or accept input operation depending on whether the TERMINAL option is specified.

For SSP-ICF input operations, the maximum amount of data that can be received by a program is 4075 bytes for all subsystems except Intra, IMS, and APPC, for which the maximum is 4096 bytes.

The syntax of the READ statement is:

```
READ file-name RECORD
```

```
[ INTO identifier-1 ] [ TERMINAL IS { identifier-2 }  
                        { literal-1 } ]
```

```
[ NO DATA imperative-statement-1 ]
```

```
[ AT END imperative-statement-2 ] .
```

S7910028-

TERMINAL Option: If the TERMINAL option is specified, a get operation is performed for the session specified. If the TERMINAL option is not specified, an accept input operation is performed.

NO DATA Option: If specified, this option allows the statement specified in the imperative-statement-1 parameter to be processed if data is not available for this READ statement.

AT END Option: If specified, this option allows the statement specified in the imperative-statement-2 parameter to be executed if the READ statement is issued and an invite operation is not currently outstanding.

For more information about the READ statement, see the manual *Programming with COBOL*.

Example of a READ Statement

```
READ TRANSACTION-FILE.
```

OR

```
READ TRANSACTION-FILE,  
  TERMINAL IS SSP-ICF-SESSION.
```

The first **READ** statement performs an accept input operation. The second **READ** statement performs a get operation.

WRITE Statement (Performing Operations within a Session)

Use the WRITE statement to perform many of the communications operations between two programs once a session has been started. The type of operation is determined by the value specified for the FORMAT parameter (which is described below).

The syntax of the WRITE statement is:

```
WRITE record-name [ FROM identifier-1 ]  
  
[ FORMAT IS { identifier-2 }  
             { literal-1 } ]  
  
[ TERMINAL IS { identifier-3 }  
              { literal-2 } ]  
  
[ { INDICATOR [ IS ] }  
  { INDICATORS [ ARE ] }  
  { INDIC } identifier-4 ] .
```

S7910029-0

record-name: Specifies the output area that contains any of the information required with the operation.

FORMAT Option: Identifies the SSP-ICF operation that is to be performed. See “COBOL Operations Summary Chart” later in this chapter for a complete list of the operations you can specify in the FORMAT parameter. Any of the SSP-ICF operations beginning with \$\$ can be used.

If you are using IDDU, identifies the format definition that externally describes the SSP-ICF operation that is to be performed. For more information about using IDDU with COBOL, refer to the manual *Programming with COBOL*.

TERMINAL Option: Specifies the identifier of the session during which the operation is to be performed. If the TERMINAL option is not used, the operation is performed for the session associated with the last READ or WRITE statement.

See the manual *Programming with COBOL* for a description of other WRITE statement parameters.

WRITE Statement Operations

The following are the primary communications operations you can perform using the WRITE statement.

- Evoke operations: To start a remote program
- Put operations: To send data to the remote program
- End of transaction operations: To end a communications transaction
- End of session operation: To end the session in which the remote program was started

These operations are described in the following pages; other types of operations that can also be specified by the WRITE statement are described under “Additional WRITE Statement Operations” later in this chapter. An example of each type of operation that can be specified in the FORMAT option is also provided.

Starting Remote Programs (Evoke Operations)

To start a remote program or procedure and to start a communications transaction, specify an **evoke** operation in the FORMAT parameter of the WRITE statement (\$\$EVOK, \$\$EVOKNI, or \$\$EVOKET). See “WRITE Statement (Performing Operations within a Session)” earlier in this chapter for the syntax of the WRITE statement.

With an evoke operation, you must send an **evoke parameter list**. If you specify the evoke operation in the FORMAT parameter of the WRITE statement, the parameters (fields) in that list must be specified in the following order:

Positions	Field Description
1 through 8	The name of the program or procedure to be evoked (left-adjusted).
9 through 16	The password you use to sign on the remote system (left-adjusted).
17 through 24	The user identifier you use to sign on the remote system (left-adjusted).
25 through 32	The name of the remote system library that contains the program or procedure to be evoked (left-adjusted).
32 through 51	Reserved.
52 through 55	The length of data, positional parameters, or keyword parameters.
56 through xxxx	User data, positional procedure parameters, or keyword parameters. (Leading blanks are ignored.)

If a field is not used, enter the correct number of blanks for the unused field.

IDDU Evoke Operation Considerations:

If you are using IDDU, you can select one of the following IDDU functions to perform an evoke operation: evoke process (EVOKE keyword), evoke process and send detach (EVOKE and DETACH keywords), or evoke process and invite (EVOKE and INVITE keywords).

Because IDDU does not reserve positions for unused parameters or duplicated parameters, one or more of the parameters in the evoke parameter list may not be specified or may be specified in a different order. Therefore, after defining an IDDU format to be used with the evoke operation, you should do the following:

1. Use IDDU to print a format definition listing for the format you defined.
2. Use the listing to determine the order and starting position of each of the parameters in the evoke parameter list.

Sending Program Data with an Evoke Operation

Data sent with an evoke operation can be **parameters** to be used by the evoked **procedure**, or it can be **data** to be used by the evoked **program**. When you are creating a procedure that is to have parameters sent to it by an evoke operation, answer *no* to the prompt **PROGRAM DATA IN THE INCLUDE STATEMENTS** on the end of job menu in the SEU procedure, or specify **PDATA-NO** on the **COPY** control statement for **\$MAINT**. If you want data to be sent to a program, answer *yes* to the SEU prompt for program data, or specify **PDATA-YES** on the **COPY** control statement.

Sending Data (Put Operations)

To send a data record to a remote system or program, specify a **put** operation in the **FORMAT** parameter of the **WRITE** statement (**\$\$SENDNI**, **\$\$SENDE**, **\$\$SENDET**, **\$\$SENDNF**, or **\$\$SENDFM**). If you are using **IDDU** functions, specify a put operation in a user-defined field, select the send detach, or specify a put operation in a user-defined field and select the invite. See “**WRITE Statement (Performing Operations within a Session)**” earlier in this chapter for the syntax of the **WRITE** statement.

Each type of put operation requires the following fields in the output area:

length field: A 4-byte field that contains, in decimal, the length of the user data. An output length of zero for a **\$\$SEND** operation performs an **invite** operation. An output length of zero can also be used for **\$\$SENDE**, **\$\$SENDET**, and **\$\$SENDNI** operations.

A maximum of 4075 bytes can be sent by a put operation for all subsystems except Intra, IMS, and APPC, for which the maximum is 4096 bytes.

data field: The field containing the user data to be sent.

For example, the following **WRITE** statement sends one data record:

```
01 DATA-RECORD.
03 RECORD-LENGTH    PIC 9(4).
03 THE-RECORD       PIC X(256).
  ●
  ●
  ●
WRITE TRANSACTION-RECORD FROM DATA-RECORD,
  FORMAT IS '$$SENDNI', TERMINAL IS SSP-ICF-SESSION.
```

Ending Communications Transactions (End of Transaction Operations)

You can end the transaction by specifying the **put end of transaction** operation or the **evoke end of transaction** operation in the **FORMAT** parameter of the **WRITE** statement (**\$\$SENDET** for the put end of transaction or **\$\$EVOKET** for the evoke end of transaction). If you are using **IDDU** functions, select the **send detach** for the put end of transaction or the **evoke process and send detach** for the evoke end of transaction.

- If your program has finished sending data, a put end of transaction operation in the **FORMAT** parameter tells the remote system that you have no more data to send and that you do not expect to receive any data.
- If your program is receiving data, check for an end of transaction return code received from the subsystem to determine when the remote system has finished sending.
- If you want to start a program or procedure at the remote system and immediately end the transaction, an evoke end of transaction operation in the **FORMAT** parameter indicates that your program does not expect to receive any data. For example, you can send data to a remote program and then start a different program on the remote system to use that data:
 - Your program starts program A at the remote system and sends data to program A.
 - Program A stores the data on disk.
 - When your program has finished sending data to program A, your program uses the put end of transaction operation to end program A.
 - Your program then uses the **\$\$EVOKET** operation to start program B at the remote system.
 - Program B processes the data that program A stored on disk previously.

Once the transaction has ended, you can end the session or start another transaction with this or another session.

The following statement sends a put end of transaction operation and, therefore, tells the remote system that this is the end of this communications transaction:

```
WRITE TRANSACTION-RECORD FROM DATA-RECORD,  
  FORMAT IS '$$SENDET', TERMINAL IS SSP-ICF-SESSION.
```

Ending Sessions (End of Session Operation)

To end a session with a remote system, use the **DROP** statement, or specify the **\$\$EOS** operation in a **WRITE** statement. (The **DROP** statement is described later in this chapter.)

To end a session using the **WRITE** statement, specify **\$\$EOS** in the **FORMAT** parameter. For example:

```
WRITE TRANSACTION-RECORD,  
    FORMAT IS '$$EOS', TERMINAL IS SSP-ICF-SESSION.
```

Additional WRITE Statement Operations

The following are additional interactive communications operations you can specify on the **WRITE** statement:

- Request to change direction operation
- Set timer operation
- Negative response operations (used only with the Intra and SNUF subsystems)
- Cancel operations, for canceling a group (chain) of data records (used only with the Intra and SNUF subsystems), or for canceling any valid invite operation for which no input has yet been received (used only with the Intra and BSCCEL subsystems)
- Fail operation (used only with the Intra, Peer, and APPC subsystems)

Request to Change Direction Operation

To request a change in the direction of transmission, specify a **request to change direction** operation in the **FORMAT** parameter of the **WRITE** statement (**\$\$RCD**). If you are using **IDDU** functions, select the send request to write and invite. After you issue the **\$\$RCD** operation, your program must continue to receive data until it receives a return code indicating that the remote system is ready to begin receiving data. No additional parameters or data is associated with the **\$\$RCD** operation.

The following **WRITE** statement shows how to request that the remote system stop sending so that your program can send data:

```
WRITE TRANSACTION-RECORD,  
    FORMAT IS '$$RCD', TERMINAL IS SSP-ICF-SESSION.
```

SSP-ICF and Work Station Timer Operations

To use the SSP-ICF and work station timer, use the **\$\$TIMER** operation in the **WRITE** statement in the format:

hhmmss

where **hh** is hours, **mm** is minutes, and **ss** is seconds.

A return code is returned to your program when the time has ended. Use an accept operation to determine when the time has ended and to determine the action to take next based on the return code received.

\$\$TIMER Operation Example

```
01  TIMER                                PIC X(6) VALUE '000030'.  
    .  
    .  
    .  
    WRITE TRANSACTION-RECORD FROM TIMER,  
      FORMAT IS '$$TIMER', TERMINAL IS SSP-ICF-SESSION.  
    .  
    .  
    .  
    READ TRANSACTION-FILE,  
      IF RETURN-CODE EQUAL '0310',  
      THEN  
        GO TO TIMER-EXPIRED.
```

This example sets the timer to 30 seconds, then issues an accept input operation and checks for a return code.

Negative Response Operations

To tell the remote system or program that your program found something wrong with the data it received (that is, to send a negative response), use the **FORMAT** parameter in the **WRITE** statement to specify one of the **negative response** operations (**\$\$NRSP** or **\$\$NRSPNI**). These operations can only be used with the Intra and SNUF subsystems.

Sense data can also be sent with the negative response. The following is the format of the sense data:

Positions	Description
1	Indicates whether sense data is being sent: 0 or blank indicates that <i>no</i> sense data is being sent; 8 indicates that sense data is being sent.
2 through 9	The sense data sent with the negative response. The sense data must begin with 10xx, 08xx, or 0000. The last four positions are user-defined.

For example, the following statements send a negative response operation that includes the sense data 08008000:

```
01  NEG-RESP-REC.  
03  REC-LEN          PIC X(4) VALUE '0008'.  
03  RESP-DATA       PIC X(08) VALUE '08008000'.  
  ●  
  ●  
  ●  
WRITE TRANSACTION-RECORD FROM NEG-RESP-REC,  
  FORMAT IS '$$NRSP', TERMINAL IS SSP-ICF-SESSION.
```


Cancel Operations

There are two types of cancel operations: those that cancel a *group* of records that has just been sent (used by the SNUF and Intra subsystems only), and the *cancel invite* operation, which cancels an invite operation for which no input has yet been received (used by the BSCCEL and Intra subsystems only).

- For the SNUF and Intra subsystems only, to cancel a group of records that has just been sent, use the **FORMAT** parameter in the **WRITE** statement to specify one of the **cancel** operations (**\$\$CANL** or **\$\$CANLNI**). The cancel operations have no additional parameters or data associated with them.

For example, the following **WRITE** statement cancels the current chain of records:

```
WRITE TRANSACTION-RECORD FROM SSP-ICF-RECORD,  
  FORMAT IS '$$CANL', TERMINAL IS SSP-ICF-SESSION-1S.
```

- For the BSCCEL and Intra subsystems only, to cancel any valid invite operation for which no input has yet been received from any invited session, use the **FORMAT** parameter in the **WRITE** statement to specify the **cancel invite** operation (**\$\$CNLINV**). The cancel invite operation has no additional parameters or data associated with it. For restrictions on the use of this operation by the BSCCEL subsystem, see “Cancel Invite Operation” in Chapter 2 of the *SSP-ICF Base Subsystems Reference* manual.

For example, the following **WRITE** statement cancels an invite operation that no session has yet responded to:

```
WRITE TRANSACTION-RECORD FROM SSP-ICF-RECORD,  
  FORMAT IS '$$CNLINV', TERMINAL IS SSP-ICF-SESSION-1S.
```

Fail Operation

To tell the remote system that your program detected an abnormal condition (for example, it received incorrect data), specify a **fail** operation in the **FORMAT** parameter of the **WRITE** statement (**\$\$FAIL**). If you are using **IDDU** functions, select the send fail. The fail operation has no additional parameters associated with it, and no data can be sent with the fail operation. The fail operation can be used only with the Intra, Peer, and **APPC** subsystems.

For example, the following **WRITE** statement sends a fail indication:

```
WRITE TRANSACTION-RECORD,  
  FORMAT IS '$$FAIL', TERMINAL IS SSP-ICF-SESSION.
```

DROP Statement (Releasing a Session)

To release a session, use the **DROP** statement, which issues a release operation for a particular session. You must specify the name of the **TRANSACTION** file associated with this session. You can specify a **literal** for the session identifier or an **identifier** that refers to a 2-character alphanumeric data item that contains the session identifier. The session identifier must correspond to the **SYMID** parameter specified on the corresponding **SESSION** statement.

The syntax of the **DROP** statement is:

```
DROP { literal  
        identifier } FROM file-name.
```

S7910030-0

All acquired sessions are automatically released when the application program ends.

COBOL Operations Summary Chart

The following chart shows the valid COBOL operations for each subsystem. An x in a subsystem column indicates that the subsystem supports the operation. A - in a column indicates that the subsystem does not support the operation.

COBOL SSP-ICF Operation	COBOL Operation	Communications Subsystems										
		Intra	BSC	CEL	CCP	CICS	IMS	3270 ¹	Finance	Peer	SNUF	APPC
Accept input ²	READ	x	x	x	x	x	x	x	x	x	x	x
Acquire	ACQUIRE	x	x	x	x	x	x	x	x	x	x	x
Cancel	\$\$CANLNI	x	-	-	-	-	-	-	-	-	x	-
Cancel invite	\$\$CNLINV	x	x	-	-	-	-	-	-	-	-	-
Cancel then invite	\$\$CANL	x	-	-	-	-	-	-	-	-	x	-
End of session	\$\$EOS	x	x	x	x	x	x	x	x	x	x	x
Evoke	\$\$EVOKNI	x	x	x	x	x	x ³	-	x	x	x	x
Evoke end of transaction	\$\$EVOKET	x	x	-	x	x	-	-	x	x	x	x
Evoke then invite	\$\$EVOK	x	x	x	x	x	x ³	-	x	x	x	x
Fail	\$\$FAIL	x	-	-	-	-	-	-	x	-	-	x
Get ²	READ	x	x	x	x	x	x	x	x	x	x	x
Get attributes ⁴	ACCEPT	x	x	x	x	x	x	x	x	x	x	x
Get status ⁶	ACCEPT	x	-	-	-	-	-	-	-	-	-	x
Invite ⁵	\$\$SEND	x	x	x	x	x	x	x	x	x	x	x
Negative response	\$\$NRSPNI	x	-	-	-	-	-	-	-	-	x	-
Negative response then invite	\$\$NRSP	x	-	-	-	-	-	-	-	-	x	-
Put	\$\$SENDNI	x	x	x	x	x	-	x	x	x	x	x
Put end of file/chain	\$\$SENDE	x	x	x	x	-	x	x	x	x	x	-
Put end of transaction	\$\$SENDET	x	x	-	x	x	-	-	x	x	x	x
Put FMH	\$\$SENDNF	x	-	-	-	-	-	x	-	x	-	-
Put FMH then invite	\$\$SENDFM	x	-	-	-	-	-	x	-	x	-	-
Put then invite	\$\$SEND	x	x	x	x	x	x	x	x	x	x	x
Release	DROP	x	x	x	x	x	x	x	x	x	x	x
Request to change direction then invite	\$\$RCD	x	x	x	-	-	-	-	x	x	x	x
Set timer	\$\$TIMER	x	x	x	x	x	x	x	x	x	x	x

1Although the BSC 3270 subsystem is not part of SSP-ICF, its operations are listed here to show its similarities to other subsystems.
2The READ statement performs either a get or an accept input operation, depending on whether the TERMINAL option is specified.
3Valid only when the DATAID and FLDLTH parameters are specified on the SESSION OCL statement, and when the HOSTNAME parameter on the SESSION statement is CICS or IMS.
4Valid only when the ATTRIBUTE-DATA keyword is used in the SPECIAL-NAMES paragraph and the SPECIAL-NAMES name is specified in the ACCEPT statement.
5Valid only when a \$\$SEND operation is issued with a record length of zero.
6The record area must be at least 128 bytes long.

Return Code Processing in COBOL

Following each operation, a return code consisting of a major code and a minor code is given to your program in the IBM-extended FILE STATUS area. In addition, a COBOL return code is given in the FILE STATUS field identifying the status of the operation. The following list shows the COBOL return code as returned in the appropriate FILE STATUS data field and the corresponding SSP-ICF return code(s).

Use this list to determine the SSP-ICF return code that corresponds to the COBOL return code. Then see the description of the SSP-ICF return code in the appropriate subsystem reference manual. (For example, the 02xx group below is described in each subsystem reference manual in the *Major Code 02* box description, which applies to all the return codes beginning with 02.) All of the return codes that are valid for that subsystem are described in that manual. A summary listing of all the codes for all the subsystems is in Appendix B.

SSP-ICF Return Code	COBOL Return Code
00xx, 03xx, 0800	00
01xx	01
02xx	9A
04xx	9I
1100	10
2800	9E
3401	9G
80xx	30
81xx	92
82xx	9C
83xx	9N or 9K

COBOL Coding Examples

For a complete example of a COBOL communications program, see "Writing a COBOL Program That Uses the Intra Subsystem" in Chapter 6 of the *SSP-ICF Guide and Examples* manual. The same programming example described in the Intra chapter is also used in the other subsystem chapters, but only the changed areas needed to allow communications with that type of remote system are shown.

Chapter 5. Programming SSP-ICF with RPG II

File Description Specifications	5-2
RPG II Communications Operations	5-3
Starting Remote Programs (Evoke Operations)	5-3
IDDU Evoke Operation Considerations	5-3
Sending Data (Put Operations)	5-5
Request to Change Direction Operation	5-6
Set Timer Operation	5-7
Negative Response Operations	5-8
Cancel Operations	5-9
Fail Operation	5-9
End of Session Operation	5-9
WORKSTN Operations	5-10
ACQ (Acquire) Operation	5-10
REL (Release) Operation	5-11
NEXT Operation	5-12
READ Operation	5-13
RPG Cycle Input	5-14
RPG II Operations Summary Chart	5-14
Return Code Processing in RPG II	5-15
INFSR Subroutine Coding Considerations	5-18
RPG II Status Values	5-19
RPG II Programming Considerations	5-20
Using Continuation Options on the File Description Specifications	5-20
NUM Continuation Option	5-20
SAVDS Continuation Option	5-21
IND Continuation Option	5-21
SLN Continuation Option	5-21
ID Continuation Option	5-21
INFSR Continuation Option	5-21
INFDS Continuation Option	5-22
FMST Continuation Option	5-22
CFILE Continuation Option	5-22
SRT and MRT Program Considerations	5-22
End-of-File Considerations	5-23
Release Considerations	5-23
Restrictions for WORKSTN Files	5-24
Input and Output Considerations	5-24
RPG II Coding Examples	5-25

The interactive communications portion of an RPG II program consists of preparing data for transmission, processing data that was received, using existing work station operations and additional SSP-ICF operations to perform communications operations, and properly handling return codes. Noncommunications data processing varies depending on the application; these noncommunications functions are not described in this manual.

The operations you use in interactive communications are similar to work station operations. The file used is a WORKSTN file, the same input operations are used, and the output operations are performed via special SSP-ICF formats.

If you are using the Intra or APPC subsystem, externally described fields, format, and file definitions (also called data definitions) can be used to send data records. Data definitions, which describe data records and communications functions, are defined separately from the application program. The interactive data definition utility (IDDU) is used to create data definitions. Refer to the manual *Getting Started with Interactive Data Definition Utility* for more information about IDDU.

The details about using IDDU with the Intra or APPC subsystem are described in the appropriate subsystem reference manual.

File Description Specifications

When you use RPG II for interactive communications, you must complete the file description specifications. These specifications should contain the same information that you would code for a WORKSTN file. File description specifications for a WORKSTN file are described in the manual *Programming with RPG II*.

File Description Specifications

For the valid entries for a system, refer to the RPG reference manual for that system

Line	Form Type	Filename	File Type		Mode of Processing				Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	File Addition/Unordered	
			File Designation	End of File	Length of Key Field or of Record Address Field	Record Address Type	Type of File Organization or Additional Area	Overflow Indicator						Key Field Starting Location	Extension Code E/L
			IO/U/C/D PS/C/R/T/D/F	E	L/R	A/P/I/K I/X/D/T/R/I or 2							A/U	R/U/N	Tape Rewind File Condition U1-U8, UC
			Sequence	File Format	Block Length	Record Length	External Record Name				Continuation Lines	Option	Entry		
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									
0 2	F	ICFILE	CD				84		WORKSTN						
0 3	F										KNUM		2		
0 4	F										KINFDS	RECD			
0 5	F										KINFSR	ERRS			
0 6	F										KFMTS	*NONE			
0 7	F										KID	MSID			
0 8	F										KCFILE	COMFILE			
0 9	F														

(Only if you are using IDDU.)

RPG II Communications Operations

To assist in coding interactive communications operations in RPG II, predefined operations are provided.

Note: Some of the operations have data fields associated with them. Space for these fields, in the locations described, must be reserved even if the field is not coded. All values in these fields must be character values.

The following sections describe the communications operations.

Starting Remote Programs (Evoke Operations)

To start a remote program or procedure and to start a communications transaction, specify an **evoke** operation (**\$\$EVOK**, **\$\$EVOKNI**, or **\$\$EVOKET**).

With an evoke operation, you must send an **evoke parameter list**. You define these parameters as fields on the output specifications.

Positions	Field Description
1 through 8	The name of the program or procedure to be evoked (left-adjusted).
9 through 16	The password you use to sign on the remote system (left-adjusted).
17 through 24	The user identifier you use to sign on the remote system (left-adjusted).
25 through 32	The name of the remote system library that contains the program or procedure to be evoked (left-adjusted).
33 through 52	Reserved.
53 through 56	The length (in decimal) of user data, if any (right-adjusted).
57 through xxxx	User data or positional parameters.

If a field is not used, enter the correct number of blanks for the unused field.

IDDU Evoke Operation Considerations:

If you are using IDDU, you can select one of the following IDDU functions to perform an evoke operation: evoke process (**EVOKE** keyword), evoke process and send detach (**EVOKE** and **DETACH** keywords), or evoke process and invite (**EVOKE** and **INVITE** keywords).


Sending Data (Put Operations)

To send a data record to a remote system or program, specify a **put** operation (\$\$SENDNI, \$\$SENDE, \$\$SENDET, \$\$SENDNF, or \$\$SENDNF). If you are using IDDU functions, specify a put operation in a user-defined field, select the send detach, or specify a put operation in a user-defined field and select the invite.

For each put operation, you specify two fields on the output specifications as follows:

Positions	Field Description
1 through 4	Length, in decimal, of the user data (right adjusted). An output record length of zero for \$\$SEND operation performs an invite operation. An output length of zero is also allowed for \$\$SENDE, ##SENDET, and \$\$SENDNI operations. A maximum of 4075 bytes can be sent by a put operation for all subsystems except Intra, IMS, and APPC, for which the maximum is 4096 bytes.
5 through xxxx	User data to be sent.

The following figure shows how you can enter a \$\$SENDET operation on the output specifications:



International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21-9090
Printed in U.S.A.
75 76 77 78 79 80

Program		Keying Instruction	Graphic	Card Electro Number	
Programmer	Date		Key		

Page 1 of 2 Program Identification 75 76 77 78 79 80

Line	Form Type	Filename or Record Name	Type (H/D/T/E)		Space		Skip		Output Indicators			Field Name or EXCPT Name	End Position in Output Record	Commas		Zero Balances to Print		No Sign		CR		-		5 - 9 = User Defined
			OR	NR	BE	AF	BE	AF	Net	And	And			Yes	No	Yes	No	1	2	3	4	A	B	
01	O	ICFILE	E									SENDET												
02	O												K8	'\$\$SENDET'										
03	O												4	'0080'										
04	O											LSTREC	84											
05	O																							
06	O																							

Request to Change Direction Operation

To request a change in the direction of transmission, specify a **request to change direction** operation (\$\$RCD). This operation has no fields associated with it; therefore, you need only to code \$\$RCD on the output specifications. If you are using IDDU functions, select the send request to write and invite.

After you issue the request to change direction operation, your program must continue to receive data until the remote system sends a change direction indication. This is indicated in the return code following the input operation.

Set Timer Operation

To specify a time interval in your program, use the **set timer** operation (\$\$TIMER). Enter the following field on the output specifications:

Positions	Field Description
1 through 6	Interval time to be set. The time should be specified in hours, minutes, and seconds (hhmmss).

Note: To use the \$\$TIMER operation, the ID field in the file must identify a display station or session that is attached to the program.

To check that the time has ended, use a READ operation *not* preceded by a NEXT operation. The NEXT operation causes input to come from a specified session (not the timer) during the READ operation (see “NEXT Operation” and “READ Operation” later in this chapter for more information about these operations).

The following figure shows how you can enter the value on the output specifications to set the timer for 2 minutes:

International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21-9090
 Printed in U.S.A.
 Program Identification 75 76 77 78 79 80

Program			Keying Instruction		Graphic		Card Electro Number		Page 1 of 2	
Programmer			Date		Key				Program Identification 	

Line	Form Type	Filename or Record Name	Type (H/D/T/E)				Space	Skip	Output Indicators			Field Name or EXCPT Name	Edit Code	End Position in Output Record	Constant or Edit Word								Commas		Zero Balances to Print		No Sign		CR		X = Remove Plus Sign	Y = Date Field Edit	Z = Zero Suppress	5 - 9 = User Defined			
			O	R	A	N			D	Before	After				Not	And	And	Not	1	2	3	4	A	B	C	D	J	K	L	M							
01	O																																				
02	O																																				
03	O	ICFILE										TIMER																									
04	O												K7																								
05	O																																				
06	O																																				
07	O																																				

Negative Response Operations

To tell the remote system or program that your program found something wrong with the data it received (that is, to send a negative response), use a **negative response** operation (\$\$NRSP or \$\$NRSPNI). These operations are for the Intra and SNUF subsystems only.

Enter the operation on the output specifications. You can also enter the following two optional fields on the output specifications:

Positions	Field Description
1	Indicates whether sense data is being sent: 0 or blank indicates that no sense data is being sent; 8 indicates that sense data is being sent.
2 through 9	The sense data to be sent with the negative response. The sense data is user-defined, but the first 4 characters must be 10xx, 08xx, or 0000. For a description of the sense data, see the <i>Systems Network Architecture Reference Summary</i> .

The following figure shows how you can enter the negative response values on the output specifications:

RPG OUTPUT SPECIFICATIONS

GX21-9090
Printed in U.S.A.
75 76 77 78 79

IBM International Business Machines Corporation

Program	Date	Keying Instruction	Graphic Key	Card Electro Number
---------	------	--------------------	-------------	---------------------

Page of

Program Identification:

Line	Filename or Record Name	Type P/D/T/E	Space	Skip	Output Indicators	Field Name or EXCPT Name	End Position in Output Record	Commas	Zero Balances to Print	No Sign	CR	X = Remove Plus Sign	5-9 = User Defined
					And			Yes	Yes	1	A	Y = Date	
					And			No	No	2	B	J = Field Edit	
					Not			No	Yes	3	C	Z = Zero Suppress	
					Not			No	No	4	D	M	
						*AUTO							
	BLCFILE	E				NRSP							
							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						

Cancel Operations

There are two types of cancel operations: those that cancel a *group* of records that has just been sent (used by the SNUF and Intra subsystems only), and the *cancel invite* operation, which cancels an invite operation for which no input has yet been received (used by the BSCCEL and Intra subsystems only).

- For the SNUF and Intra subsystems only, to cancel a group of records that has just been sent, specify a **cancel** operation (\$\$CANL or \$\$CANLND). The cancel operations have no fields associated with them; therefore, you need only enter the operation on the output specifications.
- For the BSCCEL and Intra subsystems only, to cancel any valid invite operation for which no input has yet been received from any invited session, use the FORMAT parameter in the WRITE statement to specify the **cancel invite** operation (\$\$CNLINV). The cancel invite operation has no additional parameters or data associated with it. For restrictions on the use of this operation by the BSCCEL subsystem, see “Cancel Invite Operation” in Chapter 2 of the *SSP-ICF Base Subsystems Reference* manual.

Fail Operation

To tell the remote system that your program detected an abnormal condition (for example, it received incorrect data), specify a **fail** operation (\$\$FAIL). If you are using IDDU functions, select the send fail. The fail operation has no fields associated with it, and no data can be sent by the operation. Therefore, you need only enter the operation on the output specifications. The fail operation can be used only with the Intra, Peer, and APPC subsystems.

End of Session Operation

To end a session with a remote system, use the **end of session** operation (\$\$EOS). The end of session operation has no fields associated with it; therefore, you need only to enter the operation on the output specifications.

*Note: If an RPG II program is started with an operation that includes an end of transaction indication (by an *EXEX procedure start request), the program should issue an end of session or a release operation to free the session ID entry in the internal table of identifiers for the WORKSTN file.*

WORKSTN Operations

The following WORKSTN operations and RPG cycle input are used with interactive communications:

- ACQ (acquire) operation
- REL (release) operation
- NEXT operation
- READ operation

ACQ (Acquire) Operation

The ACQ operation acquires the session specified by the 2-character session identifier (literal or variable) in factor 1 of the calculation specifications. The first character of the identifier must be numeric (1 through 9), and the second character must be alphabetic (A through Z, \$, #, or @). Factor 2 specifies the name of the WORKSTN file from the file description specifications.

If an error occurs when the program attempts to acquire the session, the indicator in positions 56 and 57 is set on, and the next calculation step is executed. If no indicator is specified, the program halts unless the INFSR subroutine is specified in the program. If the INFSR subroutine is specified, the subroutine receives control. See "Return Code Processing in RPG II" later in this chapter for more details on error handling.

The following figure shows how you can specify an ACQ operation on the calculation specifications:

IBM International Business Machines Corporation GX21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Program			Keying Instruction		Graphic		Card Electro Number								Page 12 of		Program Identification 75 76 77 78			
Programmer			Date		Key															

C	Line	Form Type Control Level (L,O,L9), LR,SR,ANVOR	Indicators					Factor 1	Operation	Factor 2	Result Field		Resulting Indicators			Comments
			Not	And	And	Not	Not				Name	Length	Plus	Minus	Zero	
0	1	C	NO				'LS'	ACQ	ICFILE						90	
0	2	C						-OR-								
0	3	C	NO				SSNID	ACQ	ICFILE						90	
0	4	C														
0	5	C														

NEXT Operation

The NEXT operation forces the next input to the program to come from the session specified in factor 1 (literal or variable) of the calculation specifications. Factor 2 contains the name of the WORKSTN file for which the operation is requested.

If more than one NEXT operation is specified between input (READ or primary file input) operations, only the last operation has any effect.

If an error occurs during the NEXT operation, the indicator in positions 56 and 57 is set on, and the next calculation step is executed. If no indicator is specified, the program halts unless the INF SR subroutine is specified in the program. If the INF SR subroutine is specified, the subroutine receives control.

See "Return Code Processing in RPG II" later in this chapter for more details on error handling.

The following figure shows how you can specify the NEXT operation on the calculation specifications:

IBM		International Business Machines Corporation		RPG CALCULATION SPECIFICATIONS																				GX21-909: Printed in I	
Program				Keying Instruction				Graphic				Card Electro Number				Page 1 2 of		Program Identification 75 76 7							
Programmer				Date				Key																	
C	Line	Form Type Control Level (L,O,L,B), L,F,SR,AN/OR	Indicators						Factor 1	Operation	Factor 2	Result Field		Resulting Indicators			Comments								
			Net	And	And	Not	Not	Not				Name	Length	Arithmetic	Plus	Minus		Zero							
												Compare													
												1 > 2 1 < 2 1 = 2													
												Lookup (Factor 2) is													
												High Low Equal													
												Half Adjust (H)													
0	1	C							'2S'	NEXT	ICFILE								90						
0	2	C								-OR-															
0	3	C							SSNID	NEXT	ICFILE								90						
0	4	C																							
0	5	C																							

RPG Cycle Input

The RPG program cycle includes a step to read a record from the primary file. If the primary file is a WORKSTN file, the input operation performed may be an accept input or a get operation. If a NEXT operation has been executed since the last RPG input cycle, the RPG input cycle performs an accept input operation. For information about how the RPG program cycle affects WORKSTN files, see "How WORKSTN Files Are Processed" in the manual *Programming with RPG II*.

RPG II Operations Summary Chart

The following chart shows the valid RPG II operations for each subsystem. An x in a subsystem column indicates that the subsystem supports the operation. A - in a column indicates that the subsystem does not support the operation.

