SC21-7751-3

File No. S34-36

IBM System/34
Interactive Communications Feature
Reference Manual

Program Number 5726-SS1

SC21-7751-3

File No. S34-36

IBM System/34
Interactive Communications Feature
Reference Manual

Program Number 5726-SS1

Fourth Edition (January 1982)

This is a major revision of, and obsoletes, SC21-7751-2 and Technical Newsletter SN21-8154. This manual has been changed extensively. A new chapter has been added to describe BASIC support for SSP-ICF, the manual has been reorganized, and the coding examples and return codes description have been completely rewritten.

This edition applies to release 8 of the IBM System/34 System Support Program Product (Program 5726-SS1) and to all subsequent releases until otherwise indicated in new editions or technical newsletters.

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

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

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

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

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

Preface

This manual is intended primarily for application programmers who must write interactive communications programs. The manual also contains information for System/34 system programmers and host system programmers. This manual serves as both a reference manual and a programmer's guide, giving detailed reference material as well as concepts, programming considerations, and examples.

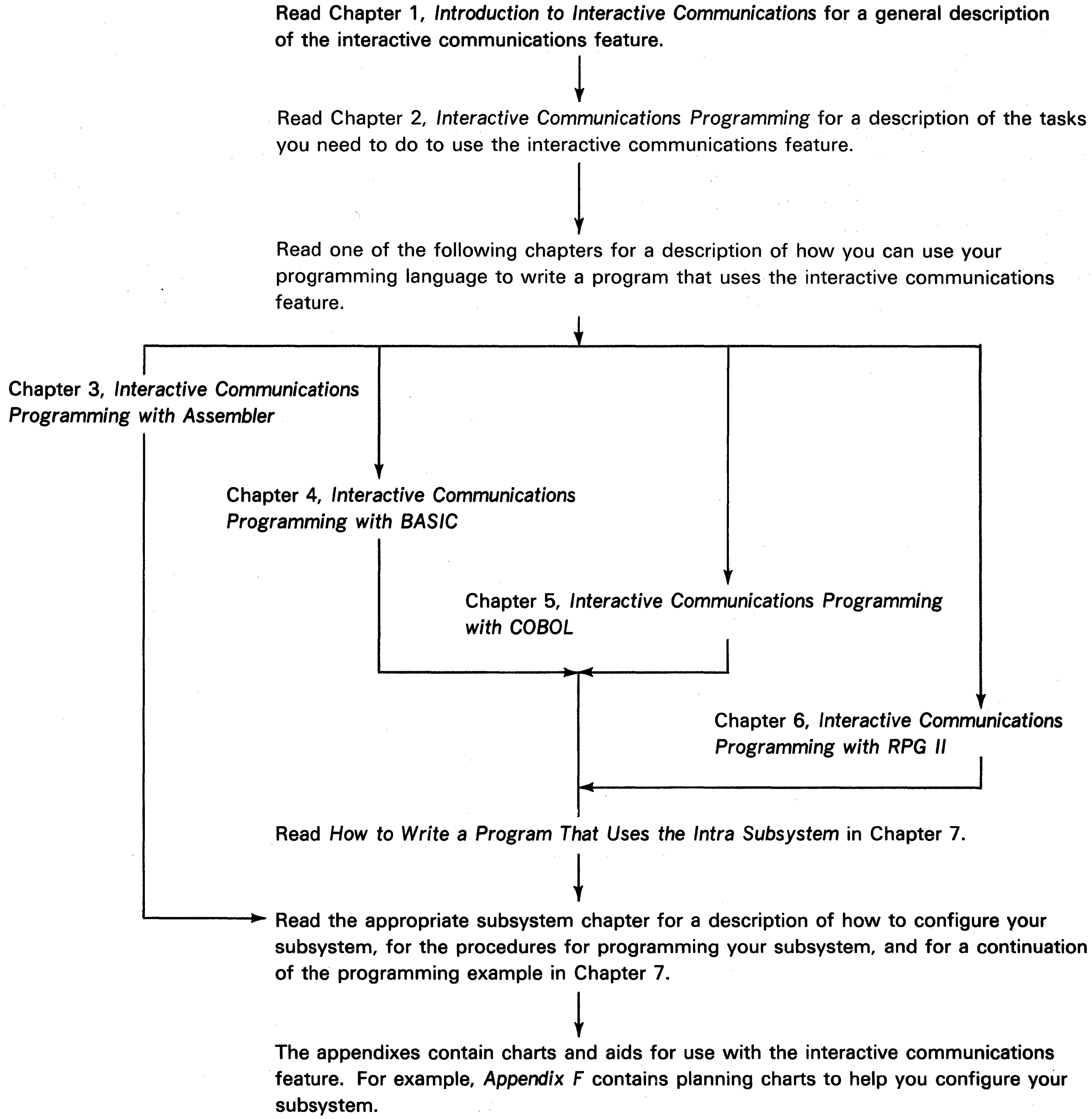
Before reading this manual, you should be familiar with System/34 programming terminology, particularly work station programming, and you should be able to program in whatever language you intend to use. In some instances, you must also be familiar with the terminology of the remote system. The terms introduced in this manual are defined in the glossary.

This manual contains the following chapters:

- *Chapter 1. Introduction to Interactive Communications,* which describes interactive communications in general.
- *Chapter 2. Interactive Communications Programming,* which gives an overview of interactive communications programs and how they are constructed and run.
- *Chapter 3. Interactive Communications Programming with Assembler,* which describes the assembler programming support for interactive communications.
- *Chapter 4. Interactive Communications Programming with BASIC,* which describes the BASIC programming support for interactive communications.
- *Chapter 5. Interactive Communications Programming with COBOL,* which describes the COBOL programming support for interactive communications.
- *Chapter 6. Interactive Communications Programming with RPG II,* which describes the RPG II programming support for interactive communications.
- *Chapter 7. The Intra Subsystem,* which provides programming information for using the Intra subsystem to communicate with programs in the same system.
- *Chapter 8. The BSCEL Subsystem,* which provides programming information for using the BSCEL (BSC equivalence link) subsystem to communicate with application programs on another System/34 and other BSC systems.
- *Chapter 9. The BSC CCP Subsystem,* which provides programming information for using the BSC CCP subsystem to communicate with System/3 Model 15 CCP application programs.
- *Chapter 10. The BSC CICS Subsystem,* which provides programming information for using the BSC CICS subsystem to communicate with CICS/VS application programs.
- *Chapter 11. The BSC IMS/IRSS Subsystem,* which provides programming information for using the BSC IMS/IRSS subsystem to communicate with IMS/VS IRSS application programs.
- *Chapter 12. The BSC 3270 Support Subsystem,* which provides programming information for using the 3270 BSC support subsystem to communicate with CICS/VS, IMS/VS, or CCP application programs using 3270 BSC protocols.
- *Chapter 13. The Finance Subsystem,* which provides programming information for using the Finance subsystem to communicate with the 3601 Finance Controller and devices attached to the 3601, and the 3694 Document Processor.
- *Chapter 14. The SNA Peer Subsystem,* which provides programming information for using the SNA peer subsystem to communicate with other System/34s.
- *Chapter 15. The SNA Upline Facility Subsystem,* which provides programming information for using the SNA upline facility (SNUF) subsystem to communicate with IMS/VS and CICS/VS application programs in SNA networks.

Note:* Throughout this manual, the term *remote system refers to the system or device with which the System/34 is communicating. When the Intra subsystem is being used, remote system refers to the same System/34 because the Intra subsystem is used to communicate with another program on the same system.

HOW TO USE THIS MANUAL



Read Chapter 1, *Introduction to Interactive Communications* for a general description of the interactive communications feature.

Read Chapter 2, *Interactive Communications Programming* for a description of the tasks you need to do to use the interactive communications feature.

Read one of the following chapters for a description of how you can use your programming language to write a program that uses the interactive communications feature.

Chapter 3, *Interactive Communications Programming with Assembler*

Chapter 4, *Interactive Communications Programming with BASIC*

Chapter 5, *Interactive Communications Programming with COBOL*

Chapter 6, *Interactive Communications Programming with RPG II*

Read *How to Write a Program That Uses the Intra Subsystem* in Chapter 7.

Read the appropriate subsystem chapter for a description of how to configure your subsystem, for the procedures for programming your subsystem, and for a continuation of the programming example in Chapter 7.

The appendixes contain charts and aids for use with the interactive communications feature. For example, *Appendix F* contains planning charts to help you configure your subsystem.

PREREQUISITE PUBLICATIONS

The following manuals should be read, or the equivalent knowledge obtained, before reading this manual:

- *IBM System/34 Introduction,* GC21-5153
- *IBM System/34 Planning Guide,* GC21-5154
- *Data Communications Concepts,* GC21-5169

The following manuals must be used in conjunction with this manual:

- *IBM System/34 System Support Reference Manual,* SC21-5155
- *IBM System/34 RPG II Reference Manual,* SC21-7667
- *IBM System/34 Basic Assembler and Macro Processor Reference Manual,* SC21-7705
- *IBM System/34 COBOL Reference Manual,* SC21-7741
- *IBM System/34 BASIC Reference Manual,* SC21-7835

RELATED PUBLICATIONS

- *IBM System/34 Installation and Modification Reference Manual,* SC21-7689
- *IBM System/34 Data Communications Reference Manual,* SC21-7703
- *IBM System/34 System Data Areas and Diagnostic Aids Manual,* LY21-0049
- *IBM System/34 Operator's Guide,* SC21-5158
- *IBM System/34 3270 Device Emulation Program Product User's Guide,* SC21-7868
- *Systems Network Architecture Reference Summary,* GA27-3136
- *3270 Information Display System Components Description Manual,* GA27-2749
- *IBM System/3 CCP Messages Manual,* GC21-5170
- *IBM System/3 Communications Control Program System Reference Manual,* GC21-7620
- *CICS/VS Messages and Codes Manual,* SC33-0081
- *Advanced Communications Function for VTAM (ACF/VTAM) System Programmer's Guide,* SC38-0258
- *IMS/VS Advanced Function for Communications,* SH20-9054
- *IMS/VS Version 1 Installation Guide,* SH20-9081
- *Customer Information Control System/Virtual Storage 3790 Guide,* SC33-0075
- *CICS/VS System Programmer's Reference Manual,* SC33-0069
- *CICS/VS Application Programmer's Reference Manual (Command Level),* SC33-0077

Notes:
1. This manual contains several references to an SNA 3270 subsystem and to 3270 device emulation. These topics apply only to the 3270 Device Emulation Program Product, and are described in the *3270 Device Emulation User's Guide.*
2. The System/34 application programmer might be concerned with line protocols and internal subsystem logic. The line sequences and control flows for each operation are documented in the subsystem chapters of the *IBM System/34 Interactive Communications Feature Program Logic Manual,* LY21-0581.

This page is intentionally left blank.

Contents

Chapter 1. Introduction to Interactive Communications

The System/34 interactive communications support allows distributed processing to be implemented on System/34. The interactive communications support is designed to be easy to learn and use. It is provided as a feature of the System/34 System Support Program Product.

The interactive communications feature includes support for program-to-program communications using BSC and SNA as well as communications between programs within the same system. The feature also allows programs on other systems to initiate System/34 procedures and allows System/34 programs to initiate programs or procedures on other systems without remote system operator intervention. These other systems include System/3, System/370, and System/34. To facilitate incoming procedure requests, the interactive communications support can maintain a connection on a communications line when no System/34 application program is active.

Past communications support required that the application program control the format of the data passed on the communications line. This meant that the programmer required familiarity with the line protocol (BSC, for example). Above the BSC or SDLC line protocol can be another level of protocols (IRSS or SNA) with still other control programs such as IMS, CICS, VTAM, and CCP above them. The interactive communications feature isolates the application program from all these levels of protocol, thereby simplifying the effort required to write System/34 application programs that use communications.

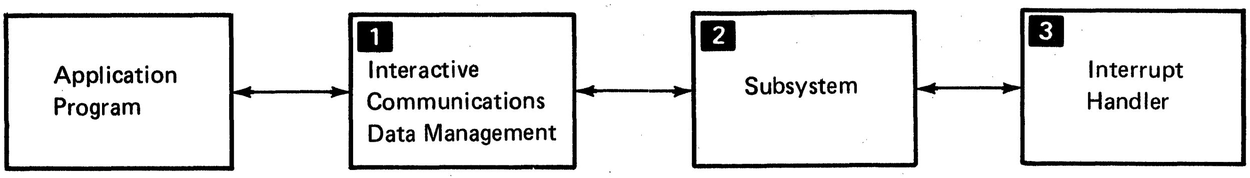
Previous System/34 data communications support (RPG BSC support, MRJE, and SRJE) is designed primarily for *batch* communications. The interactive communications feature is designed primarily for *interactive* communications. Interactive communications differs conceptually from batch communications in that in interactive communications, the sequence of events is not necessarily predetermined; either program can logically start, alter, or stop the communication. In addition, batch communications is characterized by the transfer of large quantities of data in one direction, whereas interactive communications usually consists of a brief exchange of data (such as an inquiry and a response).

The application programming interface to the interactive communications feature is provided via enhancements to the assembler macroinstructions, the BASIC work station support, the COBOL work station support, and the RPG II work station support. This interface shields the application program from most of the differences in communications protocols and remote communications support. With proper design, a user can develop programs for an existing BSC network and then move them to an SNA network without changing the communications code within the application program.

Because the interactive communications feature uses work station support, programmers familiar with work station operations will require very little training to successfully write programs that use the interactive communications feature. Also, many of the options (such as read under format and local data area) available for work station programming are also available for communications. The same statements that control input and output for display stations can also be used for communications. An inquiry program, for example, can receive input from either a display station or a remote program as a result of a single input operation.

STRUCTURE OF THE INTERACTIVE COMMUNICATIONS FEATURE

The interactive communications feature consists of interactive communications data management, specific subsystem support for communicating with different systems, and interrupt handlers.



1 *Interactive communications data management* is the interface between an application program and the subsystem. Interactive communications data management is similar to work station data management; application programs perform interactive communications operations in the same way that they perform work station operations.

2 Because communications can occur with different systems and each has different protocols, a *subsystem*, designed for a specific remote system, isolates most system-dependent considerations from the application program. The following subsystems are provided:

Subsystem	**Communicates With**
Intra	Other programs in the same System/34
BSC Equivalence Link	System/34, System/32, and others
BSC CCP	System/3 Model 15 CCP
BSC CICS	CICS/VS (BTAM)
BSC IMS/IRSS	IMS/VS via IRSS (BTAM)
BSC 3270	IMS/VS, CICS/VS, and System/3 CCP
Finance	3601 Finance Controller and 3694 Document Processor
SNA Peer	System/34
SNA Upline Facility	CICS/VS or IMS/VS
SNA 3270[1]	IMS/VS and CICS/VS

3 An *interrupt handler* is the interface between the subsystem and the communications hardware. A BSC interrupt handler controls communication on the line for the BSC subsystems. (The BSC 3270 support subsystem has a different BSC interrupt handler than the one that supports the other BSC subsystems.) For the SNA subsystems, an SDLC task controls communication on the line. The SDLC task could be primary or secondary SDLC, or both. The SNA Upline Facility (SNUF) subsystem requires an SNA task, which is the interface between the subsystem and the SDLC task. The Intra subsystem has no interrupt handler because the Intra subsystem does not use a communications line.

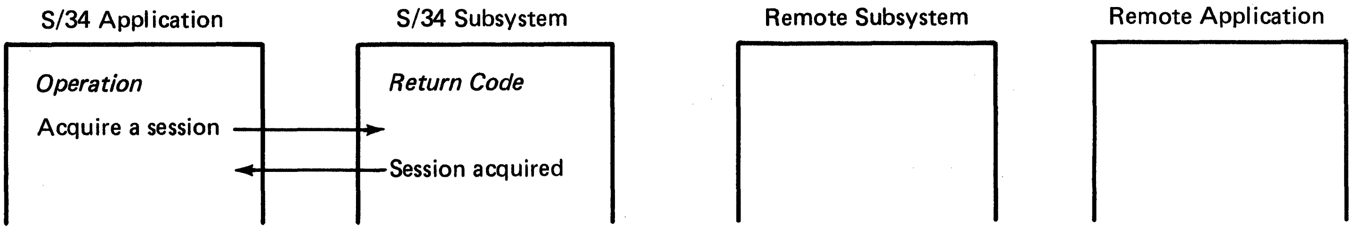
[1] The SNA 3270 subsystem is not described in this manual because it is not supported by SSP-ICF. It is supported by the 3270 Device Emulation Program Product, and it is described in the *3270 Device Emulation User's Guide*.

SESSIONS AND TRANSACTIONS

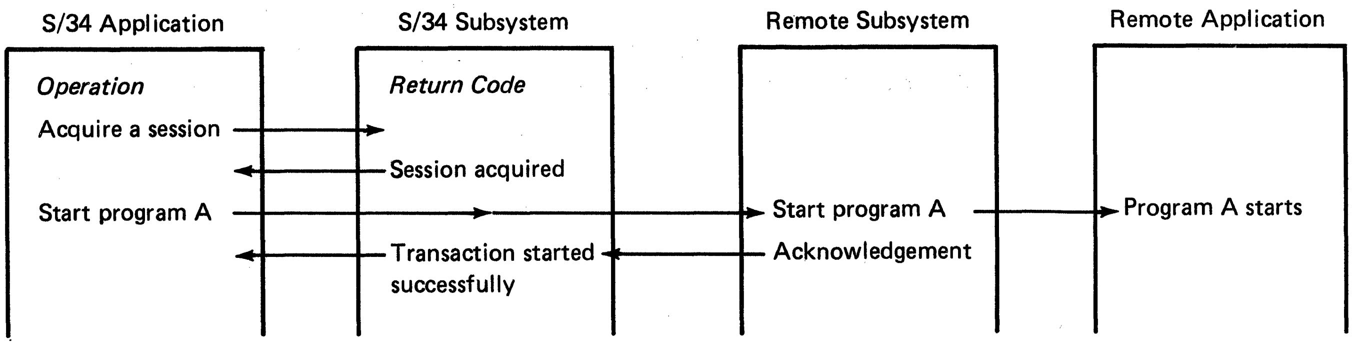
When using the interactive communications feature, each application program communicates through a session using a transaction. A session is a logical connection or pipeline to the remote system. A transaction is the communications between the System/34 application program and another application program. A session must exist before a transaction can take place.

Sessions are defined at both systems during configuration. The connection allowing these sessions is established when the subsystem is started (enabled). A session can then be started by a System/34 application program or by a remote application program. Depending on how the session was started, the characteristics of the session are different.