RPG II SSP-ICF Operation	RPG II Operation	Communications Subsystems										
		Intra	BSC	CEL	CCP	CICS	IMS	3270 ¹	Finance	Peer	SNUF	APPC
Accept input ²	READ	x	x	x	x	x	x	x	x	x	x	x
Aquire	ACQ	x	x	x	x	x	x	x	x	x	x	x
Cancel	\$\$CANLNI	x	-	-	-	-	-	-	-	-	x	-
Cancel invite	\$\$CNLINV	x	x	-	-	-	-	-	-	-	-	-
Cancel then invite	\$\$CANL	x	-	-	-	-	-	-	-	-	x	-
End of session	\$\$EOS	x	x	x	x	x	x	x	x	x	x	x
Evoke	\$\$EVOKNI	x	x	x	x	x	x ⁴	-	x	x	x	x
Evoke end of transaction	\$\$EVOKET	x	x	-	x	x	-	-	x	x	x	x
Evoke then invite	\$\$EVOK	x	x	x	x	x	x ⁴	-	x	x	x	x
Fail	\$\$FAIL	x	-	-	-	-	-	-	x	-	-	x
Get ²	READ	x	x	x	x	x	x	x	x	x	x	x
Invite ³	\$\$SEND	x	x	x	x	x	x	x	x	x	x	x
Negative response	\$\$NRSPNI	x	-	-	-	-	-	-	-	-	x	-
Negative response then invite	\$\$NRSP	x	-	-	-	-	-	-	-	-	x	-
Put	\$\$SENDNI	x	x	x	x	x	-	x	x	x	x	x
Put end of file/chain	\$\$SENDE	x	x	x	x	-	x	x	x	x	x	-
Put end of transaction	\$\$SENDET	x	x	-	x	x	-	-	x	x	x	x
Put FMH	\$\$SENDNF	x	-	-	-	-	-	x	-	x	-	-
Put FMH then invite	\$\$SENDFM	x	-	-	-	-	-	x	-	x	-	-
Put then invite	\$\$SEND	x	x	x	x	x	x	x	x	x	x	x
Release	REL	x	x	x	x	x	x	x	x	x	x	x
Request to change direction then invite	\$\$RCD	x	x	x	-	-	-	-	x	x	x	x
Set timer	\$\$TIMER	x	x	x	x	x	x	x	x	x	x	x

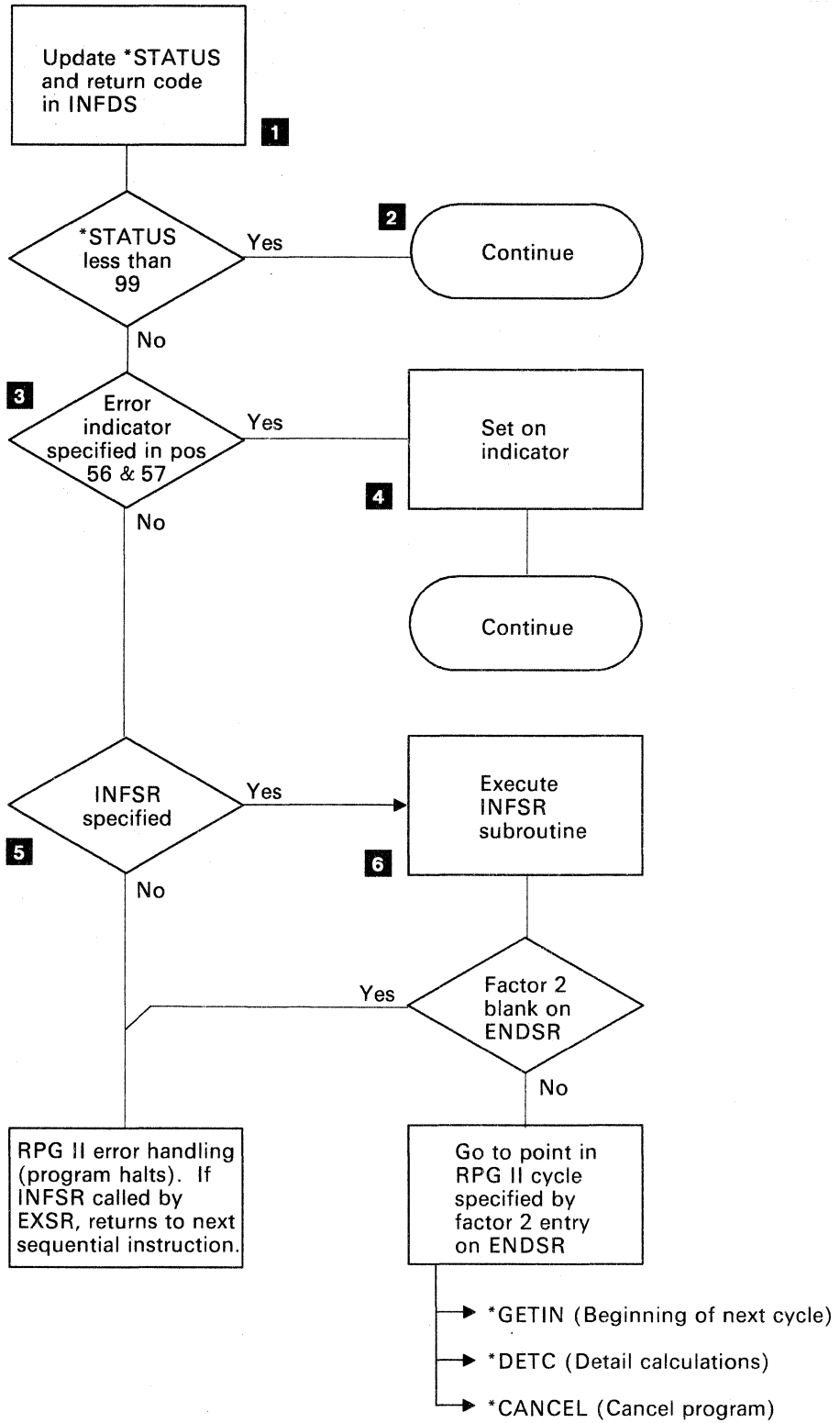
¹Although the BSC 3270 subsystem is not part of SSP-ICF, its operations are listed here to show its similarities to other subsystems.

²If a NEXT operation is executed before the READ operation, the READ operation is a get operation; otherwise, the operation is an accept input operation.

³Valid only when a \$\$SEND operation is issued with a record length of zero.

⁴Valid only when the DATAID and FLDLTH parameters are specified on the SESSION OCL statement, and when the HOSTNAME parameter on the SESSION statement is CICS or IMS.

The following chart and description show the steps for processing return codes.



S7910057-0

- 1** When an operation is completed, the status information (*STATUS and the return code, in most cases) is updated in the INFDS.
- 2** If the condition is normal, the next instruction in your program is processed.
- 3** If the condition is an exception or error (*STATUS equal to or greater than 99), a check is made to see whether an indicator was specified in positions 56 and 57 of the calculation specifications for a READ, ACQ, REL, or NEXT operation.
- 4** If an indicator was specified, that indicator is set on, and the next instruction in the program is processed. If the INFSR subroutine is to be run, you can issue an EXSR operation.
- 5** If an indicator was not specified, a check is made to see whether an INFSR was specified. If not or if factor 2 of the ENDSR is blank, RPG displays a message on the system console.
- 6** If INFSR was specified and factor 2 of the ENDSR operation contains an entry, control is passed to the point specified by factor 2 on the ENDSR operation. Factor 2 can be *GETIN to go to the beginning of the next input cycle, *DETC to perform detail calculations, *CANCL to cancel the program, or a variable that contains one of these values.

RPG II Status Values

The following shows the *STATUS values as returned in the RPG II INFDS for each major and minor return code. Use this list to determine the SSP-ICF return code or group of codes that corresponds to the *STATUS value. Then see the description of the major and minor return codes in the appropriate subsystem reference manual. All return codes that are valid for that subsystem are described in that reference manual. A summary chart in Appendix B shows which codes are valid for each subsystem.

Major Code	Minor Code	*STATUS Value
00, 01, 02	All (except 10)	00000
00, 02	10	01321
03	00	01311
03	01, 02, 03	01299
03	08	01275
03	10	01331
03	14	01311
03	15	01299
03	1C	01275
04	02, 11, 12	01299
08	00	01285
11	00	00011
28	00	00000
34	01	01201
34	31	01201
80, 81, 83	All	01251
82	All	01281

Note: RPG II performs additional error checking before passing a request to data management. If an error is found, the status value is updated, and the return code field remains unchanged.

RPG II Programming Considerations

The following topics describe items you should consider when you write an RPG II program for interactive communications. Information on these and other considerations for WORKSTN file programming is in the manual *Programming with RPG II*.

Using Continuation Options on the File Description Specifications

The following continuation options can be coded on the file description specifications for WORKSTN files:

- NUM
- SAVDS
- IND
- SLN
- ID
- INFSR
- INFDS
- FMST
- CFILE (for use with IDDU only)

NUM Continuation Option

The NUM continuation option specifies the maximum number of display stations and sessions that can be attached to the file at one time. This number should include the number of requesters as specified by the MRTMAX parameter plus the number of display stations and sessions that the program acquires at a time. The display stations and sessions specified by the MRTMAX parameter are reserved for requesters, and the remaining display stations and sessions can be acquired. For example, if the MRTMAX parameter value is 4 and the NUM value is 5, only one session can be acquired at a time. The number specified must be right-adjusted in positions 60 through 65. If no number is specified, 1 is assumed.

Note: Even if the program is an SRT program, a NUM value of 2 (or more) must be specified if the program also acquires display stations and sessions.

SAVDS Continuation Option

The SAVDS continuation option specifies the name of a data structure that can be saved and restored for each display station and each session in this file. This data structure cannot be a display station local data area, and it cannot contain a compile-time array or a pre-execution-time array.

Note: Only one copy of the data structure is available at a time. For example, if a program receives input from a session, only the data structure for the session is available; the data structure for the display station is not available. The only data structure available is that of the display station or session from which the last input came. Therefore, you should not use this data structure to save the identifier of a display station for which an interactive communications request has been made.

IND Continuation Option

The IND continuation option specifies the indicators associated with each display station and session that are to be saved and restored. The indicators numbered 01 through the number specified by the IND value are saved. The entry must be right-adjusted in positions 60 through 65.

Note: Only one copy of the indicators is available at a time. For example, if a program receives input from a session, only the indicators for the session are available; the indicators for the display station are not available. The only indicator available is that of the display station or session from which the last input came.

SLN Continuation Option

The SLN continuation option specifies the starting line number for display formats. The SLN option does not apply to sessions.

ID Continuation Option

The ID continuation option specifies the name of a 2-character field to contain the identifier of the current display station or session. Following input operations, the field contains the identifier of the display station or session from which the input was received. Any output operations are directed to the display station or session whose identifier is in the field. Thus, by changing the contents of the field, the output can be directed to any session or display station. A session identifier must be numeric-alphabetic (for example, 1S); a display station identifier must be alphabetic-alphanumeric (for example, W1).

INFSR Continuation Option

The INFSR continuation option specifies the name of a subroutine to be used for exception/error handling. "Return Code Processing in RPG II," earlier in this chapter, describes INFSR in more detail.

INFDS Continuation Option

The INFDS continuation option specifies the name of a data structure to contain information concerning exceptions and errors. "Return Code Processing in RPG II," earlier in this chapter, describes INFDS in more detail.

FMTS Continuation Option

The FMTS continuation option specifies the name of the display screen format load member containing the operations for this program. The name entered in this option overrides the name normally assumed by the RPG II compiler (the program name followed by FM). If the only operations used in the program are the interactive communications operations, *NONE must be specified for this option.

CFILE Continuation Option

The CFILE continuation option specifies the name of an IDDU file definition. This file definition contains the IDDU formats that may be used with an Intra or an APPC communications session.

SRT and MRT Program Considerations

An SRT (single requester terminal) program can have only one requesting display station or only one requesting session. SRT programs can acquire multiple sessions or display stations, using the ACQ operation. If an SRT program acquires any display stations or sessions, the NUM value on the file description specifications must reflect the maximum number of concurrently attached sessions and display stations (all those that are acquired plus one requester).

An MRT (multiple requester terminal) program can have multiple requesting display stations and/or sessions. The first requester of an MRT program causes the program to be loaded and initiated. Each succeeding requester attaches to the program at the beginning of an input cycle or when a READ operation is performed. The program is notified of the new requester via a return code on the input operation. MRT programs can also acquire additional display stations and sessions. The NUM value on the file description specifications must include the maximum number of requesters plus the number of sessions and display stations that are acquired and that are active concurrently.

End-of-File Considerations

The effects of end of file on the program depend on whether the file is a demand file or a primary file.

End of file for a demand or primary file occurs only on an input operation (not preceded by a NEXT operation) and only when no display stations or sessions have been requested for input; that is, there are no outstanding invite operations. (This second condition could occur because no invite operations were issued or because all display stations and sessions have been released.) If the program is a never-ending program (NEP), both end-of-file conditions must exist and the system operator must have entered the STOP SYSTEM command for the end-of-file condition to occur.

For a primary WORKSTN file, an end-of-file condition sets on the LR indicator, and the program goes to end of job. For a demand WORKSTN file, an end-of-file condition sets on the indicator in positions 58 and 59 of the READ operation that detected the end of file. This indicator can be the LR indicator, or the program can set on the LR indicator later. If no indicator is specified, RPG issues an error message indicating that the end of the file has been reached.

Release Considerations

You can specify a release operation by using the REL operation (described earlier in this chapter) or by coding an R in position 16 of the output specifications. If a format name is specified in the same output specification that contains an R in position 16, the format is displayed or the interactive communications operation is performed before the display station or session is released. When a program ends, display stations and sessions are automatically released.

If a session was acquired, the release operation terminates that session. If a display station was acquired, the release places the display station in standby mode.

If the session was started by a remote procedure start request or if a display station requests the program, the release operation passes the session or display station on to the next step in the procedure. If the program is an MRT program, the session or display station is released immediately. If the program is an SRT program, the session or display is released when the program terminates. If the program is the last step in the job, the display station returns to the command display, or the session is terminated when the program ends.

Restrictions for WORKSTN Files

The following restrictions apply to using a WORKSTN file in an RPG II interactive communications program:

- The WORKSTN file must be specified as a combined file (capable of both input and output).
- If the WORKSTN file is specified as a primary file, no secondary files are allowed in the program.
- Only one WORKSTN file is allowed in a program.
- A program cannot contain a KEYBOARD, CRT, or CONSOLE file if it contains a WORKSTN file.
- Control level indicators, match field values, and look-ahead fields are not allowed.
- The first page indicator (1P) is not allowed.

Input and Output Considerations

Considerations for when output can be sent and what input operations are required depend on whether the communication is with a display station or session that is acquired or is a requester.

When a requester (either a display station or a session) attaches to a program, the first operation is an input operation. The input operation fills in the ID field, which is used to direct subsequent operations to the appropriate session or display station. If data accompanied the request, the data is passed to the program on this first input operation; if no data accompanied the request, a blank record is passed to the program. If the program is an SRT program, output to the requester may precede input; however, if output precedes input, data sent with the request is lost. Output can precede input if the requester's ID or blanks are specified in the ID field and output is performed as the first operation to the file.

When a session or display station is acquired, the next input operation retrieves a blank record. If an output operation (any put or evoke with data) is performed in the same cycle as the acquire operation, the next input operation retrieves a data record.

RPG II Coding Examples

For a complete example of an RPG II communications program, see “Writing an RPG II Program That Uses the Intra Subsystem” in Chapter 6 of the *SSP-ICF Guide and Examples* manual. The same programming example described in the Intra chapter is also used in for the other subsystems, but only the changed areas needed to allow communications with that type of remote system are shown.

Chapter 6. The Intra Subsystem

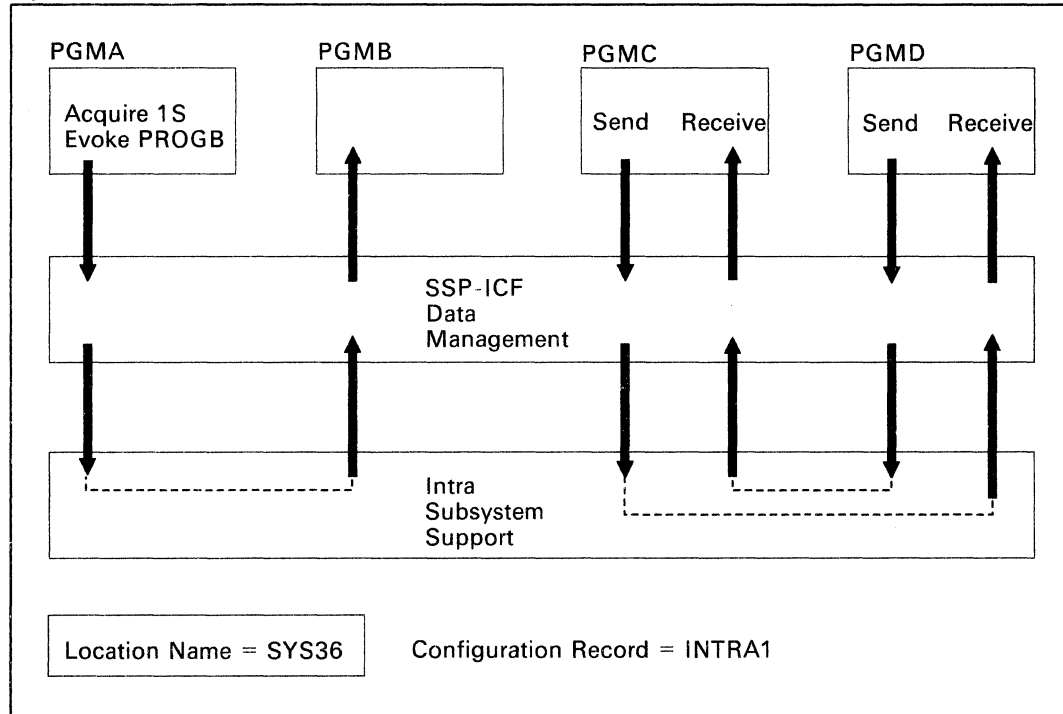
Overview of the Intra Subsystem	6-3
Setting Up an Intra Subsystem	6-5
CNFIGICF Procedure	6-5
Explanation of Displays	6-6
Subsystem Member Definition	6-7
Display 1.0 SSP-ICF Configuration Member Definition	6-7
Display 2.0 SSP-ICF Configuration Member Type	6-8
Display 22.0 Subsystem Member Definition	6-9
Modifying a Subsystem Configuration	6-10
Enabling and Disabling the Intra Subsystem	6-10
Starting Communications Sessions That Use the Intra Subsystem	6-11
SESSION Statement	6-12
Procedure Start Requests	6-13
Communications Operations for the Intra Subsystem	6-14
Accept Input Operation	6-17
Acquire Operation	6-18
Acquire Operation Examples	6-18
Assembler	6-18
BASIC (Normal Acquire)	6-18
BASIC (Special Acquire)	6-19
COBOL	6-19
RPG II	6-19
Cancel Operations	6-20
SNUF-Related Cancel Operations	6-20
BSCEL-Related Cancel Invite Operation	6-21
End of Session Operation	6-22
Ending Sessions Started by an Evoke Operation from Another Program	6-22
Evoke Operations	6-23
Assembler Evoke Operation (Macroinstructions)	6-25
\$WSIO Macro	6-25
Example of \$WSIO Macro	6-25
\$EVOK Macro	6-26
Example of \$EVOK Macro	6-26
BASIC Evoke Operation Parameters	6-27
IDDU Format Considerations	6-27
BASIC Example (Evoke Operation)	6-28
COBOL Evoke Operation Parameters	6-29
IDDU Format Considerations	6-29
COBOL Example (Evoke Operation)	6-30
RPG II Evoke Operation Parameters	6-31
IDDU Format Considerations	6-31
RPG II Example (Evoke Operation)	6-32
Fail Operation	6-33

Get Operations	6-35
Invite Operation	6-37
Negative Response Operations	6-38
Put Operations	6-39
Release Operation	6-41
Request to Change Direction Operations	6-42
Set Timer Operation	6-44
Intra Subsystem Return Codes	6-45

Overview of the Intra Subsystem

The Intra subsystem allows interactive communications between two application programs in the same System/36. Multiple pairs of application programs can communicate concurrently in the same subsystem.

System/36



S7910003-0

The Intra subsystem can also be used in the following ways:

- It can be used in a limited way to test new communications programs that are to be run with any of the other subsystem types, such as BSCEL or Peer. You can test a program in the Intra subsystem to help debug the program before you attempt to use it to communicate with its intended remote system over a communications line.

During testing, you can check only the return codes returned by the Intra subsystem. After testing, if you want to check other return codes not supported by the Intra subsystem, you can add the necessary coding changes before actually running the program with the intended subsystem.

- It can be used to help train programmers in writing SSP-ICF programs. Complete communications programs (written in assembler, BASIC, COBOL, and RPG II) that use the Intra subsystem are shown and described in the *SSP-ICF Guide and Examples* manual.
- It can be used with one program to control access to a critical file. All programs attempting to access the file would have to communicate with that program via the Intra subsystem before the file could be accessed. Security can be in effect on System/36, and journaling could be done by the program performing the actual input or output operations to the file.
- It can be used with externally described field, format, and file definitions (also called data definitions) to send data records. Data definitions, which describe data records and communications functions, are defined separately from the application program. The interactive data definition utility (IDDU) is used to create data definitions.

For more information about using IDDU, refer to the manual *Getting Started with Interactive Data Definition Utility*. For more information about IDDU and communications files, refer to the online information for IDDU.

Setting Up an Intra Subsystem

Before an Intra subsystem can be used for communications, the SSP-ICF support, the Intra subsystem support, and configuration members must be put on System/36. This is accomplished with the following procedures:

1. The CNFIGSSP (SSP configuration) procedure must be used to copy (install) the SSP-ICF support and Intra subsystem support from diskette to the system. (See the manual *Changing Your System Configuration* for the description of the CNFIGSSP procedure.)
2. The CNFIGICF (SSP-ICF configuration) procedure must be used to create a particular Intra subsystem configuration member. The name of the created subsystem configuration member is then used when that subsystem is started by the ENABLE procedure command.

Multiple Intra configuration members can be created and stored in the system, and more than one Intra subsystem can be active and in use at the same time. (The name of each member must be unique within a library.) Subsystems of other types can also be active.

CNFIGICF Procedure

This section describes the displays and parameters (shown in prompt form) needed to define and create an Intra subsystem configuration. To help you define the attributes of a particular Intra subsystem member, use the prompting facilities in the CNFIGICF procedure to specify the parameter values that define the subsystem attributes and create the subsystem member.

A general description of the configuration process is contained in the *SSP-ICF Guide and Examples* manual.

Explanation of Displays

On the following displays for an Intra configuration member:

- All the prompts that *can be displayed* for the Intra subsystem are shown on the displays and are described in the text. The prompts are shown for all the parameters that are needed either to create a new Intra configuration member or to change (edit), delete, or review an existing member.

*Note: The prompt lines that are **actually displayed** on succeeding displays depend on the task specified on display 1.0 and on the options that you select on other displays. Prompt lines not shown do not apply to the task or options previously selected.*

- For this set of example displays only, the values to the right of the prompts are shown with:
 - Default values, supplied by the system. If the system provides a default value, that value is shown here. (You can enter a different value if you wish.)
 - Sample values, as typical examples. If fewer characters are shown than the field allows, the remaining positions in the field are underscored. Note that once a value has been entered in a field, it becomes the default value for any related fields on the succeeding displays.
 - Underscored fields represent fields in which a value can be specified or a default value (if any) is assumed when a value is not specified. The length of the underscore represents the length of the field.
 - Fields filled with asterisks (*) indicate that the field contains a value that the CNFIGICF procedure duplicates from a value that was entered on a previous display.
 - Fields with values in them indicate that the value shown is the only value that can be specified for this subsystem.

Subsystem Member Definition

Display 1.0 SSP-ICF Configuration Member Definition

On display 1.0, specify the name of the subsystem configuration member you are creating or using in some way, and specify what is to be done with the member.

```
1.0                SSP-ICF CONFIGURATION MEMBER DEFINITION

1. Configuration member name . . . . . INTRA__
2. Library name . . . . . #LIBRARY
3. Select one of the following:
   1. Create new member
   2. Edit existing member
   3. Create new member from existing member
   4. Remove a member
   5. Review a member
   Option . . . . . 1-5 _
4. Existing member name . . . . . _____
5. Existing member library name . . . . . #LIBRARY

Cmd7-End      Cmd19-Cancel
```

- 1. Configuration member name:** Enter the name that identifies this configuration of the subsystem. This name is used to store the subsystem configuration member in a library, and it is also used in the ENABLE and DISABLE procedure commands to start and stop the subsystem.
- 2. Library name:** Enter the name of the user library in which the configuration member is to be stored. The default is the library that you are currently using.
- 3. Select one of the following:** Select one of the five options. For example, if you want to modify an existing member then store the modified member as a new member (without changing the existing member), select option 3.
- 4. Existing member name:** This prompt is displayed only if you select option 3 for prompt 3. Enter the name of the existing subsystem configuration member that is to be used to create the new member. (The existing member is not changed.)
- 5. Existing member library name:** This prompt is displayed only if you selected option 3 for prompt 3. Enter the name of the library that contains the existing member. The default is the library name specified for prompt 2.

Display 2.0 SSP-ICF Configuration Member Type

On display 2.0, specify the type of member you want to define or redefine.

```
2.0                      SSP-ICF CONFIGURATION MEMBER TYPE          INTRA

Select one of the following options:
  1. INTRA
  2. BSC
  3. SNA
  4. Async
  5. PC Support/36

Option: 1

Cmd3-Previous display      Cmd5-Restart CNFIGICF
Cmd7-End                   Cmd19-Cancel          COPR IBM Corp. 1986
```

Select one of the following options: Because the Intra subsystem member is being defined, enter a 1. The Intra subsystem does not require a line member to be defined.

Display 22.0 Subsystem Member Definition

On display 22.0, specify the remote location name with which your program is to communicate. For the Intra subsystem, the location is within your System/36; therefore, you can use the other program name as the location name.

```
22.0                SUBSYSTEM MEMBER DEFINITION                INTRA

1. Remote location name . . . . . INTRA

Cmd5-Restart CNFIGICF                Cmd7-End
Cmd19-Cancel                          COPR IBM Corp. 1986
```

- 1. Remote location name:** Specifies the name of the remote location with which your program will be communicating. Enter a name of no more than 8 characters. This name is used to match a **SESSION** statement with a subsystem configuration; therefore, to use this configuration, you must also enter this name in the **LOCATION** parameter of a **SESSION** statement. See “**SESSION Statement**” later in this chapter for a description of the **SESSION** statement. If you do not enter a location name, the subsystem configuration member name is also used as the location name.

This location name also appears in system messages to help operators identify the particular subsystem configuration.

Modifying a Subsystem Configuration

To change one or more of the attributes defined in a subsystem member of an existing Intra subsystem configuration, you can use the CNFIGICF procedure to change (edit) the member. (For the changed attributes to take effect, the subsystem using the configuration must be disabled and enabled again.) After the CNFIGICF procedure is completed, the updated member definition is used each time any subsystem associated with the changed member is enabled.

Enabling and Disabling the Intra Subsystem

The ENABLE and DISABLE procedure commands are used to start and end the Intra subsystem on System/36. The **ENABLE procedure command** associates the subsystem support with a particular subsystem configuration member.

The **DISABLE procedure command** ends (disables) a specified subsystem. When the DISABLE procedure command is performed, the disabled subsystem cannot handle program communications requests because it no longer exists in main storage.

See the section “Enabling and Disabling Subsystems” in Chapter 1 for a description of the ENABLE and DISABLE procedures.

Starting Communications Sessions That Use the Intra Subsystem

System/36 communications sessions using the Intra subsystem can be started in one of the following ways:

- Your program can issue an **acquire operation** to start (acquire) the session. The acquire operation identifies the session to be started and must match the session identifier specified in an associated SESSION statement, when one is used.
- Another program in System/36 can start your program indirectly with a **procedure start request** (actually, an evoke operation). The request in the Intra subsystem is made when the program issues an evoke operation that identifies a procedure which then starts your program. (The evoke operation does in the Intra subsystem what a procedure start request does in other types of subsystems when the request is sent by a remote system to System/36.)

The following sections describe the SESSION statement and procedure start requests for the Intra subsystem.

SESSION Statement

Each program (except BASIC programs) that is to acquire a session must have at least one SESSION statement included in the procedure that loads the program. The SESSION statement must be placed between the LOAD and RUN OCL statements used for the program. The SESSION statement specifies three things:

- It identifies, on the SYMID parameter, the session to be acquired later in the program.
- It identifies, on the LOCATION parameter, the location with which the program is to communicate (in this case, another program in System/36). Before the SESSION statement is executed, the Intra subsystem associated with the specified location must have already been started. (The location name was specified in the subsystem configuration member.)
- It specifies, on the BATCH parameter, whether the session is to be used for batch processing. If BATCH-NO is specified or assumed, several of the communications operations (for example, the cancel, negative response, and request to change direction operations) are *not* valid for an Intra-supported (interactive) session. If BATCH-YES is specified, CONFIRM is ignored on an end of transaction operation.

The SESSION statement, then, identifies the session and location with which your program is to communicate, and indirectly identifies the subsystem that has the necessary attributes needed to communicate in the session.

Note: A BASIC program requires a SESSION statement if interactive session operations are performed.

The syntax of the SESSION statement for the Intra subsystem is:

```
// SESSION LOCATION-name,SYMID-session id [ ,BATCH- $\left\{ \begin{array}{l} \text{NO} \\ \text{YES} \end{array} \right\} ]$ 
```

S9020325-0

LOCATION Parameter: Specifies the location name to be associated with this session. The location name, specified on display 22.0 during subsystem configuration, refers to the location with which your program is to communicate. (If the location name parameter was not specified during subsystem configuration, the location name was assumed to be the same as the subsystem configuration member name. The subsystem configuration member name is the name that was specified on the ENABLE procedure command to enable the subsystem now being used for this session and location.) This parameter has no default.

SYMID Parameter: Specifies the symbolic identifier of the session with which this SESSION statement is associated. Your program uses this identifier when it acquires the session and whenever it issues any operation in the session. The identifier must be 2 characters: the first character must be numeric (0 through 9), and the second character must be alphabetic (A through Z, \$, #, or @). This parameter has no default.

BATCH Parameter: Specifies whether batch-oriented operations (request to change direction, negative response, cancel, and function management header operations) can be issued for this session. YES indicates that they can be issued; NO indicates that they cannot and is the default.

Procedure Start Requests

To initiate another procedure on System/36, your program must issue an evoke operation. The subsystem starts (evokes) the specified System/36 procedure, which then starts a program with which your program can communicate. For a description of the four types of evoke operations that can be used to start another program, see “Evoke Operations” later in this chapter.

Communications Operations for the Intra Subsystem

This section describes all the input and output operations that can be coded in a program that is to communicate, using the Intra subsystem, with another program in System/36.

For complete details about how many of these operations work and how to use them, see the *SSP-ICF Guide and Examples* manual. Also, for complete coding examples of two Intra communications programs in assembler, BASIC, COBOL, and RPG II, refer to the examples given in the Intra subsystem chapter in the *SSP-ICF Guide and Examples* manual.

The following summary chart presents *all* the Intra subsystem operations and their operation codes in all four languages. Then, in the topics that follow, each operation or group of related operations is described. Each operation is described, its operation codes in all four languages are shown in a smaller chart, and coding examples (if appropriate) are given.

Note: In the operation charts in this section, the codes that are valid for each operation are listed to the right of the operation in their respective language columns. If an operation is not valid in one or more languages, dashes (—) are shown in the columns instead.

Intra Subsystem Operations	Language Operation			
	Assembler	BASIC	COBOL	RPG II
Accept input	ACI	WAITIO and READ ¹	READ ²	READ ³
Acquire	ACQ	OPEN	ACQUIRE	ACQ
Cancel ⁴	CAN	\$\$CANLNI	\$\$CANLNI	\$\$CANLNI
Cancel invite	CNI	\$\$CNLINV	\$\$CNLINV	\$\$CNLINV
Cancel then get ⁴	CANG	—	—	—
Cancel then invite ⁴	CANI	\$\$CANL	\$\$CANL	\$\$CANL
End of session	EOS	\$\$EOS	\$\$EOS	\$\$EOS
Evoke ¹⁰	EVK ⁵	\$\$EVOKNI	\$\$EVOKNI	\$\$EVOKNI
Evoke end of transaction ¹⁰	EVE ⁵	\$\$EVOKET	\$\$EVOKET	\$\$EVOKET
Evoke then get ¹⁰	EVG ⁵	—	—	—
Evoke then invite ¹⁰	EVI ⁵	\$\$EVOK	\$\$EVOK	\$\$EVOK
Fail	FAIL	\$\$FAIL	\$\$FAIL	\$\$FAIL
Get ¹⁰	GET	READ	READ ⁶	NEXT and READ ⁷
Get attributes	GTA	ATTRIBUTE\$	ACCEPT	—
Get status ⁹	GST	ATTRIBUTE\$	ACCEPT	—
Invite ¹⁰	INV	\$\$SEND ⁸	\$\$SEND ⁸	\$\$SEND ⁸
Negative response ⁴	NRP	\$\$NRSPNI	\$\$NRSPNI	\$\$NRSPNI
Negative response then get ⁴	NRPG	—	—	—
Negative response then invite ⁴	NRPI	\$\$NRSP	\$\$NRSP	\$\$NRSP
Put ¹⁰	PUT	\$\$SENDNI	\$\$SENDNI	\$\$SENDNI
Put end of chain	PEC	\$\$SENDE	\$\$SENDE	\$\$SENDE
Put end of transaction ¹⁰	PEX	\$\$SENDET	\$\$SENDET	\$\$SENDET
Put then get ¹⁰	PTG	—	—	—
Put then invite ¹⁰	PTI	\$\$SEND ⁸	\$\$SEND ⁸	\$\$SEND ⁸
Put FMH	PFM	\$\$SENDNF	\$\$SENDNF	\$\$SENDNF
Put FMH then get	PFMG	—	—	—
Put FMH then invite	PFMI	\$\$SENDFM	\$\$SENDFM	\$\$SENDFM
Release	REL	CLOSE	DROP	REL
Request to change direction then get ⁴	RCDG	—	—	—
Request to change direction then invite ⁴	RCDI	\$\$RCD	\$\$RCD	\$\$RCD
Set timer	STM	\$\$TIMER	\$\$TIMER	\$\$TIMER

¹In BASIC, an accept input operation is performed only if the WAITIO operation is followed by a READ operation.
²In COBOL, an accept input operation is performed only if the TERMINAL option of the READ statement is not specified or is specified with blanks.
³In RPG II, an accept input operation is performed only if the READ operation is *not* preceded by a NEXT operation.
⁴This operation is valid only in *batch* sessions, when BATCH-YES is specified on the SESSION statement.
⁵In assembler, a function management header can be sent with this operation by specifying OPM-FMH on the \$WSIO macro.
⁶In COBOL, a get operation is performed only if the TERMINAL option of the READ statement is specified with nonblanks.
⁷In RPG II, a get operation is performed only if a NEXT operation is executed before the READ operation.
⁸In BASIC, COBOL, or RPG II, only an invite operation is performed if \$\$SEND is issued with a record length of 0 bytes. Otherwise, \$\$SEND performs a *put then invite* operation.
⁹The record area must be at least 128 bytes long.
¹⁰In assembler, a confirm indicator can be sent with this operation by specifying OPM-CONFIRM on the \$WSIO macro. If OPM is not CONFIRM it must be ZERO. In other languages, a confirm indicator is specified in an IDDU format definition.

The following chart presents the Intra subsystem operations and their related functions in IDDU.

Intra Subsystem Operations	IDDU Keywords ¹	IDDU Functions
Evoke Evoke end of transaction	EVOKE EVOKE and DETACH	Evoke process Evoke process and send detach
Evoke then invite	EVOKE and INVITE	Evoke process and invite
Fail	FAIL	Send fail
Invite	INVITE	Invite
Put	Field ²	Put
Put end of transaction	DETACH ³	Send detach
Put then invite	Field and INVITE ²	Put and invite
Request to change direction then invite	RQSWRT and INVITE	Send request to write and invite
¹ A confirm indicator can be specified in addition to each function (except RQSWRT and INVITE, and FAIL). A confirm indicator is specified in an IDDU format definition. ² Data for these operations is specified in a separate user-defined field. ³ In addition, a user-defined field can be specified with this operation.		

In the topics that follow, each operation or group of related operations is described; its operation codes in all four languages are shown, corresponding IDDU keywords (if appropriate) are shown, and coding examples (if appropriate) are given.

For a description of how IDDU processes functions, refer to Appendix A, "Processing IDDU Functions," in the *SSP-ICF Base Subsystems Reference* manual.

Accept Input Operation

Your program can use the **accept input** operation to perform three different functions. You can use it to:

- Obtain data from any program or any display station that has responded to an invite operation that was previously issued in your program. If data becomes available to your program from more than one program or display station before the accept input operation is issued, your program receives the data that was *first* made available, whether it was from another program (via the Intra subsystem) or from a display station.
- Check whether the timer that was set by the set timer operation has expired. (For a description of the set timer operation, see “Set Timer Operation” later in this chapter.)
- Wait for a new requester.
 - If your program was evoked, it should issue an accept input operation as its first operation to determine the identifier of the new requester. (Your program is notified of the new requester by the resulting 01xx return code at the end of the accept input operation.)
 - If your program is an MRT NEP program and no previous invite operation or set timer operation is in effect, it should issue an accept input operation so it can wait for a new requester.

Except for the first accept input operation in evoked programs or in MRT NEP programs, all accept input operations in all programs should be issued to receive data only after an invite operation is issued by itself or in combination with another operation, or after a set timer operation is issued.

Operation	Assembler	BASIC	COBOL	RPG II
Accept input	ACI	WAITIO and READ ¹	READ ²	READ ³
¹ In BASIC, an accept input operation is performed only if the WAITIO operation is followed by a READ operation. ² In COBOL, an accept input operation is performed only if the TERMINAL option of the READ statement is not specified or is specified with blanks. ³ In RPG II, an accept input operation is performed only if the READ operation is <i>not</i> preceded by a NEXT operation.				

Acquire Operation

Your program uses the **acquire** operation to establish a session between your program and the Intra subsystem in System/36. The session being established is identified in the acquire operation statement, and its identifier must match the session identifier given in the SYMID parameter of your program's SESSION statement for this session.

The session started by the acquire operation is initialized with the parameters specified in the SESSION statement. (For the Intra subsystem, there are no parameters on the SESSION statement that override any parameters defined during subsystem configuration.)

Note: In BASIC, a SESSION statement is not needed if a special acquire operation is performed. In this case, the location name is specified in the LOC parameter of the OPEN statement to indicate which location is to communicate with this session.

Operation	Assembler	BASIC	COBOL	RPG II
Acquire	ACQ	OPEN	ACQUIRE	ACQ

Acquire Operation Examples

Assembler:

```
$WSIO DTF-ICDTF2,TERMID-2S,OPC-ACQ
```

This \$WSIO macro is used to acquire the session identified as 2S. The DTF to be used for sending or receiving data is identified as ICDTF2. (For a complete description of the \$WSIO macro's communications parameters, see "\$WSIO Macro" in Chapter 2.) A SESSION statement that specifies SYMID-2S is required.

BASIC (Normal Acquire):

```
OPEN #1: "SESSION,ID=1S,RECL=255" IOERR ICFERR
```

This OPEN statement opens interactive communications file #1 and acquires the session identified as 1S. The maximum record length that can be sent or received is 255 bytes. If the acquire operation is not successful, the program branches to the statement labeled ICFERR. A SESSION statement that specifies SYMID-1S is required.

BASIC (Special Acquire):

OPEN #1: "SESSION,LOC=CHICAGO,RECL=255" IOERR ICFERR

This OPEN statement opens interactive communications file #1 and acquires a session with the remote location identified as CHICAGO. No SESSION statement is used. For this acquire operation to be successfully performed, a subsystem configuration specifying the location name CHICAGO must already be enabled.

COBOL:

ACQUIRE COMM-SESSION FOR COMMUNICATIONS-FILE.

This ACQUIRE statement acquires the session that has the same session identifier as the value in the COMM-SESSION field. The COMM-SESSION field must be defined as a 2-character field with a valid session identifier (such as PIC XX VALUE, '1S'). The session is acquired for the TRANSACTION file named COMMUNICATIONS-FILE, which has been opened as I-O. A SESSION statement is required.

RPG II:

Field:	Factor 1	Operation	Factor 2	Indicator
Positions:	18 - 27	28 - 32	33 - 42	56 - 57
Value:	'1S'	ACQ	ICFILE	90

This ACQ operation acquires the session specified by the identifier '1S' in factor 1 of the calculation specifications. Factor 2 specifies the name of the WORKSTN file from the file description specifications. A SESSION statement that specifies SYMID-1S is required.

Cancel Operations

There are two types of cancel operations: those that cancel a *group* of records that has just been sent (used by the Intra subsystem, particularly when simulating a SNUF subsystem environment), and the *cancel invite* operation, which cancels an invite operation for which no input has yet been received (used by the Intra subsystem, particularly when simulating a BSC/CEL subsystem environment).

SNUF-Related Cancel Operations

Your program can use a SNUF subsystem-related cancel operation to cancel the current chain of data (group of records) that it is sending to the other program. The cancel operation causes a return code to be returned to the other program, which is in receive state. The return code indicates to the receiving program that the sending of the current chain is being terminated abnormally (possibly because your program detected an error in the data). The receiving program should disregard all the records in the current chain that have been sent (that is, all records sent since the previous end of chain indication).

The SNUF-related cancel operation is valid only when three conditions exist: the operation must be issued only in a batch session, *within* a chain of records, and while your program is in send state. The operation does not end the session.

The following are the three types of SNUF-related cancel operations.

- **Cancel:** Cancels the current chain of data.
- **Cancel then get** (assembler only): Cancels the current chain of data being sent, and then waits for the other program to send its own data.
- **Cancel then invite:** Cancels the current chain of data being sent, invites the other program to send its own data, and then regains control without having to wait for the invited input to be received. (An accept input or get operation must be issued later to receive the invited input.)

The cancel and negative response operations can be considered as a pair. Cancel is the appropriate response when a negative response indication is received. However, if the sending program detects an error, cancel can be sent without waiting to receive a negative response indication.

Operation	Assembler	BASIC	COBOL	RPG II
Cancel ¹	CAN	\$\$CANLNI	\$\$CANLNI	\$\$CANLNI
Cancel then get ¹	CANG	—	—	—
Cancel then invite ¹	CANI	\$\$CANL	\$\$CANL	\$\$CANL

¹This operation is valid only in *batch* sessions, when BATCH=YES is specified on the SESSION statement.

BSCCEL-Related Cancel Invite Operation

Your program can use a BSCCEL subsystem-related *cancel invite* operation to cancel any valid invite operation for which no input has yet been received from any invited session. (The cancel invite operation is the only valid cancel operation for the BSCCEL subsystem.)

When used by the Intra subsystem, the BSCCEL-related cancel invite operation is valid only when it is issued after any valid invite operation. Normally, no data will be in the subsystem's input buffer when the cancel invite operation is issued. If data *is* in the input buffer, the operation fails and return code 0412 is received by the program. Your program must issue an input operation to receive the data.

Operation	Assembler	BASIC	COBOL	RPG II
Cancel invite	CNI	\$\$CNLINV	\$\$CNLINV	\$\$CNLINV

Note: When this operation is used with a BSCCEL subsystem, there are some restrictions about which operations this operation can follow and there are some switched line restrictions. Refer to "Cancel Invite Operation" in Chapter 2 of the SSP-ICF Base Subsystems Reference manual for that information.

End of Session Operation

Your program uses the **end of session** operation to terminate a session. Unlike the release operation, the end of session operation always terminates the session (if it still exists), and it *always* gives a normal completion return code. However, if the operation is issued during an active transaction, both the transaction and the session are terminated abnormally by the Intra subsystem, and the other program may also be terminated abnormally. For example, your program could issue the end of session operation after an error has occurred on one of its previous operations; it may be an error from which your program cannot easily recover.

Ending Sessions Started by an Evoke Operation from Another Program

The end of session operation can be issued in a session that was started by an evoke operation issued by another program in System/36. In this case, your program should issue the end of session operation after it receives an end of transaction indication. The end of session operation frees that session so that it can be started again by another program.

If your program does not issue an end of session operation, the session exists until your program (or multiple-program procedure) terminates. To prevent your program from terminating abnormally because of a communications error, you may want to code the end of session operation in your program as a general recovery action for all unexpected errors that you have not handled individually in your program. The end of session operation could be used to terminate the session rather than retrying the failing operation in that session or specifying some special recovery action for each error.

Operation	Assembler	BASIC	COBOL	RPG II
End of session	EOS	\$\$EOS	\$\$EOS	\$\$EOS

Note: If an RPG II program is started with an evoke end of transaction operation, the program should issue an end of session or release operation to free the session ID entry in the internal table of identifiers for the WORKSTN file.

Evoked Operations

The evoke operation starts a procedure (and a transaction) on System/36. The procedure then starts a program that will handle the transaction. You can issue an evoke operation in your program only after a session has been acquired. Multiple evoke operations can be issued in an Intra session. (However, only one transaction at a time can be active; the previous transaction must have ended before the next evoke operation can be issued.)

The evoke operation must include an **evoke parameter list**, and can optionally include either **procedure parameters** for the procedure being started or user-supplied **data** for one of the programs started by the procedure. The parameters specified in the evoke parameter list (including the name of the procedure being started) are described for each language later in the topic.

The following types of evoke operations can be used in an Intra session to start another procedure on System/36.

- **Evoke:** Evokes the specified procedure, sends data to the subsystem (if specified by the user), and then waits until that procedure has been started before control is returned to your program.
- **Evoke end of transaction:** Evokes the specified procedure, sends any data specified by the user to one of the programs started by that procedure, and then ends the transaction without allowing the program to communicate in return.
- **Evoke then get (assembler only):** Evokes the specified procedure, sends any data specified by the user, and then waits for input to be received from one of the programs started by the procedure.
- **Evoke then invite:** Evokes the specified procedure, sends any data specified by the user, invites one of the programs started by that procedure to send data, and regains control without having to wait for the evoke operation to be completed or for the invited data to be received. (An accept input or get operation must be issued later in this transaction to receive the data in your program's input buffer.)

Operation	Assembler	BASIC	COBOL	RPG II
Evoke	EVK ^{1,2}	\$\$EVOKNI	\$\$EVOKNI	\$\$EVOKNI
Evoke end of transaction	EVE ^{1,2}	\$\$EVOKET	\$\$EVOKET	\$\$EVOKET
Evoke then get	EVG ^{1,2}	—	—	—
Evoke then invite	EVI ^{1,2}	\$\$EVOK	\$\$EVOK	\$\$EVOK

¹In assembler, a function management header can be sent with this operation by specifying OPM-FMH on the \$WSIO macro.
²In assembler, a confirm indicator can be sent with this operation by specifying OPM-CONFIRM on the \$WSIO macro. If OPM is not CONFIRM, it must be ZERO.

The evoke parameter list associated with each type of evoke operation contains the name of the procedure to be started, the name of the library in which the procedure is located, and the password and user identifier associated with that procedure. (The password and user identifier are needed only if security is being used on System/36.) The evoke operation can optionally include either (but not both) parameters to be sent to the evoked procedure or data to be passed to one of the programs started by the procedure.

A function management header can be included with the data sent on an evoke operation. If function management headers are included in the data passed in a session, BATCH-YES must be specified on the SESSION statement associated with that session. (For a description of function management headers, see “Function Management Headers (CICS and IMS)” in Chapters 1 and 2 of the *SSP-ICF Upline Subsystems Reference* manual.)

If you are using assembler, a confirm indicator can also be included with the data sent on an evoke operation. If the confirm indicator is to be included, OPM-CONFIRM must be specified. If OPM is not CONFIRM, it must be ZERO. In other languages, a confirm indicator is specified in an IDDU format definition.

The total length of the procedure name and data (or procedure parameters) specified in the program to be sent to the subsystem cannot exceed 508 bytes. (This does not include the other three evoke list parameters, each of which can be 8 bytes long.)

The evoke, evoke end of transaction, and evoke then invite operations can also be used with externally described field, format, and file definitions (also called data definitions). The interactive data definition utility (IDDU) is used to create data definitions. The following IDDU keywords are used for each language (in place of the operation codes):

Operation	IDDU Keywords
Evoke	EVOKE ¹
Evoke end of transaction	EVOKE and DETACH ¹
Evoke then invite	EVOKE and INVITE ¹
¹ A confirm indicator can be specified in addition to each function. A confirm indicator is specified in an IDDU format definition.	

Refer to the manual *Getting Started with Interactive Data Definition Utility* for more information about IDDU and creating data definitions.

Notes:

- 1. If you are using IDDU, information associated with the EVOKE can be passed in a user-defined field or separately, in the following order: procedure name, name of the library in which the procedure is located, user ID and password associated with the procedure, and user data (if any). When user data is passed, the result is an evoke operation followed by a put operation and any additional keywords (for example, CONFIRM, DETACH, or INVITE).*
- 2. If you are using IDDU to send an evoke parameter list that contains procedure parameters of which some may not be specified, you should define an IDDU format that specifies only one parameter. The length of the parameter should be defined as the total length of all the passed parameter values (including the commas that are used to separate the values or to indicate any unspecified parameters).*
- 3. If you are using IDDU to create a format definition, you should also use IDDU to print it. Use the resulting format definition listing to determine the order and starting positions of the parameters in the evoke parameter list and, optionally, the starting position of the user-supplied data or procedure parameters that follow.*

Assembler Evoke Operation (Macroinstructions)

\$WSIO Macro: To perform an evoke operation in assembler, use the \$WSIO macro. You specify the desired evoke **operation code** (EVK, EVE, EVG, or EVI) in the OPC parameter of the macro (for example, OPC-EVK). You must use another macro, \$EVOK, to specify the evoke **parameters** needed to perform the evoke operation specified on the \$WSIO macro. The function management header modifier (specified as OPM-FMH in the \$WSIO macro) is valid only if BATCH-YES was specified on the SESSION statement of the program that acquired the session. (For a complete description of the communications parameters for the \$WSIO macro, see “\$WSIO Macro” in Chapter 2.) The confirm indicator is valid only if OPM-CONFIRM was specified in the \$WSIO macro. If OPM is not CONFIRM, it must be zero.

User data or procedure parameters (in either positional or keyword form) to be passed to the other program or procedure are specified in the RCAD and OUTLEN parameters on the \$WSIO macro. The RCAD parameter can be used for both input and output when the evoke then get operation is used.

Example of \$WSIO Macro:

```
EVOK  $WSIO  DTF-ICDTF1,INLEN-256,RCAD-IOBUFF,OPC-EVG,  
        OPM-CONFIRM,PL@-EVKLST,OUTLEN-111
```

This \$WSIO macro (in your program) evokes a procedure on System/36, starts a transaction in the acquired session, and then waits for input because the operation is an **evoked then get** operation. The parameters to be used in the operation are those identified by the label EVKLST (shown in the following \$EVOK example). There are 111 bytes of output data or procedure parameters in your program buffer named IOBUFF that are to be sent to the other program or procedure. Note that, because 111 bytes are specified for the output buffer, the procedure name can have no more than 8 characters (the two cannot exceed the specified maximum of 119 bytes). Then, when input is received from the program, the data is placed in your program's buffer (IOBUFF), which is 256 bytes long.

\$EVOK Macro: The \$EVOK macro builds a parameter list to be associated with an evoke operation. The label on this macro should be the label specified on the PL@ parameter of the \$WSIO macro performing the evoke operation. (For a complete description of the \$EVOK macro and its parameters, see "\$EVOK Macro" in Chapter 2.)

Example of \$EVOK Macro:

```
      •  
      •  
EVKLST $EVOK  V-ALL,PNAME-ICPROC,LNAME-ICLIB,  
          UID-USERID,PWORD-PASS,SYNCL-1  
      •  
      •  
ICPROC EQU  *  
      DC  CL8 'ICFPROC '  
ICLIB  EQU  *  
      DC  CL8 'COMMLIB '  
USERID EQU  *  
      DC  CL8 'JJOHNSON '  
PASS   EQU  *  
      DC  CL4 'J4AG '
```

This \$EVOK example shows an evoke parameter list, used by a \$WSIO macro (such as the previous \$WSIO macro example), that causes the procedure named ICFPROC in the library named COMMLIB to be evoked. The user identifier JJOHNSON is located at the address labeled USERID, and the user's password J4AG is at the address PASS.

BASIC Evoke Operation Parameters

The following parameters are associated with BASIC evoke operations; System/36 uses the first four parameters to form the evoke parameter list. If you don't use a parameter (defined as a field in the BASIC evoke operations), enter the correct number of blanks for the unused field.

Positions	Field Description
1 through 8	The name of the procedure in System/36 to be evoked (left-adjusted).
9 through 16	Your password (left-adjusted), to be checked by System/36 (if security is being used) to ensure that your program is allowed to start the specified procedure.
17 through 24	Your user identifier (left-adjusted), to be checked by System/36 (if security is being used).
25 through 32	The name of the library that contains the procedure to be started (left-adjusted).
33 through xxxx	User data or procedure parameters (leading blanks are ignored).

Note: To see how the positions described here correspond to the values shown in the following example, see "Starting Remote Programs (Evoke Operations)" in Chapter 3.

IDDU Format Considerations:

Because IDDU does not reserve positions for unused parameters or duplicated parameters, one or more of the parameters in the evoke parameter list may not be specified or may be specified in a different order. Therefore, after defining an IDDU format to be used with the evoke operation, you should do the following:

1. Use IDDU to print a format definition listing for the format you defined.
2. Use the listing to determine the order and starting position of each of the parameters in the evoke parameter list.

BASIC Example (Evoke Operation):

```
030 WRITE #1, USING 40, FORMAT "$$EVOK": "BASICR", PASS$, USERID$, &  
      &"#LIBRARY", "ICFPROG, USERLIB" IOERR ICFERR  
040 FORM 4*C 8, C 15
```

The WRITE statement at line 30 writes data to communications file #1 using the FORM statement at line 40. The WRITE statement issues a \$\$EVOK (evoke then invite) operation to evoke the BASICR procedure, which is in #LIBRARY in System/36. The variable PASS\$ and the intrinsic function USERID\$ contain the password and user identifier needed to sign on to the system. The BASICR procedure calls the program ICFPROG that is in the user library USERLIB. The FORM statement at line 40 indicates that the \$\$EVOK operation is to send four fields (evoke parameters) of 8 characters (4*C 8) each and 15 bytes of positional parameters (C 15). If an error occurs, the program branches to the statement labeled ICFERR.

COBOL Evoke Operation Parameters

The following parameters are associated with COBOL evoke operations; System/36 uses the first four parameters to form the evoke parameter list. All the parameters must be defined by your program in the output area for COBOL evoke operations. All values in these fields must be character values. If a field is not used, space must still be reserved for it in the output area.

Bytes	Field Description
8	The name of the procedure in System/36 to be evoked.
8	The password you use to sign on the system if security is being used.
8	The user identifier you use to sign on the system if security is being used.
8	The name of the library containing the procedure to be started.
20	Reserved.
4	Length (in decimal) of user data or procedure parameters, if any.
xxxx	User data or procedure parameters.

IDDU Format Considerations:

Because IDDU does not reserve positions for unused parameters or duplicated parameters, one or more of the parameters in the evoke parameter list may not be specified or may be specified in a different order. Therefore, after defining an IDDU format to be used with the evoke operation, you should do the following:

1. Use IDDU to print a format definition listing for the format you defined.
2. Use the listing to determine the order and starting position of each of the parameters in the evoke parameter list.

COBOL Example (Evoke Operation):

```
*****
*   EVOKE PARAMETER LIST                               *
*****
57 01 EVOKE-RECORD.
58   03 PROCEDURE-NAME      PIC X(8)  VALUE 'ICFREM  '.
59   03 PASSWORD            PIC X(8)  VALUE 'T123   '.
60   03 USERID              PIC X(8)  VALUE 'OURSYSM'.
61   03 LIBRARY             PIC X(8)  VALUE 'THEIRLIB'.
62   03 FILLER              PIC X(20) VALUE SPACES.
63   03 EVOKE-DATA-LENGTH  PIC 9(4)  VALUE 0.
●
●
●
95  WRITE SCREEN-SSP-ICF-RECORD FROM EVOKE-RECORD,
     FORMAT IS '$$EVOK', TERMINAL IS ICF-SESSION.
```

The WRITE statement at line 95 issues the \$\$EVOK (evoke then invite) operation to evoke a procedure (ICFREM) in the session identified by ICF-SESSION. Lines 57 through 63 give the values of the parameters used in the evoke operation performed by the WRITE statement.

RPG II Evoke Operation Parameters

The following parameters are associated with RPG II evoke operations; System/36 uses the first four parameters to form the evoke parameter list. These parameters are defined as fields for the RPG II evoke operations. For any parameters that are not used, enter the correct number of blanks in the fields.

Positions	Field Description
1 through 8	The name of the procedure in System/36 to be evoked (left-adjusted).
9 through 16	The password (left-adjusted) you use to sign on the system if security is being used.
17 through 24	The user identifier (left-adjusted) you use to sign on the system if security is being used.
25 through 32	The name of the library in the system containing the procedure to be started (left-adjusted).
33 through 52	Reserved.
53 through 56	Length (in decimal) of user data or procedure parameters, if any (right-adjusted).
57 through xxxx	User data or procedure parameters.

IDDU Format Considerations:

Because IDDU does not reserve positions for unused parameters or duplicated parameters, one or more of the parameters in the evoke parameter list may not be specified or may be specified in a different order. Therefore, after defining an IDDU format to be used with the evoke operation, you should do the following:

1. Use IDDU to print a format definition listing for the format you defined.
2. Use the listing to determine the order and starting position of each of the parameters in the evoke parameter list.

RPG II Example (Evoke Operation):

IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21-9090
Printed in U.S.A.
75 76 77 78 79 80

Program	Keying Instruction	Graphic	Card Electro Number
Programmer	Date	Key	

Page 1 of 2

Program Identification

Line	Form Type	Filename or Record Name	Type (H/D/E)				Space	Skip	Output Indicators			Field Name or EXCPT Name	End Position in Output Record	Commas	Zero Balances to Print	No Sign	CR	-	X = Remove Plus Sign	Y = Date	Z = Zero Suppress	5-9 = User Defined
			O	R	D	E			And	And	And											
Constant or Edit Word																						
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24																						
01	O	OWSICF										*AUTO										
02	O												K6									
03	O												8	'MRTINV'								(procedure name)
04	O												16	'P7H3'								(password)
05	O												24	'TRW'								(user identifier)
06	O												30	'ICFLIB'								(library name)
07	O												56	'8'								(length of user data)
08	O												64	'12345678'								(user data)

This example shows a \$\$EVOK (evoke then invite) operation being used to evoke an MRT procedure named MRTINV that is located in the library ICFLIB. The user identifier is TRW, and the password is P7H3. Eight bytes of user data are to be passed to one of the programs that is started by the MRTINV procedure.

Fail Operation

Your program uses the **fail** operation to indicate that it has detected an abnormal condition while it was sending or receiving data. The fail operation causes a return code to be sent to the other program, indicating that the fail operation was issued.

If a program that is in the **send** state issues a fail operation, it may indicate that the data just sent was in error or that some other condition occurred. (The last record before the fail operation was issued is still sent to the other program.)

If a program that is in the **receive** state issues a fail operation, it indicates that the data received was in error. The program issuing the fail operation should immediately do at least one output operation so it can indicate why it sent the fail operation. (No data can be sent *with* a fail operation.) The record sent by the output operation should identify what the error is and where the other program should restart.

In either case, the program that issued the fail operation should send, and the program that receives the fail return code *must* receive. Otherwise, the program that was sending cannot determine which record failed or with which record it should begin sending again.

If both programs issue a fail operation at the same time, the program that was receiving will be successful and should send. The program that was sending will receive the fail return code indicating that its next operation *must* be an input operation.

Operation	Assembler	BASIC	COBOL	RPG II
Fail	FAIL	\$\$FAIL	\$\$FAIL	\$\$FAIL

Note: When a program that is in the receive state issues a fail operation, any other records following the record that failed are ignored by the receiving subsystem.

The fail operation can also be used with externally described field, format, and file definitions (also called data definitions). The interactive data definition utility (IDDU) is used to create data definitions. The following IDDU keyword is used for each language (in place of the operation code):

Operation	IDDU Keyword
Fail	FAIL

Refer to the manual *Getting Started with Interactive Data Definition Utility* for more information about IDDU and creating data definitions.

Note: If you are using IDDU, the IDDU keywords DETACH and INVITE may also be specified with FAIL. When FAIL and DETACH are specified, the result is a fail operation followed by an end of transaction. When keywords FAIL and INVITE are specified, the result is a fail operation followed by an invite operation.

Get Operations

Your program uses the **get** operation to obtain data from either a specific program or a specific display station. In an SSP-ICF session, the get operation causes the subsystem to get data from the program with which your program is communicating (and which has already been evoked). The get operation also causes your program to wait for the data if it is not immediately available. Your program receives control when the data is available. (Note that the *get* operation obtains data from a specific program or display station, and the *accept input* operation allows the data to come from *any* previously invited program or display station.)

In the Intra subsystem, the get operation can be issued by itself or, in assembler only, it can be issued in combination with another operation (such as cancel then get or put then get).

The **get attributes** operation (assembler, BASIC, and COBOL only) can be issued at any time during a session to determine the status of that session. (In BASIC, the ATTRIBUTE\$ intrinsic function is used.) The operation gets the current status information about the session in which your program is communicating.

The status information received by the get attributes operation contains (in 10 bytes) the following fields:

Position	Value	Meaning
1	A C R	Session not yet acquired by the program. Session is an acquired session. Session is a remotely started session.
2	N I O	Input not invited for this session. Input invited for this session, but no input is available. Invited input is available for this session.
3 through 10	Name	Location name (specified during configuration and on the SESSION OCL statement).

The **get status** operation (assembler, BASIC, and COBOL only) can also be issued at any time during a session to determine the status of a session (a 128-byte buffer is required). The get status operation receives the same status information as the get attributes operation along with the following additional status information:

Position	Value	Meaning
11	I A	Intra subsystem is being used. APPC subsystem is being used.
12	0 1	Synchronization level is NONE. Synchronization level is CONFIRM.
13	M B	Mapped conversation. Basic conversation.
14 through 16	Blanks	Reserved.
17 through 33	Name	Own fully qualified LU name.
34 through 41	Name	Partner LU name.
42 through 58	Name	Partner fully qualified LU name.
59 through 66	Name	Session group name.
67 through 74	Name	User ID.
75 through 128	Blanks	Reserved.

Operation	Assembler	BASIC	COBOL	RPG II
Get	GET	READ	READ ¹	NEXT and READ ²
Get attributes	GTA	ATTRIBUTE\$	ACCEPT	—
Get status	GST	ATTRIBUTE\$	ACCEPT	—
¹ In <i>COBOL</i> , a get operation is performed only if the <i>TERMINAL</i> option of the <i>READ</i> statement is specified with nonblanks. ² In <i>RPG II</i> , a get operation is performed only if a <i>NEXT</i> operation is executed before the <i>READ</i> operation.				

Invite Operation

Your program uses the **invite** operation to request input data from another program (via the associated session), but it receives control without waiting for the input. To obtain the data, your program must issue an accept input or a get operation later in this transaction.

The invite operation can be issued by itself or in combination with another operation (such as evoke then invite or put then invite).

Operation	Assembler	BASIC	COBOL	RPG II
Invite	INV ²	\$\$SEND ¹	\$\$SEND ¹	\$\$SEND ¹
¹ In BASIC, COBOL, or RPG II, only an invite operation is performed if \$\$SEND is issued with a record length of 0 bytes. Otherwise, \$\$SEND performs a <i>put then invite</i> operation. ² In assembler, a confirm indicator can be sent with this operation by specifying OPM-CONFIRM on the \$WSIO macro. If OPM is not CONFIRM, it must be ZERO.				

The invite operation can also be used with externally described field, format, and and file definitions (also called data definitions). The interactive data definition utility (IDDU) is used to create data definitions. The following IDDU keyword is used for each language (in place of the operation code):

Operation	IDDU Keyword
Invite	INVITE ¹
¹ A confirm indicator can be specified in addition to this function. A confirm indicator is specified in an IDDU format definition.	

Refer to the manual *Getting Started with Interactive Data Definition Utility* for more information about IDDU and creating data definitions.

Negative Response Operations

Your program uses the negative response operation to indicate that it detected something wrong with the data it received. Eight bytes of user data (sense information), indicating the reason for the negative response, are passed with the negative response indication to the program that sent the data.

The following are the three types of negative response operations.

- **Negative response:** Sends a negative response indication with sense data to the other program, and then returns control to your program without waiting for a response from the other program. This operation must be followed by an input operation to determine the action taken by the other program.
- **Negative response then get (assembler only):** Sends a negative response indication with sense data to the other program, and then waits for input to be received from the other program.
- **Negative response then invite:** Sends a negative response indication with sense data to the other program, and regains control without having to wait for the other program to respond. (An accept input or a get operation must be issued later to receive the invited input.)

Operation	Assembler	BASIC	COBOL	RPG II
Negative response ¹	NRP	\$\$NRSPNI	\$\$NRSPNI	\$\$NRSPNI
Negative response then get ¹	NRPG	—	—	—
Negative response then invite ¹	NRPI	\$\$NRSP	\$\$NRSP	\$\$NRSP

¹This operation is valid only in *batch* sessions, when BATCH-YES is specified on the SESSION statement.

The negative response operation can be issued only when: (1) the program is in the receive state; (2) the data received is within a chain, or the operation is the first operation after the end of a chain; and (3) this session is a batch session (BATCH-YES was specified on the SESSION statement associated with this program).

The 8 characters of user data should contain user-defined sense codes to indicate the reason for the negative response. To be sent to the other program, the sense codes must be put into your program's output buffer. The system sense code must be the first 4 characters in the buffer, and the next 4 characters make up the user sense code. The Intra subsystem checks to ensure that the system sense code is 10xx, 08xx, or 0000. If the system sense code is not one of these, the operation is rejected. If no sense code is supplied by the program, the default system sense code of 0811 (break) is used.

The program that receives the negative response gets a return code indicating the condition. That program must then do an input operation to receive the sense data. The program that received the negative response indication should issue a cancel operation.

Put Operations

The put operation passes data records from the issuing program to the other program in this transaction. Each put operation sends only one record to the subsystem. You can issue put operations only during a transaction. You can issue any of the following put operations without sending any data by specifying an output record length of zero.

- **Put:** Issues a put operation to the subsystem to send a data record to the other program, and returns control to your program without waiting for the operation to complete.
- **Put end of chain:** Issues a put operation that indicates to the other program that this is the last record in the chain. This operation also indicates to the subsystem that your program is not requesting any input. Control is returned to your program after the subsystem determines that the other program has received the end of chain indication.
- **Put end of transaction:** Issues a put operation to send a record to the other program, and then indicates to that program that this transaction has ended. (The session is still active, but communications between the two programs has ended.) Control is returned to your program after the subsystem acknowledges that the other program has received the end of transaction indication.

With the successful completion of this operation, communications between the two programs is ended. However, the session is still active between System/36 and the program that started the session. If *your* program started the session, it can start another transaction by evoking another program. If the *other* program started the session, it can start another transaction by evoking a different procedure on System/36.

- **Put then get (assembler only):** Issues a put operation to send a record to the other program, and then waits for the other program to send data to your program. Control is not returned to your program until the data is received. (See “Get Operations” earlier in this chapter.)
- **Put then invite:** Issues a put operation to send a record to the other program, followed by an invite operation so it can receive data from that program. (See “Invite Operation” earlier in this chapter.) Control is returned to your program without waiting for the other program to send the data. An accept input or get operation must be issued later in this transaction to receive the invited input.

There are three different **put function management header (FMH)** operations that issue a put operation with a function management header in the data. (For a description of function management headers, see “Function Management Headers (CICS and IMS)” in Chapters 1 and 2 of the *SSP-ICF Upline Subsystems Reference* manual.) One of the following operations must be used to indicate to the other program that the data includes a function management header:

- **Put FMH:** Sends a record that includes an FMH.
- **Put FMH then get** (assembler only): Sends a record that includes an FMH and then waits for the other program to send data.
- **Put FMH then invite:** Sends a record that includes an FMH and then invites (without waiting for) the other program to send data.

The put FMH operations are valid only if BATCH-YES was specified on the SESSION statement for the program that acquired the session. Any put FMH operation causes the receiving program to get a return code indicating that a function management header is included with the record.

The Intra subsystem does not check the format or contents of function management headers. (Function management headers have no particular use in the Intra subsystem; they are supported for compatibility with the SNUF subsystem.)

If you are using assembler, a confirm indicator can also be included with the data sent on a put, put end of transaction, put then get, and put then invite operations. If confirm indicators are included, OPM-CONFIRM must be specified. If OPM is not CONFIRM, it must be ZERO. In other languages, a confirm indicator is specified in an IDDU format definition.

Note: If a put end of transaction operation is specified with a confirm indicator and BATCH-YES is specified on the SESSION statement, the confirm indicator is ignored.