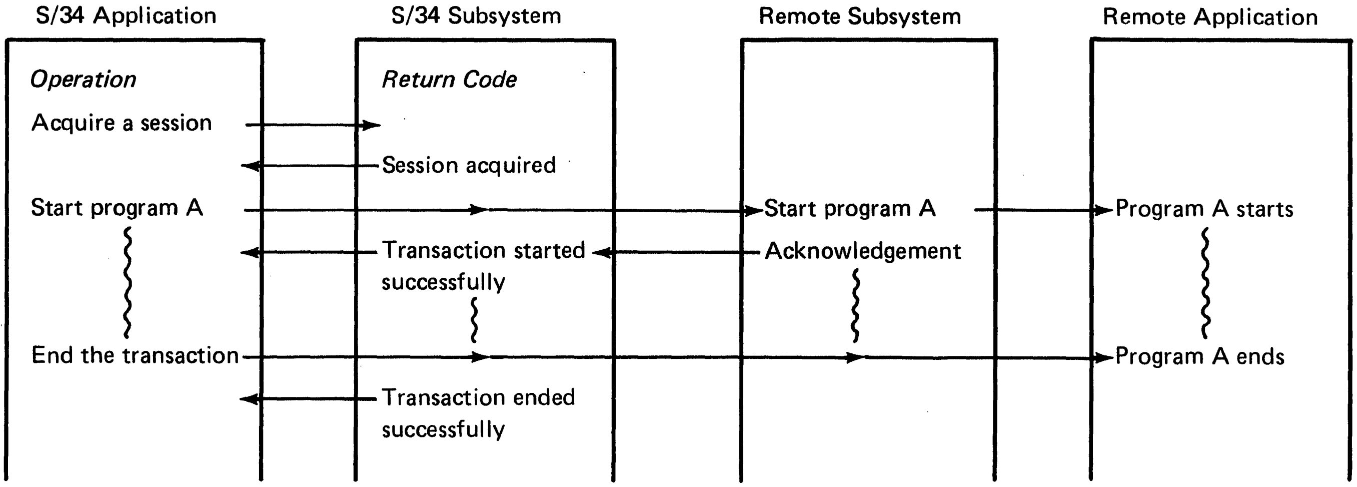
If a System/34 program requests (acquires) a session, the session is called an *acquired session.* When the System/34 application program acquires a session from the subsystem through interactive communications data management (not illustrated), the subsystem assigns or reserves a session for that application program. An indication (return code) is then given to the application program indicating the success or failure of the acquire.



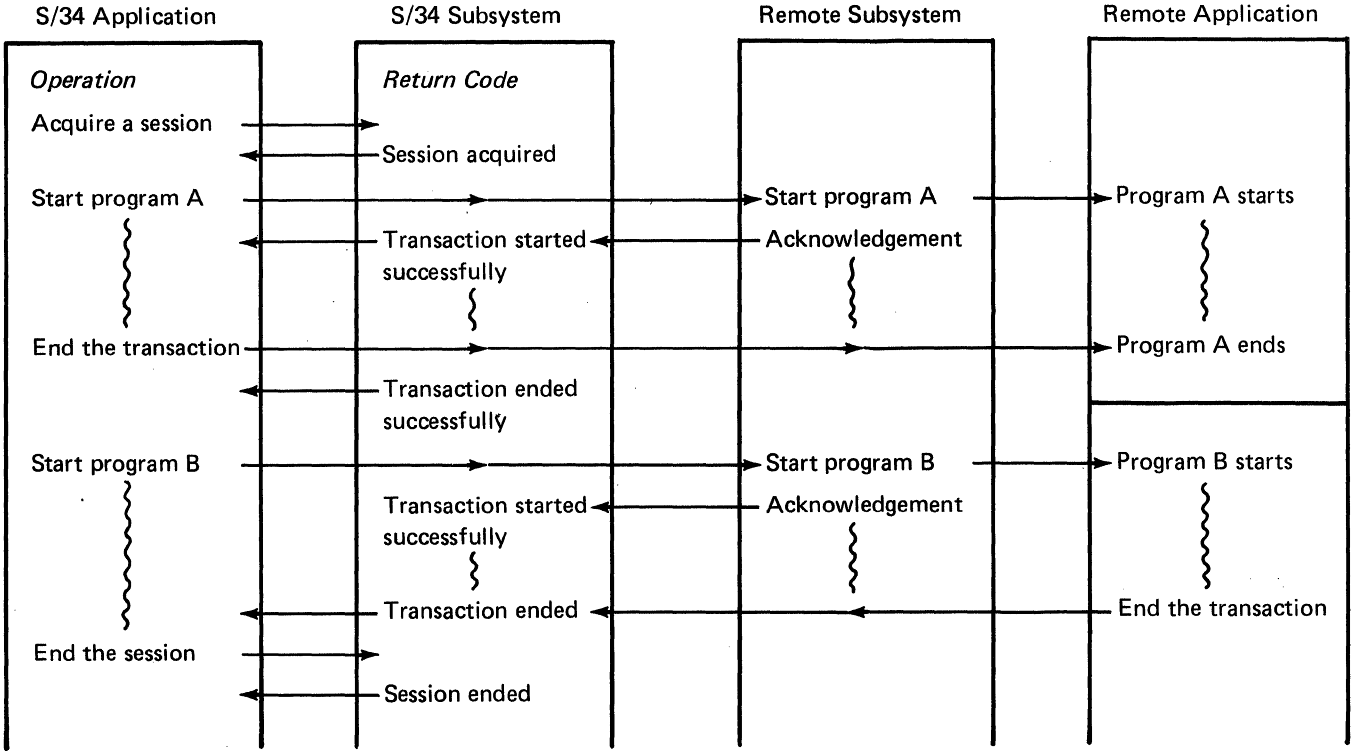
After the session has been started, communications with the remote application can begin. The System/34 application program requests the subsystem to start the remote application and thereby begin a transaction with that application program.



The transaction remains active as long as communication between the two application programs continues. Either program can end the transaction.

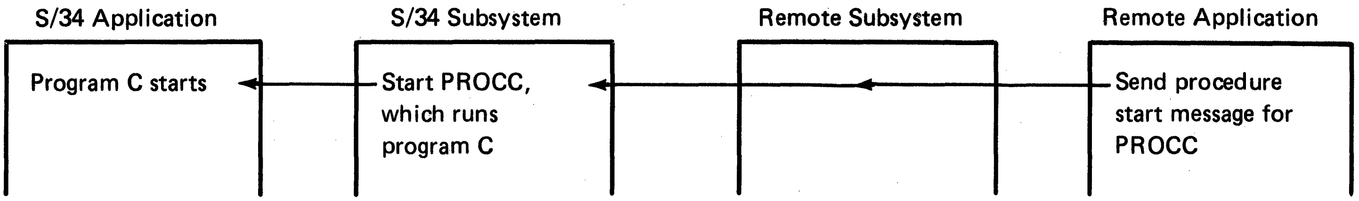


When the transaction ends, the session still exists. The System/34 application program can start another transaction on the same session, and can continue to request transactions consecutively until all desired transactions are complete and the session is ended by either application program.

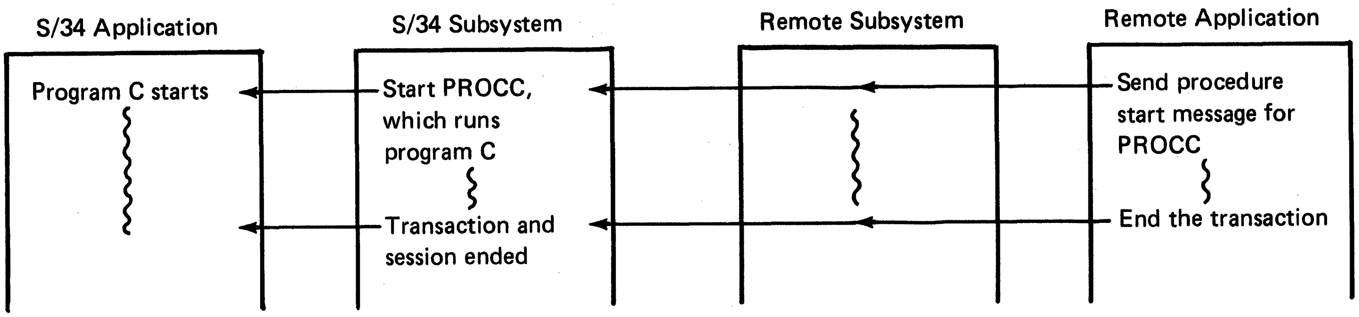


An acquired session remains active for as many transactions as desired until the application program requests that it be terminated (or an error causes the session to be abnormally terminated).

A session can also be started by an incoming procedure request. For this type of session, an application sends a specially formatted message to the System/34 subsystem requesting that a procedure be started. As a result, an application program on the System/34 begins running and communication between the two applications can begin.



When a session is started by an incoming procedure request, the transaction begins when the session begins. The transaction ends whenever either program requests an end of transaction. The session consists of only one transaction and ends when the transaction ends.


S/34 Application
S/34 Subsystem
Remote Subsystem
Remote Application
Program C starts
Start PROCC, which runs program C
Send procedure start message for PROCC
Transaction and session ended
End the transaction

STORAGE REQUIREMENTS

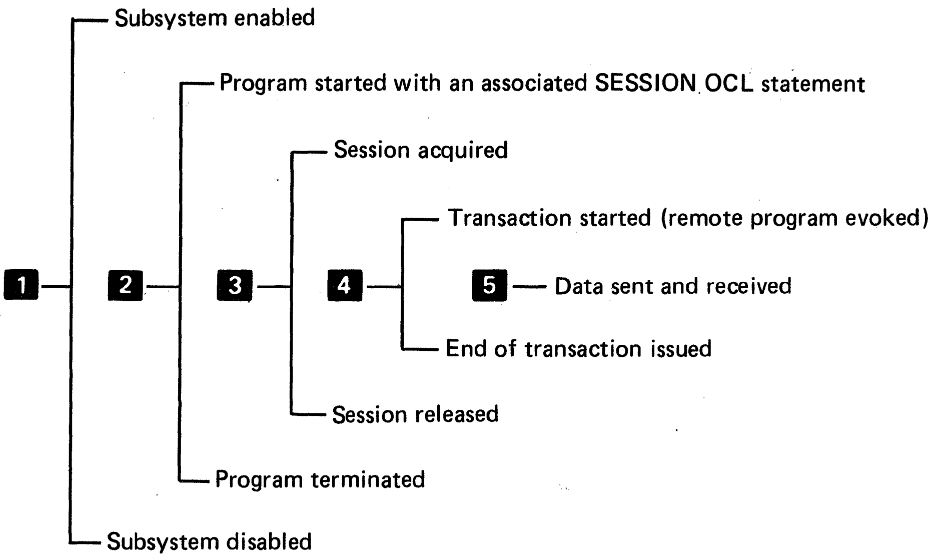
To use any of the interactive communications feature subsystems that require a communications line, you must have a minimum of 64 K bytes of main storage; if you use only the Intra subsystem, however, the minimum is 48 K bytes of main storage. To make full use of the capabilities of the interactive communications feature, such as multiple concurrent sessions, or to improve performance, you should have at least 96 K bytes of main storage. Specific information on main storage and secondary storage requirements of the interactive communications feature is available in the *Planning Guide* and the subsystem chapters of this manual.

Chapter 2. Interactive Communications Programming

This chapter describes the elements of interactive communications programming that concern the application programmer. These elements are:

- Configuring the interactive communications environment
- Enabling the subsystem
- SESSION OCL statement
- Issuing interactive communications operations
- Starting a session
- Performing input and output
- Performing other operations
- Checking return codes
- Ending a session
- Remote initiation of procedures
- Disabling the subsystem

The following illustration shows the hierarchy of levels that exist during normal operation of System/34 interactive communications programs. Each of these levels is described in more detail later in this chapter.


Subsystem enabled
Program started with an associated SESSION OCL statement
Session acquired
Transaction started (remote program evoked)
1
2
3
4
5
Data sent and received
End of transaction issued
Session released
Program terminated
Subsystem disabled

1 Before any interactive communications programs begin sessions, a subsystem configuration must be enabled. Certain configuration parameters can be modified during the enable. The enabled configuration remains active until it is disabled.

2 Associated with each session to be started by the program is a SESSION OCL statement. This statement defines subsystem-dependent parameters as well as the session ID and location name (described later in this section) associated with the session. The parameters remain in effect until the program terminates.

3 Within each application program, sessions can be started (acquired). A session allows communication between the System/34 application program and its subsystem. The session remains active until it is released.

4 Within each session, transactions can be started (evoked) to allow communications with the remote application. Transactions are started by evoking a remote application program. Communication between the two programs continues until an end of transaction occurs.

5 Within each transaction, data can be sent and received by the program.

Each level 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 reenabling the subsystem configuration.

Levels 2, 3, and 4 are specifically for acquired sessions. Sessions started by incoming procedure requests do not require SESSION OCL statements, cannot be acquired, and cannot evoke transactions. These functions are performed by the remote application with its procedure start request. In this case, the remote application fulfills the role of a display station operator.

The connection between the levels is maintained by several parameters. The subsystem configuration name identifies the particular subsystem configuration. The ENABLE procedure specifies the subsystem configuration name to be enabled. The location name, specified during configuration, is included on the SESSION OCL statement to identify the location being referenced. Also specified on the SESSION OCL statement is the symbolic session ID. This is the same session ID specified when the session is acquired by the application program. Thus, the configuration can be changed without affecting the application program.

CONFIGURING THE INTERACTIVE COMMUNICATIONS ENVIRONMENT

The first step in preparing to run interactive communications programs is configuration. During system configuration, you specify whether you want interactive communications in your system. If you do, you must run a special interactive communications procedure called CNFIGICF. This procedure prompts you for the subsystem types (CCP, for example) and for specific parameters for each subsystem type. You can define multiple configurations of each type, each identified by a unique name. The details, including the prompts and parameter descriptions, are described in the appropriate subsystem chapter later in this manual.

ENABLING THE SUBSYSTEM

To run an application program that uses interactive communications, you must activate (enable) the particular subsystem configuration that you want to use. An application program that uses interactive communication can be loaded before the subsystem is activated, but no sessions can be started. The Peer subsystem also allows enabling of a specific location. See Chapter 14 for details.

An ENABLE procedure activates the subsystem and allows some modification of configuration variables. The ENABLE procedure performs the following functions:

- Determines whether the line requested is available.
- Loads and attaches the subsystem if it is not already active.
- Loads and attaches any other required tasks (BSC or SNA and SDLC) if they are not already active.
- Assigns storage for required data areas and buffers.

After the subsystem is enabled, programs can begin using that subsystem. You can enable a subsystem by having the ENABLE procedure automatically run after IPL. See the *Installation and Modification Reference Manual* for a description of how to specify a procedure to be run automatically after IPL.

Note: If a subsystem is enabled, procedures that require a dedicated system, such as COMPRESS, cannot be run.

The format of the ENABLE procedure command is:

```
ENABLE name , ┌#LIBRARY    ┐ , line number , ┌SHOW  ┐ , ┌location┐
              └library name┘                 └NOSHOW┘   └        ┘
```

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

library name: Specifies the name of the library that contains the specified subsystem configuration. The default is #LIBRARY.

line number: Specifies the number of the communications line for which this subsystem is to be enabled. This parameter can be omitted for the Intra subsystem.

SHOW/NOSHOW: Specifies whether subsystem configuration parameters are to be displayed before the subsystem is enabled. If SHOW is specified, they are displayed, and some of the configuration parameters can be changed.

location: Specifies the name of the remote location to be enabled. Location name can be specified only if the subsystem being enabled is a Peer subsystem. This name must have been specified as a remote location name during subsystem configuration.

SESSION OCL STATEMENT

Each program that acquires an interactive communications session must have a SESSION OCL statement associated with that session.[1] SESSION OCL statements are similar to WORKSTN OCL statements and must appear between the LOAD and RUN statements. The SESSION statement references the subsystem configuration that the application program uses and the ID of the session to be acquired.

Each subsystem has its own specific parameters on the SESSION OCL statement.

The format of the SESSION OCL statement is:

```
// SESSION LOCATION-name,SYMID-session-id
   [optional subsystem-dependent parameters]
```

LOCATION: Specifies the location name associated with this session. The location name is defined during subsystem configuration, and refers to the name of the remote location with which communication is to take place.

SYMID: Specifies the symbolic ID of the session with which this OCL statement is associated. The symbolic ID must be two characters, with the first character numeric (0 through 9) and the second character alphabetic (A through Z, #, $, or @). This is the same ID that the application program uses when referring to this session. This ID is the equivalent of a symbolic display station ID as specified on the WORKSTN OCL statement. This parameter has no default.

The optional subsystem-dependent parameters are described in each of the subsystem chapters.

[1] If the program is a BASIC program and the subystem is Intra, Peer, or CCP, a SESSION statement might not be required. See Chapter 4, *Interactive Communications Programming with BASIC* for more information.

ISSUING INTERACTIVE COMMUNICATIONS OPERATIONS

Interactive communications data management has a set of operations to establish and control sessions and transactions. The following sections include a general description of each operation. The operations are issued differently in BASIC, COBOL, and RPG II than they are in assembler. For more details on how to use these operations, see the programming language chapters. For specific subsystem considerations about each operation, see the chapter on the pertinent subsystem.

Each language chapter also contains a language summary chart that lists all the operations that are valid for that language and indicates all the subystems for which each operation is valid. The chart also shows the operation codes (mnemonics) that must be coded in that language to perform the operations. You will probably want to refer to one of these language summary charts when you are coding a program in a particular language and subsystem.

Appendix A contains a different type of operations summary chart. The *Input/Output Operations Summary Chart* shows all of the operations (for all the languages) as input, output, or combined input/output operations, and indicates in which subystem each operation is valid.

STARTING A SESSION

After the subsystem has been enabled, communications can be started. Two operations are required to begin communicating:

- Acquire, to start the session

- Evoke, to start the transaction

Acquire Operation

The acquire operation establishes a session. Associated with the acquire is a session ID (corresponding to the SYMID parameter given on the SESSION OCL statement) that identifies this session. When the acquire operation completes successfully, a session with this ID exists. Communication with the remote system might or might not have been started (depending on the subsystem), but the session is reserved for this program.

Evoke Operation

The evoke operation starts a procedure or an application program on the remote system and, thereby, begins a transaction. An evoke can occur only after a session has been acquired, but multiple evokes can be issued in each session if the previous transaction has ended before the next evoke is issued. Several types of evokes can be issued:

- *Evoke* issues an evoke operation and then waits until the remote system acknowledges the operation. The issuing application program can then begin transmitting data.
- *Evoke then invite* issues an evoke followed by an invite. The invite enables the remote application program to send data first for this transaction. (See *Invite Operation* later in this chapter.) Control is returned to the issuing application program without waiting for completion of the evoke.
- *Evoke then get* issues an evoke and then waits for input from the evoked application. (See *Get Operation* later in this chapter.)
- *Evoke end of transaction* issues an evoke and then ends the transaction. This means that no further communication takes place between the issuing program and the evoked program. See the special considerations in *Remote Initiation of Procedures* later in this chapter if you will be using evoke end of transaction to initiate a System/34 procedure.

Associated with each evoke is an evoke parameter list, which contains the procedure name, library name, password, and user ID associated with the program to be evoked. The evoke operation can optionally include data to be sent to the remote application. See *Remote Initiation of Procedures* later in this chapter for more information about evoking System/34 procedures.