Operation	Assembler	BASIC	COBOL	RPG II
Put	PUT ³	\$\$\$SENDNI	\$\$\$SENDNI	\$\$\$SENDNI
Put end of chain	PEC	\$\$\$SENDE	\$\$\$SENDE	\$\$\$SENDE
Put end of transaction	PEX ³	\$\$\$SENDET	\$\$\$SENDET	\$\$\$SENDET
Put then get	PTG ³	—	—	—
Put then invite ¹	PTI ³	\$\$\$SEND	\$\$\$SEND	\$\$\$SEND
Put FMH ²	PFM	\$\$\$SENDNF	\$\$\$SENDNF	\$\$\$SENDNF
Put FMH then get ²	PFMG	—	—	—
Put FMH then invite ²	PFMI	\$\$\$SENDFM	\$\$\$SENDFM	\$\$\$SENDFM
¹ In BASIC, COBOL, or RPG II, if this operation is issued with a record length of 0 bytes, only the invite operation is performed. ² This operation is valid only in <i>batch</i> sessions, when BATCH-YES is specified on the SESSION statement. ³ In assembler, a confirm indicator can be sent with this operation by specifying OPM-CONFIRM on the \$WSIO macro. If OPM is not CONFIRM, it must be ZERO.				

The put, put end of transaction, and put then invite operations can also be used with externally described field, format, and file definitions (also called data definitions). The interactive data definition utility (IDDU) is used to create data definitions. The following IDDU keywords are used for each language (in place of the operation codes):

Operation	IDDU Keywords
Put Put end of transaction Put then invite	Field ^{1,2} DETACH ^{2,3} Field and INVITE ^{1,2}
¹ Data for these operations is specified in a separate user-defined field. ² A confirm indicator can be specified in addition to each function. A confirm indicator is specified in an IDDU format definition. ³ In addition, a user-defined field can be specified with this operation.	

Refer to the manual *Getting Started with Interactive Data Definition Utility* for more information about IDDU and creating data definitions.

Release Operation

Your program uses the **release** operation to attempt to terminate a session. Depending on how the session was started, the release operation produces different results:

- If the session was *acquired* by your program, the release operation terminates the session immediately (unless some error condition occurs). The operation frees the resources that were used during the session. (If the release operation is not successful, the **end of session** operation can be issued to terminate the session.) The same or another session can then be acquired.

If a release operation is issued for an acquired session during an active transaction, an error return code is received; the release operation can be performed only after your program successfully issues a put end of transaction operation or after your program receives an end of transaction return code.

- If the session was started when your program was *evoked* by another program, and your program is:
 - An MRT program, the release operation passes the session to the next step in your procedure. The system then executes any additional OCL statements in the procedure.
 - An SRT program, the release operation is delayed until your program terminates. (If your program issues another communications operation for that session, error return code 2800 is received.)

Operation	Assembler	BASIC	COBOL	RPG II
Release	REL	CLOSE	DROP	REL

Request to Change Direction Operations

Your program can use the request to change direction operation to indicate that it wants to send something to the other program (or it wants to end the session in a controlled manner) rather than continue receiving data. The other program, however, must decide whether to stop sending and when to stop.

After issuing a request to change direction operation, your program should continue to receive data until it receives a return code that indicates the other program has stopped sending. Your program, in response to the return code, can then begin sending its data, perform other processing, or terminate.

Your program can issue this operation only when three conditions exist: the operation must be issued only in a *batch* session, *during* a transaction, and while your program is in the *receive* state. If your program is neither receiving nor sending (that is, it is between chains), the operation has no effect.

The following are the two types of request to change direction operations.

- **Request to change direction then get** (assembler only): Sends a request to change direction *indication* to the other program, and waits for the other program to either send the next data record or end the transaction. Then, when the record or return code becomes available, your program receives control from the subsystem, and it receives the results of the get operation.
- **Request to change direction then invite**: Sends a request to change direction *indication* to the other program, performs an invite operation to continue receiving the data still being sent, and returns control to your program without having it wait for the invited input to be received. (An accept input or a get operation must be issued later in the transaction to receive the invited input.)

If additional request to change direction operations are issued before the direction has changed, the request to change direction portion of each one is ignored, but the input portion (get or invite) is performed.

When *your* program issues the request to change direction operation, the other program receives the request to change direction *indication* by means of return code 0010. It can then allow or ignore the request.

If the *other* program issues a request to change direction operation, your program receives (at the end of any type of put operation) a change direction *indication* by return code 0010. Your program should stop sending and issue an input operation as soon as possible.

Operation	Assembler	BASIC	COBOL	RPG II
Request to change direction then get ¹	RCDG	—	—	—
Request to change direction then invite ¹	RCDI	\$\$RCD	\$\$RCD	\$\$RCD
¹ This operation is valid only in <i>batch</i> sessions, when BATCH-YES is specified on the SESSION statement.				

The request to change direction operation can also be used with externally described field, format, and and file definitions (also called data definitions). The interactive data definition utility (IDDU) is used to create data definitions. The following IDDU keyword is used for each language (in place of the operation code):

Operation	IDDU Keyword
Request to change direction then invite	RQSWRT and INVITE

Refer to the manual *Getting Started with Interactive Data Definition Utility* for more information about IDDU and creating data definitions.

Set Timer Operation

Your program can use the **set timer** operation to set a timer before performing some specified function, such as an accept input operation. The set timer operation specifies an interval of time (in hours, minutes, and seconds) to wait before your program receives a timer expired return code. Your program continues to execute, and all operations are valid during the time interval. Your program must issue an accept input operation some time after it has issued the set timer operation, so that it can accept the return code after the timer has expired.

Only one time interval can be maintained for your program. If a previous set timer operation has been issued and the timer has not yet expired, the old time interval is replaced by the new interval.

You can use the set timer operation to retry other operations that may not be successful, possibly because of a temporary lack of resources (for example, during an acquire operation). To do this, issue the set timer operation and then perform accept operations until the timer expires. (The accept operations allow the program to continue receiving input from other invited programs and display stations while waiting for the timer.)

Note: If your program is a BASIC or RPG II program, a set timer (\$\$TIMER) operation is not valid unless at least one display station or session is attached to your program. (This restriction does not apply to the TIMER intrinsic function in BASIC.)

Operation	Assembler	BASIC	COBOL	RPG II
Set timer	STM	\$\$TIMER	\$\$TIMER	\$\$TIMER

Intra Subsystem Return Codes

This section describes all the return codes that are valid for the Intra subsystem. These are interactive communications return codes that are sent at the end of each subsystem operation to indicate the results of that operation. The appropriate return code is sent by the subsystem to the application program that issued the operation; the program can then check the results and act accordingly.

The return code is a 4-digit value; the first 2 digits contain the major code, and the last 2 digits contain the minor code. Assembler programs receive the return codes in binary form (2 bytes long). BASIC, COBOL, and RPG II programs receive the return codes in EBCDIC hexadecimal form (4 bytes).

Notes:

- 1. In the return code descriptions, **your program** refers to the local System/36 application program that initiates the operation and receives the return code from the subsystem. The **other program** refers to the other application program in the same System/36 with which this program is communicating through SSP-ICF.*
- 2. Several references are also made in the descriptions to **input** and **output** operations. The following chart shows all the input, output, and combined input/output operations that are valid for the Intra subsystem. Although all the operations shown are valid for Intra, the primary purpose for a few of them is to allow operations that are valid in other subsystems to be tested in the Intra subsystem. The validity of these operations also depends on the logical sequence of communications events occurring between the two programs in System/36.*
- 3. Appendix B contains brief descriptions of all the return codes for all subsystems, and it identifies all the subsystems for which each return code is valid. This appendix can be useful when you want to make changes to a program so it can be used with a different subsystem.*

Input Operations to Your Program	Output Operations from Your Program	Combined Operations in Your Program
Accept input		
	Acquire ¹	
	Cancel Cancel invite	Cancel then get ² Cancel then invite
	End of session	
	Evoke ^{3,6} Evoke end of transaction ^{3,6}	Evoke then get ^{2,3,6} Evoke then invite ^{3,6}
	Fail	
Get Get attributes ^{4,7} Get status ⁷		
Invite ⁶		
	Negative response	Negative response then get ² Negative response then invite
	Put ⁶ Put end of chain Put end of transaction ⁶	Put then get ^{2,6} Put then invite ⁶
	Put FMH	Put FMH then get ² Put FMH then invite
	Release	
		Request to change direction then get ² Request to change direction then invite
	Set timer ⁵	
<p>¹Normally, the acquire operation should be followed by an evoke operation in order to establish a transaction. However, it can also be followed by a set timer or get attributes (ATTRIBUTE\$, in BASIC) operation.</p> <p>²Valid only in assembler language.</p> <p>³Evoke operations in assembler can have OPM-FMH specified with the \$WSIO macro.</p> <p>⁴Valid only in assembler, BASIC, and COBOL languages.</p> <p>⁵For BASIC and RPG II programs, the set timer (\$\$TIMER) operation can only be issued in a session that is currently active, or to an acquired device that is currently attached to the program.</p> <p>⁶In assembler, this operation can have OPM-CONFIRM specified with the \$WSIO macro. In other languages, a confirm indicator is specified using an IDDU format definition.</p> <p>⁷The record area must be at least 128 bytes long.</p>		

Major Code 00 – Operation completed successfully.

General Description: The input or output operation issued by your program was completed successfully. The operation sent or received some data.

General Considerations: Check the minor return code for an end of transaction indication, and continue with the next operation.

Code Indication/Action

0000 Normal Indication: For input operations performed by your program, 0000 indicates that some data and a change direction indication were received on a successful input operation. The other program now wants to receive some data; your program must send it.

For output operations performed by your program, 0000 indicates that the last output operation was completed successfully and that your program can continue to send data.

Normal Action: If a change direction indication was received on an input operation, issue an output operation.

For the actions that can be taken (in this session) after 0000 is returned for an output operation, refer to the following chart:

In This Session, If Your Program:	And the Last Output Operation Was:	Then (in This Session):
Started the session (this is an acquired session)	Acquire operation	Issue another output (except acquire) operation, or issue an input operation.
	End of transaction (evoke or put) operation	Issue an(other) evoke operation, issue a release operation, continue local processing, or terminate your program.
	Any other output operation	Issue another output (except evoke) operation, or issue an input operation.
Was evoked ¹ (by another program)	Put end of transaction operation	Your session has ended; continue local processing, or terminate your program.
	Any other output operation	Issue another output (except evoke) operation, or issue an input operation.

¹An evoked program (started by an evoke operation issued by another System/36 program) cannot issue an evoke operation in this session; it can issue an evoke only in a different session, that it has first acquired. An evoked program that is part of a multiple-program procedure can issue a release operation at any time to pass the session on to the next program in the procedure. (An end of session operation would end the session, not pass it.) If the evoked program is an SRT program and it issues another communications operation after it issues the release operation, error code 2800 is returned to that program. Subsequent communicating operations in the next program, however, are processed normally.

Code Indication/Action

0001 Normal Indication: Your program has received some data on a successful input operation. It must continue to receive input until SSP-ICF returns a change direction indication, which allows your program to send data, or an end of transaction indication.

Normal Action: Issue another input operation. If your program can detect something equivalent to a change direction indication, indicating that the last of the data in the chain was just received, it can issue an output operation.

0003 Normal Indication: An end of chain (SNA) indication was received with some data on a successful input operation; the last record in the chain has been received.

Normal Action: Issue another input operation to receive the next chain.

0004 Normal Indication: A function management header and a change direction indication were received with some data on a successful input operation. The other program wants to receive some data.

Normal Action: Your program now has control of the session; process the function management header and issue an output operation.

0005 Normal Indication: A function management header was received with some data on a successful input operation. Your program must continue to receive input until SSP-ICF returns a change direction indication or an end of transaction indication.

Normal Action: Process the function management header and issue another input operation.

0008 Normal Indication: An end of transaction indication was received with the last of the data on a successful input operation. The transaction has ended, and the session with your program has ended.

Normal Action: If your program initiated the transaction, it can issue another evoke operation (to start another program), it can issue a release operation (to either perform local processing or start another session), or it can terminate. If a remote procedure start request initiated the transaction, your program can either issue an end of session operation or terminate.

000C **Normal Indication:** A function management header was received with an end of transaction indication and the last of the data on a successful input operation. The transaction has ended, and the session with your program has ended.

Normal Action: If your program initiated the transaction, it can issue another evoke operation (to start another program), it can issue a release operation (to either perform local processing or start another session), or it can terminate. If a remote procedure start request initiated the transaction, your program can either issue an end of session operation or terminate.

0010 **Normal Indication:** A request to change direction was received from the other program on a successful output operation for your program; the other program wants to send data as soon as possible. You should allow the other program to send its data.

Normal Action: Issue an input operation as soon as possible.

0014 **Normal Indication:** On a successful input operation, a change direction indication and some data was received. In addition, the other program requested confirmation.

Normal Action: Process any data received with the request. If no errors were detected by your program, issue an output operation. If an error was detected by your program, issue a fail operation, or terminate your program.

0015 **Normal Indication:** On a successful input operation, some data was received. In addition, the other program requested confirmation.

Normal Action: Process any data received with the request. If no errors were detected by your program, issue another input operation. If an error was detected by your program, issue a fail operation, or terminate your program.

001C **Normal Indication:** On a successful input operation, an end of transaction indication and some data was received. In addition, the other program requested confirmation.

Normal Action: If no errors were detected by your program and your program initiated the transaction, it can issue another evoke operation (to start another program), it can issue a release operation (to perform local processing), or it can terminate. If no errors were detected by your program and a remote procedure start initiated the transaction, your program can either issue an end of session operation, or terminate. If an error was detected by your program, issue a fail operation, or terminate your program.

0028 Normal Indication: An end of transaction indication was received with a system message on a successful input operation. The message, now in your program's input buffer, describes the status of the transaction that has ended. The session with your program has ended.

Normal Action: Handle the message in the input buffer (possibly display it). Also, if your program initiated the transaction, it can issue another evoke operation (to start another program), it can issue a release operation (to either perform local processing or start another session), or it can terminate. If your program was evoked, either issue an end of session operation or terminate your program.

0038 Normal Indication: An end of transaction indication was received with a *truncated* system message on a successful input operation. The message, truncated because it was too long for your program's input buffer, describes the status of the transaction that has ended. The session with your program has ended.

Normal Action: Handle the truncated message (possibly display it) in your program's input buffer. Also, if your program initiated the transaction, it can issue another evoke operation (to start another program), it can issue a release operation (to either perform local processing or start another session), or it can terminate. If your program was evoked, issue an end of session operation, or terminate your program.

Major Code 01 – Successful operation with a new requester.

A new requester return code indicates to your program that it was started by another program in System/36. Your program was started by an evoke operation (EVK or \$\$EVOKNI) that was sent by the other program. The request caused your program to be evoked if it is an SRT program or if it is an MRT program that was not already loaded and active. The request may have included some data for your program.

Normal Description: Each of the following return codes indicates that either the *input* operation issued by your program and responded to by a new requester completed successfully, or the output operation issued by your program in response to a new requester completed successfully.

If the operation was an input operation and data was included with the evoke operation, then that data is in your program's input buffer.

If your program is an SRT program that was evoked and the initial operation is an output operation, the operation sent some data to the new requester. However, although the operation did complete successfully, if the evoke request also included data for your program, that data is lost. Or, if an end of transaction indication was sent with the request, the data sent by your output operation is lost and the requesting program is released from your program.

If your program is an assembler program, the length of the data is returned in the input length field of the program's DTF. If the input length in the DTF is zero, no data was sent by the requester; if the input length is greater than zero, data was sent.

Note: The new requester return codes are returned only to evoked SRT programs and to active or evoked MRT programs.

General Considerations: Check the minor return code for an end of transaction indication, and continue with the next operation.

Code Indication/Action

0100 Normal Indication: On a successful input operation from a new requester, a procedure start request and a change direction indication were received, and a data record may have been received with the request. The other program now wants to receive some data; your program must send it.

For output operations performed by an evoked SRT program, the operation completed successfully.

Normal Action: For an input operation, handle any data that may have been passed with the request. For both input and output operations, perform any necessary record keeping for the new requester, and issue an output operation or an input operation.

0101 Normal Indication: On a successful input operation from a new requester, a procedure start request was received and some data may have been received. Your program must continue to receive input until SSP-ICF returns a change direction indication or an end of transaction indication.

Normal Action: Handle any data passed with the request, perform any necessary record keeping for the new requester, and issue another input operation. If your program can detect something equivalent to a change direction indication, indicating that the last of the data in the chain was just received, it can issue an output operation.

0104 Normal Indication: On a successful input operation from a new requester, a procedure start request, a function management header, and a change direction indication were received with some data. The other program now wants to receive some data.

Normal Action: Your program now has control of the session: process the function management header, handle the data passed with the request, perform any necessary record keeping for the new requester, and issue an output operation to the other program.

0105 Normal Indication: On a successful input operation from a new requester, a procedure start request and a function management header were received with some data. Your program must continue to receive input until SSP-ICF returns a change direction indication or an end of transaction indication.

Normal Action: Process the function management header, handle the data passed with the request, perform any necessary record keeping for the new requester, and issue an input operation.

0108 Normal Indication: On a successful input operation from a new requester, a procedure start request and an end of transaction indication were received, and some data may have been received. (A complete transaction was started and ended by the other program.) The session with your program has ended.

If your program is an SRT program (evoked by a new requester) that issued an output operation as its first operation, no data was sent to the requester even though the output operation completed successfully. Because an end of transaction indication was also received with the procedure start request, the requester is released from your program, and any data sent by the initial output operation is lost. And, if any data was sent by the requester, that data is lost also.

*Note: Return code 0118 is returned only to the **first** program in a multiple-program procedure (and only for the first operation). Return code 0108 is returned only to each one of the **succeeding** programs in that procedure (and only for the first operation in each program).*

Normal Action: Perform any necessary record keeping for the new requester of the transaction that has ended. Then, either issue an end of session operation or terminate your program.

010C Normal Indication: On a successful input operation from a new requester, a procedure start request and a function management header were received with data and an end of transaction indication. (A complete transaction was started and ended by the other program.) The session with your program has ended.

Normal Action: Process the function management header, handle the data passed with the request, and perform any necessary record keeping for the new requester. Then, either issue an end of session operation or terminate your program.

0114 Normal Indication: On a successful input operation from a new requester, a procedure start request, a change direction indication, and a confirm indication were received. In addition, the other program requested confirmation.

Normal Action: Perform any necessary record keeping for the new requester. Process any data received with the request. If no errors were detected by your program, issue an output operation. If an error was detected by your program, issue a fail operation.

0115 Normal Indication: On a successful input operation from a new requester, a procedure start request and a confirm indication were received, and some data may have been received. In addition, the other program requested confirmation.

Normal Action: Perform any necessary record keeping for the new requester. Process any data received with the request. If no errors were detected by your program, issue another input operation. If an error was detected by your program, issue a fail operation.

0118 Normal Indication: On a successful input operation from a new requester, a procedure start request was received with an end of transaction indication, and some data may have been received. (A complete transaction was started and ended by the other program.) The session has been ended.

If your program is an SRT program (evoked by a new requester) that issued an output operation as its first operation, no data was sent to the requester even though the output operation completed successfully. Because an end of transaction indication was also received with the procedure start request, the requester is released from your program, and any data sent by the initial output operation is lost.

*Note: Return code 0118 is returned only to the **first** program in a multiple-program procedure (and only for the first operation).*

Normal Action: Handle any data passed with the request, and perform any necessary record keeping for the new requester of the transaction that has ended. Then, because your program was evoked, either issue an end of session operation, or terminate your program.

011C Normal Indication: On a successful input operation from a new requester, a procedure start request and an end of transaction indication were received, and some data may have been received. In addition, the other program requested confirmation.

Normal Action: Perform any necessary record keeping for the new requester. Process any data received with the request. If no errors were detected by your program, issue an end of session operation, or terminate your program. If an error was detected by your program, issue a fail operation or terminate.

Major Code 02 – Successful operation, but a stop system request or a disable subsystem request is pending.

Normal Description: The input operation issued by your program was completed successfully. Your program received some data. However, because a stop system request or a disable subsystem request is pending, no new sessions using the subsystem can be initiated.

General Considerations: Your program should complete its communications processing as soon as reasonably possible so that the pending request to stop the system or to disable the subsystem can be completed in an orderly manner. (For example, you can issue a *request to change direction* operation to stop receiving input, or you can issue an *end of session* operation at the earliest logical stopping point.) Also, check the minor return code for an end of transaction indication, and continue with the next operation.

Code Indication/Action

0200 Normal Indication: On a successful input operation, an indication was received that a stop system request or a disable subsystem request is pending; no new sessions using the subsystem can be initiated. Also, 0200 indicates that some data and a change direction indication were received. The other program now wants to receive some data; your program must send it.

Normal Action: Issue an output operation.

0201 Normal Indication: Your program has received some data on a successful input operation. Also, a stop system request or a disable subsystem request is pending; no new sessions using the subsystem can be initiated. Your program must continue to receive input until SSP-ICF returns a change direction indication or an end of transaction indication.

Normal Action: Issue another input operation. If your program can detect something equivalent to a change direction indication, indicating that the last of the data in the chain was just received, it can issue an output operation.

0203 Normal Indication: An end of chain (SNA) indication was received with some data on a successful input operation; the last record in the chain has been received. Also, a stop system request or a disable subsystem request is pending; no new sessions using the subsystem can be initiated.

Normal Action: Issue another input operation to receive the next chain.

0204 Normal Indication: A function management header and a change direction indication were received with some data on a successful input operation. The other program now wants to receive some data. Also, a stop system request or a disable subsystem request is pending; no new sessions using the subsystem can be initiated.

Normal Action: Your program now has control of the session; process the function management header and issue an output operation.

0205 Normal Indication: A function management header was received with some data on a successful input operation. Also, a stop system request or a disable subsystem request is pending; no new sessions using the subsystem can be initiated. Your program must continue to receive input until SSP-ICF returns a change direction indication or an end of transaction indication.

Normal Action: Process the function management header and issue another input operation.

0208 Normal Indication: An end of transaction indication was received with the last of the data on a successful input operation. The transaction has ended, and the session with your program has ended. Also, a stop system request or a disable subsystem request is pending; no new sessions using the subsystem can be initiated.

Normal Action: If your program initiated the transaction, it can issue another evoke operation (only if a disable subsystem request condition is pending) to start another program, it can issue a release operation (to perform local processing), or it can terminate. If a remote procedure start request initiated the transaction, your program can either issue an end of session operation or terminate.

020C Normal Indication: A function management header and an end of transaction indication were received with the last of the data on a successful input operation. The transaction has ended, and the session with your program has ended. Also, a stop system request or a disable subsystem request is pending; no new sessions using the subsystem can be initiated.

Normal Action: If your program initiated the transaction, it can issue another evoke operation (only if a disable subsystem request condition is pending) to start another program, it can issue a release operation (to perform local processing), or it can terminate. If a remote procedure start request initiated the transaction, your program can either issue an end of session operation or terminate.

0214 Normal Indication: On a successful input operation, a change direction indication and some data was received. In addition, the other program requested confirmation. Also, a stop system request or a disable subsystem request is pending; no new sessions using the subsystem can be initiated.

Normal Action: Process any data received with the request. If no errors were detected by your program, issue an output operation. If an error was detected by your program, issue a fail operation, or terminate your program.

0215 Normal Indication: On a successful input operation, some data was received. In addition, the other program requested confirmation. Also, a stop system request or a disable subsystem request is pending; no new sessions using the subsystem can be initiated.

Normal Action: Process any data received with the request. If no errors were detected by your program, issue another input operation. If an error was detected by your program, issue a fail operation, or terminate your program.

021C Normal Indication: An end of transaction indication was received with the last of the data on a successful input operation. In addition, the other program requested confirmation. Also, a stop system request or a disable subsystem request is pending; no new sessions using the subsystem can be initiated.

Normal Action: If no errors were detected by your program and your program initiated the transaction, it can issue another evoke operation (to start another program), it can issue a release operation (to perform local processing), or it can terminate. If no errors were detected by your program and a remote procedure start initiated the transaction, your program can either issue an end of session operation, or terminate. If an error was detected by your program, issue a fail operation (the transaction remains active).

0228 **Normal Indication:** An end of transaction indication was received with a system message on a successful input operation. The message (now in your program's input buffer) describes the status of the transaction that has ended. The session with your program has ended. Also, a stop system request or a disable subsystem request is pending; no new sessions using the subsystem can be initiated.

Normal Action: Handle the message in the input buffer (display it, for example). If your program initiated the transaction, it can issue another evoke operation (only if a disable subsystem request condition is pending) to start another program, it can issue a release operation (to perform local processing), or it can terminate. If your program was evoked, issue an end of session operation, or terminate your program.

0238 **Normal Indication:** An end of transaction indication was received with a *truncated* system message on a successful input operation. The message, truncated because it was too long for your program's input buffer, describes the status of the transaction that has ended. The session with your program has ended. Also, a stop system request or a disable subsystem request is pending; no new sessions using the subsystem can be initiated.

Normal Action: Handle the truncated message in your program's input buffer (display it, for example). If your program initiated the transaction, it can issue another evoke operation (only if a disable subsystem request condition is pending) to start another program, it can issue a release operation (to perform local processing), or it can terminate. If your program was evoked, issue an end of session operation, or terminate your program.

Major Code 03 – Successful operation, but no data received.

Normal Description: The input or set timer (output) operation just performed was completed successfully, but no data was sent or received.

General Considerations: Check the minor return code for an end of transaction indication, and continue with the next operation.

Code Indication/Action

0300 Normal Indication: A change direction indication with *no* data was received on a successful input operation.

Normal Action: Issue an output operation or continue to issue input operations.

0301 Normal Indication: On a successful input operation, *no* data was received. Your program must continue to receive input until SSP-ICF returns a change direction indication, which allows your program to send data, or an end of transaction indication.

Normal Action: Issue another input operation.

0302 Normal Indication: A fail indication was received with *no* data on a *successful* input operation. The other program has issued a fail operation to indicate, for example, that the previous data that it sent was in error.

Normal Action: Issue another input operation.

0303 Normal Indication: An end of chain indication was received *without* data on a successful input operation; the last record in the chain has already been received.

Normal Action: Consider the data chain complete and issue another input operation to receive the next chain.

0308 Normal Indication: An end of transaction indication was received *without* data on a successful input operation. The transaction has ended, and the session with your program has ended.

Normal Action: If your program initiated the transaction, it can issue another evoke operation (to start another program) or it can issue a release operation (to either perform local processing or start another session). If a remote procedure start request initiated the transaction, your program can either issue an end of session operation or terminate.

0310 Normal Indication: The time interval specified by a set timer operation in your program has expired.

Note: If your program has an exception handling routine, you should check for the 0310 return code before you make any checks based on the WSID field.

Normal Action: Issue the operation that is to perform the intended function (such as displaying a message) after the specified time interval has expired.

0314 Normal Indication: On a successful input operation, a change direction indication without data was received. In addition, the other program requested confirmation.

Normal Action: If no errors were detected by your program, issue an output operation. If an error was detected by your program, issue a fail operation, or terminate your program.

0315 Normal Indication: On a successful input operation, no data was received. In addition, the other program requested confirmation.

Normal Action: If no errors were detected by your program, issue another input operation. If an error was detected by your program, issue a fail operation, or terminate your program.

031C Normal Indication: On a successful input operation, an end of transaction indication without data was received. In addition, the other program requested confirmation.

Normal Action: If no errors were detected by your program and your program initiated the transaction, it can issue another evoke operation (to start another program), it can issue a release operation (to perform local processing), or it can terminate. If no errors were detected by your program, and a remote procedure start initiated the transaction, your program can either issue an end of session operation, or terminate. If an error was detected by your program, issue a fail operation, or terminate your program.

Major Code 04 – Output exception occurred.

Normal (Exception) Description: An output exception occurred because your program attempted to send output when it should be receiving the output that has already been sent by the other program. Your output was not sent and should be sent later.

Note: If your program issues another output operation, an error return code of 831C or 8323 will be received.

General Recovery Actions: Issue an input operation to receive data or a message from the other program, or to allow the other program to send a change direction indication.

Code Indication/Action

0402 Normal Indication: A fail operation was issued by the other program to indicate that the data sent by your program was in error and caused an exception condition in the other program.

Recovery Action: Issue an input operation to begin performing the recovery actions that were previously agreed to by the programmers of both this program and the other program.

0411 Normal Indication: The other program has sent a *message* for your program, but because your program also attempted an output operation, the message is still in the subsystem input buffer, waiting to be received. Your program must receive the message before it can perform an output operation.

Normal Action: Issue an input operation to receive the message.

0412 Normal Indication: The other program has sent *data* for your program, but because your program also attempted an output operation, the data is still in the subsystem input buffer, waiting to be received. Or, a cancel invite operation failed either when the other program sent a message or data for your program, or when a return code was received from the subsystem. Your program must receive the data, message, or return code before it can issue an output operation.

Normal Action: Issue an input operation or an accept input operation to receive the data, message, or return code.

Major Codes 08-34 – Miscellaneous program errors.

Error Description: An operation attempted by your program failed. The error may have occurred because an operation was issued at the wrong time or because a data record was too long.

Recovery Action: Refer to the individual return code descriptions for the appropriate recovery actions.

Code Indication/Action

0800 Error Indication: The acquire operation just performed was not successful. It tried to acquire a session that has already been acquired by your program and that is still active.

Recovery Action: If the session requested by the original acquire operation is the one needed, your program can begin communicating in the session because it is already available. If a different session is desired, issue another acquire operation for a different session by specifying a different session ID. (The identifier must have been specified in the SYMID parameter of a SESSION statement that preceded the program.)

1100 Error Indication: The accept operation just performed in your program was not successful for one of the following reasons: (1) Your MRT program may have just released its last requester, indicating that your program can begin to terminate normally. (2) Your program may have attempted to accept input when no invite operations have been issued and the program is *not* an MRT or NEP program. (3) Your program *is* both an MRT and an NEP program, and a stop system condition is in effect, which suppresses the implied invites to all potential requesters.

Recovery Action: If you still have a requester or an acquired session, issue an invite operation (or a combined operation that includes an invite) followed by an accept input operation. This return code indicates the logical end of file for WORKSTN files in RPG II programs and TRANSACTION files in COBOL programs.

2800 Error Indication: Your program (which is an SRT program that has been evoked by another program) has issued a release operation in the session in which it was evoked, and is now attempting to communicate with the evoking program. Because that session was released from your program, this operation was not performed, and any further attempts to communicate with that program results in another 2800 return code. (The session is ended for your program only, if it is part of a multiple-program procedure.)

Recovery Action: Continue local processing or terminate your program. Your program may be in error; you should correct it so that the release operation is issued after all communications with the other program have been completed.

3401 Error Indication: This input operation was rejected because the record length of the data sent by the other program exceeds the length of your program's input buffer.

Recovery Action: Issue a message about the error to the local system and terminate your program. Then, in your program, change the record length of the input buffer to be at least as long as the longest data record to be received. For assembler programs only, the record length of the rejected data is contained in the DTF, at offset \$WSEFFL. For other program types, the length is not available; only the error indication is received.

Major Code 80 – Permanent (nonrecoverable) subsystem error.

Error Description: A nonrecoverable error has occurred in the subsystem; the subsystem has been (or is being) disabled, and your session has been terminated. The error indication has been sent as a message to the display station or to the system console; the operator can refer to the *System Messages* manual for additional information. The error indication is also returned to your program as a return code; the minor code portion indicates the specific cause. (Each return code is described on the following pages.) The subsystem must be enabled again before communications can resume.

General Recovery Actions: The following general actions can be taken for each 80xx return code. Other specific actions are given in each return code description.

- Issue, to the system operator or to the display station operator who started the program, a message requesting that the subsystem be enabled again.
- Issue an end of session (EOS or \$\$EOS) operation for the session that has terminated. Your program can: (1) wait for the subsystem to be enabled by issuing a set timer (\$\$TIMER) operation or by using the TIMER intrinsic function (in BASIC only); (2) continue local processing; or (3) terminate. Note that if your program is written in BASIC or RPG II, the \$\$TIMER operation is not valid unless at least one display station or session is attached to your program.
- If the session should be started again after the subsystem is enabled, it must be reacquired by your program or restarted by the other program.

Note: If the session is started again, it starts from the beginning, not at the point where the session error occurred.

Code Indication/Action

8081 Error Indication: An SSP-ICF error caused a processor check in this subsystem.

Recovery Action: This subsystem has been disabled; it must be enabled again before communications can resume. Your program can continue local processing, wait to reissue the acquire operation, or terminate.

8082 Error Indication: This session is being terminated immediately because the subsystem controlling the session is currently being disabled; the subsystem is not waiting for any of its active sessions to be completed normally.

Recovery Action: Communications with the other program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, wait to reissue the acquire operation, or terminate.

Major Code 82 – Acquire operation failed.

Error Description: An attempt to acquire a session was not successful; the session was not started. An error indication was returned to your program as a return code; the minor portion of the code indicates the specific cause. (Each return code is described on the following pages.) The error indication has also been sent as a message to the display station or to the system console; the operator can refer to the *System Messages* manual for additional information.

General Recovery Actions: The following general actions can be taken for each 82xx return code. Other specific actions are given in each return code description.

1. Determine why the 82xx error code was returned to your program. Read the description of that return code to determine what action is needed.
2. If a parameter value must be changed in the subsystem configuration record or in the SESSION statement for your program:
 - a. To change a parameter value in the subsystem configuration, disable the subsystem first, make the change in the subsystem's configuration record, then enable the subsystem again to make the change effective.
 - b. To change a parameter value in the SESSION statement associated with your program, terminate only your program to change your SESSION statement.

Note: When a parameter can be specified both in the SESSION statement and in the subsystem configuration, the value in the SESSION statement overrides the value in the subsystem configuration record (for your program only). Therefore, in some cases, you may choose to make a change in the SESSION statement rather than disabling the subsystem to make the change in its configuration record.

3. If no change is needed in your program or in the subsystem, simply reissue the acquire operation. It could be successful if the error was temporary (for example, if the maximum number of acquired sessions had been reached when the acquire operation was first issued). If the acquire operation is again unsuccessful, it should be retried only a limited number of times. (The limit for retries should be specified in your program.)

(continued)

Major Code 82 – Acquire operation failed.

General Recovery Actions: (continued)

4. Issue a set timer operation in your program so it can wait for a specified time interval before reissuing the acquire operation. However, for RPG II and BASIC programs, the set timer (\$TIMER) operation is valid only if at least one display station or session is attached to your program. (This restriction does not apply to COBOL and assembler programs or to the TIMER intrinsic function in BASIC.)

Code Indication/Action

8233 Error Indication: On an unsuccessful acquire operation, an invalid session identifier was detected. Either no SESSION statement was specified between the LOAD and RUN statements for this program, or the session identifier in your program does not match the identifier specified on the SESSION statement for the session being acquired. The session was not started.

Recovery Action: If the error is in your program, specify the correct session identifier in your program. If an incorrect identifier was specified on the SESSION statement, specify the correct value in the SYMID parameter.

8281 Error Indication: On an unsuccessful acquire operation, an SSP-ICF error condition was detected. The error caused a processor check in this subsystem.

Recovery Action: This subsystem has been disabled; it must be enabled again before communications can resume. Your program can continue local processing, wait to reissue the acquire operation, or terminate.

8282 Error Indication: The acquire operation just performed was unsuccessful because the subsystem controlling the session is currently being disabled; no sessions can be acquired in the subsystem.

Recovery Action: Communications with the other program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, wait to reissue the acquire operation, or terminate.

82A8 Error Indication: The acquire operation was not successful because the maximum number of active sessions allowed in the system has been reached. No more than 360 sessions can be active in System/36 at one time. The session was not started.

If this acquire operation is associated with a SESSION statement (normal acquire), the maximum of 260 normally acquired sessions are already active at this time. If this acquire operation is *not* associated with a SESSION statement (BASIC special acquire), the maximum of 100 specially acquired and/or evoked sessions are already active at this time.

Recovery Action: Your program can wait for another session to end and then reissue the acquire operation. Otherwise, your program can continue local processing or terminate.

82AA Error Indication: The acquire operation just performed was not successful because the specified subsystem has not been enabled or has been disabled. The subsystem that must be enabled is the one whose subsystem configuration member contains the same remote location name as that specified by the location parameter in the SESSION statement or on the OPEN statement in BASIC. That location name must also be specified in the subsystem configuration record. The session was not started.

Recovery Action: Verify that the subsystem name was specified correctly on the LOCATION parameter of the SESSION statement or on the OPEN statement in BASIC. If the correct name was specified, enable the specified subsystem by entering the ENABLE procedure command. Then reissue the acquire operation. Otherwise, your program can continue local processing, wait to reissue the acquire operation, or terminate.

82AB Error Indication: The acquire operation just performed was not successful because the specified subsystem is currently *being* enabled. The session was not started.

Recovery Action: Your program can wait until the subsystem has been enabled, then reissue the acquire operation to start the session.

82B0 Error Indication: The acquire operation just performed was not successful either because the specified subsystem is currently being disabled, or because it has a disable subsystem request pending. No new sessions can be started.

Recovery Action: Your program can wait until the subsystem is enabled again, and then reissue the acquire operation. Otherwise, your program can continue local processing, or it can terminate.

Major Code 83 – Session error occurred.

Error Description: An error has occurred in the session, but the session is still active. Recovery might be possible; the error indication was returned to your program as a return code. The minor portion of the code indicates the specific cause. (Each return code is described on the following pages.) The error indication has also been sent as a message to the display station or to the system console; the operator can refer to the *System Messages* manual for additional information.

General Recovery Actions: The following general actions can be taken for each 83xx return code. Other specific actions are given in each return code description.

1. Determine why the 83xx error code was returned to your program. Read the description of that return code to determine what action is needed.
2. If a parameter value must be changed in the subsystem configuration record or in the SESSION statement for your program:
 - a. To change a parameter value in the subsystem configuration, disable the subsystem first, make the change in the subsystem's configuration record, then enable the subsystem again to make the change effective.
 - b. To change a parameter value in the SESSION statement associated with your program, terminate only your program before correcting your SESSION statement.

Note: When a parameter can be specified both in the SESSION statement and in the subsystem configuration, the value in the SESSION statement overrides the value in the subsystem configuration record (for your program only). Therefore, in some cases, you may choose to make a change in the SESSION statement rather than disabling the subsystem to make the change in its configuration record.

3. If no change is needed in your program or in the subsystem, (and depending on what the return code description says):
 - a. Check the other program to see if a change is required in it to correct the error received.
 - b. Retry the operation, if possible; it could be successful. If it is not successful, it should be retried only a limited number of times. (The limit for retries should be specified in your program.)

Code Indication/Action

830B Error Indication: Your program has attempted to execute a communications input or output operation either before the session was acquired or after it has ended. Your program may have (1) issued an input or output operation either *before* it issued an acquire operation or *after* it has released the session (by a release or end of session operation), or it may have (2) improperly handled a session was not acquired error return code.

Recovery Action: Check your program to ensure that no input or output operation is attempted without an active session and to ensure that the return code is handled properly. If you want your program to recover from an improperly handled error condition, issue another acquire operation.

8319 Error Indication: A negative response to the previous output operation was issued by the other (receiving) program. Sense data was sent with the negative response and it is in the subsystem input buffer waiting to be received by your program.

Recovery Action: Issue an input operation to receive the sense data.

831A Error Indication: The evoked program terminated abnormally or an evoke operation failed to complete successfully. A message describing the error condition is waiting in the subsystem input buffer. If an evoke operation failed, it could have been either the operation just performed or a previous operation (when the evoke operation was combined with another operation, such as evoke then invite, or when the evoke was followed by an accept input operation).

Recovery Action: Your program should issue an input operation to receive the message so you can print or display it. Then it can reissue the evoke operation to reestablish the transaction, it can issue an end of session operation, or it can terminate.

831B Error Indication: On the previous negative response operation issued by your program, invalid sense data was included. The data was not sent.

Recovery Action: Correct your program so that, on a negative response operation, valid sense data is sent. The sense data can be no longer than 8 bytes, and it must begin with 10xx, 08xx, or 0000.

831C Error Indication: The output operation issued before this output operation received a return code indicating that the other program sent a message or data for your program, but that return code was not properly handled in your program. *This* output operation was rejected as invalid at this time because your program must first issue an input operation to receive the message or data.

Recovery Action: Issue an input operation to receive the message or data.

831E Error Indication: The operation just issued by your program was invalid. Either the operation code is an unrecognized code, or the operation specified by the code is not supported by the subsystem. Or, you may have attempted a batch operation, but BATCH-NO was specified in the SESSION statement for your program. The session is still active.

Recovery Action: Your program can try a different operation, issue a release or end of session operation, or terminate. Correct the error in your program or in the SESSION statement before attempting to communicate with the other program.

831F Error Indication: On an output operation, an indication was received that your program tried to send a data record having a length that exceeds the maximum user record length specified for this session. The session is still active.

Recovery Action: Change the record length in your program and recompile it. The maximum user record length must be large enough for the longest record to be sent or received. Reissue the acquire operation to restart the session after making these changes.

8322 Error Indication: A put with no invite operation was followed by a *request to change direction then get* operation, a *request to change direction then invite* operation, or a negative response operation. None of these operations are valid while your program is in the send state. The session is still active.

Recovery Action: Your program can issue an output operation to continue sending, issue an input operation to begin receiving, issue an end of session operation to continue local processing, or terminate. Correct the error in your program before attempting to communicate with another program.

8323 Error Indication: Either a cancel operation was issued while your program was in receive state (the cancel operation is valid only in send state); or your program received a fail indication while it was in send state, and it issued another output operation (an input operation should follow a received fail operation). The session is still active.

Recovery Action: Before attempting to communicate with another program, correct the error in your program.

8326 Error Indication: Following an output operation, an invalid cancel operation was issued by your program. The cancel operation is valid only within a chain, not preceding a chain or between chains. The session is still active.

Recovery Action: Either continue local processing by ignoring the error, or correct the error in your program before attempting to communicate with another program.

8327 Error Indication: An invalid input or output operation was issued when no transaction existed; your program may have expected more data when there is none. Either the other program has already ended the transaction, your program has ended the transaction, or your program has not issued an evoke operation to start communicating with the other program. The session is still active.

Recovery Action: If you want your program to recover from this error, issue an evoke operation to start a transaction. Otherwise, issue an end of session operation; then continue local processing or terminate your program. If a coding error in your program caused the error, correct your program.

8329 Error Indication: An invalid evoke operation was detected in this session. Your program was evoked by an evoke operation issued by another program, and cannot, therefore, issue any evoke operations in this session.

Recovery Action: If you want your program to recover from this error, issue a different operation. If you want to issue the evoke in another session, issue an acquire operation, then issue the evoke operation. Otherwise, you can issue an end of session operation to terminate this session; then continue local processing or terminate your program. If a coding error in your program caused the error, correct your program.

832A Error Indication: An indication that both programs were attempting to receive input was detected by the subsystem. The program that was in control of the transaction (in send state) issued an input operation without indicating a change of direction, or the program that was in receive state ignored the change direction indication and issued another input operation. The session is still active.

Recovery Action: Either issue an output operation to send data, or issue a request to change direction operation so the transaction can be synchronized. If a coding error in your program caused the error, correct your program.

832C Error Indication: An invalid release operation, following an invite operation, was detected in your program. Because your program issued the invite operation, it cannot issue a release operation to terminate the invited session.

Recovery Action: Issue an accept or get operation to satisfy the invite operation. Otherwise, issue an end of session operation to terminate the session. If a coding error in your program caused the error, correct your program.

832D Error Indication: An invalid operation following an invite operation was detected in your program. Once you have issued an invite operation, the next subsystem operation must be a get or accept operation.

Recovery Action: Issue a get operation or an accept input operation to receive the input that was invited. Otherwise, issue an end of session operation to terminate the session. If a coding error in your program caused the error, correct your program.

832F Error Indication: An invalid evoke or release operation was issued before a transaction was completed. The operation was not performed. The session is still active.

Recovery Action: Your program can terminate the transaction by issuing a put end of transaction operation; then it should issue an evoke operation to start another transaction or issue a release operation. If a coding error in your program caused the error, correct your program.

8330 Error Indication: On an input operation performed by this program, a cancel operation and a change direction indication were received. The other program canceled the transaction it was sending and now wants to receive some data; your program must send it. (The session is still active.)

Recovery Action: Issue an output operation.

8331 Error Indication: On an input operation performed by this program, a cancel operation was received *without* a change direction indication. The other program canceled the transaction it was sending (possibly because it detected an error in the data), but it wants to send the data again or send different data. (The session is still active.) Your program must continue to receive input until SSP-ICF returns a change direction indication or an end of transaction indication.

Recovery Action: Discard the previously received input (or perform any other agreed-to activity), then issue another input operation.

8333 Error Indication: On an input or output operation, an invalid session identifier was detected. The session is still active.

Recovery Action: Reissue the operation with the correct session identifier. Otherwise, issue an end of session operation, then terminate the program and correct the programming error that caused the communications error.

83E0 Error Indication: Your program attempted to execute an operation using an IDDU format definition that was not found.

Recovery Action: Check the name of format definition in your program to be sure it is correct. Then check the format definition to see whether it is defined in the file definition.

83E1 Error Indication: Your program attempted to execute an operation using an IDDU format definition that could not be retrieved from disk.

Recovery Action: Continue with another operation, or a different format definition. Then check the data dictionary to see if the format definition is defined. The format definition could have been deleted using another method besides IDDU.

83E8 Error Indication: A cancel invite operation either was issued to a session that was not invited or was issued before the initial accept input operation in the evoked program was completed. The cancel invite operation is valid only when it is issued after any valid invite operation. The session and the transaction, however, are still active.

Recovery Action: Your program can issue an output operation to continue sending, issue an input operation to begin receiving, issue an end of session operation to continue local processing, or terminate. Correct the error in your program before attempting to communicate with another program on your system.

Appendix A. Subsystem Operation Codes (by Language)

Assembler Operations	A-3
BASIC Operations	A-4
COBOL Operations	A-5
RPG II Operations	A-6

The following charts are duplicates of those shown in each language chapter. The charts show all the communications operations that are valid for each programming language supported by System/36 (assembler, BASIC, COBOL, and RPG II). Each chart shows the valid operations for that language, their operation codes (mnemonics), and the subsystems supporting each operation. An **x** in a subsystem column means that the subsystem supports the operation, and a **-** means that it does not.

Assembler Operations

The following communications operations are valid in the assembler language; each operation, however, is valid only for those subsystems containing an x in the corresponding subsystem column. (For a description of each operation, refer to the subsystem operations topic in the reference manual describing your subsystem.)

Assembler SSP-ICF Operation	Assembler Operation	Communications Subsystems									
		Intra	BSC ¹	CCP	CICS	IMS	3270 ¹	Finance	Peer	SNUF	APPC
Accept input	ACI	x	x	x	x	x	x	x	x	x	x
Aquire	ACQ	x	x	x	x	x	x	x	x	x	x
Cancel	CAN	x	-	-	-	-	-	-	-	x	-
Cancel invite	CNI	x	x	-	-	-	-	-	-	-	-
Cancel then get	CANG	x	-	-	-	-	-	-	-	x	-
Cancel then invite	CANI	x	-	-	-	-	-	-	-	x	-
End of session	EOS	x	x	x	x	x	x	x	x	x	x
Evoke	EVK	x	x	x	x	x	x ²	-	x	x	x
Evoke end of transaction	EVE	x	x	-	x	x	-	-	x	x	x
Evoke then get	EVG	x	x	x	x	x	x ²	-	x	x	x
Evoke then invite	EVI	x	x	x	x	x	x ²	-	x	x	x
Fail	FAIL	x	-	-	-	-	-	-	x	-	x
Get	GET	x	x	x	x	x	x	x	x	x	x
Get attributes	GTA	x	x	x	x	x	x	x	x	x	x
Get status ³	GST	x	-	-	-	-	-	-	-	-	x
Invite	INV	x	x	x	x	x	x	x	x	x	x
Negative response	NRP	x	-	-	-	-	-	-	-	x	-
Negative response then get	NRPG	x	-	-	-	-	-	-	-	x	-
Negative response then invite	NRPI	x	-	-	-	-	-	-	-	x	-
Put	PUT	x	x	x	x	x	-	x	x	x	x
Put end of file/chain	PEF/PEC	x	x	x	x	-	x	x	x	x	-
Put end of transaction	PEX	x	x	-	x	x	-	-	x	x	x
Put then get	PTG	x	x	x	x	x	x	x	x	x	x
Put then invite	PTI	x	x	x	x	x	x	x	x	x	x
Put FMH	PFM	x	-	-	-	-	-	x	-	x	-
Put FMH then get	PFMG	x	-	-	-	-	-	x	-	x	-
Put FMH then invite	PFMI	x	-	-	-	-	-	x	-	x	-
Release	REL	x	x	x	x	x	x	x	x	x	x
Request to change direction then get	RCDG	x	x	x	-	-	-	-	x	x	x
Request to change direction then invite	RCDI	x	x	x	-	-	-	-	x	x	x
Set timer	STM	x	x	x	x	x	x	x	x	x	x

¹Although the BSC 3270 subsystem is not part of SSP-ICF, its operations are listed here to show its similarities to other subsystems.

²Valid only when the DATAID and FLDLTH parameters are specified on the SESSION OCL statement, and when the HOSTNAME parameter on the SESSION statement is CICS or IMS.

³The record area must be at least 128 bytes long.

BASIC Operations

The following communications operations are valid in the BASIC language; each operation, however, is valid only for those subsystems containing an x in the corresponding subsystem column. (For a description of each operation, refer to the subsystem operations topic in the chapter describing your subsystem.)

BASIC SSP-ICF Operation	BASIC Operation	Communications Subsystems									
		Intra	BSC ¹	CCP	CICS	IMS	3270 ¹	Finance	Peer	SNUF	APPC
Accept input	WAITIO ²	x	x	x	x	x	x	x	x	x	x
Aquire	OPEN	x	x	x	x	x	x	x	x	x	x
Cancel	\$\$CANLNI	x	-	-	-	-	-	-	-	x	-
Cancel invite	\$\$CNLINV	x	x	-	-	-	-	-	-	-	-
Cancel then invite	\$\$CANL	x	-	-	-	-	-	-	-	x	-
End of session	\$\$EOS	x	x	x	x	x	x	x	x	x	x
Evoke	\$\$EVOKNI	x	x	x	x	x	x ³	-	x	x	x
Evoke end of transaction	\$\$EVOKET	x	x	-	x	x	-	-	x	x	x
Evoke then invite	\$\$EVOK	x	x	x	x	x	x ³	-	x	x	x
Fail	\$\$FAIL	x	-	-	-	-	-	-	x	-	x
Get	READ	x	x	x	x	x	x	x	x	x	x
Get attributes	ATTRIBUTE	x	x	x	x	x	x	x	x	x	x
Get status ⁶	ATTRIBUTE	x	-	-	-	-	-	-	-	-	x
Invite ⁴	\$\$SEND	x	x	x	x	x	x	x	x	x	x
Negative response	\$\$NRSPNI	x	-	-	-	-	-	-	-	x	-
Negative response then invite	\$\$NRSP	x	-	-	-	-	-	-	-	x	-
Put	\$\$SENDNI	x	x	x	x	x	-	x	x	x	x
Put end of file/chain	\$\$SENDE	x	x	x	x	-	x	x	x	x	-
Put end of transaction	\$\$SENDET	x	x	-	x	x	-	-	x	x	x
Put FMH	\$\$SENDNF	x	-	-	-	-	-	x	-	x	-
Put FMH then invite	\$\$SENDFM	x	-	-	-	-	-	x	-	x	-
Put then invite	\$\$SEND	x	x	x	x	x	x	x	x	x	x
Release	CLOSE	x	x	x	x	x	x	x	x	x	x
Request to change direction then invite	\$\$RCD	x	x	x	-	-	-	-	x	x	x
Set timer	\$\$TIMER ⁵	x	x	x	x	x	x	x	x	x	x

¹Although the BSC 3270 subsystem is not part of SSP-ICF, its operations are listed here to show its similarities to other subsystems.

²Valid only when it is followed by a READ operation or when it follows a timer operation.

³Valid only when the DATAID and FLDLTH parameters are specified on the SESSION OCL statement, and when the HOSTNAME parameter on the SESSION statement is CICS or IMS.

⁴Valid only when a \$\$SEND operation is issued with a record length of ZERO.

⁵The timer can also be set by the TIMER intrinsic function.

⁶The record area must be at least 128 bytes long.

COBOL Operations

The following communications operations are valid in the COBOL language; each operation, however, is valid only for those subsystems containing an x in the corresponding subsystem column. (For a description of each operation, refer to the subsystem operations topic in the chapter describing your subsystem.)

COBOL SSP-ICF Operation	COBOL Operation	Communications Subsystems									
		Intra	BSCCL	CCP	CICS	IMS	3270 ¹	Finance	Peer	SNUF	APPC
Accept input ²	READ	x	x	x	x	x	x	x	x	x	x
Acquire	ACQUIRE	x	x	x	x	x	x	x	x	x	x
Cancel	\$\$CANLNI	x	-	-	-	-	-	-	-	x	-
Cancel invite	\$\$CNLINV	x	x	-	-	-	-	-	-	-	-
Cancel then invite	\$\$CANL	x	-	-	-	-	-	-	-	x	-
End of session	\$\$EOS	x	x	x	x	x	x	x	x	x	x
Evoke	\$\$EVOKNI	x	x	x	x	x	x ³	-	x	x	x
Evoke end of transaction	\$\$EVOKET	x	x	-	x	x	-	-	x	x	x
Evoke then invite	\$\$EVOK	x	x	x	x	x	x ³	-	x	x	x
Fail	\$\$FAIL	x	-	-	-	-	-	-	x	-	x
Get ²	READ	x	x	x	x	x	x	x	x	x	x
Get attributes ⁴	ACCEPT	x	x	x	x	x	x	x	x	x	x
Get status ⁶	ACCEPT	x	-	-	-	-	-	-	-	-	x
Invite ⁵	\$\$SEND	x	x	x	x	x	x	x	x	x	x
Negative response	\$\$NRSPNI	x	-	-	-	-	-	-	-	x	-
Negative response then invite	\$\$NRSP	x	-	-	-	-	-	-	-	x	-
Put	\$\$SENDNI	x	x	x	x	x	-	x	x	x	x
Put end of file/chain	\$\$SENDE	x	x	x	x	-	x	x	x	x	-
Put end of transaction	\$\$SENDET	x	x	-	x	x	-	-	x	x	x
Put FMH	\$\$SENDNF	x	-	-	-	-	-	x	-	x	-
Put FMH then invite	\$\$SENDFM	x	-	-	-	-	-	x	-	x	-
Put then invite	\$\$SEND	x	x	x	x	x	x	x	x	x	x
Release	DROP	x	x	x	x	x	x	x	x	x	x
Request to change direction then invite	\$\$RCD	x	x	x	-	-	-	-	x	x	x
Set timer	\$\$TIMER	x	x	x	x	x	x	x	x	x	x

¹Although the BSC 3270 subsystem is not part of SSP-ICF, its operations are listed here to show its similarities to other subsystems.

²The READ statement performs either a get or an accept input operation, depending on whether the TERMINAL option is specified.

³Valid only when the DATAID and FLDLTH parameters are specified on the SESSION OCL statement, and when the HOSTNAME parameter on the SESSION statement is CICS or IMS.

⁴Valid only when the ATTRIBUTE-DATA keyword is used in the SPECIAL-NAMES paragraph and the SPECIAL-NAMES name is specified in the ACCEPT statement.

⁵Valid only when a \$\$SEND operation is issued with a record length of ZERO.

⁶The record area must be at least 128 bytes long.

RPG II Operations

The following communications operations are valid in the RPG II language; each operation, however, is valid only for those subsystems containing an x in the corresponding subsystem column. (For a description of each operation, refer to the subsystem operations topic in the chapter describing your subsystem.)

RPG II SSP-ICF Operation	RPG II Operation	Communications Subsystems									
		Intra	BSC ¹	CCP	CICS	IMS	3270 ¹	Finance	Peer	SNUF	APPC
Accept input ²	READ	x	x	x	x	x	x	x	x	x	x
Aquire	ACQ	x	x	x	x	x	x	x	x	x	x
Cancel	\$\$CANLNI	x	-	-	-	-	-	-	-	x	-
Cancel invite	\$\$CNLINV	x	x	-	-	-	-	-	-	-	-
Cancel then invite	\$\$CANL	x	-	-	-	-	-	-	-	x	-
End of session	\$\$EOS	x	x	x	x	x	x	x	x	x	x
Evoke	\$\$EVOKNI	x	x	x	x	x	x ⁴	-	x	x	x
Evoke end of transaction	\$\$EVOKET	x	x	-	x	x	-	-	x	x	x
Evoke then invite	\$\$EVOK	x	x	x	x	x	x ⁴	-	x	x	x
Fail	\$\$FAIL	x	-	-	-	-	-	-	x	-	x
Get ²	READ	x	x	x	x	x	x	x	x	x	x
Invite ³	\$\$SEND	x	x	x	x	x	x	x	x	x	x
Negative response	\$\$NRSPNI	x	-	-	-	-	-	-	-	x	-
Negative response then invite	\$\$NRSP	x	-	-	-	-	-	-	-	x	-
Put	\$\$SENDNI	x	x	x	x	x	-	x	x	x	x
Put end of file/chain	\$\$SENDE	x	x	x	x	-	x	x	x	x	-
Put end of transaction	\$\$SENDET	x	x	-	x	x	-	-	x	x	x
Put FMH	\$\$SENDNF	x	-	-	-	-	-	x	-	x	-
Put FMH then invite	\$\$SENDFM	x	-	-	-	-	-	x	-	x	-
Put then invite	\$\$SEND	x	x	x	x	x	x	x	x	x	x
Release	REL	x	x	x	x	x	x	x	x	x	x
Request to change direction then invite	\$\$RCD	x	x	x	-	-	-	-	x	x	x
Set timer	\$\$TIMER	x	x	x	x	x	x	x	x	x	x

¹Although the BSC 3270 subsystem is not part of SSP-ICF, its operations are listed here to show its similarities to other subsystems.

²If a NEXT operation is executed before the READ operation, the READ operation is a get operation; otherwise, the operation is an accept input operation.

³Valid only when a \$\$SEND operation is issued with a record length of ZERO.

⁴Valid only when the DATAID and FLDLTH parameters are specified on the SESSION OCL statement, and when the HOSTNAME parameter on the SESSION statement is CICS or IMS.

Appendix B. Summary Listing of Return Codes

This appendix lists all the return codes that are returned by one or more of the SSP-ICF subsystems to indicate the results of a communications input or output operation. The abbreviated information in this appendix can help you to convert a program that uses one subsystem type so that it can use a different subsystem type for communications.

Also included in this appendix are the BSC 3270 return codes. (The BSC 3270 subsystem is part of the 3270 Device Emulation feature.) The 3270 codes are listed here to help you convert a program from an SSP-ICF subsystem to the BSC 3270 subsystem, or vice versa.

For each return code, the following information is given:

- All the subsystems for which the code is valid.
- A brief description of the results of the operation; that is, the normal or error indication. (No recovery actions are given here.)

The detailed information about each return code is given in each of the subsystem chapters for which the return code is valid. Each chapter describes the indications of the code and the normal or error recovery actions that should be taken as a result of the code. The description of the code may vary from one chapter to another, because each description is tailored to the characteristics of that subsystem.

Major Code 00 – Operation completed successfully.

General Description: The input or output operation issued by your program was completed successfully. The operation may have sent or received some data, and one of the following conditions was indicated.

Code Subsystems Affected/Primary Indication

0000 All subsystems

Change direction indication was received on successful input operation, or an output operation was successful.

0001 Intra BSCCL CCP CICS IMS Peer SNUF APPC

Successful input operation. Continue to receive.

0003 Intra Peer SNUF

End of chain indication received on successful input operation.

0004 Intra SNUF

Function management header and a change direction indication received on successful input operation.

0005 Intra SNUF

Function management header received on successful input operation.

0007 SNUF only

Function management header and an end of chain indication received on successful input operation.

0008 Intra BSCCL IMS SNUF APPC

End of transaction indication received on successful input operation.

000C Intra SNUF

Function management header and an end of transaction indication received on successful input operation.

0010 Intra BSCEL CCP CICS Peer SNUF APPC

Request to change direction received on successful output operation.

0012 3270 only

One or more types of unsupported 3270 orders received on successful input operation.

0014 Intra APPC

Remote process requested confirmation and allowed the local application to begin sending data.

0015 Intra APPC

Remote process requested confirmation. The local application program should continue to receive data.

001C Intra APPC

Remote process requested confirmation and that the transaction be ended.

0020 BSCEL CCP 3270 SNUF

System message and an end of transmission (or change direction) indication received on successful input operation.

0021 BSCEL CICS SNUF

System message received on successful input operation. Continue to receive.

0024 APPC only

Acquire operation was successful, a remotely controlled session was allocated to the local application program.

0025 SNUF only

Function management header received with a host system message.

0028 Intra BSCEL IMS Peer SNUF

System message and an end of transaction indication received on successful input operation.

0030 BSCEL 3270 SNUF

Truncated system message and end of transmission (or change direction) indication received on successful input operation.

0031 BSCEL CICS

Truncated system message received on successful input operation. Continue to receive.

0038 Intra BSCEL IMS Peer SNUF

Truncated system message and an end of transaction indication received on successful input operation.

Major Code 01 – Successful operation with a new requester.

General Description: The input or output operation issued by your program was completed successfully. The operation may have either sent data to a new requester or received data from a new requester.

Code Subsystems Affected/Primary Indication

0100 Intra BSCCL CCP IMS Finance Peer SNUF APPC

Change direction indication received on successful input operation with a new requester, or output operation with a new requester was successful.

0101 Intra BSCCL CCP CICS IMS Peer SNUF APPC

Successful input operation with a new requester. Continue to receive.

0103 Peer SNUF

End of chain indication received on successful input operation with a new requester.

0104 Intra SNUF

Function management header and a change direction indication received on successful input operation with a new requester.

0105 Intra SNUF

Function management header received on successful input operation with a new requester.

0107 SNUF only

Function management header and an end of chain indication received on successful input operation with a new requester.

0108 Intra BSCEL IMS Peer SNUF APPC

End of transaction indication received on successful input operation with a new requester.

010C Intra SNUF

Function management header and an end of transaction indication received on successful input operation with a new requester.

0110 APPC only

Remote process requested that data be sent.

0114 Intra APPC

Remote process requested confirmation and allowed the local application program to begin sending data.

0115 Intra APPC

Remote process requested confirmation. The local application program should continue to receive data.

0118 Intra BSCEL IMS Peer SNUF APPC

End of transaction indication received on successful input operation with a new requester.

011C Intra APPC

Remote process requested that a confirmation indication be sent and that the transaction be ended.

Major Code 02 – Input operation completed successfully, but a stop system request or a disable subsystem request is pending.

General Description: The input operation issued by your program was completed successfully. The operation may have received some data or a remote system message. However, because a stop system request or a disable subsystem request is pending, no new sessions using the subsystem can be started.

Code Subsystems Affected/Primary Indication

0200 Intra Finance Peer SNUF APPC

Change direction indication received with a stop system or disable subsystem indication on successful input operation.

0201 Intra BSCEL CCP CICS IMS Peer SNUF APPC

A stop system or disable subsystem indication received on a successful input operation. Continue to receive.

0203 Intra Peer SNUF

End of chain indication received with a stop system or disable subsystem indication on successful input operation.

0204 Intra SNUF

Function management header and a change direction indication received with a stop system or disable subsystem indication on successful input operation.

0205 Intra SNUF

Function management header received with a stop system or disable subsystem indication on successful input operation.

0207 SNUF only

Function management header and an end of chain indication received with a stop system or disable subsystem indication on successful input operation.

0208 Intra BSCCL IMS SNUF APPC

End of transaction indication received with a stop system or disable subsystem indication on successful input operation.

020C Intra SNUF

Function management header and an end of transaction indication received with a stop system or disable subsystem indication on successful input operation.

0210 APPC only

Remote process requested that data be sent.

0212 3270 only

One or more types of unsupported 3270 orders received with a stop system or disable subsystem indication on successful input operation.

0214 Intra APPC

Remote process requested confirmation and allowed the local application program to begin sending data.

0215 Intra APPC

Remote process requested confirmation. The local application program should continue to receive data.

021C Intra APPC

Remote process requested that a confirmation indication be sent and that the transaction be ended.

0220 BSCCL CCP 3270 SNUF

System message and an end of transmission (or change direction) indication received with a stop system or disable subsystem indication on successful input operation.

0221 BSCEL CICS SNUF

System message received with a stop system or disable subsystem indication on successful input operation. Continue to receive.

0228 Intra BSCEL IMS Peer SNUF

System message and an end of transaction indication received with a stop system or disable subsystem indication on successful input operation.

0230 BSCEL CCP 3270 SNUF

Truncated system message and end of transmission (or change direction) indication received with a stop system or disable subsystem indication on successful input operation.

0231 BSCEL CICS

Truncated system message received with a stop system or disable subsystem indication on successful input operation. Continue to receive.

0238 Intra BSCEL IMS Peer SNUF

Truncated system message and an end of transaction indication received with a stop system or disable subsystem indication on successful input operation.

Major Code 03 – Successful operation, but no data received.

General Description: The input operation or set timer output operation just performed was completed successfully, but no data was sent or received.

Code Subsystems Affected/Primary Indication

0300 Intra BSCEL CCP CICS IMS Peer SNUF APPC

Change direction or end of transmission indication received with no data on successful input operation.

0301 Intra BSCEL Peer SNUF APPC

No data received on successful input operation. Continue to receive.

0302 Intra Peer

Fail indication with no data was received on successful input operation.

0303 Intra Peer SNUF

End of chain indication received with no data on successful input operation.

0308 Intra BSCEL CCP CICS IMS Peer SNUF APPC

End of transaction indication received with no data on successful input operation.

0310 All subsystems

Timer interval (specified by a set timer operation) has expired.

0314 Intra APPC

Remote process requested confirmation and allowed the local application program to begin sending data.

0315 Intra APPC

Remote process requested confirmation. The local application program should continue to receive data.

031C Intra APPC

Remote process requested that a confirmation indication be sent and that the transaction be ended.

Major Code 04 – Output exception occurred.

General Description: An output exception occurred because your program attempted to send output when it should have been receiving the output already sent by the other (remote) program.

Code Subsystems Affected/Primary Indication

0402 Intra only

Fail operation issued by other program.

0411 Intra BSCEL CCP IMS

Message for your program waiting to be received.

0412 All subsystems

Data for your program waiting to be received.

Major Code 08-34 – Miscellaneous program errors.

Error Description: An operation attempted by your program failed. The error may have occurred because an operation was issued at the wrong time or because a data record was too long.

Code Subsystems Affected/Primary Indication

0800 All subsystems

Acquire operation attempted to acquire an active session.

1100 All subsystems

Accept operation was not successful.

2800 All subsystems

Communications was attempted with a session that has been released.

3401 All subsystems

Input operation rejected because data to be received was too long for your program's input buffer.

3431 APPC only

An input exception occurred because the program received data that exceeded its maximum record length. The local application program should continue to receive data.

Major Code 80 – Permanent (nonrecoverable) subsystem error.

Error Description: A nonrecoverable error has occurred in the subsystem; the subsystem has been (or is being) disabled, and your session has been terminated.

Code Subsystems Affected/Primary Indication

8081 All subsystems

An SSP-ICF error caused the abnormal termination of the subsystem.

8082 All subsystems

The subsystem controlling the session is being disabled immediately.

8083 CCP IMS 3270 Finance Peer

Communications adapter controller check occurred on an output operation.

8084 CCP IMS 3270 Finance Peer

Communications adapter controller check occurred on an input operation.

80BD CCP CICS

A connection was attempted on an inactive X.21 communications line.

80C0 APPC only

Session failed.

80C1 APPC only

Evoke operation failed. Mapped conversation not allowed.

80D0 APPC only

Remote transaction program not available. No retry allowed.

Major Code 81 – Permanent (nonrecoverable) session error.

Error Description: A nonrecoverable error has occurred in the session; the session has been terminated.

Code Subsystems Affected/Primary Indication

8136 BSCEL only

Invalid remote identifier received in response to your output operation.

8137 BSCEL only

Invalid remote identifier received in response to your input operation.

8183 BSCEL CCP IMS Finance Peer

Communications adapter controller check occurred on an output operation.

8184 BSCEL CCP IMS Finance Peer

Communications adapter controller check occurred on an input operation.

8185 BSCEL Peer

Attempt to automatically call a remote location was not successful; all available numbers were tried.

8186 BSCEL only

Attempt to automatically call a remote location was not successful; all numbers were already marked as called.

8187 Finance only

Your program was not synchronized with the remote program.

8191 BSCEL CCP SNUF APPC

Permanent line error occurred on an output operation.

8192 BSCEL CCP

Permanent line error occurred on an input operation.

8193 BSCEL CCP CICS

Disconnect indication received on an output operation.

8194 BSCEL CCP CICS

Disconnect indication received on an input operation.

8196 Peer SNUF APPC

SNA unbind command received from remote system.

8197 BSCEL APPC

Remote system abnormally terminated the transaction on an output operation.

8198 BSCEL only

Remote system abnormally terminated the transaction on an input operation.

8199 BSCEL CCP

Time between successive data blocks sent on output operations exceeded specified wait time.

819A BSCEL CCP

Time between successive data blocks received on input operations exceeded specified wait time.

819B BSCCEL only

In a put-versus-put situation, a block size error in the remote program's data was detected on an output operation.

819C BSCCEL CCP SNUF

A block size error in the remote program's data was detected on an input operation.

819D BSCCEL Peer SNUF APPC

Unexpected data was received from the remote system.

819E CCP only

Abnormal shutdown indication received from the CCP host system.

819F CCP only

Normal shutdown indication received from the CCP host system.

81A3 Finance only

Synchronization was lost between the subsystem and the remote system communications equipment.