The evoke operation can include a function management header. See Chapter 15 for a description of function management headers.

PERFORMING INPUT AND OUTPUT

After the session and transaction have been established, input and output operations can be performed. The following operations are available for performing input and output:

- Put, to send records
- Invite, to allow input and accept, to obtain invited input
- Get, to request input

Put Operation

The put operation passes data from the issuing program to the remote application program. The following types of put operations can be issued:

- *Put* issues a put to the subsystem and returns control to the application program without waiting for the operation to complete. If multiple put operations are issued, the current put operation is not started until the previous put operation is complete. If the previous put operation failed, the current put is not performed and the application program is informed via the appropriate return code.
- *Put then invite* issues a put followed by an invite. The invite allows the remote application program to begin sending data on this session. (See *Invite Operation* later in this chapter.) Control is returned to the application program without waiting for the remote system to send the data.
- *Put then get* issues a put operation and then waits for the remote application program to send data. (See *Get Operation* later in this chapter.)
- *Put end of chain or put end of file* issues a put operation that indicates to the remote system that this is the last record in a group of data. The put end of chain operation is used for the Intra, SNUF, and Peer subsystems. The put end of file operation is used for the BSC subsystems. Control is returned to the user program after the remote system acknowledges receipt of the end of chain, or after end of file is sent by the BSC interrupt handler. The put end of file and put end of chain operations translate to the same operation code (and are issued the same way in BASIC, COBOL, and RPG II programs); therefore, these operations need not be recoded when going from BSC to SNA or vice versa. See Appendix C for more information concerning BSC to SNA migration.

- *Put function management header* issues a put, put then invite, or put then get operation and indicates to the remote application program that a function management header is included in the data. Put function management header is valid only for the Intra and SNUF subsystems.

- *Put end of transaction* issues a put operation and then indicates to the remote application program that this transaction is ended and no more communication will take place between the two applications. Control is returned to the user program after the remote system acknowledges receipt of the end of transaction.

Invite Operation

The invite operation asks for input data, but the issuing program receives control without waiting for the input. To obtain the data, the user program must subsequently issue an accept or get operation. The invite operation can be issued alone or as a modifier with an evoke, put, cancel, negative response, or request change direction operation. If the invite operation is issued as a modifier for an operation that is not supported by a subsystem, the invite is ignored.

Accept Operation

The accept operation allows the issuing program to obtain data from any previously invited program or display station, to allow new requesters, or to check whether the timer has expired (see *Set Timer Operation* later in this chapter). If data is available from more than one display station or program, the data received by the program is the data from the first display station or program that sent it. An accept to receive data should be issued only after a previous invite or set timer request; however, if the program is a MRT NEP, an accept without a previous invite or set timer request allows the program to wait for a new requester.

Get Operation

The get operation provides the issuing program with data from a specific program. The issuing program receives control when data is available. The get operation differs from the accept operation in that the get operation is directed to a specific program (or display station), whereas an accept operation allows input to be obtained from any previously invited session or display station.

PERFORMING OTHER OPERATIONS

Most programs can be written using only the previous input and output operations. However, if additional functions are required, the following operations are available:

- Request to change direction
- Negative response
- Fail
- Cancel
- Set timer
- Get attributes
- Pass-through operations

Request to Change Direction Operation

The request to change direction operation indicates that the issuing program wants to transmit data. The operation can only be issued by a program that is receiving. The request to change direction operation has two forms: request to change direction then invite and request to change direction then get. After issuing either operation, the issuing program should continue to receive data until it receives a return code indicating that the remote program is ready to begin receiving. The issuing program can then begin sending data.

A user program that receives a request to change direction is notified via a return code following a put operation. (See *Checking Return Codes* later in this chapter for a description of return codes.)

Negative Response Operation

The negative response operation sends a negative response to the remote application. A negative response indicates that the application program detected something wrong with the data received. The response can include eight characters of sense information to inform the remote system of the reason for the negative response. The negative response operation can be issued alone or with a get or invite. The negative response operation should be used only when receiving data from the remote program.

A user program that receives a negative response is notified via a return code following a put operation. (See *Checking Return Codes* later in this chapter for a description of return codes.) The only valid response to a negative response is a cancel.

Fail Operation

The fail operation indicates to the remote program that an abnormal condition has occurred within the application program. The fail operation can be issued while the program is sending or receiving. If a program issues a fail operation while sending, it indicates that the data just sent was in error. All data sent before the fail operation is transmitted to the receiving program, and a return code indicating the fail is given to the receiving program. If a program issues a fail operation while receiving, it indicates that the data received was in error. The subsystem discards all subsequent data until the transmitting subsystem acknowledges receipt of the fail operation. In either case, the program that issued the fail operation must transmit, and the program that receives the fail return code must receive. If both programs issue a fail operation simultaneously, the program that was receiving will be successful and must transmit. The program that was transmitting will receive an unsuccessful return code and must begin receiving. No data can accompany the fail operation.

Cancel Operation

The cancel operation sends a cancel command to the remote program. The cancel command indicates to abnormally end this group (chain) of data records and to disregard previous records in this group (all records sent since the previous end of chain). The cancel operation can be issued alone or with an invite or a get. The cancel operation should be issued only while transmitting data. A cancel operation does not end a session; recovery from a cancel operation depends on the subsystem.

A user program that receives a cancel is notified via a return code. (See *Checking Return Codes* later in this chapter for a description of return codes.)

The cancel and negative response operations can be considered as a pair. Cancel is the appropriate response when a negative response is received. However, if the transmitting program discovers an error, cancel can be sent without first receiving a negative response.

Set Timer Operation

The set timer operation specifies an interval of time (in hours, minutes, and seconds) to wait before issuing a timer expired return code. The issuing program continues to execute, and all operations are valid during the time interval. When the time interval expires, the issuing program receives a return code from an accept operation indicating that the time interval has expired. The session or work station ID field is not changed when the accept operation for the timer has completed.

The set timer operation can be useful in retrying other operations that fail because of a temporary lack of resources. To do this, issue the set timer operation and then continue to do accepts until the timer expires. The accepts allow the program to continue to receive input from other programs and display stations while waiting for the timer. Only one time interval can be maintained for a program. If a previous set timer operation has been issued and has not yet expired, the old time interval is replaced by the new interval. If you are using RPG II or the BASIC $$TIMER operation, at least one requester or acquired device must be attached to the program before the program issues the set timer operation.

Get Attributes Operation

The get attributes operation returns status information about a specific session to the issuing program. The status information includes the session status, the invite status, and the 8-character location name (specified during configuration) associated with this session.

The session status is one of the following:

- The session has a SESSION OCL statement but has not yet been acquired.
- The session has been acquired.
- The session is an evoked session; that is, a procedure start request has been issued by the remote system, and the transaction has not yet ended.

The invite status is one of the following:

- This session has not been invited.
- This session has been invited, but no data has been received.
- This session has been invited, and data is ready.

Pass-Through Operations

Pass-through operations (either pass-through put or pass-through invite) are issued for a pass-through session with either the SNUF subsystem or the Intra subsystem. Pass-through operations indicate that data management and the subsystem are not to translate the user program data stream (including SNA control information), but are to pass it to the user program. The pass-through support is described in Appendix B.

ENDING A SESSION

When all required communications are complete, the session can be ended. Depending on whether the session is an acquired session or was started by an incoming procedure request, the following operations either end the session or pass it on to the next step in the job:

- Put end of transaction
- Release
- End of session

The following sections describe the effect of these operations.

Put End of Transaction Operation

A put end of transaction operation can be issued by either program. This operation always ends the transaction. If it is issued for a session started by an incoming procedure request, put end of transaction also ends the session.

Release Operation

The release operation is an attempt by the issuing program to terminate communication with the session. Release performs two different actions depending on the type of session:

- If the session was acquired by the issuing program, the release operation requests the subsystem to end the session. If the release operation was successful (return code less than 0402), the session is terminated. If the release operation was not successful, the end of session operation can be issued to terminate the session. The same or another session can then be acquired. (See *Acquire Operation* earlier in this chapter.)

- If the session was started by an incoming procedure request and the issuing program is an MRT program, the release operation passes the session to the next step in the procedure. The SSP then executes any further OCL in the procedure.

- If the session was started by an incoming procedure request and the issuing program is an SRT program, the release is delayed until the issuing program terminates. However, any subsequent operations by the issuing program to the session result in an error return code indicating that communications is being attempted to a released requester (return code 2800). After the issuing program terminates, the session is passed to the next step in the procedure.

Note: A release for an acquired session can be performed only if no transaction is active; that is, end of transaction has been successfully sent or received.

End of Session Operation

The end of session operation terminates a session. End of session can also be issued after a session that was started by an incoming procedure request receives an end of transaction return code (see *Special Considerations* in this section). The end of session operation frees that session for subsequent procedure start requests. If the end of session is not used, the session remains allocated until the job terminates. End of session can also be issued after an error resulted from the previous operation.

The end of session operation always terminates the session and gives a normal completion return code.

When end of session is issued for a session, one of the following occurs:

- If no transaction is active and no error occurred, the session is terminated normally.
- If a transaction is active or an irrecoverable error occurred during the session, the session is terminated abnormally.

If the procedure was started by an incoming procedure request, all subsequent job steps run without a requester whether the session ended normally or abnormally. If the session ended abnormally, a return code of 8158 is placed in the OCL condition code (CD) for the job. You can prevent the subsequent job steps from running by adding the following OCL statement to each job step:

```
// IF ?CD?/8158 CANCEL
```

For more information on the IF statement, see the *SSP Reference Manual*.

If the end of session operation is issued for a session that does not exist or is not being used by the issuing program, no session is terminated; however, a normal return code is given to the issuing program.

Note About Ending an Acquired Session

You should end a session by using either the release operation or the end of session operation before your terminate your program. If you do not end the session, the system will end it when your program ends. If an error occurs while the system is ending the session, your program cannot handle the return code. A message is displayed on the system console, and the operator must handle the error.

Use the end of session operation if you want the session ended and do not want to check the session status. A normal completion return code is issued to your program regardless of whether the session ended normally or abnormally, and, although a message is displayed on the system console, no operator intervention is required.

Use the release operation if you want your program to check whether the session ended normally or abnormally.

CHECKING RETURN CODES

After each operation, a return code is passed to the application program. This return code should be checked by the program to determine the status of the operation just performed, and to determine which operation should be performed next. Each code returned to an assembler program is 2 bytes long and in binary form. Each code returned to a BASIC, COBOL, or RPG II program is 4 bytes long and in hexadecimal (EBCDIC) form.

Every return code has four digits, and consists of two parts: a major code (2 digits), and a minor code (2 digits). The major code identifies the general condition for a group of return codes, and is usually sufficient to determine the action to be taken. The minor code identifies the specific condition and indicates the specific action that should be taken next.

Usually, the application program can determine what action to take by checking the major code only. Most programs might check only a few minor codes for specific conditions that might occur in that particular application or communications configuration. At a minimum, when the code is returned as a result of an *input* operation, it should be checked to see if the last of the 4 digits is even (bit 7 is off). This check determines whether an input operation is allowed as the next operation.

The description of each return code that is valid for a subsystem is contained in the chapter describing that subsystem. (If the code can be issued by more than one subsystem, its description occurs in more than one chapter and may vary from one chapter to another.) A summary chart in Appendix A shows all the return codes, and indicates all the subsystems for which each code is valid.

Major Codes

All major codes that represent *normal* or *output exception* conditions have values less than 0800, and those that represent *error* conditions have values equal to or greater than 0800. This division enables the application program to make a quick comparison to determine the type of action required.

The main groups of major return codes are:

- Operation was completed successfully (00xx, 01xx, 02xx)
- Successful operation, but no data was received (03xx)
- Output exception occurred (04xx)
- Subsystem error occurred; session has been terminated (80xx)
- Nonrecoverable session error occurred; session has been terminated (81xx)
- Acquire operation failed; session was not started (82xx)
- Session error occurred; recovery might be possible (83xx)

Minor Codes

The minor part of a return code identifies the specific condition within the general condition identified by the major part of the code. Some examples of the minor codes are:

- Some data was received on an input operation (xx01)
- End of transaction indication was received (xx08)
- Invalid evoke operation was issued (xx29)
- Maximum number of sessions are already active (xxA8)

Some return codes occur in pairs: one resulting from an input operation, and the other resulting from an output operation. The purpose of each pair is to help determine, during a combined input/output operation, which part of the operation the error occurred in. For example, the return codes 8183 and 8184 are valid for most of the subsystems. Both codes indicate that an MLCA controller check occurred: 8183 indicates the check occurred on an output operation, and 8184 indicates the check occurred on an input operation. If 8184 was returned after a put then get operation, the put portion worked correctly and the error occurred during the get portion of the combined operation.

REMOTE INITIATION OF PROCEDURES

If you expect to have procedures initiated on the System/34 from remote locations, you must have the subsystem enabled before the remote request arrives. You can enable the subsystem by having a procedure automatically run following IPL. (See *Enabling the Subsystem* earlier in this chapter for more information.)

To initiate procedures on the System/34, the remote application (if not on another System/34 with the interactive communications feature) must send a procedure start request. The procedure start request, sent by the remote system, initiates a session and starts a System/34 procedure by evoking the first program in that procedure. There are four types of procedure start request statements:

- *EXEC
- *EXEX
- *EXNC
- *EXNX

Note: If the system attempting to start a procedure on the System/34 is another System/34 with the interactive communications feature, the remote program can issue an evoke operation instead of these procedure start requests.

The format of each of these procedure requests is:

```
*EXxx procedure  [data or parameters]  [user id]

[library name]  [password]  [record length]  [block length]

                          C
[record separator]  [I]  [T]  [X]
                    [N]  [N]  [N]
```

The *procedure* name must begin in position 7 and must be separated from the data and parameters by one or more blanks. The parameters following the *password* are valid for the BSCEL subsystem only.

The *data and parameters* are considered to be everything from the first nonblank character following the procedure name through position 127, and are available as data to the application program or as positional procedure parameters.

The *user id* begins in position 128, the *library name* begins in position 136, and the *password* begins in position 144; the password must be 4 bytes long. If a library is not specified, #LIBRARY is assumed. The user id, library name, and password fields are positional and must be padded on the right with blanks if any field follows. If the System/34 does not use security, then the user ID and password are not required.

The remaining parameters are used with the *EXEC and *EXNC procedure start requests for the BSCEL subsystem. Parameters specified with the procedure start request are used for the session being started instead of parameters specified during subsystem configuration or with the ENABLE procedure.

The *record length* is the maximum user record length (4-digit decimal, right-justified). The record length begins in position 148.

The *block length* is the length of the block of data records to be transmitted or received (4-digit decimal, right-justified). The block length begins in position 152. If the block length is 0000, data records will not be blocked.

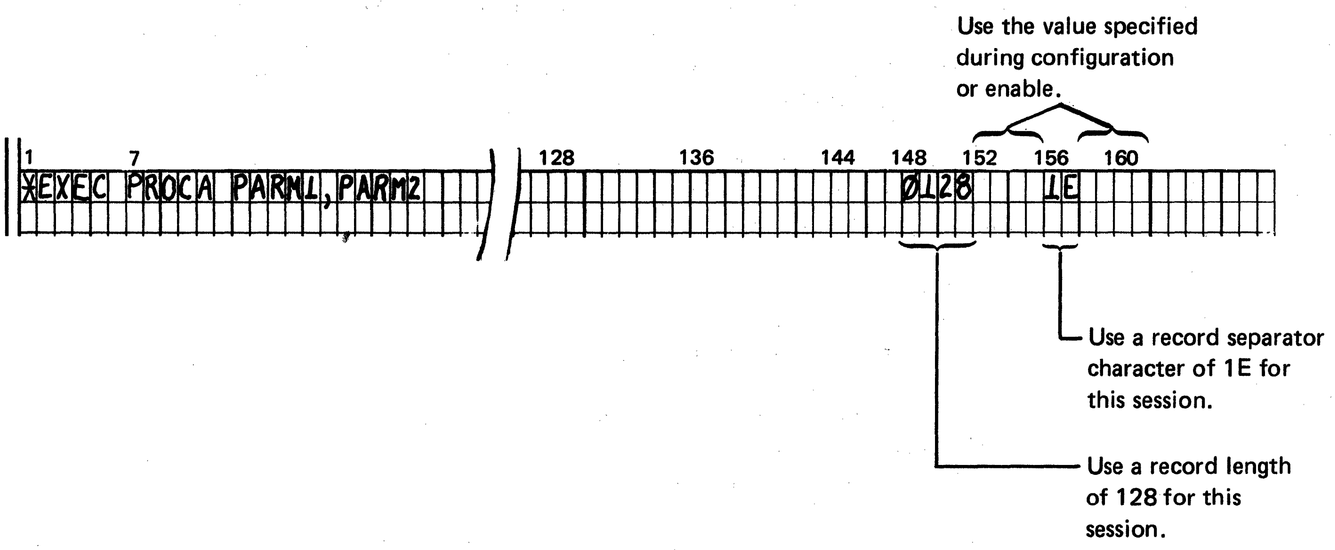
The *record separator* is the hexadecimal value for the record separator character. The record separator character begins in position 156. If you specify 00, no record separator character is used.

I is the indicator for ITB mode. The indicator is in position 158. If you specify I, ITB mode is used (ITB characters are used to separate data records in a block). If you specify N, ITB mode is not used.

CT is the indicator for blank compression or blank truncation. This indicator is in position 159. If you specify C, blank compression is used. If you specify T, blank truncation is used. If you specify N, no blank compression or blank truncation is used.

X is the indicator for transparency. This indicator begins in position 160. If you specify X, data is transmitted in transparent mode. If you specify N, data is transmitted in nontransparent mode.

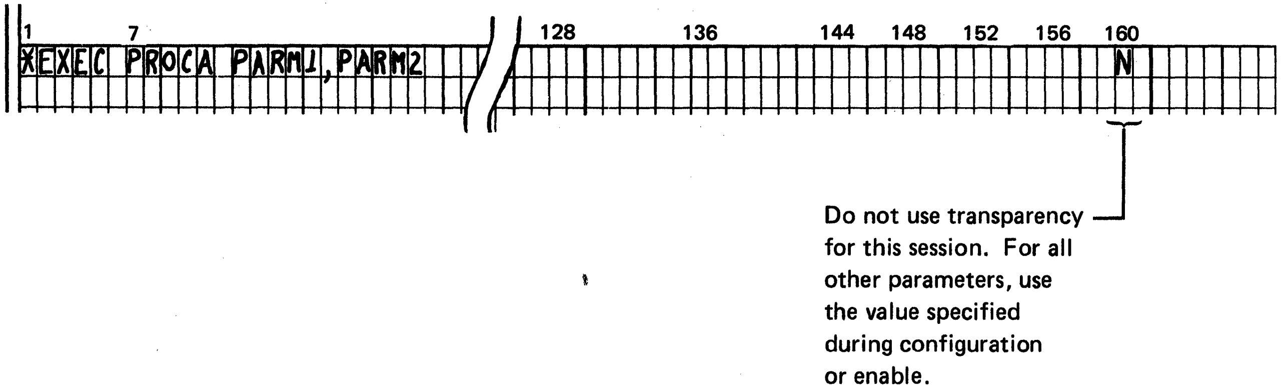
If a parameter is blank, the parameter specified during configuration or when the subsystem was enabled is used. For example:



Use the value specified during configuration or enable.