81B6 SNUF only

The host system has quiesced the session; the session has been terminated.

81B8 BSCCEL only

In a put-versus-put situation, a record that exceeds the maximum user record length was received from the remote system on an output operation.

81B9 BSCCEL only

A record that exceeds the maximum user record length was received on an input operation.

81BA Finance only

A record longer than 512 bytes was received on an input operation.

81BC BSCEL CICS

Wrong type of list was used to automatically call a remote location.

81C2 APPC only

Operation failed. Session no longer available.

81C4 APPC only

Session group not active.

81C5 APPC only

Deallocate ABEND(SVC) received.

81C6 APPC only

Deallocate ABEND(TIMER) received.

Major Code 82 – Acquire operation failed.

Error Description: An attempt to acquire a session was not successful; the session was not started.

Code Subsystems Affected/Primary Indication

820A BSCCL only

Invalid combination of values detected during an acquire operation; both ASCII and TRANSP-YES were specified.

820D IMS only

Normal shutdown indication received from the IMS host system.

8213 IMS only

For the maximum message amount specified (or defaulted) in the SESSION statement, not enough buffer space was available during an acquire operation.

821E Finance only

No SESSION statement found between LOAD and RUN statements for this BASIC program.

8233 All subsystems

Session identifier is either missing or invalid.

8236 BSCCL CCP

Invalid remote identifier received from the remote system or device.

8281 All subsystems

An SSP-ICF error caused the abnormal termination of the subsystem.

- 8282** All subsystems
The subsystem in which the acquire operation was attempted is currently being disabled.
- 8283** BSCEL CCP
Communications adapter controller check occurred during an acquire operation.
- 8285** BSCEL CCP Peer SNUF APPC
Attempt to automatically call a remote location and acquire the session was not successful; all available numbers were tried.
- 8286** BSCEL CCP
Attempt to automatically call a remote location and acquire the session was not successful; all numbers were already marked as called.
- 8288** Finance only
The specified remote location was not active when the acquire operation was attempted.
- 8289** BSCEL only
Invalid combination of values detected during an acquire operation; a record separator and transparent mode were both specified.
- 828A** BSCEL only
Invalid combination of values detected during an acquire operation; a record separator and ITB mode were both specified.
- 828B** BSCEL only
Invalid combination of values detected during an acquire operation; the maximum user record length specified was greater than the block length.
- 828C** BSCEL only
Invalid combination of values detected during an acquire operation; 3740 multiple files and ITB mode were both specified.

- 828D** BSCEL only
Invalid combination of values detected during an acquire operation; blank compression and ITB mode were both specified.
- 828E** BSCEL only
Invalid combination of values detected during an acquire operation; blank truncation and ITB mode were both specified.
- 828F** BSCEL only
Invalid block length specified; for the other parameters specified, block length must not be zero.
- 8290** BSCEL only
Invalid combination of values detected during an acquire operation; blank compression and transparent mode were both specified.
- 8291** BSCEL CCP
Permanent line error occurred during an acquire operation.
- 8293** BSCEL CCP
Disconnect indication received from remote system during an acquire operation.
- 8296** SNUF only
SNA unbind command received from remote system during an acquire operation.
- 8297** BSCEL only
Remote system abnormally terminated the attempted acquire operation.

- 829F** CCP only
Normal shutdown indication received from the CCP host system during an acquire operation.
- 82A0** BSCEL only
Invalid record separator character was specified in the SESSION statement.
- 82A1** SNUF only
Log-on portion of acquire operation failed; either the host system was inactive, or an invalid remote program name was specified in the APPLID parameter.
- 82A2** CCP only
Invalid CCP sign-on password was specified during an acquire operation.
- 82A5** SNUF only
Invalid combination of values detected during an acquire operation; *yes* was specified both for MSGPROT and BATCH parameters.
- 82A6** Peer SNUF APPC
SNA bind command failed.
- 82A7** BSCEL CCP CICS Peer SNUF APPC
The specified communications line was already in use when the acquire operation was attempted.
- 82A8** All subsystems
The maximum number of active sessions allowed in System/36 were already active when the acquire operation was attempted.

82A9 IMS only

IMS host system was not active when the acquire operation was attempted.

82AA All subsystems

Specified subsystem was not enabled when the acquire operation was attempted. For SNA subsystems, the subsystem may be enabled, but communications with the specified remote location has not been activated.

82AB All subsystems

Specified subsystem was being enabled when the acquire operation was attempted. For SNA subsystems, the subsystem may have been enabled; however, communications is being activated with the specified remote location.

82AC IMS only

Batch session could not be acquired because the line is currently being used by interactive sessions.

82AD IMS only

Interactive session or another batch session could not be acquired because the line is currently being used by a batch session.

82AE IMS 3270

Invalid PTERM address detected during an acquire operation; address in SESSION statement is not specified in subsystem configuration.

82AF IMS only

Incorrect PTERM address detected during an acquire operation; a remote address reserved for procedure start requests from the IMS host system was specified.

82B0 All subsystems

Specified subsystem was being disabled, or a disable subsystem request was pending, when the acquire operation was attempted. For SNA subsystems, if the subsystem is not being disabled, communications with the specified remote location is being deactivated.

82B1 CCP CICS IMS 3270

The session to be acquired was identified as already being in use.

82B2 CCP CICS

All the sessions in the address pool were already in use when the acquire operation was attempted.

82B3 IMS 3270 Peer SNUF APPC

All the sessions specified in the subsystem configuration were already in use when the acquire operation was attempted.

82BB Finance SNUF

Logical work station specified by the LWSID parameter was not available when the acquire operation was attempted.

82BC BSCEL CCP

Wrong type of list was used to automatically call a remote location during the acquire operation.

82C3 APPC only

Session group name was not found.

82C4 APPC only

Session group name is not active.

Major Code 83 – Session error occurred.

Error Description: An error has occurred in the session, but the session is still active. Recovery might be possible.

Code Subsystems Affected/Primary Indication

830B All subsystems

An input or output operation was attempted in an unacquired session.

830D IMS SNUF

Normal shutdown indication received from host system.

830E IMS only

Normal shutdown indication received from IMS host system on an input operation.

8311 SNUF only

Output operation was attempted while a message was waiting to be received.

8313 IMS only

Insufficient task work space available on an output operation.

8316 Peer APPC

Evoke operation failed because evoked program was not found.

8317 IMS only

Maximum user record length was exceeded on an output operation.

8319 Intra SNUF

Negative response to your program's previous output operation was issued by the remote program.

831A Intra BSCCL CCP IMS Peer

Evoke operation failed to complete successfully, or the evoked program terminated abnormally.

831B Intra SNUF

Invalid sense data included on a negative response operation issued by your program.

831C All subsystems

Operation is invalid at this time; a return code for the previous output operation was improperly handled by your program.

831D IMS only

Output operation invalid in this session because it was started by a remote procedure start request.

831E All subsystems

Invalid operation code, or operation not supported by the subsystem.

831F All subsystems

Attempted to send a data record whose length exceeds the maximum user record length specified for this session.

8320 IMS only

On a batch output operation, your program attempted to send a record longer than 256 bytes to the IMS host system.

8322 Intra BSCCL CCP IMS Finance Peer SNUF APPC

The attempted operation is not valid after a put with no invite operation.

8323 Intra SNUF

Cancel operation invalid while in receive state, or output operation attempted after receiving a fail indication.

- 8324** SNUF only
Function management header sent by your program at wrong time; valid only with the *first* record in the chain.
- 8326** Intra SNUF
Invalid cancel operation was issued; valid only within a chain.
- 8327** Intra BSCCEL IMS Peer SNUF APPC
Invalid input or output operation issued when no transaction existed.
- 8329** Intra BSCCEL CCP CICS IMS Finance Peer SNUF APPC
Evoke operation not valid in this session; your program was evoked by a remote procedure start request.
- 832A** Intra Peer
Both programs attempted to receive input.
- 832B** BSCCEL CCP CICS IMS
Record length of 0 detected on first output operation attempted.
- 832C** All subsystems
Release operation invalid after an invite operation.
- 832D** All subsystems
Attempted operation is not valid following an invite operation.
- 832E** CCP only
Program cancel indication received from the CCP host system when no transaction was active.

- 832F** Intra BSCCL CCP 3270 Peer SNUF APPC
Invalid evoke or release operation was issued before a transaction was completed.
- 8330** Intra SNUF
Cancel indication and change direction indication received on an input operation.
- 8331** Intra SNUF
Cancel indication received *without* change direction indication on an input operation.
- 8332** SNUF only
Cancel indication and end of transaction indication received on an input operation.
- 8333** All subsystems
Invalid session identifier detected on input or output operation.
- 8334** BSCCL IMS 3270 Peer APPC
Procedure name missing on an evoke operation issued by your program.
- 8336** CICS only
Invalid remote identifier received in response to your output operation.
- 8338** 3270 only
Invalid or unsupported 3270 command received from the remote system.
- 8339** 3270 only
Data rejected by subsystem because it was still processing the last operation.

- 833C** BSCCL only
- Invalid combination of data modes detected on output operation; transparency mode and ITB mode were both specified.
- 8383** CCP CICS 3270
- Communications adapter controller check occurred on an output operation.
- 8384** CCP CICS 3270
- Communications adapter controller check occurred on an input operation.
- 8385** CICS only
- Attempt to automatically call a remote location was not successful; all available numbers were tried.
- 8386** CICS only
- Attempt to automatically call a remote location was not successful; all numbers were already marked as called.
- 8391** CICS IMS 3270
- Permanent line error occurred on an output operation.
- 8392** CICS IMS 3270
- Permanent line error occurred on an input operation.
- 8397** CICS 3270 Peer
- Remote system abnormally terminated the transaction during an output operation.
- 8398** CICS 3270
- Remote system abnormally terminated the transaction during an input operation.
- 8399** CICS 3270
- Delay count exceeded at this location on an output operation.

839A CICS 3270

Delay count exceeded at this location on an input operation.

839B CICS SNUF

In a put-versus-put situation, a block size error in the remote program's data was detected on an output operation.

839C CICS 3270

The remote program's data block length exceeded the subsystem input buffer length on an input operation.

83A7 CCP only

Attempted operation not performed because specified communications line was already in use.

83B0 Peer APPC

Operation was not successful because subsystem is being disabled.

83C7 APPC only

Fail operation received. No data truncated.

83C8 APPC only

Fail operation received. No data truncated.

83C9 APPC only

Fail operation received. Data being purged.

83CA APPC only

Fail operation received. Data being purged.

83CB APPC only

Fail operation received. Data truncated.

83CC APPC only

Fail operation received. Data truncated.

83CD APPC only

Operation invalid. Confirmation request not allowed.

83CE APPC only

Security invalid. Request rejected by remote system.

83CF APPC only

Conversation types mismatched. Request rejected by remote system.

83D0 APPC only

Remote transaction program not available.

83D1 APPC only

Process initialization parameters (PIPs) not allowed. Request rejected by remote system.

83D2 APPC only

Process initialization parameters (PIPs) not specified correctly. Request rejected by remote system.

83D3 APPC only

Synchronization level not supported by remote program.

83D4 APPC only

Maximum input length exceeded.

83E0 Intra APPC

Externally described format definition not found.

83E1 Intra APPC

Externally described format definition not retrieved.

83E8 Intra BSCCL

Invalid cancel invite operation.

Appendix C. Conversion Considerations

BSCCL to Peer Conversion Considerations	C-3
CICS to SNUF Conversion Considerations	C-3
IMS/VS to SNUF Conversion Considerations	C-4
Uninvited Data	C-4
End of Transaction Operations	C-5
BATCH Parameter on the SESSION Statement	C-6
IMS/IRSS Subsystem	C-6
SNUF Subsystem	C-6
IMS/IRSS to SNUF Conversion Considerations	C-7
Peer to APPC Conversion Considerations	C-8
SNUF to APPC Conversion Considerations	C-9

This appendix describes several considerations for converting a program written for a BSC subsystem so that it can be used with an SNA subsystem. It also describes considerations for converting a program written for a Peer or SNUF subsystem so that it can be used with an APPC subsystem. The following conversions are described:

- BSC to Peer
- CICS to SNUF (CICS/VS on remote system)
- IMS/IRSS to SNUF (IMS/VS on remote system)
- Peer to APPC
- SNUF to APPC

BSCEL to Peer Conversion Considerations

A System/36 program that uses the BSCEL subsystem can be changed so it can use the Peer subsystem. The following items describe considerations for converting the program to use the Peer subsystem.

- Configuration parameters and SESSION statement parameters for the Peer subsystem are different from those for the BSCEL subsystem. See the BSCEL and Peer chapters in the *SSP-ICF Base Subsystems Reference* manual for a description of each set of configuration displays and each SESSION statement.
- The program using the BSCEL subsystem should be running with a partner attribute of NORM.
- The cancel invite operation supported by the BSCEL subsystem is *not* supported by the Peer subsystem.
- The put end of *file* operation performed by the BSCEL subsystem is functionally the same as the put end of *chain* operation performed by the Peer subsystem.
- Minor return codes returned by the BSCEL subsystem might be different from those returned by the Peer subsystem. If the program checks minor return codes, changes might be required to handle the Peer minor return codes. If the program checks only major return codes, no changes in return code checking are required.
- A System/36 program using the BSCEL subsystem must be synchronized with the program on the remote system. That is, in situations where either program can perform input or output operations, one program must perform input and the other must perform output. A program using the Peer subsystem can achieve synchronization with the remote program by checking the return codes. This added flexibility causes the Peer subsystem to return different minor return codes than those returned by the BSCEL subsystem. Appendix B contains a summary listing that identifies the return codes that are used by each subsystem.

CICS to SNUF Conversion Considerations

System/36 programs that use the CICS subsystem can be run without change on the SNUF subsystem if no minor return codes are checked. The minor return codes issued by the two subsystems differ somewhat, especially in indicating the end of a transaction. Therefore, coding changes might be required if minor return codes are checked.

Another difference between the two subsystems is that the CICS subsystem allows only the *EXEX procedure start request; the SNUF subsystem allows either *EXEX or *EXEC.

IMS/VS to SNUF Conversion Considerations

Consider the following items when writing interactive communications programs that communicate with IMS/VS:

- Uninvited data
- End of transaction operations
- BATCH parameter on the SESSION statement

These items must be considered whether you use the SNUF subsystem or the IMS/IRSS subsystem; however, the considerations are different. The following sections compare the two subsystems in these areas. The final section describes the considerations for writing a program that uses the IMS/IRSS subsystem so that the program can be run using the SNUF subsystem with as little change as possible.

Uninvited Data

Uninvited data is data received in a session that has no program associated with it. (Data that begins with *EXEC or *EXEX is a procedure start request and is not considered uninvited data.) A procedure can be specified (during the configuration of the subsystem) so that it begins running when uninvited data is received. The subsystem passes the data to a program started by the procedure; the program can process the data, save it in a file, or discard it.

Uninvited data can be received for any of the following reasons:

- A message switch or broadcast message is received.
- A System/36 program releases a session that has data on the IMS/VS output queue.
- An IMS/VS status message is sent to a session.

The program that receives the uninvited data cannot perform any output operations in the session. The System/36 program is treated as a remotely started program. If the program is using the SNUF subsystem, the program must receive 256-byte records and should be prepared to handle function management headers.

End of Transaction Operations

End of transaction operations (put end of transaction or evoke end of transaction) are treated differently by the IMS/IRSS subsystem than they are by the SNUF subsystem.

When an end of transaction operation is sent to the IMS/IRSS subsystem, the subsystem indicates to IRSS that the message has ended.

When an end of transaction operation is sent to the SNUF subsystem, the subsystem sends a change direction indication and expects to receive a null record, terminating the transaction from IMS/VS. If anything other than a null record is received, the session is terminated abnormally, and an error return code is sent to the System/36 program. Something other than a null record can be received in any of the following cases:

- The IMS/VS remote program has generated output to be sent to this session.
- An output message was placed on the IMS/VS output queue by an IMS/VS program other than the one evoked by the System/36 program.
- A message switch or broadcast message was placed on the IMS/VS output queue for this session.

Because of these situations, end of transaction operations are not recommended for SNUF sessions that communicate with IMS/VS.

BATCH Parameter on the SESSION Statement

The BATCH parameter on the SESSION statement determines how the SNUF and IMS/IRSS subsystems handle input and output operations.

IMS/IRSS Subsystem

When BATCH-NO is specified on the SESSION statement, the IMS/IRSS subsystem accumulates records as they are received from the System/36 program. The records are then sent as one message when the program indicates the message is complete (by issuing either an end of transaction operation or an input operation). The total length of all the records submitted within a message cannot be greater than the specified maximum record length.

When BATCH-YES is specified on the SESSION statement, the IMS/IRSS subsystem sends each record as it is received from the System/36 program. The subsystem sends each record as a segment of a message until it receives an end of transaction operation or an input operation from the program. A session specified as BATCH-YES cannot be acquired while another session is active because of the delays in line turnaround that may occur. BATCH-YES should be specified when large amounts of data must be sent without intervening responses from the host system.

SNUF Subsystem

When BATCH-NO is specified on the SESSION statement, the SNUF subsystem handles each record from the System/36 program as a complete chain. IMS/VS requires that when a chain ends, a change direction indication must also be sent. This means that the System/36 program cannot issue multiple output operations consecutively. To assure that the change direction indication is sent, each put or evoke operation must be accompanied by an invite, get, or end of transaction modifier (that is, the operation must be a *combined* operation, such as a *put then get* operation). Note that an evoke operation without a modifier that is followed by a get or invite operation is not acceptable because the change direction indication would not accompany the evoke operation, and IMS/VS would, therefore, reject it.

When BATCH-YES is specified on the SESSION statement, the SNUF subsystem sends each record from the System/36 program as an element of a chain. This allows the program to perform consecutive output operations; however, a change of direction must still be indicated at the end of each chain. Therefore, a chain should not be terminated by a put end of chain operation. The last output operation should include an invite, get, or end of transaction modifier, or the output operation should be followed by an invite or get operation.

IMS/IRSS to SNUF Conversion Considerations

If the following guidelines are followed, programs that communicate with IMS/VS through the IMS/IRSS subsystem can be converted to use the SNUF subsystem:

- If more than one output operation is performed consecutively, the BATCH-YES parameter must be specified on the SESSION statement.
- If a get operation is used (one that is not combined with an evoke or put operation), the BATCH-YES parameter must be specified on the SESSION statement.
- An end of transaction operation (put end of transaction or evoke end of transaction) used with the SNUF subsystem fails when output remains on the IMS/VS output queue for this session. The operation will not fail when using the IMS/IRSS subsystem.
- Return codes for permanent errors (80xx or greater, or *STATUS values greater than 99) returned by the SNUF subsystem are different from those returned by the IMS/IRSS subsystem.

Peer to APPC Conversion Considerations

A System/36 program that uses the Peer subsystem can be changed so it can use the APPC subsystem. The following paragraphs describe considerations for converting the program to use the APPC subsystem.

Configuration parameters and SESSION statement parameters for the APPC subsystem are different from those for the Peer subsystem. See the Peer and APPC chapters in the *SSP-ICF Base Subsystems Reference* manual for a description of each set of configuration displays and each SESSION statement.

A put with end of chain operation is not supported in the APPC subsystem. In addition, the return code indicating an end of chain was received is not supported.

The application may receive a return code indicating an end of transaction was received with the data record. An end of transaction indication, when using the Peer subsystem, does not get returned to the application with the data record.

The notification of an evoke failure may not occur until a subsequent operation. With the Peer subsystem, an evoke operation does not complete until an acknowledgement is received from the remote Peer subsystem. With the APPC subsystem, a confirm indicator can be included with the evoke operation to verify that the evoke operation completed.

An evoke operation may fail because a remotely controlled session was lost. To avoid this, specify N on the acquire remotely controlled sessions prompt (display 44.0) when defining your configuration.

Minor return codes returned by the Peer subsystem may be different from those returned by the APPC subsystem. If the program checks minor return codes, changes might be required to handle the APPC minor return codes. Return codes for fail errors returned by the Peer subsystem are different from those returned by the APPC subsystem. Appendix B contains a summary listing that identifies the return codes that are used by each subsystem.

Externally defined field, format, and file definitions (also called data definitions) can be used instead of SSP-ICF operations. This allows you to use the new functions of APPC without converting the entire program.

Security considerations are described in the manual *Using System/36 Communications*.

SNUF to APPC Conversion Considerations

A System/36 program that uses the SNUF subsystem can be changed so it can use the APPC subsystem. The following paragraphs describe considerations for converting the program to use the APPC subsystem.

Configuration parameters and SESSION statement parameters for the APPC subsystem are different from those for the SNUF subsystem. See the SNUF and APPC chapters in the appropriate reference manual for a description of each set of configuration displays and each SESSION statement.

The APPC subsystem does not communicate with IMS.

The APPC subsystem has a maximum of one LU (logical unit) with one session when communicating with CICS/VS.

No cancel, negative response, and put FMH operations are supported by the APPC subsystem.

Minor return codes returned by the SNUF subsystem may be different from those returned by the APPC subsystem. If the program checks minor return codes, changes might be required to handle the APPC minor return codes. Return codes for fail errors (03xx or 04xx) returned by the SNUF subsystem are different from those returned by the APPC subsystem. Appendix B contains a summary listing that identifies the return codes that are used by each subsystem.

Externally defined field, format, and file definitions (also called data definitions) can be used instead of SSP-ICF operations. This allows you to use the new functions of APPC without converting the entire program.

Appendix D. EBCDIC and ASCII Character Sets


EBCDIC/DP (Data Processing) Character Set D-2
EBCDIC/WP (Word Processing) Character Set D-3
ASCII Character Set D-4

The following charts show all the EBCDIC and ASCII character sets. The charts are provided to show the data link control characters that are used in BSC data communications. Certain parameters for some BSC subsystems should not specify values that contain any of the data link control characters shown in these charts. Those parameters are identified in the individual subsystem chapters.

EBCDIC/DP (Data Processing) Character Set

First Hexadecimal Digit
↓
Second Hexadecimal Digit
↓

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	PF	HT	LC	DEL		RLF	SMM	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	RES	NL	BS	IL	CAN	EM	CC		IFS	IGS	IRS	IUS
2	DS	SOS	FS		BYP	LF	EOB ETB	PRE ESC			SM			ENQ	ACK	BEL
3			SYN		PN	RS	UC	EOT					DC4	NAK		SUB
4	SP										¢	.	<	(+	
5	&										!	\$	*)	;	⌋
6	-	/									!	,	%	-	>	?
7										\	:	#	@	"	=	"
8		a	b	c	d	e	f	g	h	i						
9		j	k	l	m	n	o	p	q	r						
A		~	s	t	u	v	w	x	y	z						
B																
C	{	A	B	C	D	E	F	G	H	I						
D	}	J	K	L	M	N	O	P	Q	R						
E	\		S	T	U	V	W	X	Y	Z						
F	0	1	2	3	4	5	6	7	8	9						

 Duplicate Assignment

ACK 0 = 1070
 ACK 1 = 1061
 TTD = 022D
 RVI = 107C
 WACK = 106B

EBCDIC/WP (Word Processing) Character Set

First Hexadecimal Digit
Second Hexadecimal Digit

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX		HT	RNL	DEL		SPS	RPT	VT	FF	CR		
1	DLE	DC1	DC2	DC3		NL	BS				UBS			IGS	IRS	ITB
2				WUS		LF	ETB	ESC			SW	CU2		ENQ		BEL
3			SYN	IRT			NBS	EOT	SBS	IT	RFF			NAK		SUB
4	SP	RSP									¢	.	¢ <	(+	!
5	&									↓	!	\$	*)	;	¢
6	Req	/										,	%	-	¢ >	?
7									± \		:	#	@	'	=	"
8		a	b	c	d	e	f	g	h	i						
9		j	k	l	m	n	o	p	q	r						
A		~	s	t	u	v	w	x	y	z						
B																
C	2 {	A	B	C	D	E	F	G	H	I	DHY					
D	3 }	J	K	L	M	N	O	P	Q	R						
E	1/4	NSP	S	T	U	V	W	X	Y	Z						
F	0	1	2	3	4	5	6	7	8	9						

 Duplicate Assignment

ACK 0 = 1070
 ACK 1 = 1061
 TTD = 022D
 RVI = 107C
 WACK = 106B

ASCII Character Set

First Hexadecimal Digit

Second Hexadecimal Digit

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	! or	"	#	\$	%	&	'	()	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^ or 7	_
6	'	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

Glossary

#LIBRARY. The library, provided with the system, that contains the System Support Program Product. See *system library*.

abnormal termination. A system failure or operator action that causes a job to end unsuccessfully.

access method. The way that records in files are referred to by the system. The reference can be consecutive (records are referred to one after another in the order in which they appear in the file), or it can be random (the individual records can be referred to in any order).

acknowledgment character (ACK). In binary synchronous communications, a transmission control character sent as a positive response to a data transmission.

ACK0. In binary synchronous communications, the even-numbered positive acknowledgment character. See *acknowledgment character (ACK)*.

ACK1. In binary synchronous communications, the odd-numbered positive acknowledgment character. See *acknowledgment character (ACK)*.

acquire. To assign a display station or session to a program.

acquired session. A session that has been started by a System/36 program using an acquire operation, or in BASIC, using an OPEN statement.

adapter. See *communications adapter*.

address pool. In data communications, a collection of multipoint addresses. Each address can be associated with an individual SSP-ICF session.

addressing. (1) In data communications, the way that the sending or control station selects the station to which it is sending data. (2) A means of identifying storage locations.

Advanced Peer-to-Peer Networking (APPN). A communications feature that routes data in a network between two or more APPC systems that are not directly attached.

advanced program-to-program communications (APPC). Communications support that allows System/36 to communicate with other systems having the same support. APPC is the way that System/36 puts the IBM SNA LU-6.2 protocol into effect.

alert. An error message sent to the system services control point (SSCP) at a host system. On System/36, the problem management portion of the Communications and Systems Management feature is used to generate and send alerts.

allocate. To assign a resource, such as a disk file or a diskette file, to perform a specific task.

alphabetic character. Any one of the letters A through Z (uppercase and lowercase). Some program products extend the alphabet to include the special characters #, \$, and @.

alphameric. Consisting of letters, numbers, and often other symbols, such as punctuation marks and mathematical symbols.

alphanumeric. See *alphameric*.

American National Standard Code for Information Interchange (ASCII). The code developed by ANSI for information interchange among data processing systems, data communications systems, and associated equipment. The ASCII character set consists of 7-bit control characters and symbolic characters.

APAR. See *authorized program analysis report (APAR)*.

APPC. See *advanced program-to-program communications (APPC)*.

application. (1) A particular business task, such as inventory control or accounts receivable. (2) A group of related programs that apply to a particular business area, such as the Inventory Control or the Accounts Receivable application.

application program. A program used to perform an application or part of an application.

APPN. See *Advanced Peer-to-Peer Networking*.

ASCII. See *American National Standard Code for Information Interchange (ASCII)*.

assembler. A program that converts assembler language statements to machine instructions.

assembler instruction statement. A statement that controls what the assembler does, rather than what the user program does.

assembler language. A symbolic programming language in which the set of instructions includes the instructions of the machine and whose data structures correspond directly to the storage and registers of the machine.

asynchronous transmission. In data communications, a method of transmission in which the bits included in a character or block of characters occur during a specific time interval. However, the start of each character or block of characters can occur at any time during this interval. Contrast with *synchronous transmission*.

attribute. A characteristic. For example, an attribute for a displayed field could be blinking.

authorized program analysis report (APAR). A request for correction of a defect in a current release of an IBM-supplied program.

autoanswer. In data communications, the ability of a station to receive a call over a switched line without operator action. Contrast with *manual answer*.

autocall. In data communications, the ability of a station to place a call over a switched line without operator action. Contrast with *manual call*.

autocall unit. A common carrier device that allows System/36 to automatically call a remote location.

automatic reconnect. An option specified during system configuration that allows a remote work station controller to be reconnected automatically on a switched or nonswitched line.

BASIC (beginner's all-purpose symbolic instruction code). A programming language designed for interactive systems and originally developed at Dartmouth College to encourage people to use computers for simple problem-solving operations.

basic data exchange. A file format for exchanging data on diskettes between systems or devices.

batch. Pertaining to activity involving little or no operator action. Contrast with *interactive*.

batch BSC. The System Support Program Product support that provides data communications with BSC computers and devices via the RPG T specification or the assembler \$DTFB macroinstruction.

batch processing. A processing method in which a program or programs process records with little or no operator action. Contrast with *interactive processing*.

binary synchronous communications (BSC). A form of communications line control that uses transmission control characters to control the transfer of data over a communications line. Compare with *synchronous data link control (SDLC)*.

bind command. An SNA command used to define the protocols for a session. Contrast with *unbind command*.

block. (1) A group of records that is recorded or processed as a unit. Same as *physical record*. (2) Ten sectors (2560 bytes) of disk storage. (3) In data communications, a group of records that is recorded, processed, or sent as a unit. (4) In DW/36, a sequential string of text (defined using the cursor movement keys or line commands) that is treated as a unit.

BSC. See *binary synchronous communications (BSC)*.

BSCCL (binary synchronous communications equivalence link) subsystem. The SSP-ICF subsystem that provides BSC communications with another System/36 and many other BSC computers and devices.

buffer. (1) A temporary storage unit, especially one that accepts information at one rate and delivers it at another rate. (2) An area of storage, temporarily reserved for performing input or output, into which data is read or from which data is written.

byte. The amount of storage required to represent one character; a byte is 8 bits.

C & SM. See *Communications and Systems Management (C & SM)*.

call. (1) To activate a program or procedure at its entry point. Compare with *load*. (2) In data communications, the action necessary in making a connection between two stations on a switched line.

CCITT. Consultative Committee on International Telegraphy and Telephone.

CCP. See *communications control program (CCP)*.

CCP subsystem. The SSP-ICF subsystem that provides data communications with a System/3 Model 15D.

chain. (1) A group of logically linked records. (2) In SNA, a group of logically linked records that are transferred over a communications line.

character key. A keyboard key that allows the user to enter the character shown on the key. Compare with *command key* and *function key*.

check. (1) An error condition. (2) To look for a condition.

CICS subsystem. The SSP-ICF subsystem that allows binary synchronous communications with CICS/VS.

CICS/VS. Customer Information Control System, which operates on a host system such as a System/370, or a 30XX or 43XX processor.

close. To end the processing of a file.

closed user group. A group of DTEs that can access only one another. DTEs outside of the group can neither access nor be accessed by members of the group.

COBOL (common business-oriented language). A high-level programming language, similar to English, that is used primarily for commercial data processing.

command. A request to the system to perform an operation or a procedure.

command key. A keyboard key that is used to request specific programmed actions. Compare with *character key* and *function key*.

communications. See *data communications*.

communications adapter. A hardware feature that enables a computer or device to become a part of a data communications network.

Communications and Systems Management (C & SM). A feature of the System Support Program Product that contains the remote management support (also referred to as DHCF), the change management support (referred to as DSNX), and the problem management support (referred to as alerts).

communications control program (CCP). An IBM System/3 Model 15 program that allows communications between System/3 and the SSP-ICF CCP subsystem.

communications file. A file that describes an advanced program-to-program communications (APPC) subsystem session between a System/36 program and a remote device, another program, or another system.

communications file definition. The format in the communications file that contains the APPC subsystem session description.

communications line. The line over which data communications takes place; for example, a telephone line.

communications link. See *data link*.

compress. (1) To move files, libraries, or folders together on disk to create one continuous area of unused space. (2) To replace repetitive characters in a file or folder with control characters so that the file or folder takes up less space when saved on diskette.

compression. In data communications, a technique for removing strings of duplicate characters and for removing trailing blanks before transmitting data.

configuration. The group of machines, devices, and programs that make up a data processing system. See also *system configuration*.

configuration member. In data communications, a member that defines the attributes of a communications subsystem or line.

configure. (1) To describe (to the system) the devices, optional features, and program products installed on a system. (2) To describe to SSP-ICF both the communications facilities connected to System/36 and the attributes of the subsystem and remote system.

console. A device used for communication between an operator and the system.

constant. A data item with a value that does not change. Contrast with *variable*.

control station. The primary or controlling computer on a multipoint line. The control station controls the sending and receiving of data.

control storage. Storage in the computer that contains the programs used to control input and output operations and the use of main storage. Contrast with *main storage*.

convention. A general agreement about basic principles. A rule.

current library. The first library searched for any required members. The current library can be specified during sign-on or while running programs and procedures.

data circuit-terminating equipment (DCE). The equipment installed at the user's location that provides all the functions required to establish, maintain, and terminate a connection, and the signal conversion and coding between the data terminal equipment (DTE) and the line.

data communications. The transmission of data between computers and/or remote devices (usually over a long distance).

data definition. Information that describes the contents and characteristics of a field, format (record), or file. A data definition can include such things as field names, lengths, and data types. See also *field definition*, *file definition*, and *format definition*.

data dictionary. A folder that contains field, format, and file definitions.

Data Encryption Subroutine. A feature of the System Support Program Product that codes and decodes data for security purposes. This subroutine is only used by the SSP-ICF Finance subsystem.

data link. The equipment and rules (protocols) used for sending and receiving data.

data link escape (DLE) character. In BSC, a transmission control character usually used in transparent text mode to indicate that the next character is a transmission control character.

data management. See *disk data management*.

data mode. In data communications, a time during which BSC is sending or receiving characters on the communications line.

data stream. All information (data and control information) transmitted over a data link.

data terminal equipment (DTE). The data processing unit that uses communications lines.

DCE. See *data circuit-terminating equipment (DCE)*.

DDM. See *Distributed Data Management (DDM)*.

default value. A value stored in the system that is used when no other value is specified.

define-the-file (DTF). A control block containing information that is passed between data management routines and users of the data management routines.

delete. To remove. For example, to delete a file.

demodulate. To set a modulated signal to its original state.

DHCF. See *Distributed Host Command Facility (DHCF)*.

disable. In interactive communications, to end a subsystem and free the area of main storage used by that subsystem. Contrast with *enable*.

disk data management. The System Support Program Product support that processes a request to read or write data.

display format. Data that defines (or describes) a display.

Distributed Data Management (DDM). A feature of the System Support Program Product that allows an application program to work on files that reside on a remote system.

Distributed Host Command Facility (DHCF). Another name for the remote management support offered by the Communications and Systems Management feature. This support allows HCF host system users to operate System/36s in an HCF network.

Distributed Systems Executive (DSX). A program product available for IBM host systems (System/370, 43XX, and 30XX) that allows the host system to get, send, and remove files, programs, formats, and procedures in a network of computers.

Distributed Systems Node Executive (DSNX). Another name for the change management support offered by the Communications and Systems Management feature. This support processes changes sent by a DSX host system.

DLE. See *data link escape (DLE) character*.

DSNX. See *Distributed Systems Node Executive (DSNX)*.

DSX. See *Distributed Systems Executive (DSX)*.

DTE. See *data terminal equipment (DTE)*.

DTF. See *define-the-file (DTF)*.

duplex. Pertains to communications in which data can be sent and received at the same time. Same as full duplex. Contrast with *half duplex*.

EBCDIC. See *extended binary-coded decimal interchange code (EBCDIC)*.

EBCDIC character. Any one of the symbols included in the 8-bit EBCDIC set.

emulation. Imitation; for example, the imitation of a computer or device.

enable. In interactive communications, to load and start a subsystem. Contrast with *disable*.

encryption. The process of scrambling information according to predefined rules so that it cannot be read without knowledge of them, as in the transmission of data over communications lines.

end-of-text (ETX) character. In binary synchronous communications, the transmission control character used to end a logical set of records that began with the start-of-text character.

end-of-transmission (EOT) character. In binary synchronous communications, the transmission control character usually used to end communications.

end-of-transmission-block (ETB) character. In binary synchronous communications, the transmission control character used to end a block of records that began with the start-of-text character.

ENQ. See *enquiry (ENQ) character*.

enquiry (ENQ) character. In binary synchronous communications, the transmission control character usually used to request a response from the remote system or device.

enter. To type in information from a keyboard and press the Enter key in order to send the information to the computer.

enter/update mode. The mode that is used to enter new statements into a source or procedure member, or to change statements that already exist in a source or procedure member.

EOT. See *end-of-transmission (EOT) character*.

ETB. See *end-of-transmission-block (ETB) character*.

ETX. See *end-of-text (ETX) character*.

evoke. In SSP-ICF and DDM, to start a program or procedure so that it can communicate with your program.

expression. A representation of a value. For example, variables and constants appearing alone or in combination with operators.

extended binary-coded decimal interchange code (EBCDIC). A set of 256 eight-bit characters.

feature. A programming or hardware option, usually available at an extra cost. For example, Communications is a feature of the System Support Program Product.

field definition. Information that describes the characteristics of data in a field. A field definition is contained in a data dictionary.

file. A set of related records treated as a unit.

file definition. (1) In RPG, file description and input specifications that describe the records and fields in a file. (2) In IDDU, information that describes the contents and characteristics of a file. A file definition is contained in a data dictionary.

file name. The name used by a program to identify a file. See also *label*.

Finance subsystem. The SSP-ICF subsystem that allows System/36 to communicate with the 3601 and 4701 Finance Controllers and the 3694 Document Processor.

folder. A named area on disk that contains documents, profiles, mail, or data definitions. Compare with *library*.

format. (1) A defined arrangement of such things as characters, fields, and lines, usually used for displays, printouts, files, or documents. (2) To arrange such things as characters, fields, and lines. (3) In BASIC, a representation of the correct form of a command or statement. (4) In IDDU, a group of related fields, such as a record, in a file.

format definition. Information that describes the contents and characteristics of data within a group of related fields, such as a record in a file. A format definition is contained in a data dictionary.

full duplex. Same as *duplex*.

function key. A keyboard key that requests an action but does not display or print a character. The cursor movement and Help keys are examples of function keys. Compare with *command key* and *character key*.

function management header. In SNA, a special record or part of a record that contains control information for the data that follows.

generation. For some remote systems, the translation of configuration information into machine language.

half duplex. Pertains to communications in which data can be sent in only one direction at a time. Contrast with *duplex*.

HCF. See *Host Command Facility (HCF)*.

hex. See *hexadecimal*.

hexadecimal. Pertaining to a system of numbers to the base sixteen; hexadecimal digits range from 0 (zero) through 9 (nine) and A (ten) through F (fifteen).

Host Command Facility (HCF). A feature available for IBM host systems (System/370, 43XX, and 30XX) that allows host system users to operate System/36s from their 3270-type display stations as though they were using remotely attached 5250-type display stations.

host system. The primary or controlling computer in a communications network. See also *control station*.

IDDU. See *interactive data definition utility (IDDU)*.

identifier. (1) A sequence of bits or characters that identifies a program, device, or system to another program, device, or system. (2) In COBOL, a data name that is unique or is made unique by the correct combination of qualifiers, subscripts, or indexes.

IF expressions. Expressions within a procedure that are used to test for a condition.

IGS. See *interchange group separator (IGS)*.

imperative statement. A statement that specifies that an action is always to be taken. An imperative statement can consist of a sequence of imperative statements.

indicator. An internal switch that communicates a condition between parts of a program or procedure.

informational message. A message that provides information to the operator, but does not require a response.

initial program load (IPL). The process of loading the system programs and preparing the system to run jobs.

initialize. To prepare for use. For example, to initialize a diskette.

inquiry. (1) A request for information in storage. (2) A request that puts a display station into inquiry mode. (3) In data communications, a request for information from another system.

interactive. Pertaining to activity involving requests and replies as, for example, between an operator and a program or between two programs. Contrast with *batch*.

Interactive Communications Feature (SSP-ICF). A feature of the System Support Program Product that allows a program to interactively communicate with another program or system.

interactive data definition utility (IDDU). The part of the System Support Program Product used to define the characteristics of data and the contents of files.

interactive processing. A processing method in which each operator action causes a response from the program or the system. Contrast with *batch processing*.

interchange group separator (IGS). A character used to indicate that blanks have been removed from a string of data and are to be reinserted.

intermediate block check. In binary synchronous communications, an option that permits checking each record, instead of checking the contents of the total buffer, when large buffers of data are received.

intermediate-text-block (ITB) character. In binary synchronous communications, the transmission control character used to indicate the end of a section of data to be checked. See *intermediate block check*.

interrupt. (1) To temporarily stop a process. (2) In data communications, to take an action at a receiving station that causes the sending station to end a transmission.

Intra subsystem. An SSP-ICF subsystem that enables programs to communicate with other programs on the same system without the use of communication lines.

intrinsic function. A function supplied by the program product. Contrast with *user-defined function*.

invite. To ask for input data from either a display station or an SSP-ICF session.

IPL. See *initial program load (IPL)*.

IRS (interchange record separator). Same as *record separator*.

ITB. See *intermediate-text-block (ITB) character*.

job. (1) A unit of work to be done by a system. (2) One or more related procedures or programs grouped into a procedure.

job step. A unit of work represented by a single program or a procedure that contains a single program. A job consists of one or more job steps.

K-byte. 1024 bytes.

label. (1) The name in the disk or diskette volume table of contents or on a tape that identifies a file. See also *file name*. (2) The name that identifies a statement. The name that identifies a BASIC program line.

LAN. See "local area network."

left-adjust. To place or move an entry in a field so that the leftmost character of the field is in the leftmost position. Contrast with *right-adjust*.

library. (1) A named area on disk that can contain programs and related information (not files). A library consists of different sections, called library members. Compare with *folder*. (2) The set of publications for a system.

library member. A named collection of records or statements in a library. The types of library members are *load member*, *procedure member*, *source member*, and *subroutine member*.

licensed application program. A set of licensed programs used to perform a particular data processing task, such as a distribution management application or a construction management application.

licensed program. An IBM-written program that performs functions related to processing user data.

link. In data communications, the connection between two systems.

literal. A symbol or a quantity in a source program that is itself data, rather than a reference to data.

load. (1) To move data or programs into storage. (2) To place a diskette into a diskette drive or a magazine into a diskette magazine drive. (3) To insert paper into a printer. (4) To mount a tape or insert a tape cartridge into a tape drive.

load member. A library member that contains information in machine language, a form that the system can use directly. Contrast with *source member*.

local. Pertaining to a device, file, or system that is accessed directly from your system, without the use of a communications line. Contrast with *remote*.

local area network (LAN). An information transport system for information transfer between devices located on the same premises, such as an office building, a manufacturing plant, a hospital area, a university campus, or any other geographically confined area.

location name. In interactive communications, the identifying name associated with a particular system or device.

logical record. The most inclusive data item. The level number for a logical record is 01.

logical unit (LU). The part of a system or device in an SNA network that allows a user or program to use the communication network.

lowercase. Pertaining to a letter having as its typical form *a f g* rather than *A F G*.

LU. See *logical unit (LU)*.

macro. See *macroinstruction*.

macroinstruction. A single instruction that represents a set of instructions.

main storage. The part of the processing unit where programs are run. Contrast with *control storage*.

manual answer. In data communications, a line type requiring operator actions to receive a call over a switched line. Contrast with *autoanswer*.

manual call. In data communications, a line type requiring operator actions to place a call over a switched line. Contrast with *autocall*.

master configuration record. Information, stored on disk, that describes system devices, programming, and characteristics.

member. See *library member*.

menu. A displayed list of items from which an operator can make a selection.

message. (1) Information sent to one or more users or display stations from a program or another user. A message can be either displayed or printed. (2) An indication of the condition of the system sent by the system. (3) For IMS/IRSS, a unit of data sent over the communications line.

message identification. A field in the display or printout of a message that directs the user to the description of the message in a message guide or a reference manual. This field consists of up to four alphabetic characters, followed by a dash, followed by the message identification code.

message identification code (MIC). A four-digit number that identifies a record in a message member. This number can be part of the message identification.

mnemonic. An identifier or symbol, using characters intended to assist memory, that is associated with an instruction, statement, or command.

mode. A method of operation. For an example, see *enter/update mode*.

modem. See *modulator-demodulator (modem)*.

modulation. Changing the frequency or size of one signal by using the frequency or size of another signal.

modulator-demodulator (modem). A device that converts data from the computer to a signal that can be transmitted on a communications line, and converts the signal received to data for the computer.

monitor. Programming or hardware that observes, supervises, controls, or verifies the operation of a system.

MRT procedure. See *multiple requester terminal (MRT) procedure*.

MRT program. See *multiple requester terminal (MRT) program*.

MSRJE. See *Multiple Session Remote Job Entry (MSRJE)*.

multiple requester terminal (MRT) procedure. A procedure that calls a multiple requester terminal program.

multiple requester terminal (MRT) program. A program that can process requests from more than one display station or SSP-ICF session at the same time using a single copy of the program. Contrast with *single requester terminal (SRT) program*.

Multiple Session Remote Job Entry (MSRJE). A feature of the System Support Program Product that allows one or more remote job entry sessions to operate on a host system (such as a System/370, or a 30XX or 43XX processor) at the same time.

multipoint. In data communications, pertains to a network that allows two or more stations to communicate with a single system on one line.

NAK. See *negative acknowledgment character (NAK)*.

NCP. See *network control program (NCP)*.

negative acknowledgment character (NAK). In binary synchronous communications, a transmission control character sent as a negative response to data received.

negative response. In data communications, a reply indicating that data was not received correctly or that a command was incorrect or unacceptable.

NEP. See *never-ending program (NEP)*.

network. A collection of data processing products connected by communication lines for information exchange between stations.

network control program (NCP). A program, generated by the user from a library of IBM-supplied modules, that controls the operation of a communications controller.

never-ending program (NEP). A long-running program that does not share system resources, except for shared files and the spool file.

node. (1) An addressable location in a communications network that provides host processing services. (2) A point where packets are received, stored, and forwarded to another node (or DTE) according to a routing method the network has defined.

nonswitched line. A connection between computers or devices that does not have to be established by dialing. Contrast with *switched line*.

null. See *null character*.

null character. The character hex 00, used to represent the absence of a printed or displayed character.

null character string. Two consecutive single quotation marks that specify a character constant of no characters.

null record. In binary synchronous communications, a record that contains no data; only the data link control characters STX ETX.

numeric. Pertaining to any of the digits 0 through 9.

OCL. See *operation control language (OCL)*.

offline. Neither controlled directly by, nor communicating with, the computer, or both. Contrast with *online*.

online. Being controlled directly by, or directly communicating with, the computer, or both. Contrast with *offline*.

open. To prepare a file for processing.

operation. A defined action, such as adding or comparing, performed on one or more data items.

operation code. (1) A code used to represent the operations of a computer. (2) In SSP-ICF, a code used by a System/36 application program to request SSP-ICF data management and/or the subsystem to perform an action. For example, the operation \$\$SEND asks that data be sent.

operation control language (OCL). A language used to identify a job and its processing requirements to the System Support Program Product.

optional network facilities. Facilities a packet switching data network user may request when establishing a virtual circuit. See also *reverse charging*, *closed user group*, and *throughput class negotiation*.

override. (1) A parameter or value that replaces a previous parameter or value. (2) To replace a parameter or value.

packet. A data transmission information unit. It has a header on the front that indicates the destination of the packet. Commonly used data field lengths in packets are 128 or 256 bytes.

packet switching. The act of transferring and routing packets from source to destination based on information contained in their headers.

packet switching data network (PSDN). A communications network that uses packet switching as a means of transmitting data.

parameter. A value supplied to a procedure or program that either is used as input or controls the actions of the procedure or program.

partner. In data communications, the remote application program or the remote computer.

password security. A System Support Program Product option that helps prevent the unauthorized use of a display station, by checking the password entered by each operator at sign-on.

Peer subsystem. The SSP-ICF subsystem that allows System/36 to communicate with another System/36 or System/34 using SNA/SDLC.

pending. Waiting, as in an operation is pending.

permanent virtual circuit (PVC). A virtual circuit that has a logical channel permanently assigned to it at each DTE. The usual call establishment protocol is therefore not required.

physical record. (1) A group of records that is recorded or processed as a unit. Same as *block*. (2) A unit of data that is moved into or out of the computer.

point-to-point line. A communications line that connects a single remote station to a computer.

poll. To execute a polling sequence.

polling. A method for determining whether each of the stations on a communications line has data to send.

positional parameter. A parameter that must appear in a specified location, relative to other positional parameters.

problem determination. The process of identifying why the system is not working. Often this process identifies programs, equipment, data communications facilities, or user errors as the source of the problem.

procedure. A set of related operation control language statements (and, possibly, utility control statements and procedure control expressions) that cause a specific program or set of programs to be performed.

procedure command. A command that runs a procedure.

procedure member. A library member that contains the statements (such as operation control language statements) necessary to perform a program or set of programs.

procedure start request. A message from the remote system asking an SSP-ICF subsystem to start a System/36 procedure.

program product. A licensed program for which a fee is charged.

program temporary fix (PTF). A temporary solution to or bypass of a defect in a current release of a licensed program.

programmable. Designating a device whose operation can be controlled by a program.

prompt. A displayed request for information or operator action.

protocol. A set of rules governing the communication and transfer of data between two or more devices in a communications system.

PSDN. See *packet switching data network (PSDN)*.

PTF. See *program temporary fix (PTF)*.

public data network. A communications common carrier network that provides data communications services over switched or nonswitched lines.

queue. A line or list formed by items waiting to be processed.

Recommendation X.25. A document, CCITT Recommendation X.25, that outlines standards for the connection of processing equipment to a packet switching data network.

record separator. In binary synchronous communications, a character used to indicate the end of one record and the beginning of another.

recovery procedure. (1) An action performed by the operator when an error message appears on the display screen. Usually, this action permits the program to continue or permits the operator to run the next job. (2) The method of returning the system to the point where a major system error occurred and running the recent critical jobs again.

remote. Pertaining to a device, file, or system that is accessed by your system through a communications line. Contrast with *local*.

remotely started session. A session started by an incoming procedure start request from the remote system. Contrast with *acquired session*.

request unit. In SNA, the record transmitted to the other system. This record can contain a request, data, or both.

requester. A display station or interactive communications session that requests a program to be run.

response unit. In SNA, the record sent to respond to a request. The response can be either positive or negative and can include control information.

restore. Return to an original value or image. For example, to restore a library from diskette.

return code. In data communications, a value generated by the system or subsystem that is returned to a program to indicate the results of an operation issued by that program.

reverse charging. A packet switching data network optional facility. It enables the DTE to request that the cost of a communications session it initiates be charged to the DTE that is called. See also *optional network facilities*.

reverse-interrupt character (RVI). In binary synchronous communications, a request by the receiving station to the sending station to stop sending and begin receiving a message.

right-adjust. To place or move an entry in a field so that the rightmost character of the field is in the rightmost position. Contrast with *left-adjust*.

routine. A set of statements in a program that causes the system to perform an operation or a series of related operations.

RPG. A programming language specifically designed for writing application programs that meet common business data processing requirements.

RU. See *request unit* and *response unit*.

RVI. See *reverse-interrupt character (RVI)*.

SDLC. See *synchronous data link control (SDLC)*.

security. The protection of data, system operations, and devices from accidental or intentional ruin, damage, or exposure. See also *system security*.

segment. A part of a program that can be run without the entire program being in main storage.

sense data. In SNA, the data sent with a negative response, indicating the reason for the response.

separator character. In data communications, the character that is used with some autocal units to separate the digits to be dialed.

session. (1) The logical connection by which a System/36 program or device can communicate with a program or device at a remote location. (2) The length of time that starts when an operator signs on the system and ends when the operator signs off the system.

SEU. See *source entry utility (SEU)*.

single requester terminal (SRT) program. A program that can process requests from only one display station or SSP-ICF session from each copy of the program. Contrast with *multiple requester terminal (MRT) program*.

SNA. See *systems network architecture (SNA)*.

SNA Upline Facility (SNUF). The SSP-ICF subsystem that allows System/36 to communicate with CICS/VS and IMS/VS application programs on a host system. Also, using this subsystem, DHCF communicates with HCF and DSNX communicates with DSX.

SNUF. See *SNA Upline Facility (SNUF)*.

SOH. See *start-of-header (SOH) character*.

source entry utility (SEU). The part of the Utilities Program Product used by the operator to enter and update source and procedure members.

source member. A library member that contains information in the form in which it was entered, such as RPG specifications. Contrast with *load member*.

SRT program. See *single requester terminal (SRT) program*.

SSCPID. System services control point identifier.

SSP. See *System Support Program Product (SSP)*.

SSP-ICF. See *Interactive Communications Feature (SSP-ICF)*.

start-of-header (SOH) character. In binary synchronous communications, the transmission control character indicating that the information that follows is a header.

start-of-text (STX) character. In binary synchronous communications, a transmission control character used to begin a logical set of records that will be ended by the end-of-text character or end-of-transmission-block character.

statement. An instruction in a program or procedure.

station. A computer or device that can send or receive data.

status. A condition. For example, the status of a printer, a job, or a communications line.

STX character. See *start-of-text (STX) character*.

subroutine. A group of instructions that can be called by another program or subroutine.

subroutine member. A library member that contains information that must be combined with one or more members before being run by the system.

subsystem. The part of communications that handles the requirements of the remote system, isolating most system-dependent considerations from the application program.

switched line. In data communications, a connection between computers or devices that is established by dialing. Contrast with *nonswitched line*.

switched network backup (SNBU). In data communications, a technique that provides a switched line connection when a nonswitched line fails.

switched virtual circuit. A virtual circuit that is requested from the network through a virtual call. It is released when the virtual circuit is cleared.

SYN. See *synchronization (SYN) character*.

synchronization (SYN) character. In binary synchronous communications, the transmission control character that provides a signal to the receiving station for timing.

synchronous. Occurring in a regular or predictable sequence.

synchronous data link control (SDLC). A form of communications line control that uses commands to control the transfer of data over a communications line. Compare with *binary synchronous communications (BSC)*.

synchronous transmission. In data communications, a method of transmission in which the sending and receiving of characters is controlled by timing signals. Contrast with *asynchronous transmission*.

syntax. The rules for the construction of a command or statement.

system configuration. A process that specifies the machines, devices, and programs that form a particular data processing system.

system library. The library, provided with the system, that contains the System Support Program Product and is named #LIBRARY.

system monitor session. In SSP-ICF, a session started by the Finance subsystem to load the applications into a finance controller.

system services control point (SSCP). A focal point within an SNA network for managing the configuration, coordinating network operator and problem determination requests, and providing directory support and other session services for network users.

System Support Program Product (SSP). A group of licensed programs that manage the running of other programs and the operation of associated devices, such as the display station and printer. The SSP also contains utility programs that perform common tasks, such as copying information from diskette to disk.

systems network architecture (SNA). A set of rules for controlling the transfer of information in a data communications network.

task. A unit of work (such as a user program) for the main storage processor.

temporary-text-delay (TTD) character. A BSC transmission control character that indicates to the receiving station that there is a temporary delay in the transmission of data.

terminal. In data communications, a device, usually equipped with a keyboard and a display device, capable of sending and receiving information over a communications line.

throughput class negotiation. A packet switching data network optional facility. Allows a DTE to negotiate the speed at which its packets travel through the packet switching data network.

Token-Ring Network, IBM. The local area network designed to run on the IBM Cabling System.

transaction. (1) An item of business. The handling of customer orders and customer billing are examples of transactions. (2) In interactive communications, the communication between the application program and a specific item (usually another application program) at the remote system.

transaction code. For the IMS subsystem, the first one to eight characters of the first segment of a message sent to IMS/VS. The transaction code identifies the application program for which the message is intended.

TRANSACTION file. An input/output file used to communicate with display stations and SSP-ICF sessions.

transmission control characters. In data communications, special characters that are included in a message to control communication over a data link. For example, the sending station and the receiving station use transmission control characters to exchange information; the receiving station uses transmission control characters to indicate errors in data it receives.

transparent data. Data that can contain any hexadecimal value.

transparent text mode. A mode that allows BSC to send and receive messages containing any of the 256 character combinations in hexadecimal, including transmission control characters.

tributary station. In data communications, a secondary device on a multipoint line.

truncate. To shorten a field or statement to a specified length.

TTD character. See *temporary-text-delay (TTD) character*.

unbind command. An SNA command used to reset the protocols for a session. Contrast with *bind command*.

uppercase. Pertaining to capital letters.

UPSI switch. See *user program status indicator (UPSI) switch*.

user program status indicator (UPSI) switch. One of a set of eight switches that can be set by and passed between application programs and procedures.

user-defined function. In BASIC, a function defined by the user in a function definition (in the DEF statement). Contrast with *intrinsic function*.

variable. A name used to represent a data item whose value can change while the program is running. Contrast with *constant*.

virtual telecommunications access method (VTAM). A set of programs that control communications between terminals and application programs running under VSE, OS/VS1, and OS/VS2.

VTAM. See *virtual telecommunications access method (VTAM)*.

WACK. See *wait-before-transmitting-acknowledgment character (WACK)*.

wait-before-transmitting-acknowledgment character (WACK). In BSC, the transmission control character indicating that the station is temporarily not ready to receive data.

work station. A device that lets people transmit information to or receive information from a computer; for example, a display station or printer.

work station ID. A two-character identifier assigned to each display station and printer on your system.

World Trade. (1) Pertains to the distinction between the US and the rest of the world. (2) Pertains to the combination of:

- IBM World Trade Americas/Far East Corporation
- IBM World Trade Europe/Middle East/Africa Corporation

X.21. In data communications, a specification of the CCITT that defines the connection of data terminal equipment to an X.21 (public data) network.

X.21 feature. The feature that allows System/36 to be connected to an X.21 network.

X.25. In data communications, a specification of the CCITT that defines the interface to an X.25 (packet switching) network.

X.25 feature. The feature that allows System/36 to connect to an X.25 network.

X.75. A standard that defines ways of interconnecting two X.25 networks.

3270 BSC Support subsystem. The subsystem that provides program-to-program communications with IMS/VS, CICS/VS, TSO, VM, or system application programs using 3270 BSC protocols, and provides support for the BSC portion of the 3270 Device Emulation feature.

3270 Device Emulation. A feature of the System Support Program Product that allows a System/36 local or remote device to appear as a 3270 device to another system.

3270 SNA Support subsystem. The subsystem that provides support for the SNA portion of the 3270 Device Emulation feature.

Index

Special Characters

\$\$TIMER operation

BASIC example 3-21

\$ALOC macro (assembler) 2-10

\$CLOS macro (assembler) 2-11

\$DTFO macro (assembler) 2-8

\$DTFW macro (assembler) 2-4

\$EVOK macro (assembler) 2-18

coding examples

Intra subsystem 6-26

description

Intra subsystem 6-26

\$OPEN macro (assembler) 2-11

\$WSIO macro (assembler)

coding examples

Intra subsystem 6-26

communications operations

charts of 2-16, 2-21, A-3

getting status of session 2-14

modified by OPM parameter 2-15

specified by OPC parameter 2-14

description

Intra subsystem 6-25

description of 2-12

parameter matrix chart 2-16

*** (asterisk)**

on configuration displays

Intra 6-6

_ (underscores)

on configuration displays

Intra 6-6

A

accept input operation

BASIC 3-8

description

Intra 6-17

required for timer operations

Intra 6-44

TERMINAL option (COBOL) 4-10

ACCEPT statement (COBOL)

description 4-7

examples 4-9

FOR option

get attributes operation 4-7

ACQ operation

description

RPG 5-10

example

languages 2-16, 5-10

Intra 6-18

acquire operation

description

Intra 6-18

error return codes

Intra 6-66

example

languages 5-10

Intra 6-18

starts a session 1-5

Intra 6-11

ACQUIRE statement (COBOL)

description 4-6

examples 4-6

identifies TRANSACTION file 4-6

acquired sessions

Intra 6-11

elements used in 1-4

acquiring a session

ACQ operation (RPG) 5-10

ACQUIRE statement (COBOL) 4-6

example

BASIC 3-6

COBOL 4-6

introduction to 1-4

OPEN statement (BASIC) 3-4

addresses, session

See session addresses

APPC subsystem

communications line

specifying line member name 1-18

DISABLE procedure command

syntax diagram 1-20

appendixes

glossary of terms G-1

APPN subsystem

communications line

specifying line member name 1-18

DISABLE procedure command

syntax diagram 1-20

ENABLE procedure command

syntax diagram 1-18

assembler

coding examples 2-24

Intra 6-18, 6-26

evoke operations

Intra description 6-25

macros

\$ALOC 2-10

assembler (continued)

macros (continued)

- \$CLOS 2-11
- \$DTFO 2-8
- \$DTFW 2-4
- \$EVOK 2-18
- \$OPEN 2-11
- \$WSIO 2-12
- \$WSIO macro parameter chart 2-16
- for controlling communications files 2-10
- list of 2-3
- operations parameter chart 2-16
- operations summary chart 2-21, A-3
- return codes
 - offset locations 2-23
 - processing 2-23
- sending data
 - evoke operations 2-20
- subroutine DTF considerations 2-23
- writing subroutines for HLL programs 2-23

asterisks (*)

- on configuration displays
- Intra 6-6

ATTRIBUTE\$ intrinsic function (BASIC) 3-25

- examples 3-25

B

BASIC

- canceling sent records 3-23
- coding examples
 - See also WRITE statement (BASIC), coding examples 3-15
 - complete program 3-31
 - Intra 6-28
- communications operations
 - list of, WRITE statement 3-13, 3-20
 - similar to work station operations 3-2
- ending a session 3-20
- ending a transaction 3-19
- ERR codes
 - chart of 3-29
 - unsuccessful operations 3-28
- evoke operations
 - general description 3-14
 - Intra description 6-27
 - procedure to send data to a procedure 3-17
 - procedure to send data to a program 3-16
- evoke parameter list 3-14
- examples
 - ATTRIBUTE\$ intrinsic function 3-25
 - CLOSE statement 3-24
 - OPEN statement 3-6
 - WAITIO statement 3-11

BASIC (continued)

- EXIT clause needed 3-31
 - FILENUM intrinsic function
 - note 3-31
 - indicating error conditions
 - fail operation 3-23
 - introduction 3-2
 - IOERR parameter needed 3-28
 - notes
 - receiving data 3-10
 - writing BASIC programs 3-31
 - OPEN statement
 - specially acquired sessions 3-6
 - operations summary chart 3-27, A-4
 - requesting change of direction 3-20
 - REREAD statement, note 3-10
 - RETCODE\$
 - chart of SSP-ICF return codes 3-30
 - description 3-30
 - status of last operation 3-28
 - return codes
 - checking 3-28
 - ERR codes, chart 3-29
 - ERR codes, meaning 3-28
 - RETCODE\$ chart of SSP-ICF return codes 3-30
 - RETCODE\$ description 3-30
 - RETCODE\$ meaning 3-28
 - sending
 - negative response 3-22
 - sense data 3-22
 - sending data
 - evoke operations 3-16
 - put operations 3-18
 - SESSION statement not needed
 - Intra 6-12
 - session status information 3-26
 - set timer operations 3-21
 - specially acquired sessions 1-14
 - examples 3-6
 - starting remote programs 3-14
 - statement descriptions
 - ATTRIBUTE\$ intrinsic function 3-25
 - CLOSE statement 3-24
 - introduction to 3-3
 - OPEN statement 3-4
 - READ statement 3-8
 - WAITIO statement 3-11
 - WRITE statement 3-12
 - WSID\$ intrinsic function
 - note 3-31
 - purpose of 3-4
- ### BASICP procedure
- changes needed to send program data 3-16
 - example using 3-17

BASICR procedure
 changes needed to send program data 3-16
 example using 3-16

BSC MSRJE
 non-SSP-ICF subsystem 1-9

BSC 3270
 limit on subsystem 1-10
 non-SSP-ICF subsystem 1-9

BSCSEL subsystem
 communications line
 specifying line number 1-18

DISABLE procedure command
 functions performed by 1-19
 syntax diagram 1-20

ENABLE procedure command
 functions performed by 1-15
 line number 1-18
 syntax diagram 1-18

return codes
 summary descriptions of B-1, B-2

C

cancel invite operation
 description
 Intra 6-21

cancel operations
 description
 languages 3-23, 4-20, 5-9
 Intra 6-20

examples
 BASIC WRITE statement 3-23
 COBOL WRITE statement 4-20

types of
 Intra 6-20, 6-21

canceling sent records
 BASIC description 3-23
 cancel operations
 RPG description 5-9
 COBOL description 4-20

CCP subsystem
 communications line
 specifying line number 1-18

DISABLE procedure command
 functions performed by 1-19
 syntax diagram 1-20

ENABLE procedure command
 functions performed by 1-15
 line number 1-18
 syntax diagram 1-18

return codes
 summary descriptions of B-1, B-2

CFILE continuation option (RPG II) 5-22

changing a member
 See modifying a configuration member

charts
 See also summary charts

communications line features, summary of 1-11

communications operations
 assembler 2-21, A-3
 BASIC 3-27, A-4
 COBOL 4-22, A-5
 RPG II 5-14, A-6

parameters, \$WSIO macro 2-16

return code processing, RPG II 5-16

SSP-ICF subsystems
 types of 1-9

status values
 RPG II 5-19

charts, summary
 See also sequence diagrams

input/output operations
 Intra 6-46

checking for return code 02xx
 STOP\$ intrinsic function (BASIC) 3-10

CICS subsystem
 communications line
 specifying line number 1-18

DISABLE procedure command
 functions performed by 1-19
 syntax diagram 1-20

ENABLE procedure command
 functions performed by 1-15
 line number 1-18
 syntax diagram 1-18

return codes
 summary descriptions of B-1, B-2

CLOSE statement (BASIC)
 description 3-24
 examples 3-24

CNFIGICF procedure
 See also configuration displays

default values
 Intra 6-6

description
 Intra subsystem 6-5

prompting facilities
 Intra 6-6

CNFIGSSP procedure
 installs subsystem support
 Intra 6-5

COBOL
 canceling sent records 4-20
 coding examples
 See also WRITE statement (COBOL), coding
 examples 4-15
 complete program 4-23
 Intra 6-30

communications operations
 list of, WRITE statement 4-13, 4-17

COBOL (continued)

- communications operations (continued)
 - not used in display formats 4-4
 - similar to work station operations 4-2
- ending a session 4-17
- ending a transaction 4-16
- evoke operations
 - general description 4-14
 - Intra description 6-29
- evoke parameter list 4-14
- examples
 - ACCEPT statement 4-9
 - ACQUIRE statement 4-6
 - READ statement 4-11
- file status
 - FILE STATUS clause 4-5
 - FILE STATUS field 4-23
 - values 4-23
- indicating error conditions
 - fail operation 4-20
- introduction 4-2
- operations summary chart 4-22, A-5
- requesting change of direction 4-17
- return codes
 - chart of 4-23
 - processing 4-23
- sending
 - negative response 4-19
 - sense data 4-19
- sending data
 - evoke operations 4-15
 - put operations 4-15
- session status information 4-8
- set timer operation 4-18
- starting remote programs 4-14
- statement descriptions
 - ACCEPT statement 4-7
 - ACQUIRE statement 4-6
 - DROP statement 4-21
 - READ statement 4-10
 - SELECT statement 4-4
 - WRITE statement 4-12
- TRANSACTION file 4-4, 4-6

COBOL statement descriptions
introduction to 4-3

coding examples, complete program
assembler 2-24
BASIC 3-31
COBOL 4-23
RPG II 5-25

combinations of subsystems 1-10

combined input/output operations
summary charts
Intra 6-46

communications lines

- introduction to 1-10
- chart of subsystem features 1-11
- line number specified on ENABLE command 1-18
- System/36 support 1-10

communications networks

- program-to-program communications
Intra example 6-3

communications operations

- assembler
 - indicating function management header 2-15
 - modified by OPM parameter 2-15
 - specified in \$WSIO macro (OPC parameter) 2-13
 - summary chart of 2-21, A-3

BASIC

- CLOSE statement 3-24
- OPEN statement 3-4
- READ statement 3-8
- summary chart of 3-27, A-4
- WAITIO statement 3-11
- WRITE statement 3-12

COBOL

- ACCEPT statement 4-7
- ACQUIRE statement 4-6
- DROP statement 4-21
- READ statement 4-10
- summary chart of 4-22, A-5
- WRITE statement 4-12

general introduction

- all subsystems 1-8

getting status of session

- ATTRIBUTE\$ function (BASIC) 3-25

introduction

- languages 3-3, 4-3, 5-3

- Intra 6-14

list of, performed in

- BASIC 3-13, 3-20

- COBOL 4-13, 4-17

- RPG 5-10

- not used in display formats 4-4

- purpose of return codes 1-8

RPG II

- ACQ operation 5-10

- NEXT operation 5-12

- READ operation 5-13

- REL operation 5-11

- summary chart of 5-14, A-6

- similar to work station operations 3-2, 4-2, 5-2

summary chart

- assembler 2-21, A-3

- BASIC 3-27, A-4

- COBOL 4-22, A-5

- RPG II 5-14, A-6

- Intra 6-14

communications, ending
 DISABLE procedure command
 Intra 6-10

communications, establishing
 ENABLE procedure command
 Intra 6-10

communications, program-to-program
(Intra) 6-3

configuration
 modifying attributes of
 Intra 6-10
 prompting facilities
 Intra 6-6

configuration displays
 asterisks (*) on
 Intra 6-6
 default values
 Intra 6-6
 explanation of
 Intra 6-6
 introduction
 Intra 6-5
 subsystem member displays
 Intra 6-7

configuration member
 activated by ENABLE procedure command
 Intra 6-5
 modifying its attributes
 Intra 6-10
 name must be unique
 Intra 6-5

configuration worksheets
See worksheets, configuration display

configurations
 examples of
 Intra 6-3

continuation options, RPG II
 CFILE 5-22
 FMTS 5-22
 ID 5-21
 IND 5-21
 INFDS 5-22
 INFSR 5-21
 list of, for WORKSTN files 5-20
 NUM 5-20
 SAVDS 5-21
 SLN 5-21