Use a record separator character of 1E for this session.

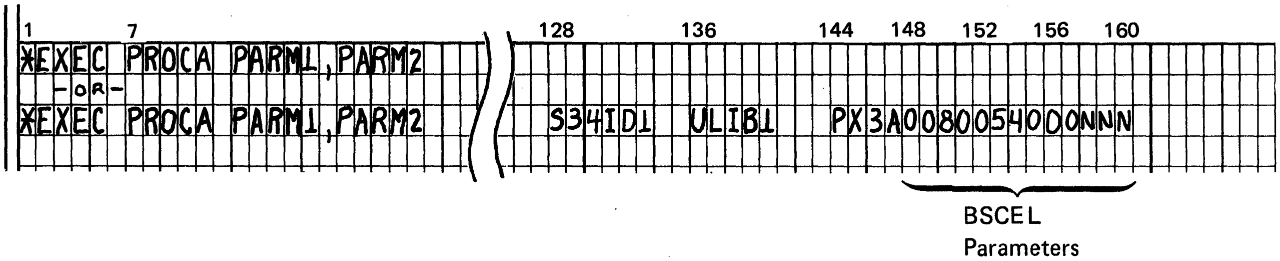
Use a record length of 128 for this session.



Do not use transparency for this session. For all other parameters, use the value specified during configuration or enable.

See *Data Formats* in Chapter 8 for more information about blocking, record separators, ITB characters, and blank truncation and compression.

The record containing the procedure start request can end with the last usable character. For example, if no parameter is required after the library name and the library name is 4 bytes long, the record could be 139 bytes long; if a 6-character procedure name is the only field sent, the record can be 12 bytes long. Depending on the subsystem, procedure start requests longer than 147 bytes could cause errors. The BSCEL subsystem accepts procedure start requests up to 160 bytes long.



```
*EXEC PROCA PARM1,PARM2
  -OR-
*EXEC PROCA PARM1,PARM2   S34ID1  ULIB1   PX3A0080054000NNN
```

The *EXEC statement requests that a procedure be started and that a session be held with a program in the procedure.

The *EXEX statement requests that a noncommunicating procedure be started; a program within that procedure can, however, request that a session be started. See *Special Considerations* later in this chapter for more information.

The *EXNC and *EXNX statements are valid only with the BSCEL subsystem. These statements serve the same functions as *EXEC and *EXEX respectively, except that no messages generated during program initiation are sent to the remote system. The *EXNC and *EXNX statements are normally used by a device, such as a 3741, that wants to start a procedure on the System/34 but cannot process messages.

Note: Because some systems (for example, the 3741) cannot transmit records longer than 128 bytes, the BSCEL subsystem allows a procedure start request to be broken into two records. The first record has the same format as described previously through column 127. If a second record is required to include the user ID, library name, password, and/or BSCEL parameters, the first record must contain a C in column 6. The user ID begins in column 7 of the second record, the library name begins in column 15, and so on. Column 128 of the first record and columns 1 through 6 of the second record are ignored.

If you are communicating between System/34s, the evoke operation causes the subsystem to build and send the procedure start request. The receiving subsystem handles the request and, if the procedure was coded to accept data (PDATA-YES), passes any data to the started program on its first input operation. An evoked System/34 application program can perform an input or output operation as its first interactive communications operation. See the following paragraph, *Writing Procedures to be Started by Incoming Procedure Requests*, for more information.

Writing Procedures to be Started by Incoming Procedure Requests

When writing procedures that are to be started by incoming procedure requests, keep several considerations in mind.

No SESSION OCL statement is allowed for sessions that are started by incoming procedure requests. If the program that the procedure runs acquires any sessions, SESSION OCL statements must be included for them.

If data is to be sent with an SRT procedure start request, PDATA-YES must be specified on the COPY utility control statement for $MAINT when the procedure is created, or the prompt for data on the end of job option menu (PROGRAM DATA IN INCLUDE STATEMENTS) must be answered yes when using SEU. MRT procedures always allow data. If data is sent with the procedure start request and the program is not prepared to receive it, the data is treated as procedure parameters. This could result in error messages, because only 11 parameters are allowed, each 8 characters long and separated by commas.

An SRT program started by an incoming procedure start request can do an output operation as the first operation. When doing an output operation first, consider the following items:

- The session ID for the session to which the output operation is sent must be ƀƀ (blanks).
- If data was sent with the incoming procedure start request, the data is lost.
- If the incoming procedure start request was an end of transaction, the requester is released, and any data the SRT program sends with the output operation is lost.

A procedure started by an incoming procedure request has all the capabilities of other procedures. For example, the procedure has:

- Access to and update of its own local data area
- Full IF statement capability
- Ability to place jobs in the input job queue
- Ability to evoke other jobs (which could initiate communication back to the system that started the job)
- Ability to change the user library for requesting procedures and programs
- Full OCL substitution capability

The only restriction is that functions that display screen formats to the requester cannot be included in the procedure. These are:

- // PROMPT
- // MENU
- // *
- SEU
- SDA
- DFU
- BASIC

Special Considerations

The following chart and notes describe what happens when a program is started with end of transaction (by *EXEX, *EXNX, or evoke end of transaction).

	SRT		
	PDATA-YES	**PDATA-NO**	**MRT**
Data include with the request	Data passed to program[1]	Procedure runs without requester[3]	Data passed to program[1]
No data included with the request	Procedure runs without a requester[3]	Procedure runs without a requester[3]	Session ID and no data passed to the program[2]

[1] After the program performs its initial input operation to receive the data, the SSP frees the session. All subsequent job steps run without a requester. (No subsequent steps can exist in an MRT procedure.) The program receives a return code indicating new requester and end of transaction (0118) on its initial input operation. If the program is coded in RPG II, the program should issue a release or end of session operation to free the entry in the internal table of IDs for the WORKSTN file.

[2] The conditions are the same as those in Note 1, but no data is passed to the program with the initial input operation.

[3] If the program issues an initial input operation for the work station file, a return code indicating no invites outstanding (1100) is given to the program. If the program is an RPG II program, this return code sets on the end of file condition, which terminates the program if the work station file is the primary file.

DISABLING THE SUBSYSTEM

To terminate a subsystem, the DISABLE procedure must be run. The Peer subsystem allows disabling of a specific location. See Chapter 14 for details. When a disable is requested for a subsystem configuration, the following functions are performed:

- If no active sessions are using this subsystem, the subsystem configuration is disabled, and the main storage being used is freed. If this subsystem is not active on any line, the subsystem is terminated; any interrupt handler (BSC or SNA and SDLC) is also terminated if it is not currently in use.

- If active sessions are using this subsystem, a message is issued to the system operator with the following options:
  - Hold the disable. New sessions are prevented from being started and, when all sessions complete, a normal disable occurs.[1]
  - Retry the disable. Check again for any active sessions.
  - Cancel active sessions and disable. Active sessions are immediately terminated, and the disable is performed.
  - Ignore the disable request. The DISABLE procedure is canceled and must be run again when a disable is desired.

- If a disable is pending or in progress, a message is issued to the system operator. The message indicates either that the disable request is not allowed, or that the operator can request an immediate disable or wait for the current disable to complete.

The format of the DISABLE procedure is:

```
DISABLE name , [location]
```

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

location: Specifies the name of the remote location to be disabled. The SNA location can be specified only if the subsystem being disabled is a Peer subsystem. This name must have been specified as a remote location name during subsystem configuration.

[1] When a disable is held, each successful operation will receive a major return code indicating that a disable is pending (02).

INTERACTIVE COMMUNICATIONS PROGRAMMING TECHNIQUES

Before writing programs that issue interactive communications operations, you should understand the work station file operations and how they relate to display stations and sessions.

Session Types

On a System/34, there are two ways to configure a display station: (1) command mode (that is, jobs can be initiated from the display station), and (2) data mode (that is, a program must acquire the display station before an operator can interact with an application).

As indicated in Chapter 1, there are two types of sessions: (1) an acquired session, and (2) a session initiated via an incoming procedure start request. An acquired session has many of the attributes of a data mode display station, and likewise the session started by an incoming procedure request has many of the attributes of a command mode display station.

An acquired session, like an acquired data display station, is active only for the duration of the program that acquired the session, or until that program issues a release or end of session operation. A session started by an incoming procedure request, however, can exist for the duration of an entire job, which can consist of many steps (programs). The parallel between this type of session and a command display station is that either can serve as a requester for multiple programs. You can take advantage of this parallel by breaking the procedure into simpler logical units of work (programs). As with a command display station, the session serves as a requester for the next job step whenever the previous job step ends or releases the session. The session is terminated when the job ends or when any program issues an end of transaction or an end of session operation. After the session is terminated, the remaining job steps, if any, run without a requester.

IDs

Most work station file operations require a symbolic ID to direct the data management to the program or display station for which the output operation is intended. The ID is a 2-character field that corresponds to the following naming conventions:

- If the ID is ƀƀ (two blanks) and if the program is an SRT program, the operation is directed to the program's requester (either the display station or the session that requested the start of the procedure). This ID is invalid if the program is an MRT program or has no requester. ƀƀ is a valid ID for an accept operation or set timer operation. RPG, however, requires a nonblank ID for any output or timer request.

- If the ID is *nn* (where *nn* is from 00 through 99), the operation is directed to a session.

- If the ID is *na* where *n* is from 0 through 9 and *a* is not numeric), the operation is directed to an acquired session with the corresponding SYMID value from the SESSION OCL statement.

- If the ID is *ax* (where *a* is not numeric and *x* is any character), the operation is directed to a display station whose physical ID is *ax* or to the display station with the corresponding SYMID value on the WORKSTN OCL statement.

On returning to the application program following a work station file operation, data management returns an ID with one of the following values:

- An ID of *ax*, which represents the physical ID of a display station or the ID of a display station with the corresponding SYMID value on the WORKSTN OCL statement.

- An ID of *nn*, which represents the physical ID assigned by data management to a session that was started by an incoming procedure request.

- An ID of *na*, which represents an acquired session with the corresponding SYMID value on the SESSION OCL statement.

The ID field is unchanged by data management for an input operation whose return code indicates that no invites are outstanding or that the time interval has expired.

SRT and MRT Program Considerations

The concept of SRT and MRT programs applies to interactive communications programs as well as to other display station programs. An incoming procedure request is treated as a requester just as if a display station were requesting the procedure.

Programs that are coded as SRT programs that want to handle only display station requesters can prevent programs from inadvertently requesting the program by modifying the OCL for that procedure. The IF EVOKED OCL test can be used to determine whether the procedure request was the result of an incoming procedure start request or an EVOKE OCL statement. The IF EVOKED OCL test can also be included in a MRT procedure, but the test is performed only for the original requester.

If an MRT program is to be started by a session, be certain the MRTMAX parameter is large enough. If the maximum number of requesters is already attached to the program, the incoming procedure request is queued until the program releases one of its active requesters. This queuing could result in unacceptable delays.

An SRT program can have only one requester but can acquire more display stations and sessions. The acquired display stations and sessions can be released explicitly within the program, but the requester is not completely released until the program ends.

A requester can be released before the program begins if RELEASE-YES is specified on the ATTR OCL statement. MRT programs can release requesters within the program. MRT programs should release the display stations and sessions when processing completes or they will remain attached to the program until it ends.

Interprogram/Interprocedure Communication

Program information can be passed from program to program and from program to procedure within a work session (from display station sign on to display station sign off or for the duration of an evoked session). A full screen of information or an entire transaction record can be passed from program to program using the read under format technique. The work station local data area and user program switches (UPSI) can also be used to pass up to 256 bytes of data or switches from program to program or from program to procedure. This can be done by using the programming language and OCL support provided to access and update these areas.

If an application is written as a series of related programs, the local data area or UPSI can be used to control program flow for a given procedure. An example of this would be the execution of a nested procedure based on program logic. Assume that at some point a program determines that the next procedure to be executed is one named SPECIAL. To run this procedure, the program first reads the local data area. This ensures that the areas that are not to be changed by the program will retain their values. Next, the program moves the literal SPECIAL into a field that is located (for this example) in the first seven positions of the program-defined local data area buffer. The program then calls the appropriate subroutine to write the program data area to the work station local data area. When the program ends, the OCL is read. If the next statement is // INCLUDE ?L'1,7'?, the next procedure executed will be the procedure named SPECIAL.

Read Under Format

A read under format allows one program in a procedure to display a format and a subsequent program in the procedure to read it. The first program displays the format using a normal output operation ($$SEND, or without suppressing input) and then goes to end of job or releases the display station or session. While the second program is initiating, the operator keys information into the displayed format, or the remote program sends data. The input information is then sent to the second program. This technique can be effective in hiding the program load time because it can be overlapped with the input operation.

Read under format can also be used with sessions started by incoming procedure requests. In this case, one program step in the procedure invites the session for input and then releases it. The process of initiating the next program step is overlapped with data arrival on the invited session.

The following steps occur in a read under format:

1. With a normal output operation, the first program displays a format at the work station or invites a session.

2. The first program either ends if the program is an SRT program or releases the requester if the program is an MRT program.

3. The second program is initiated. (Data cannot be passed to the second program from an INCLUDE OCL statement.)

4. The second program performs a normal input operation as the first operation.

Chapter 3. Interactive Communications Programming with Assembler

The interactive communications portion of an assembler program consists of preparing data for transmission, processing data that was received, using macroinstructions to define control blocks and perform operations, and checking and reacting to the return codes. Because the data preparation and processing vary greatly by application, those functions are not described in this chapter.

MACROINSTRUCTIONS

To assist the assembler language programmer in writing interactive communications programs, the following macroinstructions are provided as part of the Basic Assembler and Macro Processor Program Product:

- $DTFW (define an interactive communications DTF)

- $WSIO (perform an interactive communications operation)

- $EVOK (define an evoke parameter list)

The $DTFW and $WSIO macroinstructions are also used for work station input and output. The $ALOC, $OPEN, $CLOS, and $DTFO macroinstructions are also required for interactive communications programs. The WS-Y and FIELD-Y parameters must be specified on the $DTFO macroinstruction; the other macroinstructions remain unchanged.

The ICRTC-Y parameter must be specified on the $DTFO macroinstruction to generate the labels for the interactive communications return codes.

The following sections describe the $DTFW, $WSIO, and $EVOK macroinstructions as used for interactive communications only. More parameters and operations are available on $DTFW and $WSIO for display station input and output; for the complete description, see the *Basic Assembler Reference Manual.* Each description includes an example or examples of correctly coded macroinstructions. For examples of how macroinstructions can be used with specific subsystems, see the appropriate subsystem chapter.

The $DTFW and $WSIO macroinstructions define and modify fields in the work station DTF. The complete format of the DTF, including field labels is in the *Data Areas and Diagnostic Aids Manual.*

The following is the format of the $DTFW macroinstruction as used to generate an interactive communications DTF:

$$\left[\text{label}\right]\ \text{\$DTFW}\ \left[\text{NAME-}\begin{Bmatrix}\underline{\text{FILENAME}}\\ \text{file name}\end{Bmatrix}\right]\ \left[\text{, UPSI-}\begin{Bmatrix}\underline{00000000}\\ \text{8-bit UPSI}\end{Bmatrix}\right]$$

$$\left[\text{, CHAIN-}\begin{Bmatrix}\underline{\text{X'FFFF'}}\\ \text{DTF address}\end{Bmatrix}\right]\ \left[\text{, INLEN-}\begin{Bmatrix}\underline{0000}\\ \text{input length}\end{Bmatrix}\right]$$

$$\left[\text{, TERMID-}\begin{Bmatrix}\underline{\text{ƀƀ}}\\ \text{session id}\end{Bmatrix}\right]\ \left[\text{, TIDTAB-}\begin{Bmatrix}\underline{0000}\\ \text{session id table address}\end{Bmatrix}\right]$$

$$\left[\text{, ENTLEN-}\begin{Bmatrix}\underline{00}\\ \text{length}\end{Bmatrix}\right]\ \left[\text{, TNUM-}\begin{Bmatrix}\underline{1}\\ \text{number of entries}\end{Bmatrix}\right]\ \left[\text{, HALTS-}\begin{Bmatrix}\text{Y}\\ \underline{\text{N}}\end{Bmatrix}\right]$$

NAME: Specifies the file name associated with this DTF. Interactive communications data management does not use this name. If this parameter is not specified, FILENAME is assumed.

UPSI: 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 operand is omitted, zeros are assumed, and no test is done.

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

INLEN: Specifies, in decimal, the maximum amount of input data that the user program is prepared to receive. This number must not be greater than 65535. Note that although the macroinstruction allows any value up to 65535, the subsystem restriction is 4075 for all BSC and SNA subsystems and 40952 for the Intra subsystem. If this parameter is omitted, zeros are assumed, and no data can be transferred unless this field is modified (by the $WSIO macroinstruction).

TERMID: Specifies the ID of the session to communicate with. This ID must be the same as that specified on the SESSION OCL statement if this session is acquired as a SRT. If this parameter is omitted, blanks are assumed, but communications on the session can still take place normally if this program is evoked remotely. Data management assigns an ID to the session with the remote program that evoked this program, and places that ID in this field ($WSNAME).

TIDTAB: Specifies the address of the session and work station ID table. Programs that support multiple display stations and sessions typically want to maintain a list of IDs and associated status indicators. By specifying the TIDTAB, TNUM, and ENTLEN parameters, an area is reserved for this list. During open, the ID 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 on. For each WORKSTN and SESSION OCL statement, an entry is created with each SYMID for the first 2 bytes. The first bit of the next byte is set on if REQD-YES was specified; the second bit is set off. The table must be large enough to contain each of these IDs plus enough additional entries up to the MRTMAX value. After open is complete, the user program must maintain the list. If an end of session operation is issued or if a return code of hex 80 or 81 is received, interactive communications data management places zeros 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 this parameter is omitted, zeros are assumed, and no table is built.

ENTLEN: Specifies, the decimal length (in bytes) of each entry in the session and work station ID table. If the TIDTAB parameter was specified, ENTLEN must be specified and should be 3 or greater.

TNUM: Specifies the number of entries in the session and work station ID table. The TNUM 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, TNUM must also be specified.

HALTS: Specifies whether interactive communications data management should halt for any permanent communications error (major return code greater than or equal to hex 80). If Y or YES is specified, data management issues a system message that allows the operator the option of returning control to the user program with a permanent error return code or ending the job. If N or NO is specified, data management logs an informational message to the system console, and the user program receives control with the permanent error return code.

Example

The following example shows a DTF that can be used for multiple sessions. This DTF is part of a DTF chain; the next DTF is a printer DTF. Any permanent communications errors result in a system message.

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

The examples shown later in this chapter under *$WSIO* use this DTF.

$WSIO

The following is the format of the $WSIO macroinstruction, as used to perform interactive communications operations:

```
[label] $WSIO [DTF-label] [,OUTLEN-length] [,INLEN-length]

                  [,RCAD-address] [,TERMID-id] [,OPM-modifier]

                  [,OPC-code] [,PL@-address]
```

DTF: Specifies the address of the leftmost byte of the DTF. This is the label specified on the $DTFW macroinstruction. If this parameter is omitted, the address of the DTF is assumed to be in index register 2.

OUTLEN: Specifies, in decimal, the length of the data in the buffer pointed to by the RCAD parameter. If this 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. Therefore, whenever the output operation follows an input operation, this parameter should be specified.

INLEN: Specifies, in decimal, the maximum amount of input data that the user program is prepared to receive. This number must not be greater than 65535. Note that although the macroinstruction allows any value up to 65535, the subsystem restriction is 4096 for all BSC and SNA subsystems and 40952 for the Intra subsystem. If this parameter is omitted, the DTF remains unchanged.

RCAD: Specifies the address of the leftmost byte of the user program logical record buffer. This parameter must be specified in the first $WSIO issued that requires a record area. Thereafter, if the operand is omitted, the DTF remains unchanged. If the buffer is also to be used for display station input, it must be on an 8-byte boundary.

TERMID: Specifies the 2-character ID of the session for which this operation is intended. This ID should be the same as the SYMID on the corresponding SESSION OCL statement for acquired sessions or the same as the ID returned in the DTF following the initial accept operation for a remotely started session. 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 operand is omitted, the DTF remains unchanged. Following each accept operation, data management returns the ID of the session from which data was received in this field ($WSNAME).

OPM: 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**
FMH	Indicates that a function management header is with the data associated with the evoke. This modifier is valid only with the evoke operation for the SNUF or Intra subsystems.
ZERO	Resets the operation modifier to be no modifier.
PTH	Indicates that the specified operations (either GET or PTI) is a pass-through operation.

OPC: Specifies the operation desired. If this parameter is omitted, the DTF remains unchanged. For a chart of all the valid assembler operations and the operation codes that can be specified in this parameter, see *Assembler Operations Summary Chart* later in this chapter.

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

Position	**Value**	**Meaning**
1	A	Session not yet acquired
	C	Session is an acquired session
	R	Session is an evoked 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
3-10	name	Location name (specified during configuration and on the SESSION OCL statement)

Note: If the ID for the operation is not that of a session, the format of the attribute information is somewhat different. See the *Basic Assembler Reference Manual* for the format of attribute data for display stations.

The set timer (STM) operation 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 following expiration of the timer. The TERMID returned with the timer expired return code has no meaning. If a set timer operation has been issued and has not yet expired, the old value is discarded and the new interval is set.

PL@: Specifies the address of an associated evoke parameter list. This is the label on the $EVOK macroinstruction. This parameter must be specified on the first evoke, and remains unchanged if not specified thereafter.

The following chart shows the operation codes and the corresponding parameters on the $WSIO macroinstruction. An R indicates the parameter is required, O indicates optional, I indicates ignored, and the X in the OPM column indicates that ZERO must be specified; even though a parameter is required, it does not have to be specified if the parameter was previously specified and the value is the same.

	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	X	R	I
CANG	O	I	R	R	R	X	R	I
CANI	O	I	I	I	R	X	R	I
EOS	O	I	I	I	R	X	R	I
EVE	O	R[1]	I	R[2]	R	O[3]	R	R
EVG	O	R[1]	R	R[2]	R	O[3]	R	R
EVI	O	R[1]	I	R[2]	R	O[3]	R	R
EVK	O	R[1]	I	R[2]	R	O[3]	R	R
FAIL	O	O	I	O	R	X	R	I
GET	O	I	R	R	R	O[7]	R	I
GTA	O	I	I	R[4]	R	I	R	I
INV	O	I	I	I	R	O[7]	R	I
NRP	O	O[5]	I	O[5]	R	X	R	I
NRPG	O	O[5]	R	R	R	X	R	I
NRPI	O	O[5]	I	O[5]	R	X	R	I
PEC	O	R[1]	I	R[2]	R	X	R	I
PEF	O	R[1]	I	R[2]	R	X	R	I
PEX	O	R[1]	I	R[2]	R	X	R	I
PFM	O	R	I	R	R	X	R	I
PFMG	O	R	R	R	R	X	R	I
PFMI	O	R	I	R	R	X	R	I
PTG	O	R[1]	R	R[2]	R	X	R	I
PTI	O	R[1]	I	R[2]	R	O[7]	R	I
PUT	O	R[1]	I	R[2]	R	O[7]	R	I
RCDG	O	I	R	R	R	X	R	I
RCDI	O	I	I	I	R	X	R	I
REL	O	I	I	I	R	I	R	I
STM	O	I	I	R[6]	I	I	R	I

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

[2] Only required if the OUTLEN value is not zero.

[3] An 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.

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

[5] Up 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.

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

[7] OPM-PTH can be specified.

Examples

The following are examples of the $WSIO macroinstruction. All use the DTF as defined in the $DTFW example shown previously.

The following example issues an acquire for session 1S.

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

The following example evokes a transaction on the session that was acquired and then waits for input. The evoke parameter list begins at label EVKLST (and is shown in the $EVOK example later in this chapter). INLEN is not specified, because it was specified on the $DTFW macroinstruction.

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

The following example puts data to the session and transaction previously begun. The data is 256 bytes long and is stored at label OTBUFF.

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

$EVOK

The $EVOK macroinstruction builds a parameter list to be associated with an evoke operation. The label on this macroinstruction should be the label specified on the PL@ parameter of the $WSIO macroinstruction. The following is the format of the $EVOK macroinstruction:

$$[\text{label}]\ \text{\$EVOK}\ \left[\text{V-}\begin{Bmatrix}\text{EQU}\\ \text{ALL}\\ \underline{\text{DC}}\end{Bmatrix}\right]\ [\text{, PNAME-address}]$$

$$[\text{, LNAME-address}]\ [\text{, UID-address}]$$

$$[\text{, PWORD-address}]$$

V: Specifies the type of expansion for the parameter list. If EQU is specified, only the displacement labels are generated, and all other parameters are ignored. If DC 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. A $EVOK that includes the equates must be specified once and only once within a program.

PNAME: Specifies the address of the first character of the name of the remote procedure to be evoked. The procedure 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 procedure name is passed on the evoke.

LNAME: Specifies the address of the first character of the library name associated with the 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.

UID: Specifies the address of the first character of the user ID. The ID 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 ID is passed on the evoke.

PWORD: 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.

Examples

The following example shows a $EVOK macroinstruction as used by the evoke operation shown in a $WSIO example earlier in this chapter. The procedure name stored as ICPROC is started from the library ICLIB.

```
EVKLST   $EVOK V-ALL,PNAME-ICPROC,
               LNAME-ICLIB,UID-S34ID,
               PWORD-PASS
```

The following example is a second evoke parameter list for the same program. The procedure name stored as R3PR in ICLIB is started on a system that does not require security.

```
EVKL2 $EVOK V-DC,PNAME-R3PR,LNAME-ICLIB
```

ASSEMBLER OPERATIONS SUMMARY CHART

The following chart shows all the operations that are valid for assembler, their operation codes, and all the subsystems for which each operation is valid. An *x* in a subsystem column indicates that the system supports the operation. A - indicates that the subsystem does not support the operation.

For a description of each of these operations (except GTA and STM), see Chapter 2.

		Communications Subsystem								
Assembler Operation	**Operation Mnemonic**	**Intra**	**BSCEL**	**CCP**	**CICS**	**IMS**	**3270**	**Finance**	**Peer**	**SNUF**
Accept input	ACI	x	x	x	x	x	x	x	x	x
Acquire	ACQ	x	x	x	x	x	x	x	x	x
Cancel	CAN	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
Evoke	EVK	x	x	x	x	x	x[2]	–	x	x
Evoke end of transaction	EVE	x	x	–	x	x	–	–	x	x
Evoke then get	EVG	x	x	x	x	x	x[2]	–	x	x
Evoke then invite	EVI	x	x	x	x	x	x[2]	–	x	x
Fail	FAIL	x	–	–	–	–	–	–	x	–
Get	GET	x	x	x	x	x	x	x	x	x
Get attributes	GTA	x	x	x	x	x	x	x	x	x
Invite	INV	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
Pass-through put	PUT[1]	x	–	–	–	–	–	–	–	x
Pass-through put then invite	PTI[1]	x	–	–	–	–	–	–	–	x
Pass-through invite	INV[1]	x	–	–	–	–	–	–	–	x
Put	PUT	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
Put then get	PTG	x	x	x	x	x	x	x	x	x
Put then invite	PTI	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
Request to change direction then get	RCDG	x	x	x	–	–	–	–	x	x
Request to change direction then invite	RCDI	x	x	x	–	–	–	–	x	x
Set timer	STM	x	x	x	x	x	x	x	x	x

[1] Valid only when OPM-PTH is specified with the $WSIO macro.

[2] Valid 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.

RETURN CODES

Whenever an interactive communications operation is issued (using the $WSIO macroinstruction), 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. For general information about return codes, see *Checking Return Codes* in Chapter 2.

Each return code contains two parts (a major code and a minor code), and at least the major code should be checked. The major code is located at offset $WSRTC in the DTF, and the minor code is at offset $WSRSIQ in the DTF. For a description of each return code that can be returned for a subsystem, refer to the last section of that subsystem chapter. A chart in Appendix A shows which codes are valid for each subsystem.

INTERACTIVE COMMUNICATIONS ASSEMBLER SUBROUTINES

Because of the additional capability and flexibility available in the assembler interactive communications support, you might want to write 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 to 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 the subroutine or 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 consider 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; that is, the main program has no work station file.

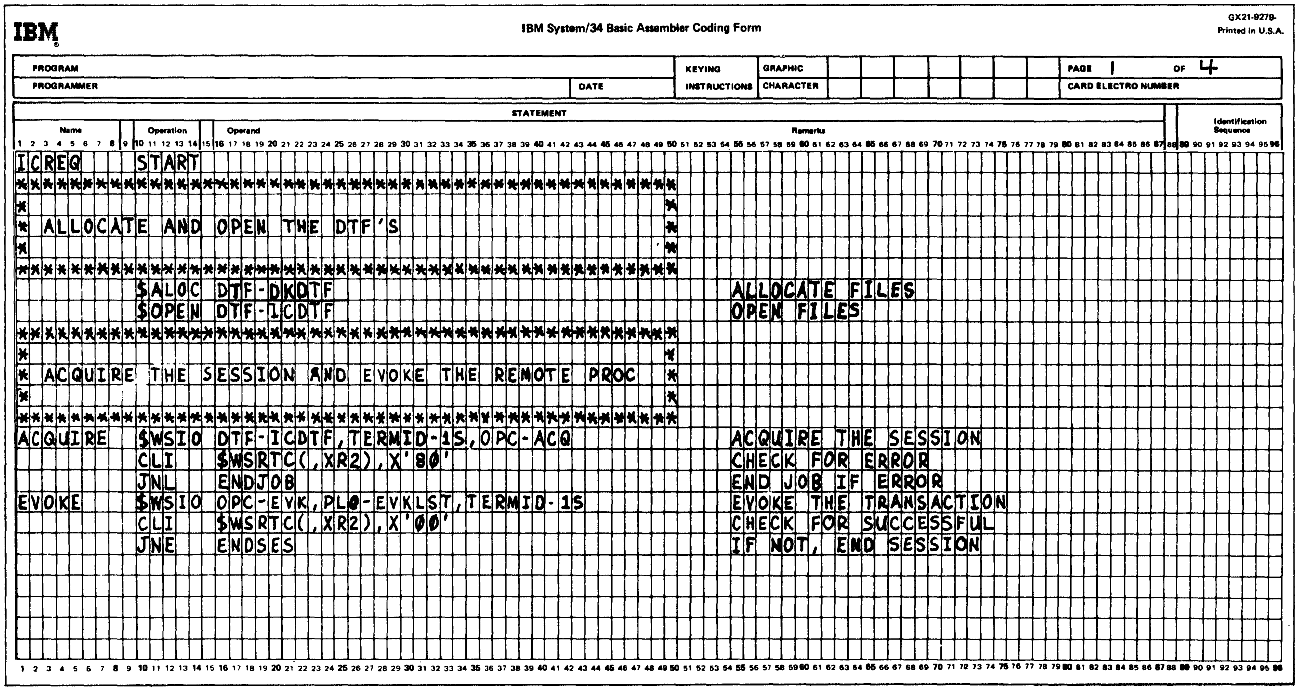
- The DTF must reside in a portion of the program that is not overlaid.

- 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.

CODING EXAMPLES

The following program is a sample interactive communications program written in assembler language. This program acquires a session with the remote system and sends a request. The program receives the data, saves it in a disk file, and then releases the session. Presumably a subsequent job or job step processes the data from the disk file. The SESSION OCL statement, assuming all subsystem-dependent parameters were configured correctly, would be:

```
// SESSION LOCATION-RW34PC,SYMID-1S
```



IBM

IBM System/34 Basic Assembler Coding Form

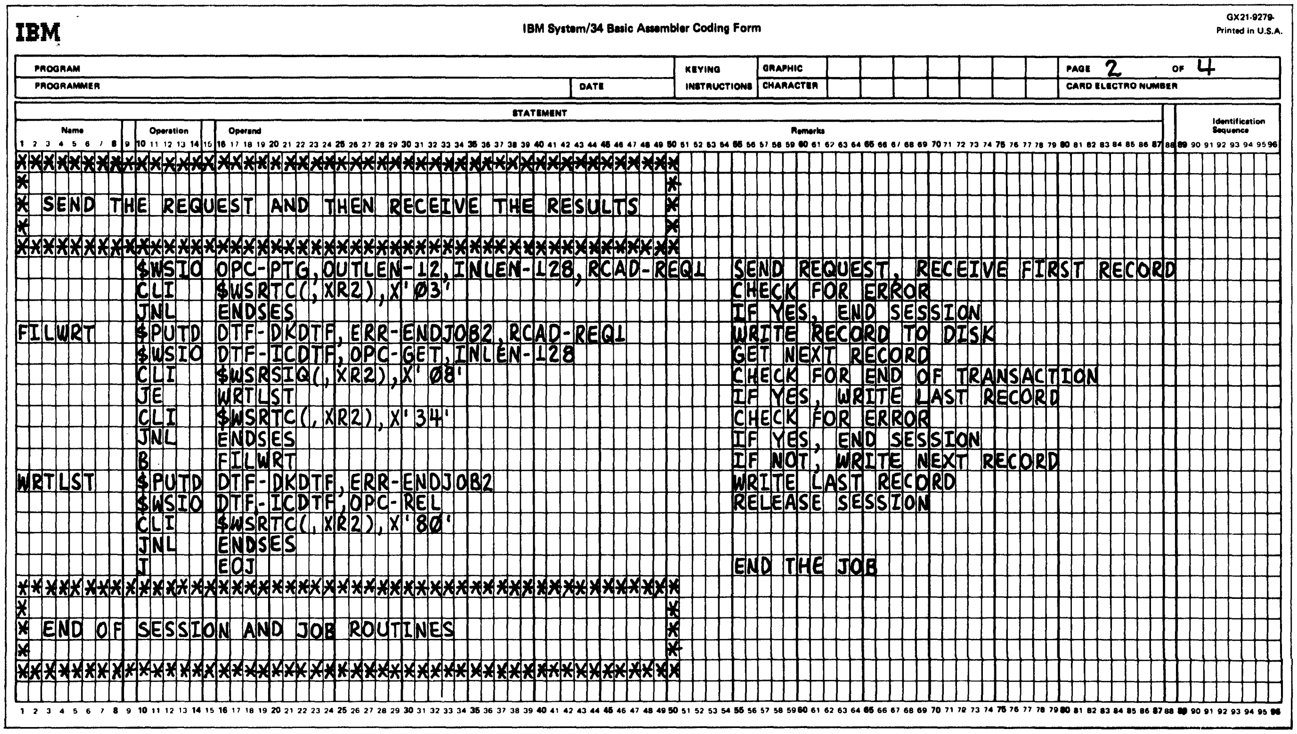
GX21-9279-
Printed in U.S.A.

PROGRAM		KEYING INSTRUCTIONS	GRAPHIC	PAGE 1	OF 4
PROGRAMMER	DATE		CHARACTER	CARD ELECTRO NUMBER	

STATEMENT

Name / Operation / Operand / Remarks / Identification Sequence

```
ICREQ    START
**************************************************
*                                                *
* ALLOCATE AND OPEN THE DTF'S                    *
*                                                *
**************************************************
         $ALOC DTF-DKDTF                              ALLOCATE FILES
         $OPEN DTF-ICDTF                              OPEN FILES
**************************************************
*                                                *
* ACQUIRE THE SESSION AND EVOKE THE REMOTE PROC  *
*                                                *
**************************************************
ACQUIRE  $WSIO DTF-ICDTF,TERMID-1S,OPC-ACQ            ACQUIRE THE SESSION
         CLI   $WSRTC(,XR2),X'8Ø'                     CHECK FOR ERROR
         JNL   ENDJOB                                 END JOB IF ERROR
EVOKE    $WSIO OPC-EVK,PLØ-EVKLST,TERMID-1S           EVOKE THE TRANSACTION
         CLI   $WSRTC(,XR2),X'ØØ'                     CHECK FOR SUCCESSFUL
         JNE   ENDSES                                 IF NOT, END SESSION
```



IBM

IBM System/34 Basic Assembler Coding Form

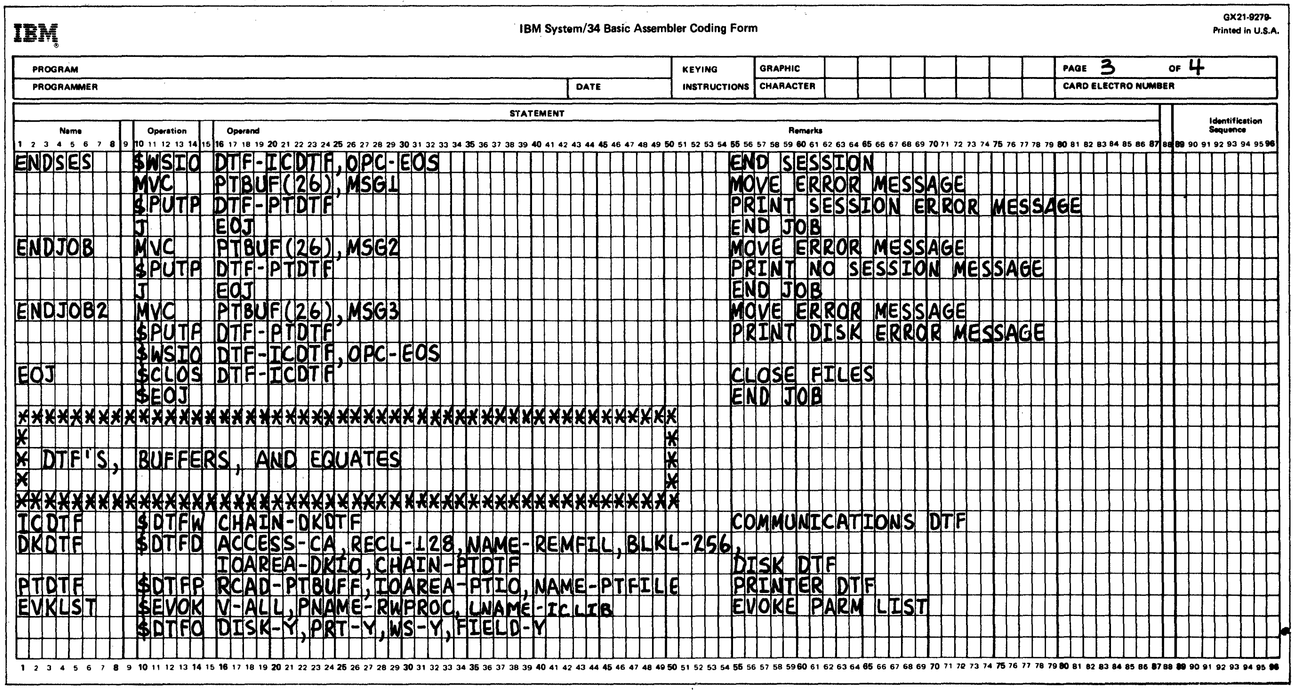
GX21-9279-
Printed in U.S.A.