COPY control statement, PDATA parameter
 assembler evoke operations 2-20
 BASIC evoke operations 3-16
 COBOL evoke operations 4-15

D

Data Encryption Feature
See encryption/decryption

debug program
 ICFDEBUG procedure 1-2

default values
 configuration displays
 Intra 6-6

DISABLE procedure command
 description
 Intra 6-10
 functions performed by 1-19
 general introduction 1-19
 syntax diagram of 1-20

disabling a subsystem 1-19

displays
See configuration displays

DROP statement (COBOL) 4-21

DTF
 address offsets for (\$DTFO macro) 2-8
 considerations in subroutines 2-23
 generated by \$DTFW macro 2-4
 specifying labels for 2-9

DTF parameter
 in \$WSIO macro 2-12
 in communications file macros 2-10

E

editing a member
 Intra 6-10

ENABLE procedure command
 description
 Intra 6-10
 functions performed by 1-15
 general introduction 1-15
 line number parameter 1-18
 location name, remote 1-16
 prepares local end for communications 1-16
 syntax diagram of 1-18
 unmatched line types 1-16

enabling a subsystem 1-15

end of job menu, SEU procedure
 assembler evoke operations 2-20
 BASIC evoke operations 3-16
 COBOL evoke operations 4-15

end of session operation
 description
 languages 3-20, 4-17, 5-9
 Intra 6-22
 examples
 BASIC WRITE statement 3-20

end of session operation (*continued*)
 examples (*continued*)
 COBOL WRITE statement 4-17

end of transaction operations
 BASIC WRITE statement 3-19
 COBOL WRITE statement 4-16
 examples
 BASIC 3-19
 COBOL 4-16

end of transaction return codes
 BASIC 3-19
 COBOL 4-16

end-of-file conditions
 RPG II 5-23

ending a session
 BASIC example 3-24
 BASIC WRITE statement 3-20
 CLOSE statement (BASIC) 3-24
 COBOL WRITE statement 4-17
 end of session operation
 RPG description 5-9

ending a transaction
 BASIC WRITE statement 3-19
 COBOL WRITE statement 4-16

ending communications
See communications, ending

EOF clause (BASIC)
 READ statement
 example 3-10
 note on 3-10

ERR codes
 BASIC return code checking 3-28
 chart of 3-29

establishing communications
See communications, establishing

evoke end of transaction operation
 description
 Intra 6-23

evoke operations
 \$EVOK macro description
 Intra 6-26
 \$EVOK macro examples
 Intra 6-26
 \$WSIO macro description
 Intra 6-25
 \$WSIO macro examples
 Intra 6-26
 assembler coding examples
 languages 2-16
 Intra 6-26
 BASIC coding examples
 languages 3-15
 Intra 6-28
 BASIC description
 Intra 6-27
 chart of
 Intra 6-23

evoke operations (*continued*)
 COBOL coding examples
 Intra 6-30
 COBOL description
 Intra 6-29
 evoke parameter list
 general description 3-14, 4-14, 5-3
 Intra 6-23
 general description
 languages 5-3
 Intra 6-23

IDDU considerations
 Intra subsystem 6-25
 BASIC 3-14
 COBOL 4-14
 RPG II 5-3

IDDU keywords
 Intra subsystem 6-24

optional data, types of
 Intra 6-23

PDATA parameter, COPY control
 statement 2-20, 3-16, 4-15

procedure for sending data
 to a BASIC procedure 3-17
 to a BASIC program 3-16

programming considerations
 for Intra 6-23

RPG coding examples
 languages 5-4
 Intra 6-32

RPG description
 languages 5-3
 Intra 6-31

RPG examples
 Intra 6-32

sending data, types of 2-20, 3-16, 4-15

starting a procedure 5-3

types of
 Intra 6-23

WRITE statement (BASIC) examples
 Intra 6-28

WRITE statement (COBOL) examples
 Intra 6-30

evoke parameter list
 assembler
 specified in \$EVOK macro 2-18
 BASIC fields
 Intra 6-27
 COBOL fields
 Intra 6-29
 description
 Intra 6-23
 general description
 languages 3-14, 4-14, 5-3
 RPG fields
 Intra 6-31

examples

ACCEPT statement
 COBOL 4-9

acquire operation
 languages 2-16, 5-10
 Intra 6-18

ACQUIRE statement
 COBOL 4-6

ATTRIBUTE\$ intrinsic function
 BASIC 3-25

cancel operations
 BASIC 3-23
 COBOL 4-20

CLOSE statement
 BASIC 3-24

end of session operation
 BASIC 3-20
 COBOL 4-17

evoke operations
 languages 2-16, 3-15, 5-4
 Intra 6-25

fail operation
 BASIC 3-23
 COBOL 4-20

negative response operation
 RPG 5-8

negative response operations
 BASIC 3-22
 COBOL 4-19

NEXT operation
 RPG 5-12

OPEN statement
 BASIC 3-6

procedure to send data on evoke operation
 BASIC 3-16, 3-17

program-to-program communications, Intra 6-3

put end of transaction operation
 BASIC 3-19
 COBOL 4-16

put operation
 BASIC 3-18
 COBOL 4-15
 RPG 5-5

READ operation
 RPG 5-13

READ statement
 COBOL 4-11

release operation
 languages 5-11

request to change direction operations
 BASIC 3-20
 COBOL 4-17

set timer operation
 BASIC 3-21
 COBOL 4-18
 RPG 5-7

WAITIO statement
 BASIC 3-11

EXIT clause (BASIC)

used with IOERR parameter 3-31

**fail operation**

description
 languages 3-23

description
 languages 4-20, 5-9
 Intra 6-33

examples
 BASIC WRITE statement 3-23
 COBOL WRITE statement 4-20

file definition

COBOL SELECT statement 4-4

file description specifications

RPG II 5-2

file status

FILE STATUS clause (COBOL) 4-5
values (COBOL) 4-23

FILE STATUS clause (COBOL) 4-5**FILE STATUS field**

COBOL return codes 4-23

file status values (COBOL)

chart of 4-23

FILENUM intrinsic function (BASIC) 3-11

note 3-31

files

controlling for communications
(assembler) 2-10

Finance subsystem

communications line
 specifying line number 1-18

DISABLE procedure command
 functions performed by 1-19
 syntax diagram 1-20

ENABLE procedure command
 functions performed by 1-15
 line number 1-18
 syntax diagram 1-18

return codes

summary descriptions of B-1, B-2

FMTS continuation option (RPG II) 5-22**FORMAT option, WRITE statement (COBOL) 4-12****FORMAT parameter, WRITE statement (BASIC) 3-12****function management header**

specifying, in assembler 2-15

function management headers

put FMH operations
 Intra 6-40

G

get attributes operation

ACCEPT statement, FOR option (COBOL) 4-7
description

Intra 6-35

OPC parameter (\$WSIO macro)

session status information 2-14

status information

Intra 6-35

get operation

description

Intra 6-35

READ statement (BASIC) 3-8

TERMINAL option (COBOL) 4-10

get status operation

status information

Intra 6-36

getting session attributes

BASIC ATTRIBUTE\$ intrinsic function 3-25

BASIC example 3-25

COBOL ACCEPT statement 4-7

H

HLL statement descriptions

assembler

See macroinstructions

BASIC 3-3

COBOL 4-3

RPG II 5-2

HLL subroutines

assembler 2-23

INFSR, RPG II 5-15, 5-18

I

ICFDEBUG procedure 1-2

ID continuation option (RPG II) 5-21

IDDU (interactive data definition utility)

in evoke operations

Intra subsystem 6-25

starting remote programs

BASIC evoke operation 3-14

COBOL evoke operation 4-14

RPG II evoke operation 5-3

IDDU keywords

evoke operations

Intra subsystem 6-24

IMS subsystem

communications line

specifying line number 1-18

DISABLE procedure command

functions performed by 1-19

syntax diagram 1-20

ENABLE procedure command

functions performed by 1-15

line number 1-18

syntax diagram 1-18

return codes

summary descriptions of B-1, B-2

IND continuation option (RPG II) 5-21

indicating error conditions

fail operation

BASIC description 3-23

COBOL description 4-20

RPG description 5-9

INFDS continuation option (RPG II) 5-22

INFSR continuation option (RPG II) 5-21

INFSR subroutine (RPG II)

return code considerations 5-18

return code processing 5-15

input operations

maximum record length

languages 3-10, 4-10

input/output operations

See also communications operations

programming considerations (RPG II) 5-24

summary charts

Intra 6-46

Interactive Communications Feature

See SSP-ICF

interactive data definition utility

See IDDU (interactive data definition utility)

Intra subsystem

accept input operation

description 6-17

acquire operation

description 6-18

starts a procedure 6-11

cancel operations

description 6-20

types of 6-20, 6-21

CNFIGICF procedure

defining members 6-5

modifying attributes 6-10

prompting facilities 6-6

CNFIGSSP procedure 6-5

coding examples

\$EVOK macro 6-26

\$WSIO macro 6-26

RPG evoke operations 6-32

WRITE statement (BASIC) 6-28

WRITE statement (COBOL) 6-30

communications operations

accept input 6-17

Intra subsystem (continued)

communications operations (*continued*)
 acquire 6-18
 cancel 6-20
 end of session 6-22
 evoke 6-23
 evoke end of transaction 6-23
 fail 6-33
 get 6-35
 get attributes 6-35
 introduction 6-14
 invite 6-37
 negative response 6-38
 put 6-39
 put end of chain 6-39
 put end of transaction 6-39
 release 6-41
 request to change direction 6-42
 set timer 6-44
 status information about 6-35
 summary chart of 6-14
 configuration
 CNFIGICF procedure 6-5
 modifying attributes of 6-10
 prompting facilities 6-6
 configuration displays
 default values 6-6
 explanation of 6-6
 subsystem member definition 6-7
 configuration members 6-5
 description and capabilities 6-3
 DISABLE procedure command
 description 6-10
 functions performed by 1-19
 syntax diagram 1-20
 displays, descriptions of 6-5
 ENABLE procedure command
 description 6-10
 functions performed by 1-15
 syntax diagram 1-18
 end of session operation
 description 6-22
 evoke end of transaction operation
 description 6-23
 evoke operations
 assembler macros 6-25
 BASIC 6-27
 chart of 6-23
 COBOL 6-29
 description 6-23
 optional data, types of 6-23
 programming considerations 6-23
 RPG 6-31
 types of 6-23
 user-supplied data 6-23
 evoke parameter list
 BASIC 6-27
 COBOL 6-29

Intra subsystem (continued)

evoke parameter list (*continued*)
 description 6-23
 RPG 6-31
 examples
 \$EVOK macro (assembler) 6-26
 \$WSIO macro (assembler) 6-26
 acquire operation 6-18
 program-to-program communications 6-3
 RPG evoke operation 6-32
 WRITE statement (BASIC evoke operation) 6-28
 WRITE statement (COBOL evoke operation) 6-30
 fail operation
 description 6-33
 function management headers 6-40
 get attributes operation
 description 6-35
 status information 6-35
 get operations
 description 6-35
 get status operation
 status information 6-36
 input/output operations
 summary chart of 6-46
 invite operation
 description 6-37
 line member 6-8
 location name
 display 22.0 6-9
 LOCATION parameter
 on SESSION statement 6-12
 modifying a configuration 6-10
 multiple configurations 6-5
 name of subsystem member
 defined on display 1.0 6-7
 negative response operations
 description 6-38
 types of 6-38
 OPEN statement example (BASIC) 6-18
 procedure start request
 starts a procedure 6-11
 procedure start requests
 description 6-13
 program-to-program communications
 example of 6-3
 put end of chain operation 6-39
 put end of transaction operation 6-39
 put FMH operations 6-40
 put operations
 description 6-39
 types of 6-39
 release operation
 description 6-41
 request to change direction operations
 description 6-42
 types of 6-42

Intra subsystem (continued)

- return codes
 - acquire operation error codes (82xx) 6-66
 - detailed descriptions of 6-45
 - miscellaneous program error codes (0800-3401) 6-62
 - new requester codes (01xx) 6-51
 - no data received codes (03xx) 6-59
 - normal completion codes (00xx) 6-47
 - output exception codes (04xx) 6-61
 - permanent subsystem error codes (80xx) 6-64
 - recoverable session error codes (83xx) 6-70
 - stop or disable pending codes (02xx) 6-55
 - summary descriptions of B-1, B-2
- SESSION statement
 - description 6-12
 - syntax diagram 6-12
- set timer operation
 - description 6-44
- setting up 6-5
- starting sessions 6-11
 - acquire operation 6-18
- subsystem attributes
 - configuration displays 6-7
- subsystem member
 - configuration displays 6-7
- SYMID parameter 6-18
 - on SESSION statement 6-12
- syntax diagrams
 - SESSION statement 6-12
- user-supplied data
 - evoke operations 6-23
 - using IDDU with 6-4

intrinsic functions, BASIC

- ATTRIBUTE\$ 3-25
- FILENUM 3-11, 3-31
- TIMER 3-21
- WSID\$ 3-4, 3-31

introductions

- acquiring sessions 1-4
- program-to-program communications 1-2
- remotely started sessions 1-6
- SESSION statement 1-5

invite operation

- description
 - Intra 6-37

IOERR parameter

- BASIC return code checking 3-28

L

LAN communications feature

- shared lines 1-13

line member name

- specified on ENABLE command 1-18

line number

- specified on ENABLE command 1-18

line sharing

- X.25
 - summary 1-12, 1-13

link verification

- ICFDEBUG procedure 1-2

location

- specified on SESSION statement
 - Intra 6-12

location name

- name of local System/36
 - Intra 6-9
 - specified on ENABLE procedure command 1-1
 - used with SESSION statement 1-8

LOCATION parameter

- on SESSION statement
 - Intra 6-12

location, remote

- See remote location(s)

LU

- See logical units (SNUF)

M

macroinstructions (macros), assembler

- \$ALOC 2-10
- \$CLOS 2-11
- \$DTFO 2-8
- \$DTFW 2-4
- \$EVOK 2-18
- \$OPEN 2-11
- \$WSIO 2-12
- \$WSIO macro parameter chart 2-16
- for controlling files 2-10
- list of 2-3

manuals

- See also remote system manuals for additional information ix, x

maximum record length

- input operations
 - languages 3-10, 4-10
- put operations
 - languages 3-18, 4-15, 5-5

member, line

See line member

member, subsystem

See subsystem member

modifying a configuration member

Intra 6-10

modifying assembler operations 2-15**MRT programs**

programming considerations (RPG II) 5-22

MSRJE, BSC

See BSC MSRJE

MSRJE, SNA

See SNA MSRJE

N**negative response operations**

description

languages 3-22, 4-19, 5-8

Intra 6-38

examples

BASIC WRITE statement 3-22

COBOL WRITE statement 4-19

RPG 5-8

sending sense data 3-22, 4-19, 5-8

types of

Intra 6-38

negotiable

Lan communications feature 1-13

X.25 shared lines 1-12

NEXT operation (RPG)

description 5-12

example 5-12

NUM continuation option (RPG II) 5-20**O****offset locations (assembler)**

return codes 2-23

OPEN statement (BASIC)

description 3-4

example

Intra 6-18

examples 3-6

identifies session 3-4

specially acquired sessions

examples 3-6

operation codes

evoke

Intra 6-23

summary charts of

assembler 2-21, A-3

operation codes (continued)

summary charts of (continued)

BASIC 3-27, A-4

COBOL 4-22, A-5

RPG II 5-14, A-6

optional data

sent with evoke operations

Intra 6-23

optional parameters, chart of

in \$WSIO macro 2-16

P**parameter list**

sent by \$EVOK macro 2-18

parameter matrix chart

\$WSIO macro 2-16

password

specifying, in assembler 2-19

PDATA parameter, COPY control statement

for sending program data 2-20, 3-16, 4-15

Peer subsystem

communications line

specifying line number 1-18

DISABLE procedure command

functions performed by 1-19

syntax diagram 1-20

ENABLE procedure command

functions performed by 1-15

line number 1-18

syntax diagram 1-18

return codes

summary descriptions of B-1, B-2

performance considerations

improving response time 1-14

problem determination

ICFDEBUG 1-2

procedure parameters

sent to the started procedure 1-7

procedure start request

description

Intra 6-13

starts a session

Intra 6-11

introduction 1-7

types of information sent with 1-7

program cycle, RPG II 5-14**program data**

sent to the started program 1-7

program parameters

sent with evoke operations

Intra 6-23

- program-to-program communications**
 - Intra subsystem 6-3
 - introduction to 1-2
- programming considerations**
 - continuation options, RPG II 5-20
 - end-of-file conditions, RPG II 5-23
 - input/output operations (RPG II) 5-24
 - MRT programs (RPG II) 5-22
 - release operations (RPG II) 5-23
 - RPG II 5-20
 - SRT programs (RPG II) 5-22
 - WORKSTN file restrictions (RPG II) 5-24
- prompting facilities**
 - CNFIGICF procedure
 - Intra 6-6
 - presents applicable lines only
 - Intra 6-6
- put end of chain operation**
 - Intra subsystem 6-39
- put end of transaction operation**
 - Intra subsystem 6-39
- put FMH operations**
 - Intra 6-40
- put operations**
 - description
 - languages 3-18, 4-15, 5-5
 - Intra 6-39
 - examples
 - BASIC WRITE statement 3-18
 - COBOL WRITE statement 4-15
 - RPG 5-5
 - maximum record length
 - languages 3-18, 4-15, 5-5
 - types of
 - Intra 6-39

R

- READ operation (RPG)**
 - description 5-13
 - example 5-13
- READ statement (BASIC)**
 - description 3-8
 - EOF clause
 - example 3-10
 - note on 3-10
 - receiving a message, example 3-9
 - STOP\$ intrinsic function 3-10
 - with REREAD statement 3-10
- READ statement (COBOL)**
 - description 4-10
 - examples 4-11
 - TERMINAL options 4-10
- receiving data**
 - example, COBOL 4-11
 - notes on, BASIC 3-10
 - READ operation
 - RPG 5-13
 - READ statement
 - BASIC 3-8
 - COBOL 4-10
- REL operation (RPG)**
 - description 5-11
 - example
 - languages 5-11
 - programming considerations 5-23
- release operation**
 - description
 - Intra 6-41
 - DROP statement (COBOL) 4-21
 - example
 - languages 5-11
 - programming considerations (RPG II) 5-23
- releasing a session**
 - DROP statement (COBOL) 4-21
 - REL operation (RPG) 5-11
- remote system**
 - See remote system manuals
- remotely started sessions**
 - Intra 6-11
 - elements used in 1-6
 - example, BASIC 3-6
 - introduction to 1-6
 - maximum allowed 1-14
- request to change direction operations**
 - description
 - languages 3-20, 4-17, 5-6
 - Intra 6-42
 - examples
 - BASIC WRITE statement 3-20
 - COBOL WRITE statement 4-17
 - types of
 - Intra 6-42
- requesting change of direction**
 - BASIC WRITE statement 3-20
 - COBOL WRITE statement 4-17
 - RPG description 5-6
- required parameters, chart of**
 - in \$WSIO macro 2-16
- REREAD statement (BASIC) 3-10**
- response time**
 - performance considerations 1-14
- RETCODE\$ (BASIC)**
 - chart of SSP-ICF return codes 3-30
 - contains status of last operation 3-28
 - description of 3-30
- retrying unsuccessful operations**
 - set timer operation
 - Intra 6-44

- return code checking**
 - end of transaction
 - BASIC 3-19
 - COBOL 4-16
- return code processing**
 - assembler 2-23
 - BASIC operations 3-28
 - COBOL 4-23
 - RPG II 5-15
- return codes**
 - acquire operation error codes (82xx)
 - Intra 6-66
 - charts of
 - COBOL 4-23
 - RPG status values 5-19
 - checking of
 - BASIC 3-28
 - COBOL FILE STATUS clause 4-5
 - detailed descriptions of
 - Intra subsystem 6-45
 - ERR codes, BASIC
 - chart of 3-29
 - meaning 3-28
 - EXIT clause needed (BASIC) 3-31
 - INFSR subroutine considerations (RPG) 5-18
 - IOERR parameter needed (BASIC) 3-28
 - miscellaneous program error codes (0800-3401)
 - Intra 6-62
 - new requester codes (01xx)
 - Intra 6-51
 - no data received codes (03xx)
 - Intra 6-59
 - normal completion codes (00xx)
 - Intra 6-47
 - offset locations (assembler) 2-23
 - output exception codes (04xx)
 - Intra 6-61
 - permanent subsystem error codes (80xx)
 - Intra 6-64
 - processing of
 - assembler 2-23
 - COBOL 4-23
 - RPG II 5-15
 - purpose of 1-8
 - recoverable session error codes (83xx)
 - Intra 6-70
 - RETCODE\$, BASIC
 - chart of SSP-ICF codes 3-30
 - contains status of last operation 3-28
 - description of 3-30
 - status values for, RPG 5-19
 - stop or disable pending codes (02xx)
 - Intra 6-55
 - summary descriptions of
 - Intra subsystem B-1, B-2
 - BSCEL subsystem B-1, B-2
 - CCP subsystem B-1, B-2
 - CICS subsystem B-1, B-2

- return codes (continued)**
 - summary descriptions of (continued)
 - Finance subsystem B-1, B-2
 - IMS subsystem B-1, B-2
 - Peer subsystem B-1, B-2
 - SNUF subsystem B-1, B-2
- RPG**
 - coding examples
 - Intra 6-32
 - evoke operations
 - Intra description 6-31
- RPG II**
 - canceling sent records 5-9
 - CFILE continuation option 5-22
 - coding examples
 - ACQ operation 5-10
 - complete program 5-25
 - negative response operation 5-8
 - NEXT operation 5-12
 - put operation 5-5
 - READ operation 5-13
 - REL operation 5-11
 - set timer operation 5-7
 - communications operations
 - ACQ operation 5-10
 - cancel 5-9
 - end of session 5-9
 - evoke 5-3
 - fail 5-9
 - introduction 5-3
 - list of 5-10
 - negative response 5-8
 - NEXT operation 5-12
 - put 5-5
 - READ operation 5-13
 - REL operation 5-11
 - request to change direction 5-6
 - set timer 5-7
 - similar to work station operations 5-2
 - continuation options
 - CFILE 5-22
 - FMTS 5-22
 - ID 5-21
 - IND 5-21
 - INFDS 5-22
 - INFSR 5-21
 - list of, for WORKSTN files 5-20
 - NUM 5-20
 - SAVDS 5-21
 - SLN 5-21
 - end-of-file considerations 5-23
 - ending a session 5-9
 - evoke operations 5-3
 - evoke parameter list 5-3
 - examples
 - evoke operation 5-4
 - file description specifications 5-2

RPG II (continued)

- FMTS continuation option 5-22
- ID continuation option 5-21
- IND continuation option 5-21
- indicating error conditions
 - fail operation 5-9
- INFDS continuation option 5-22
- INFSR continuation option 5-21
- INFSR subroutine
 - return code considerations 5-18
 - return code processing 5-15
- input/output operation considerations 5-24
- introduction 5-2
- MRT program considerations 5-22
- NUM continuation option 5-20
- operation descriptions
 - ACQ operation 5-10
 - introduction to 5-2
 - NEXT operation 5-12
 - READ operation 5-13
 - REL operation 5-11
- operations summary chart 5-14, A-6
- program cycle input 5-14
- programming considerations
 - continuation options 5-20
 - end-of-file conditions 5-23
 - input/output operations 5-24
 - introduction 5-20
 - MRT programs 5-22
 - release operations 5-23
 - SRT programs 5-22
 - WORKSTN file restrictions 5-24
- release operation considerations 5-23
- requesting change of direction 5-6
- return codes
 - INFSR subroutine considerations 5-18
 - processing of 5-15
 - status values 5-19
- SAVDS continuation option 5-21
- sending
 - negative response 5-8
 - sense data 5-8
- sending data
 - put operations 5-5
- SLN continuation option 5-21
- specifying a wait time
 - set timer operation 5-7
- SRT program considerations 5-22
- starting remote programs 5-3
- status values 5-19
 - chart of 5-19
- WORKSTN files considerations 5-24
- WORKSTN operations
 - list of 5-10

S

- SAVDS continuation option (RPG II) 5-21**
- SELECT statement (COBOL)**
 - defines TRANSACTION files 4-4
 - description 4-4
- sending data**
 - evoke operations
 - assembler description 2-20
 - BASIC description 3-16
 - COBOL description 4-15
 - put operations
 - BASIC description 3-18
 - COBOL description 4-15
 - RPG description 5-5
- WRITE statement**
 - BASIC 3-18
 - COBOL 4-15
- sending negative response**
 - BASIC description 3-22
 - COBOL description 4-19
 - RPG description 5-8
- sense data**
 - format of 3-22, 4-19, 5-8
 - sent on negative response operations 3-22, 4-19, 5-8
- sequence diagrams**
 - acquiring sessions 1-4
 - starting sessions remotely 1-6
- session errors, recoverable**
 - error return code descriptions
 - Intra 6-70
- session group name**
 - used with SESSION statement 1-8
- session id**
 - used with SESSION statement 1-8
- session identifier**
 - in SYMID parameter, SESSION statement
 - Intra 6-12
 - on acquire operation
 - Intra 6-18
- SESSION statement**
 - descriptions
 - Intra subsystem 6-12
 - in remotely started sessions 3-6
 - introduction to 1-5
 - not needed in BASIC
 - Intra 6-12
 - syntax diagrams
 - Intra 6-12
- session status**
 - information, description of
 - BASIC 3-26
 - COBOL 4-8

session status (continued)

- received by
 - ATTRIBUTE\$ function (BASIC) 3-25
 - get attributes operation 2-14

sessions

- acquiring 1-4
 - ACQ operation (RPG) 5-10
 - ACQUIRE statement (COBOL) 4-6
 - OPEN statement (BASIC) 3-4
- elements used in 1-3
- maximum active on system 1-14
- multiple levels of 1-7
- receiving data in
 - READ operation (RPG) 5-13
- releasing
 - REL operation (RPG) 5-11
- remotely starting 1-6
- SESSION statement
 - See also SESSION statement 1-5
 - introduction to 1-5
- status information about 3-26, 4-8
- types of
 - remotely started 1-14
 - specially acquired 1-14
 - user-acquired 1-14

set timer operation

- description
 - languages 3-21, 4-18, 5-7
 - Intra 6-44
- examples
 - BASIC WRITE statement 3-21
 - COBOL WRITE statement 4-18
 - RPG 5-7

setting up a subsystem

- Intra 6-5

SEU procedure, end of job menu

- assembler evoke operations 2-20
- BASIC evoke operations 3-16
- COBOL evoke operations 4-15

shared lines

- LAN communications feature 1-13
- Token-Ring Network 1-13
- X.25 1-12

SLN continuation option (RPG II) 5-21**SNA MSRJE**

- non-SSP-ICF subsystem 1-9

SNA 3270

- non-SSP-ICF subsystem 1-9

SNUF subsystem

- communications line
 - specifying line number 1-18
- DISABLE procedure command
 - functions performed by 1-19
 - syntax diagram 1-20
- ENABLE procedure command
 - functions performed by 1-15
 - line number 1-18
 - syntax diagram 1-18

SNUF subsystem (continued)

- return codes
 - summary descriptions of B-1, B-2

specially acquired sessions

- description 1-14
- examples, in BASIC 3-6
- maximum allowed 1-14

SRT programs

- programming considerations (RPG II) 5-22

SSP-ICF

- introduction to 1-2
- acquiring a session, introduction 1-4
- combinations of active subsystems 1-10
- communicating using BASIC 3-2
- communicating using COBOL 4-2
- communicating using RPG II 5-2
- debug program
 - ICFDEBUG 1-2
- elements used in sessions 1-3
- maximum active sessions 1-14
- other subsystem types 1-9
- problem determination 1-2
- procedure start requests, introduction 1-7
- remotely started sessions, introduction 1-6
- session information returned
 - BASIC 3-26
 - COBOL 4-8
- subsystems
 - chart of types 1-9
 - communications features, summary chart 1-11
 - introduction to 1-8
 - storage requirements for 1-14
- timer operations
 - BASIC 3-21
 - COBOL 4-18

starting a procedure

- BASIC WRITE statement 3-14
- COBOL WRITE statement 4-14
- RPG evoke operations 5-3

starting remote programs

- BASIC evoke operations 3-14
- COBOL evoke operations 4-14
- RPG evoke operations 5-3

starting sessions

- Intra 6-11

status information, session

- assembler 2-14
- BASIC 3-25, 3-26
- COBOL 4-8

status values

- RPG II 5-19

STOP\$ intrinsic function (BASIC) 3-10**STOPGRP procedure command 1-5, 1-7****storage requirements**

- for multiple enabled subsystems 1-14

STRGTGRP procedure command 1-7, 1-16

subroutine considerations

- assembler 2-23
- RPG II 5-15, 5-18

subsystem

- See also* subsystems
- communications line features, summary of 1-11
- configuration examples
 - Intra 6-3
- description and capabilities
 - Intra 6-3
- purpose of 1-8
- setting up
 - Intra 6-5

subsystem configuration name

- used with ENABLE command 1-8

subsystem errors, permanent

- return code descriptions
 - Intra 6-64

subsystem member

- configuration displays for
 - Intra 6-7

subsystems

- See also* subsystem
- chart of types 1-9
- combinations of 1-10
- non-SSP-ICF subsystem types 1-9
- storage requirements for 1-14

summary charts

- See also* charts
- communications line features 1-11
- communications operations
 - assembler 2-21, A-3
 - BASIC 3-27, A-4
 - COBOL 4-22, A-5
 - RPG II 5-14, A-6
 - Intra 6-14

SYMID parameter

- on SESSION statement
 - Intra 6-12
- session identifier
 - Intra 6-18

syntax diagrams

- DISABLE procedure command 1-20
- ENABLE procedure command 1-18
- SESSION statement
 - Intra 6-12

System/36

- communications line features
 - chart of 1-11
- communications lines supported 1-10
- main storage requirements 1-14
- maximum active sessions 1-14
- response time 1-14

T

TERMINAL option

- COBOL READ statement 4-10

TIMER intrinsic function (BASIC) 3-21

timer operations

- accept input operation required
 - Intra 6-44
- BASIC 3-11, 3-21
- \$\$TIMER 3-21
- TIMER intrinsic function 3-21

Token-Ring Network

- shared lines 1-13

TRANSACTION file (COBOL)

- defined for SSP-ICF 4-5
- limitation 4-4
- specifying file name 4-6

U

underscores ()

- on configuration displays
 - Intra 6-6

user-acquired sessions

- maximum allowed 1-14

user-supplied data

- evoke operations
 - Intra subsystem 6-23

using IDDU with the Intra subsystem 6-4

V

verify a program

- ICFDEBUG procedure 1-2

W

waiting for input

- BASIC example 3-11
- WAITIO statement (BASIC) 3-11

WAITIO statement (BASIC)

- description 3-11
- examples 3-11
- for timer operations 3-11
- with READ statement 3-8

work station operations for communications

- ACCEPT statement 4-7
- ACQ operation 5-10
- ACQUIRE statement 4-6
 - in BASIC 3-2
 - in COBOL 4-2
 - in RPG II 5-2
- list of, for RPG 5-10
- NEXT operation 5-12
- OPEN statement 3-4
- READ operation 5-13
- READ statement 3-8, 4-10
- REL operation 5-11
- WAITIO statement 3-11
- WRITE statement 3-12, 3-13, 3-20, 4-12, 4-13, 4-17

work station timer operations

- BASIC 3-21
- COBOL 4-18

WORKSTN file (RPG II)

- RPG cycle input 5-14

WORKSTN files

- continuation options for 5-20
- restrictions for SSP-ICF programs (RPG II) 5-24
- used for communications 5-2

WRITE statement (BASIC)

- canceling sent records 3-23
- coding examples
 - languages 3-15
 - Intra 6-28
- cancel operations 3-23
- end of session operation 3-20
- fail operation 3-23
- negative response operations 3-22
- put end of transaction operation 3-19
- put operation 3-18
- request to change direction operations 3-20
- set timer operation 3-21
- timer intrinsic function 3-21
- description of 3-12
- ending a session 3-20
- ending a transaction 3-19
- evoke operations
 - Intra 6-27
- indicating error conditions 3-23
- list of operations performed by 3-13, 3-20
- requesting change of direction 3-20
- sending a negative response 3-22
- sending data 3-18
- starting a procedure 3-14

WRITE statement (COBOL)

- canceling sent records 4-20
- coding examples
 - Intra 6-30
- cancel operations 4-20
- end of session operation 4-17
- fail operation 4-20
- negative response operations 4-19

WRITE statement (COBOL) (continued)

- coding examples (continued)
 - put end of transaction operation 4-16
 - put operation 4-15
 - request to change direction operations 4-17
 - set timer operation 4-18
- description of 4-12
- ending a session 4-17
- ending a transaction 4-16
- evoke operations
 - Intra 6-29
- indicating error conditions 4-20
- list of operations performed by 4-13, 4-17
- requesting change of direction 4-17
- sending a negative response 4-19
- sending data 4-15
- starting a procedure 4-14

writing BASIC programs

- notes on 3-31

WSID\$ intrinsic function (BASIC)

- note 3-31
- purpose of 3-4

Numerics

3270 (BSC) device emulation

- See BSC 3270

3270 (SNA) device emulation

- See SNA 3270

READER'S COMMENT FORM

Please use this form only to identify publication errors or to request changes in publications. Direct any requests for additional publications, technical questions about IBM systems, changes in IBM programming support, and so on, to your IBM representative or to your IBM-approved remarketer. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

- If your comment does not need a reply (for example, pointing out a typing error), check this box and do not include your name and address below. If your comment is applicable, we will include it in the next revision of the manual.
- If you would like a reply, check this box. Be sure to print your name and address below.

Page number(s):

Comment(s):

Please contact your IBM representative or your IBM-approved remarketer to request additional publications.

Name

Company or
Organization

Address

City

State

Zip Code

Phone No.

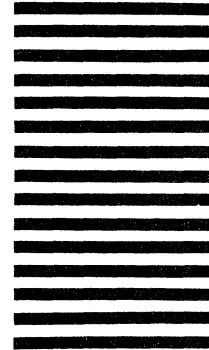
Area Code

Fold and tape. **Please do not staple.**

Cut Along Line



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST CLASS / PERMIT NO. 40 / ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Information Development
Department 245
Rochester, Minnesota, U.S.A. 55901

Fold and tape. **Please do not staple.**

Cut Along Line





Interactive Communications Feature:
Programming for Subsystems and Intra
Subsystem Reference

International Business Machines Corporation

File Number
S36-30

Order Number
SC21-9533-0

Part Number
59X4005

Printed in U.S.A.

SC21-9533-00