PROGRAM		KEYING INSTRUCTIONS	GRAPHIC	PAGE 2	OF 4
PROGRAMMER	DATE		CHARACTER	CARD ELECTRO NUMBER	

STATEMENT

Name / Operation / Operand / Remarks / Identification Sequence

```
**************************************************
*                                                *
* SEND THE REQUEST AND THEN RECEIVE THE RESULTS  *
*                                                *
**************************************************
         $WSIO OPC-PTG,OUTLEN-12,INLEN-128,RCAD-REQ1  SEND REQUEST, RECEIVE FIRST RECORD
         CLI   $WSRTC(,XR2),X'Ø3'                     CHECK FOR ERROR
         JNL   ENDSES                                 IF YES, END SESSION
FILWRT   $PUTD DTF-DKDTF,ERR-ENDJOB2,RCAD-REQ1        WRITE RECORD TO DISK
         $WSIO DTF-ICDTF,OPC-GET,INLEN-128            GET NEXT RECORD
         CLI   $WSRSIQ(,XR2),X'Ø8'                    CHECK FOR END OF TRANSACTION
         JE    WRTLST                                 IF YES, WRITE LAST RECORD
         CLI   $WSRTC(,XR2),X'34'                     CHECK FOR ERROR
         JNL   ENDSES                                 IF YES, END SESSION
         B     FILWRT                                 IF NOT, WRITE NEXT RECORD
WRTLST   $PUTD DTF-DKDTF,ERR-ENDJOB2                  WRITE LAST RECORD
         $WSIO DTF-ICDTF,OPC-REL                      RELEASE SESSION
         CLI   $WSRTC(,XR2),X'8Ø'
         JNL   ENDSES
         J     EOJ                                    END THE JOB
**************************************************
*                                                *
* END OF SESSION AND JOB ROUTINES                *
*                                                *
**************************************************
```



IBM

IBM System/34 Basic Assembler Coding Form

GX21-9279
Printed in U.S.A.

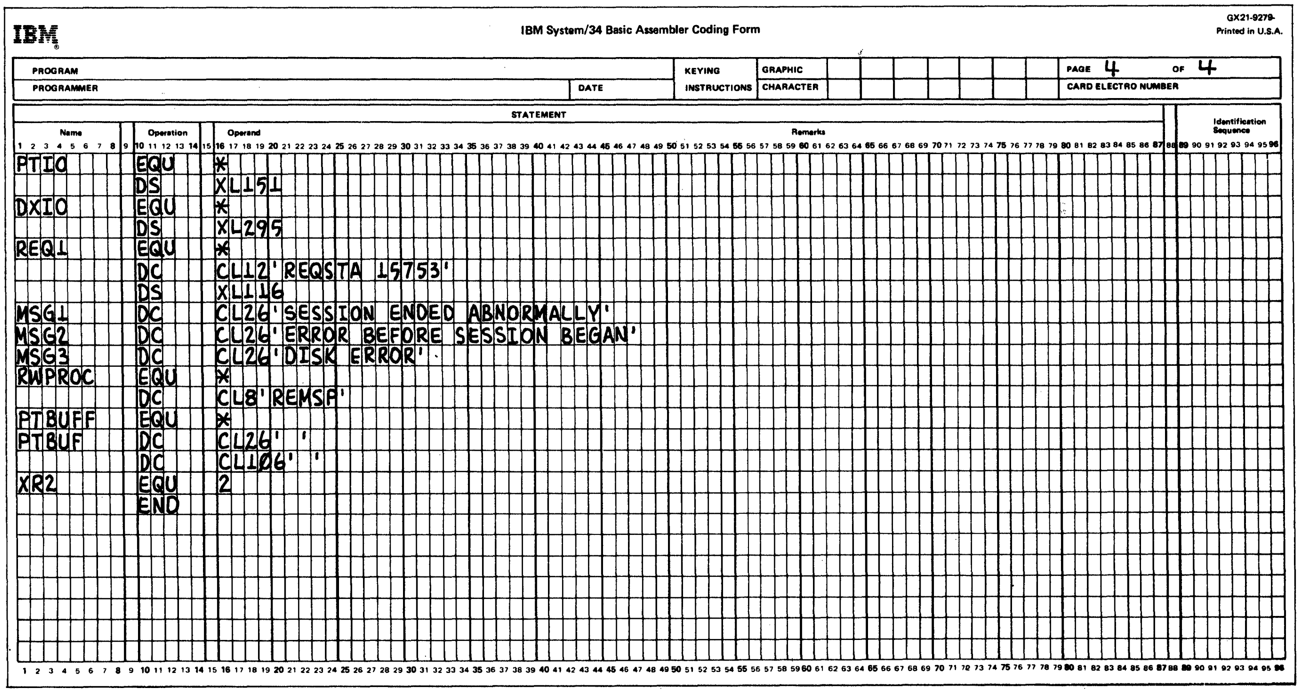
PROGRAM		KEYING INSTRUCTIONS	GRAPHIC
PROGRAMMER	DATE		CHARACTER

PAGE 3 OF 4

CARD ELECTRO NUMBER

STATEMENT

```
ENDSES   $WSIO DTF-ICDTF,OPC-EOS                       END SESSION
         MVC   PTBUF(26),MSG1                          MOVE ERROR MESSAGE
         $PUTP DTF-PTDTF                               PRINT SESSION ERROR MESSAGE
         J     EOJ                                     END JOB
ENDJOB   MVC   PTBUF(26),MSG2                          MOVE ERROR MESSAGE
         $PUTP DTF-PTDTF                               PRINT NO SESSION MESSAGE
         J     EOJ                                     END JOB
ENDJOB2  MVC   PTBUF(26),MSG3                          MOVE ERROR MESSAGE
         $PUTP DTF-PTDTF                               PRINT DISK ERROR MESSAGE
         $WSIO DTF-ICDTF,OPC-EOS
EOJ      $CLOS DTF-ICDTF                               CLOSE FILES
         $EOJ                                          END JOB
**************************************************
*                                                *
* DTF'S, BUFFERS,  AND EQUATES                   *
*                                                *
**************************************************
ICDTF    $DTFW CHAIN-DKDTF                             COMMUNICATIONS DTF
DKDTF    $DTFD ACCESS-CA,RECL-128,NAME-REMFIL,BLKL-256,
               IOAREA-DKIO,CHAIN-PTDTF                 DISK DTF
PTDTF    $DTFP RCAD-PTBUFF,IOAREA-PTIO,NAME-PTFILE     PRINTER DTF
EVKLST   $EVOK V-ALL,PNAME-RWPROC,LNAME-ICLIB          EVOKE PARM LIST
         $DTFO DISK-Y,PRT-Y,WS-Y,FIELD-Y
```



IBM

IBM System/34 Basic Assembler Coding Form

GX21-9279
Printed in U.S.A.

PROGRAM		KEYING INSTRUCTIONS	GRAPHIC
PROGRAMMER	DATE		CHARACTER

PAGE 4 OF 4

CARD ELECTRO NUMBER

STATEMENT

```
PTIO     EQU   *
         DS    XL151
DXIO     EQU   *
         DS    XL295
REQ1     EQU   *
         DC    CL12'REQSTA 15753'
         DS    XL116
MSG1     DC    CL26'SESSION ENDED ABNORMALLY'
MSG2     DC    CL26'ERROR BEFORE SESSION BEGAN'
MSG3     DC    CL26'DISK ERROR'
RWPROC   EQU   *
         DC    CL8'REMSP'
PTBUFF   EQU   *
PTBUF    DC    CL26' '
         DC    CL1Ø6' '
XR2      EQU   2
         END
```

Chapter 4. Interactive Communications Programming with BASIC

To use the BASIC language with the interactive communications feature, do the following:

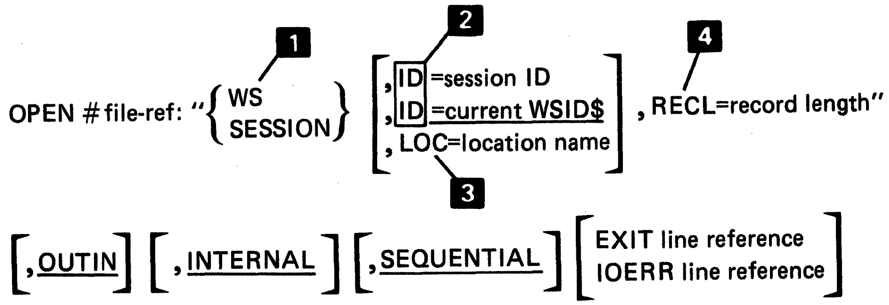
- Configure and enable the subsystem. (These operations are described in the appropriate subsystem chapter.)
- Begin a communications session by opening an SSP-ICF file.
- Begin a program or procedure at the remote system and start a communications transaction. (If you are using the Intra subsystem, the program or procedure being started is in the same System/34.)
- Send or receive data.
- Check return codes.
- End the communications transaction.
- End the communications session.

The operations you do to process data before it is sent and after it is received are the same as the operations you do when you are not using data communications. Therefore, these operations are not described in this manual. Only the operations you need for interactive communications are described here. Interactive communications operations are a lot like work station operations.

BEGINNING A SESSION

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

The following is the format of the OPEN statement and a description of each SSP-ICF parameter. See the *BASIC Reference Manual* for a description of other OPEN statement parameters.



	Parameter	Meaning
1	WS or SESSION	Input is either from a work station or an SSP-ICF session, and output is to either a work station or an SSP-ICF session.
2	ID	The 2-character identifier for the session being started. This is the same as the identifier (SYMID) on the SESSION OCL statement. If you do not enter a session ID or if the program is evoked by the remote system, the current value of WSID$ is the session ID.
3	LOC	The name of the remote location you will be communicating with during this session. If you enter a location name, a SESSION OCL statement is not required for the Intra (unless you need to specify BATCH-YES on the SESSION statement), Peer, or CCP subsystems.
4	RECL	The length of the longest record (or system message) you expect to transmit or receive. A system message is 75 bytes for the Intra subsystem.

Note: If you enter a LIBR or KEYS parameter in the OPEN statement, they are ignored.

OPEN Statement Examples

Example 1:

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

An example SESSION OCL statement for this OPEN statement is:

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

Example 2:

```
OPEN #1: "SESSION,RECL=255" IOERR ICFERR (Evoked Program)
```

A SESSION statement is not required for the evoked session; however, an evoked program can begin (acquire) sessions once evoked. In this case, a SESSION statement may be required for each additional session that the evoked program begins.

1 Open interactive communications file #1.

2 In the first example, the session ID is 1S. In the second example, the ID is set to the current value of WSID$ when your program is evoked by the remote system. The value in WSID$ is set by SSP-ICF and BASIC.

3 The maximum record length that can be sent or received is 255 bytes.

4 The program goes to ICFERR if an error occurs.

WRITE STATEMENT FORMAT

The following is the format of the WRITE statement and a description of the SSP-ICF FORMAT parameter. See the *BASIC Reference Manual* for a description of other WRITE statement parameters.

$$\text{WRITE \#file-ref} \left[,\text{USING} \begin{Bmatrix} \text{line reference} \\ \text{character-expr} \end{Bmatrix} \right] \boxed{\left[,\text{FORMAT format name} \right] :}$$

$$\begin{Bmatrix} \text{MAT array name} \\ \text{variable} \end{Bmatrix} \left[, \begin{Bmatrix} \text{MAT array name} \\ \text{variable} \end{Bmatrix} \ldots \right]$$

FORMAT is the name of the format for the SSP-ICF operation. See *BASIC Operations Summary Chart* later in this chapter for a list of the operations you can use with the FORMAT parameter. Any of the operations beginning with $$ can be used.

The WRITE statement is used in many of the examples that follow.

BEGINNING A PROGRAM OR PROCEDURE AT THE REMOTE SYSTEM

To begin a program or procedure at the remote system, and start a communications transaction, use the WRITE statement to do one of the following evoke operations. See *WRITE Statement Format* in this chapter for the format of the WRITE statement.

- *$$EVOKNI* performs an *evoke* operation, which begins a program or procedure at the remote system.

- *$$EVOK* performs an *evoke then invite* operation, which begins a program or procedure at the remote system and tells the remote system to transmit first.

- *$$EVOKET* performs an *evoke end of transaction* operation, which begins a program or procedure at the remote system, and then ends the communications transaction.

If your program is evoked by an incoming procedure start request, you do not use the evoke operation since your program is started by the remote system.

With an evoke operation you must send the following parameters:

Positions	**Meaning**
1 through 8	The name of the program or procedure to be started (left-justified)
9 through 16	The password you use to sign onto the remote system (left-justified)
17 through 24	The user ID you use to sign onto the remote system (left-justified)
25 through 32	The name of the remote system library that contains the program or procedure to be started (left-justified)
33 through xxxx	User data or positional procedure parameters; leading blanks are ignored.

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

The following is an example of starting a program or procedure at the remote system.

```
             1                        2                        3
30 WRITE #1,USING 40,FORMAT "$$EVOK": "BASICR",PASS$,USERID$,&
&"#LIBRARY","ICFPROG,USERLIB" IOERR ICFERR

40 FORM 4*C 8,C 20     4                 7
          5    6
```

1 Write data to interactive communications file #1 using the FORM statement at line 40.

2 Begin the procedure BASICR, which is in #LIBRARY.

3 The password and user ID needed to sign onto the remote system.

4 The BASIC program (ICFPROG) to be called by the procedure BASICR is in the user library (USERLIB).

5 Send four fields of 8 characters each (evoke parameters).

6 Send 20 bytes of positional parameters.

7 If an error occurs during the WRITE operation, the program goes to ICFERR.

Sending Program Data with the Evoke Operation

Data sent with the evoke operation consists of either parameters to be used by the evoked *procedure*, as shown in the previous example, or data to be used by the evoked *program*. If you send procedure parameters, answer *no* to the prompt PROGRAM DATA IN THE INCLUDE STATEMENTS on the end of the job option menu of the SEU procedure or specify PDATA-NO on the COPY control statement for $MAINT. If you send program data, answer *yes* to the SEU prompt for program data or specify PDATA-YES on the COPY control statement. See *Writing Procedures to be Started by Incoming Procedure Requests* in Chapter 2 for more information.

BASICR Procedure: You cannot use the BASICR procedure if you send program data with the evoke operation, because this procedure expects procedural parameters. The following is an example of a procedure that can be used when program data is sent:

```
// LIBRARY NAME-#BLLIB

// MEMBER PROGRAM-#BL#M1,PROGRAM2-#BL#M2,USER1-#BL#M1

// LIBRARY NAME-user library——— 1

// REGION SIZE-BASIC region size— 2

// LOAD #BLSIC

// INCLUDE procedure member— 3

// RUN

// BASIC MEMBER-name,LIBRARY-user library,WORKAREA-size,STATUS-Y or N
             \_____ _____/\________ ________/\_____ _____/\_____ _____/
                   V                V               V             V
// END             4                5               6             7
```

You must supply the information for the following parameters:

1 Enter the name of the current user library.

2 Enter the BASIC region size (24 to 64).

3 Enter the procedure name to be included. If there is no procedure name to be included, you can delete this statement.

4 Enter the name of your BASIC program.

5 Enter the name of the library that contains your BASIC program.

6 Enter the size of the work area needed for your program.

7 Enter Y if you want status information printed. Enter N if you do not want status information printed.

SENDING DATA

To send a data record to a remote system or program, use the WRITE statement and one of the following send operations. See *WRITE Statement Format* in this chapter for the format of the WRITE statement.

- *$$SENDNI* performs a *put* operation, which sends one record to the program or procedure at the remote system.

- *$$SEND* performs a *put then invite* operation, which sends one record to the program or procedure at the remote system and tells the remote system to transmit.

- *$$SENDE* performs a *put end of file* operation, which ends the file when you use BSC. It performs a put end of chain operation, which ends the chain when you use SNA.

- *$$SENDET* performs a *put end of transaction* operation, which ends the program or procedure at the remote system.

- *$$SENDNF* performs a *put function management header* operation, which sends one record containing a functior, management header to the remote system. The $$SENDNF operation is valid with the Finance, Intra, and SNUF subsystems only.

- *$$SENDFM* performs a *put function management header then invite* operation, which sends one record containing a function management header to the remote system and tells the remote system to transmit. The $$SENDFM operation is valid with the Finance, Intra, and SNUF subsystems only.

For example, the following WRITE statement is used to send one data record:

```
           1                       2                  3
30 WRITE #1, USING 40, FORMAT "$$SEND": DATA$ IOERR ICFERR
40 FORM C 255
      4
```

1 Write to interactive communications file #1 using the FORM statement at line 40.

2 Send the data record in DATA$ and tell the remote system to transmit.

3 If an error occurs, the program goes to ICFERR.

4 The length of the data record in DATA$ is 255 bytes.

RECEIVING DATA

To receive a data record, use the READ statement to get the data record from an SSP-ICF session. If you precede the READ statement with a WAITIO statement, the program waits until data is available from any work station or SSP-ICF session. If you do not precede the READ statement with the WAITIO statement, the program waits until data is available from the work station or SSP-ICF session corresponding to the file number entered in the OPEN statement. The WAITIO statement also sets the intrinsic function FILENUM to the file reference of the communications session from which data is to be read.

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

```
    1        2
30 WAITIO IOERR ICFERR
40 READ #FILENUM, USING 50: DATA$ IOERR ICFERR
50 FORM V 255
        4   3
```

1 The WAITIO statement at line 30 causes the READ statement at line 40 to read data from any work station or SSP-ICF session. Without the WAITIO statement at line 30, the READ statement waits until data is available from the work station or SSP-ICF session assigned to the file reference (#FILENUM) specified in the READ statement, then reads the data into DATA$.

2 The intrinsic function FILENUM contains the file reference of the work station or session from which the data is to be read.

3 Up to 255 characters are read into the variable DATA$.

4 Use the V parameter on the FORM statement if you do not know the length of the data record received.

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

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

A If a system message is received (ERR=70), then read the message into MESSAGE$.

B Up to 80 characters of the system message are read.

Notes About Receiving Data

1. You should use EOF with the READ statement to test for an end of transaction received from the remote system (SSP-ICF return code xx08). If you want to do another evoke operation before you close the communications file, use another READ statement with the EOF clause before you do the evoke operation.

2. You can use the STOP$ intrinsic function to test for a major return code of 02 (subsystem disable pending). If STOP$ equals Y, a 02 major return code has been issued to your program indicating that a system shut down has been requested; if not, STOP$ equals N.

3. The READ statement does an SSP-ICF get operation. And the WAITIO statement followed by the READ statement does an SSP-ICF accept operation. See Chapter 2 for more information about the get and accept operations.

CHECKING RETURN CODES

You should use the IOERR parameter on all READ, REREAD, WRITE, OPEN, CLOSE, or WAITIO statements to check the status of the input or output operation. Use the RETCODE$, ERR, or FILE intrinsic function to determine the execution status of the last operation. The meaning of the intrinsic functions are:

- RETCODE$ contains the status of the last SSP-ICF operation. The status tells whether the operation was *successful* or *unsuccessful* and gives you additional information about the results of the operation.

- ERR contains the meaning of the error for the last *unsuccessful* BASIC operation.

- FILE indicates only that the last operation was either *successful* or *unsuccessful*. If FILE is 0, the operation was successful; if file is 1, the operation was unsuccessful.

The value in RETCODE$ is the 4-digit (major and minor) SSP-ICF return code. These codes are described in each subsystem chapter. A summary chart in Appendix A shows all of 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 2.

The value in ERR depends upon the SSP-ICF return code in RETCODE$ 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 your subsystem chapter.

For an example of how to check return codes, see *How to Write a BASIC Program that Uses the Intra Subsystem* in Chapter 7.

RETCODE$	ERR	RETCODE$	ERR	RETCODE$	ERR
Value	Value	Value	Value	Value	Value
(ICF)	(BASIC)	(ICF)	(BASIC)	(ICF)	(BASIC)
0000	0	0212	0	8192	55
0001		0220	70	8193	
0003		0221		8194	
0004		0228		8195	
0005		0230		8196	
0007		0231		8197	
0008		0238		8198	
000C		0300	0	8199	
0010	69	0301		819A	
0012	0	0302	71	819B	
0020	70	0303	54	819C	
0021		0308		819D	
0028		0310	73	819E	
0030		0402	71	819F	
0031		0411	70	81A3	
0038		0412	71	81B5	
0100	68	0800	0	81B6	
0101		1100	64	81B8	
0103		2800	55	81B9	
0104		3401	55	81BA	
0105		8081	55	81BC	
0107		8082		820A	55
0108		8083		820D	
010C		8084		8213	72
0118		80BD		8215	
0200	0	8136	55	821E	55
0201		8137		8233	
0203		8183		8236	
0204		8184		8281	
0205		8185	72	8282	
0207		8186		8283	
0208		8187	55	8285	72
020C		8191		8286	

RETCODE$ Value (ICF)	ERR Value (BASIC)	RETCODE$ Value (ICF)	ERR Value (BASIC)	RETCODE$ Value (ICF)	ERR Value (BASIC)
8288	72	82B0	55	832A	71
8289	55	82B1	72	832B	
828A		82B2		832C	
828B		82B3		832D	
828C		82B4		832E	
828D		82BB	55	832F	
828E		82BC		8330	72
828F		830B	71	8331	
8290		830D		8332	
8291		830E		8333	
8293		8313		8334	
8296		8314		8336	
8297		8315		8338	
829B		8316		8339	
829F		8317		833C	
82A0		8319	70	8383	
82A1		831A		8384	
82A2		831B	72	8385	
82A5		831C	70	8386	
82A6		831D	71	8391	
82A7	72	831E		8392	
82A8		831F		8397	
82A9	55	8320		8398	
82AA	72	8322		8399	
82AB		8323		839A	
82AC		8324		839B	
82AD		8326		839C	
82AE	55	8327		83A7	
82AF		8329		83B0	

ENDING A COMMUNICATIONS TRANSACTION

Before you end your program, you must end the communications transaction by using the $$SENDET or $$EVOKET operation with the WRITE statement, or the remote system must end the transaction. For example:

- If your program is done transmitting data, use the $$SENDET operation to tell the remote system that you have no more data to send.

- If your program is receiving data, check for an end of transaction received from the remote system to determine when the remote system is done transmitting (see *Receiving Data* in this chapter).

- If you want to start a program or procedure at the remote system and then end the transaction, use the $$EVOKET operation. For example:
  - Your program starts program A at the remote system and sends data to program A.
  - Program A stores the data on disk.
  - When your program is done sending data to program A, your program uses the $$SENDET 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 an end of transaction and, therefore, tells the remote system that this is the end of this communications transaction:

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

See *Ending a Session* in Chapter 2 for more information about ending a transaction.

ENDING A SESSION

To end a session with a remote system, use the *CLOSE* statement, or use the *$$EOS* operation followed by the CLOSE statement. See *Ending a Session* in Chapter 2 for more information about ending a session.

Note: The $$REL operation cannot be used with BASIC; however, the CLOSE statement is the same as the $$REL operation.

The CLOSE statement ends the session and closes the communications file. All transactions with the program at the remote system must be complete before you end the session.

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

```
99 CLOSE #1: IOERR ICFERR
```

If an error occurs when closing the session, your program goes to ICFERR, and BASIC issues a $$EOS operation to end the session.

OTHER SSP-ICF OPERATIONS YOU CAN DO

The following are optional operations you can do with SSP-ICF. Use the WRITE statement to do these operations. See *WRITE Statement Format* in this chapter for the format of the WRITE statement.

- Ask for a change in transmission direction
- Use SSP-ICF or work station timer operations
- Send a negative response to the remote system (used only with the Intra and SNUF subsystems)
- Send a fail operation (used only with the Intra and Peer subsystems)
- Cancel a group (chain) of data records (used only with the Intra and SNUF subsystems)
- Use pass-through operations (used only with the Intra and SNUF subsystems)

Asking for a Change in Transmission Direction

If your program is receiving data, you can ask the remote system to stop sending so your program can send data. To ask for a change in the direction of transmission, use the WRITE statement to do a *request to change direction* operation ($$RCD). After you issue the $$RCD operation, your program must continue to receive data until an end of transaction or change of direction indication is received from the remote system. There are no parameters or data needed with the $$RCD operation. See *Request to Change Direction Operation* in Chapter 2 for more information about this operation.

For example, this statement asks the remote system to stop sending so that your program can send data:

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

Using SSP-ICF and Work Station Timer Operations

To use the SSP-ICF and work station timer, use the *$$TIMER* operation or the *TIMER* intrinsic function to set the timer, and use the WAITIO statement to determine when the time has ended. Return code 0310 (RETCODE$) or 73 (ERR) is returned when the time has ended.

Example of Using the $$TIMER Operation

If you use the $$TIMER operation, a work station or session must be attached to your program before you can set the time.

You specify the time in hours, minutes, and seconds in the format:

hhmmss

For example:

```
1─030 A$= "013000"
  040 WRITE #1,USING 50,FORMAT "$$TIMER": A$ IOERR ICFERR
  050 FORM C 6───2
  060 WAITIO IOERR TIME
3──  •
     •
                   4
     •
  910 TIME: IF ERR <> 73 THEN GO TO ICFERR
```

1 The timer is set to 1 hour, 30 minutes, and 00 seconds.

2 There are 6 characters used to set the time (hhmmss).

3 If an I/O error occurs when the WAITIO statement is executed, go to TIME and check for the timer return code.

4 If the return code does not indicate that the timer expired (ERR is not 73), go to ICFERR. If the return code does indicate the time expired, perform operations based upon the reason for the time-out. For example, display a message indicating that the remote system did not respond within the time allowed.

Example of Using the TIMER Intrinsic Function

If you use the TIMER intrinsic function to set the timer, a work station or session does not have to be attached to the program; however SSP-ICF must be active. To set the timer, use the TIMER intrinsic function in the format:

TIMER(time$), where time$ is the time in the format "*hhmmss*". For example:

```
                  1
030 TIME=TIMER("013000")
040 IF TIME=1 THEN PRINT "SSP-ICF IS NOT ACTIVE" ELSE WAITIO IOERR TIME1
 •       2                                             3           4
 •
 •
910 TIME1: IF ERR<>73 THEN GOTO ICFERR
                                  5
```

1 The timer is set to 1 hour, 30 minutes, and 00 seconds.

2 If SSP-ICF is not active, the timer is not set and TIME is set to 1. If TIME is 1, print a message.

3 If TIME is 0, the timer is set and the WAITIO statement is executed.

4 If a return code is returned when the WAITIO statement is executed, go to TIME1 and check the return code.

5 If the return code does not indicate that the timer expired (ERR is not 73), go to ICFERR. If the return code does indicate the time expired, perform operations based upon the reason for the time-out. For example, display a message indicating that the remote system did not respond within the time allowed.

Sending a Negative Response

To tell the remote system or program that your program found something wrong with the data it received (to send a negative response), use one of the following operations. Both types of negative response operations are for the Intra and SNUF subsystems only. See *Negative Response Operation* in Chapter 2 for more information.

- *$$NRSPNI* performs a *negative response* operation, which transmits a negative response to the remote system or program.

- *$$NRSP* performs a *negative response then invite* operation, which transmits a negative response and tells the remote system or program to transmit.

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

Data Positions	**Meaning**
1 through 8	The sense data transmitted with the negative response. The sense data must begin with 10xx, 08xx, or 0000 (for the Intra and SNUF subsystems). All other positions are user defined.

For example, the following statements send a negative response with the sense data 08008000:

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

Sending a Fail Operation

To tell the remote system that your program detected an abnormal condition (for example, received incorrect data), use the *$$FAIL* operation. The fail operation does not need any parameters, and no data can be sent with the fail operation. The fail operation is used only with the Intra and Peer subsystems. See *Fail Operation* in Chapter 2 for more information.

For example, the following statement sends a fail operation to SSP-ICF:

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

Issuing a Cancel Operation

To cancel a group of records, use one of the following cancel operations. The cancel operation does not need any parameters or data. The cancel operation is used only with the Intra and SNUF subsystems. See *Cancel Operation* in Chapter 2 for more information.

- *$$CANLNI* performs a *cancel* operation, which cancels the current group (chain) of data records.
- *$$CANL* performs a *cancel then invite* operation, which cancels the current group of data records and allows the remote system or program to transmit.

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

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

Using Pass-Through Operations

Pass-through operations are used only with the Intra and SNUF subsystems. See Appendix B for a description of pass-through operations.

BASIC OPERATIONS SUMMARY CHART

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

		Communications Subsystem								
BASIC Operation	**Operation Mnemonic**	**Intra**	**BSCEL**	**CCP**	**CICS**	**IMS**	**3270**	**Finance**	**Peer**	**SNUF**
Accept input	WAITIO[1]	x	x	x	x	x	x	x	x	x
Acquire	OPEN	x	x	x	x	x	x	x	x	x
Cancel	$$CANLNI	x	–	–	–	–	–	–	–	x
Cancel then invite	$$CANL	x	–	–	–	–	–	–	–	x
End of session	$$EOS	x	x	x	x	x	x	x	x	x
Evoke	$$EVOKNI	x	x	x	x	x	x[2]	–	x	x
Evoke end of transaction	$$EVOKET	x	x	–	x	x	–	–	x	x
Evoke then invite	$$EVOK	x	x	x	x	x	x[2]	–	x	x
Fail	$$FAIL	x	–	–	–	–	–	–	x	–
Get	READ	x	x	x	x	x	x	x	x	x
Invite[3]		x	x	x	x	x	x	x	x	x
Negative response	$$NRSPNI	x	–	–	–	–	–	–	–	x
Negative response then invite	$$NRSP	x	–	–	–	–	–	–	–	x
Pass-through put then invite	$$PTPUT	x	–	–	–	–	–	–	–	x
Pass-through invite	$$PTINV	x	–	–	–	–	–	–	–	x
Put	$$SENDNI	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
Put FMH	$$SENDNF	x	–	–	–	–	–	x	–	x
Put FMH then invite	$$SENDFM	x	–	–	–	–	–	x	–	x
Put then invite	$$SEND	x	x	x	x	x	x	x	x	x
Release	CLOSE	x	x	x	x	x	x	x	x	x
Request to change direction then invite	$$RCD	x	x	x	–	–	–	–	x	x
Set timer	$$TIMER[4]	x	x	x	x	x	x	x	x	x

[1] Valid only when followed by a READ operation.

[2] Valid 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.

[3] Valid only when used with another operation or by using a $$SEND operation with a record length of 0.

[4] The timer can also be set by the TIMER intrinsic function.

NOTES ABOUT WRITING BASIC PROGRAMS FOR SSP-ICF

1. You can use the *WSID* intrinsic function to determine the ID of the most recently accessed session. For example:

   ```
   40 WAITIO IOERR ICFERR
   50 A$=WSID$
   ```

   The value of A$ is the ID of the last SSP-ICF 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 or IOERR parameter on all SSP-ICF I/O statements. See *Checking Return Codes* in this chapter for more information.

4. You can use the ATTRIBUTE$ intrinsic function to determine the status of a session. See the *BASIC Reference Manual* for a description of the ATTRIBUTE$ intrinsic function.

5. Do not use PAUSE, BREAK, PRINT, INPUT, LINPUT, or TRACE in an evoked program (by a remote procedure start request) to put information on the display station screen.

CODING EXAMPLES

See *How to Write Programs that Use the Intra Subsystem* for an example of how to write a BASIC program that uses the Intra subsystem. The same programming example described in the Intra chapter is shown in the other subsystem chapters if any changes are needed to allow communications with that remote system.

Chapter 5. Interactive Communications Programming with COBOL

The interactive communications portion of a COBOL program consists of preparing data for transmission, processing data that was received, using the predefined formats and work station operations to perform communications operations, and properly handling return codes. Because the data preparation and processing vary greatly by application, those functions are not described in this manual. The processing of interactive communications operations is very similar to that for work station operations. The file used is a TRANSACTION file, the input operations are identical, and the output operations are performed via formats. TRANSACTION file programming considerations are in the *COBOL Reference Manual.*

FILE DEFINITION

The TRANSACTION file must be defined by a SELECT statement in the FILE-CONTROL paragraph. Only one TRANSACTION file is allowed per program. The format of the SELECT statement for a TRANSACTION file using interactive communications is:

```
SELECT file-name

     ASSIGN TO assignment-name

     ORGANIZATION IS TRANSACTION

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

    [ACCESS MODE IS SEQUENTIAL]

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

ASSIGN Clause

The ASSIGN clause associates the TRANSACTION file with devices through the use of the assignment-name. Assignment-name has the following structure:

$$\text{type-}\begin{bmatrix}\text{name} \\ \text{name-formats}\end{bmatrix}$$

The value for each field is as follows:

Type: WORKSTATION

Name: 1- to 8-character name that specifies the external name of the SFGR generated load member that contains the screen formats. This field is not required if the file is to be used with sessions only.

Formats: A two-digit numeric value that is equal to or or greater than the number of formats in the SFGR load member referenced in the name field. The maximum value and the default value for the number of formats is 32. This field is not required if the file is to be used with sessions only.

ORGANIZATION Clause

The ORGANIZATION clause specifies the logical structure of a file. TRANSACTION organization signifies user-controlled input and output of records.

FILE STATUS Clause

The FILE STATUS clause allows you to monitor the execution of input and output operations from or to a 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 resulting from TRANSACTION file processing (a major and minor return code).

ACCESS MODE Clause

The ACCESS MODE clause must always be SEQUENTIAL for TRANSACTION files.

CONTROL-AREA Clause

The 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 ID of the session or display station from which input was obtained. The symbolic ID 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 *COBOL Reference Manual*. For an example of how to code this control area, see the sample programs at the end of this chapter.

FORMATS

Access to some of the functions of interactive communications in COBOL is provided by predefined formats. These formats are used in the same way that display screen formats are used for work station operations; that is, they are issued by the FORMAT option of the WRITE statement. The predefined formats are not identical to display screen formats, however, because the predefined formats are recognized by data management, making it unnecessary to separately store and process the formats via a screen format generator. The predefined format names should not be used in any display screen format members. The following sections describe the available formats and the operations they perform. For more detail on the operations that these formats perform, see the operation descriptions in Chapter 2.

For more information on how to issue these formats, see the WRITE statement description later in this chapter.

Procedure/Program Initiation (Evoke)

To start a remote procedure or program (transaction), the evoke operation is used.

Three formats are provided for evoking a transaction:

- $$EVOKNI performs an *evoke* operation.
- $$EVOK performs an *evoke then invite* operation.
- $$EVOKET performs an *evoke end of transaction* operation.

Each evoke request can be accompanied by several parameters; the first four form the evoke parameter list. All the parameters must be defined by the application program in the output area for the evoke formats. 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.

Size	**Description**
8	The name of the remote procedure to be evoked
8	The password sent to the remote system
8	The user ID sent to the remote system
8	The library name containing the remote procedure
20	Reserved
4	Length, in decimal, of user data, if any
xxxx	User data or positional procedure parameters

Transmit Data

Four formats are provided for transmitting a record:

- $$SENDNI performs a *put* operation.
- $$SEND performs a *put then invite* operation.
- $$SENDE performs a *put end of file* operation for BSC or a *put end of chain* operation for SNA.
- $$SENDET performs a *put end of transaction* operation.
- $$SENDNF performs a *put function management header* operation.
- $$SENDFM performs a *put function manangement header then invite* operation.

The transmit requests require two fields in the output area.

Size	**Description**
4	Length, in decimal, of the user data[1]
xxxx	User data to be transmitted

Request to Change Direction

The $$RCD format is used for a *request to change direction* operation. This format has no output data associated with it.

Set Timer Interrupt Value

The $$TIMER format is used for a *set timer* operation. The following output data is required with the format.

Size	**Description**
6	Interval of time to be set. The time should be specified in hours, minutes, and seconds (hhmmss).

[1] An output length of zero for $$SEND performs an *invite* operation. An output length of zero is allowed for $$SENDE and $$SENDET, because the end of chain/file and end of transaction can be sent without data.

Send Negative Response

Two formats are provided for sending a negative response:

- $$NRSPNI performs a *negative response* operation.
- $$NRSP performs a *negative response then invite* operation.

The negative response operations can be used only for the Intra or SNUF subsystems. The negative response format can optionally have the following two fields in the output area:

Size	Description
1	Length of sense data (must be 0 or blank if no sense data, or 8 if sense data is present)
8	The sense data to be sent with the negative response

Note: The sense data is user-defined, but the first four characters must be 10xx, 08xx, or 0000.

Cancel

Two formats are provided for issuing a cancel operation:

- $$CANLNI performs a *cancel* operation.
- $$CANL performs a *cancel then invite* operation.

The cancel formats are valid only for the SNUF and Intra subsystems. These formats have no output data associated with them.

Fail

The $$FAIL format is used to issue a *fail* operation. This format has no output data associated with it. The $$FAIL format is valid only for the Intra and Peer subsystems.

End of Session

The $$EOS format is used to issue an *end of session* operation. This format has no output data associated with it.

Pass-Through Support

Two formats ($$PTPUT and $$PTINV) are provided for pass-through operations. These formats and a complete description of pass-through support are contained in Appendix B.

WORK STATION OPERATIONS

Several of the existing work station operations are used for interactive communications operations. They are:

- ACQUIRE
- READ
- WRITE
- DROP

A description of these operations as they are used for interactive communications follows.

ACQUIRE

The ACQUIRE statement acquires the specified session for the TRANSACTION file.

If a literal is specified for the session ID, it must be a 2-character alphanumeric literal with the first character numeric; if an identifier is specified, it must refer to a 2-character alphanumeric data item with the first character numeric. The session ID must correspond to the SYMID parameter specified on a SESSION OCL statement. The file name must refer to a TRANSACTION file.

The format of the ACQUIRE statement is:

$$\underline{\text{ACQUIRE}}\left\{\begin{array}{l}\text{literal}\\ \text{identifier}\end{array}\right\}\ \underline{\text{FOR}}\ \text{file-name}$$

READ

The READ statement performs either a get or accept operation depending on the TERMINAL option. If the TERMINAL option is specified, a get operation is performed for the session specified. If the TERMINAL option is not specified and only one session or display station is attached to the file, a get operation is performed for that session or display station. If the TERMINAL option is not specified and multiple sessions and display stations are attached, an accept operation is performed.

A NO DATA option is available on the READ statement that allows execution of another statement if data is not available for the READ statement.

Also available is an AT END option. This option allows a statement to be executed when the READ statement is issued with no invited display stations or sessions.

The format 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]
```

For more information on the READ statement, see the *COBOL Reference Manual.*

WRITE

The WRITE statement requests one of the formats to be performed. The FORMAT option specifies the name of the format. The record name specifies the output area that contains any of the information required with the format. The TERMINAL option can be used to specify a particular session. If the TERMINAL option is not used, the operation is performed for the session or display station associated with the last READ or WRITE.

The format of the WRITE statement is:

$$\underline{\text{WRITE}}\ \text{record-name}\left[\underline{\text{FROM}}\ \text{identifier-1}\right]$$

$$\left[\underline{\text{FORMAT}}\ \text{IS}\left\{\begin{array}{l}\text{identifier-2}\\ \text{literal-1}\end{array}\right\}\right]$$

$$\left[\underline{\text{TERMINAL}}\ \text{IS}\left\{\begin{array}{l}\text{identifier-3}\\ \text{literal-2}\end{array}\right\}\right]$$

$$\left[\left\{\begin{array}{l}\underline{\text{INDICATOR}}\\ \underline{\text{INDICATORS}}\\ \underline{\text{INDIC}}\end{array}\left[\begin{array}{l}\text{IS}\\ \text{ARE}\end{array}\right]\right\}\text{identifier-8}\right]$$

DROP

The DROP statement issues a release operation for a particular session. The name of the TRANSACTION file associated with this session must be specified. If a literal is specified for the session ID, it must be a 2-character alphanumeric literal with the first character numeric; if an identifier is specified, it must refer to a 2-character alphanumeric data item with the first character numeric. The session ID must correspond to the SYMID parameter specified on a SESSION OCL statement.

The format of the DROP statement is:

$$\underline{\text{DROP}}\left\{\begin{array}{l}\text{literal}\\ \text{identifier}\end{array}\right\}\underline{\text{FROM}}\ \text{file-name}$$

All acquired sessions are automatically released when the application program terminates normally.

COBOL OPERATIONS SUMMARY CHART

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

		Communications Subsystem								
COBOL Operation	**Operation Mnemonic**	**Intra**	**BSCEL**	**CCP**	**CICS**	**IMS**	**3270**	**Finance**	**Peer**	**SNUF**
Accept input[1]	READ	x	x	x	x	x	x	x	x	x
Acquire	ACQUIRE	x	x	x	x	x	x	x	x	x
Cancel	$$CANLNI	x	–	–	–	–	–	–	–	x
Cancel then invite	$$CANL	x	–	–	–	–	–	–	–	x
End of session	$$EOS	x	x	x	x	x	x	x	x	x
Evoke	$$EVOKNI	x	x	x	x	x	x[2]	–	x	x
Evoke end of transaction	$$EVOKET	x	x	–	x	x	–	–	x	x
Evoke then invite	$$EVOK	x	x	x	x	x	x[2]	–	x	x
Fail	$$FAIL	x	–	–	–	–	–	–	x	–
Get[1]	READ	x	x	x	x	x	x	x	x	x
Get attributes[3]	ACCEPT	x	x	x	x	x	x	x	x	x
Invite[4]		x	x	x	x	x	x	x	x	x
Negative response	$$NRSPNI	x	–	–	–	–	–	–	–	x
Negative response then invite	$$NRSP	x	–	–	–	–	–	–	–	x
Pass-through put then invite	$$PTPUT	x	–	–	–	–	–	–	–	x
Pass-through invite	$$PTINV	x	–	–	–	–	–	–	–	x
Put	$$SENDNI	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
Put FMH	$$SENDNF	x	–	–	–	–	–	x	–	x
Put FMH then invite	$$SENDFM	x	–	–	–	–	–	x	–	x
Put then invite	$$SEND	x	x	x	x	x	x	x	x	x
Release	CLOSE	x	x	x	x	x	x	x	x	x
Request to change direction then invite	$$RCD	x	x	x	–	–	–	–	x	x
Set timer	$$TIMER	x	x	x	x	x	x	x	x	x

[1] The READ statement performs either a get or an accept input operation, depending on whether the TERMINAL option is specified and depending on the number of sessions acquired.

[2] Valid 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.

[3] Valid only when the ATTRIBUTE-DATA keyword is used on the ACCEPT statement.

[4] Valid only in conjunction with another operation or by using a $$SEND with a record length of 0.

RETURN CODE PROCESSING

Following each operation, a return code consisting of a major code and a minor code is given to the user 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 file status values as returned in the appropriate FILE STATUS data field.

Use this list to determine the SSP-ICF return code (or group of return codes, if the major portion of the code is followed by *xx*) that corresponds to the file status value. Then see the description of the SSP-ICF major code in your subsystem chapter. (For example, the 02xx group below is described in each subsystem chapter 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 your subsystem are described in that chapter. A summary chart of all the codes for all the subsystems is in Appendix A.

Return Code Groups	**COBOL File Status**
00xx, 03xx, 0800	00
01xx	01
02xx	9A
04xx	9I
1100	10
2800	9E
3401	9G
80xx	30
81xx	92
82xx	9C
83xx	9N

CODING EXAMPLES

See *How to Write Programs that Use the Intra Subsystem* in Chapter 7 for an example of how to write a COBOL program that uses SSP-ICF and the Intra subsystem. The programming example described in the Intra chapter is also shown in each of the other subsystem chapters with the changes needed to allow communications with that remote system.

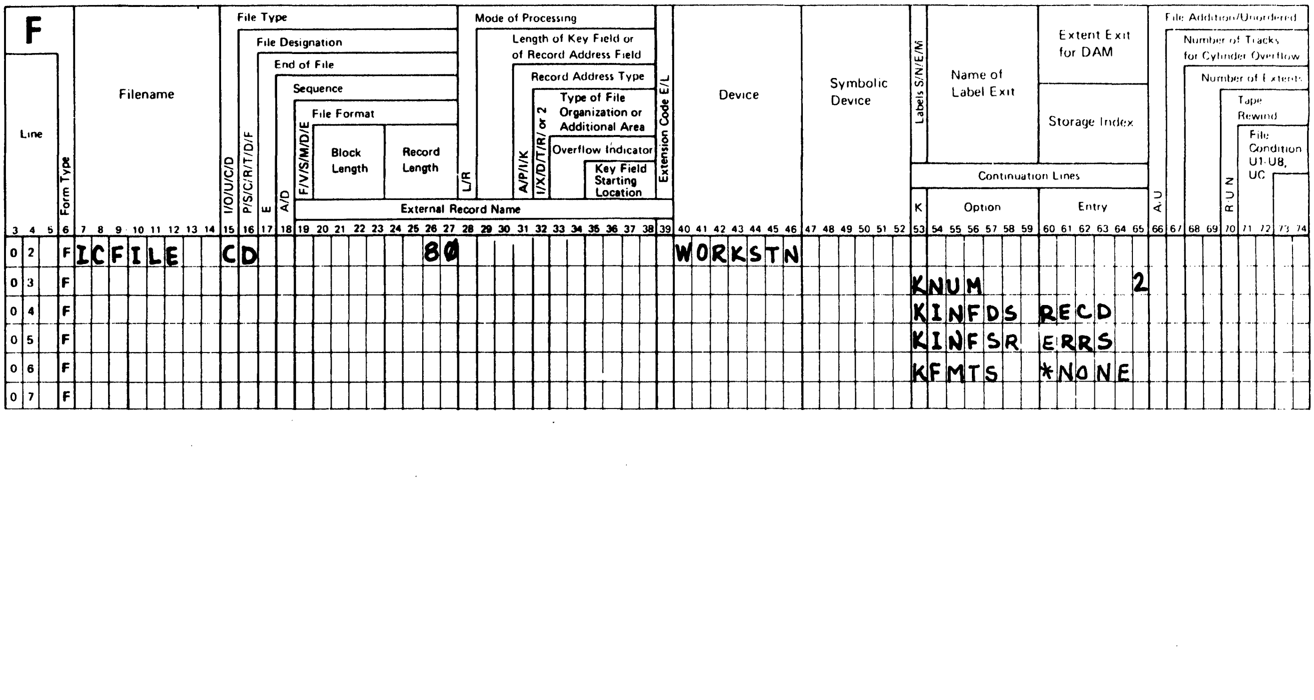
Chapter 6. Interactive Communications Programming with RPG II

The interactive communications portion of an RPG II program consists of preparing data for transmission, processing data that was received, using the predefined formats and existing work station operations to perform communications operations, and properly handling return codes. Because the data preparation and processing vary greatly by application, those functions are not described in this manual. The processing of interactive communications operations is very similar to that for work station operations. The file used is a WORKSTN file, the same input operations used, and the output operations are performed via formats.

FILE DESCRIPTION SPECIFICATION

When using RPG II for interactive communications, the file description specification must be completed. This specification should contain the same information as you would code for a WORKSTN file. The description of how to fill out the file description specification for a WORKSTN file is in the *RPG II Reference Manual*.

File Description Specifications



F
Line
Form Type
Filename
File Type
File Designation
End of File
Sequence
File Format
Block Length
Record Length
Mode of Processing
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
External Record Name
Device
Symbolic Device
Labels S/N/E/M
Name of Label Exit
Extent Exit for DAM
Storage Index
Continuation Lines
K
Option
Entry
A/U
File Addition/Unordered
Number of Tracks for Cylinder Overflow
Number of Extents
Tape Rewind
File Condition U1-U8, UC

```
02 FICFILE  CD         80            WORKSTN
03 F                                              KNUM        2
04 F                                              KINFDS RECD
05 F                                              KINFSR ERRS
06 F                                              KFMTS  *NONE
07 F
```

FORMATS

To assist in coding interactive communications operations in RPG II, predefined formats are provided. These formats are used in the same way that display screen formats are used. They are not identical, however, because the interactive communications formats are recognized by data management, making it unnecessary to separately store and process the formats via a screen format generator.

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

The following sections describe the available formats.

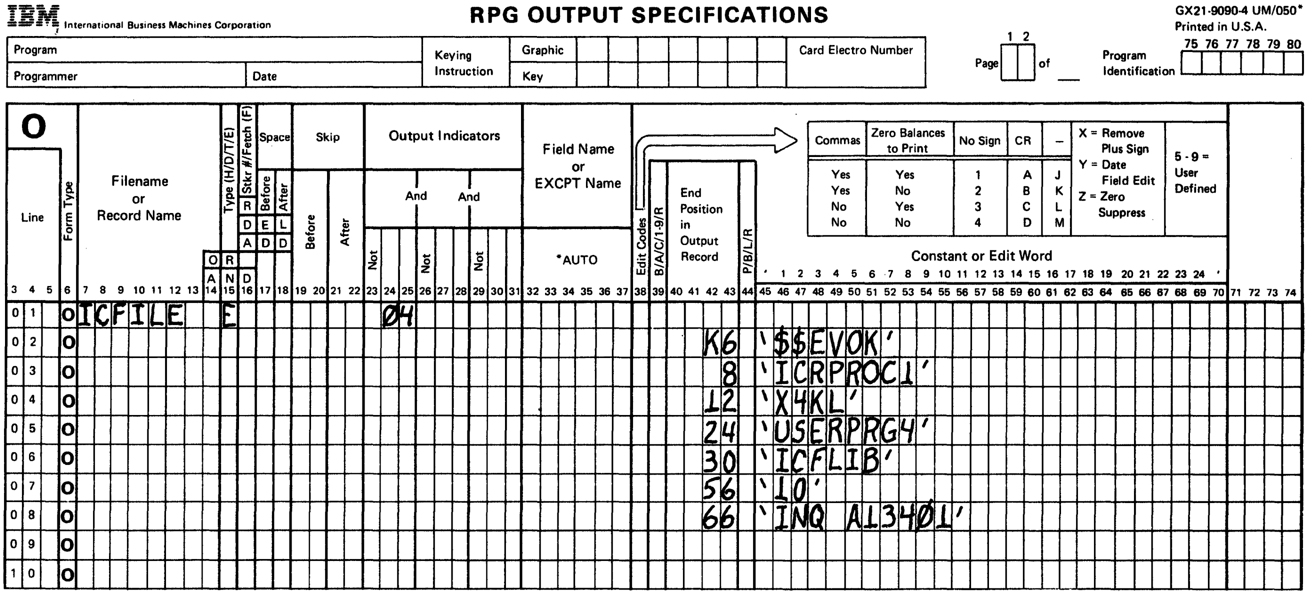
Evoke

Three formats are provided for evoking a transaction:

- $$EVOKNI performs an *evoke* operation.
- $$EVOK performs an *evoke then invite* operation.
- $$EVOKET performs an *evoke end of transaction* operation.

Each evoke request has several parameters associated with it; the first four parameters form the evoke parameter list. These parameters are defined as fields for the evoke formats.

Location	Description
1-8	The name of the remote procedure to be evoked (left-justified)
9-16	The password sent to the remote system (left-justified)
17-24	The user ID sent to the remote system (left-justified)
25-32	The library name containing the remote procedure (left-justified)
33-52	Reserved
53-56	Length, in decimal, of user data, if any (right-justified)
57-xxxx	User data or positional procedure parameters



IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21-9090-4 UM/050*
Printed in U.S.A.

```
01 OICFILE  E        Ø4
02 O                                   K6 '$$EVOK'
03 O                                    8 'ICRPROC1'
04 O                                   12 'X4KL'
05 O                                   24 'USERPRG4'
06 O                                   30 'ICFLIB'
07 O                                   56 '10'
08 O                                   66 'INQ A134Ø1'
09 O
10 O
```

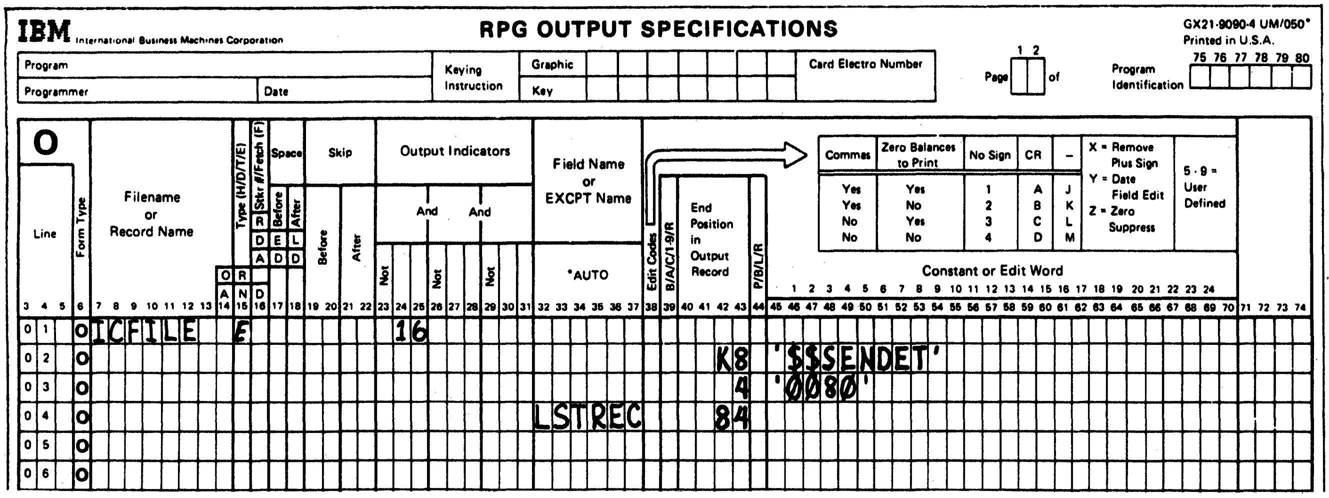
Transmit Data

Four formats are provided for transmitting a record:

- $$SENDNI performs a *put* operation.
- $$SEND performs a *put then invite* operation.
- $$SENDE performs a *put end of file* operation for BSC or a *put end of chain* operation for SNA.
- $$SENDET performs a *put end of transaction* operation.
- $$SENDNF performs a *put function management header* operation.
- $$SENDFM performs a *put function management header then invite* operation.

Each transmit request has two fields associated with it.

Location	Description
1-4	Length, in decimal, of the user data[1]
5-xx	User data to be transmitted



IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21-9090-4 UM/050*
Printed in U.S.A.

Program ______ Keying Instruction: Graphic / Key ______ Card Electro Number ______ Page □□ of ______ Program Identification □□□□□□

Programmer ______ Date ______

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Output Indicators	Field Name or EXCPT Name	End Position in Output Record	Constant or Edit Word
01	O	ICFILE	E	16			
02	O					K8	'$$SENDET'
03	O					4	'0080'
04	O				LSTREC	84	
05	O						
06	O						

[1] An output length of zero or blanks for $$SEND performs an *invite* operation. An output length of zero or blanks is also allowed for $$SENDE and $$SENDET, because the end of chain/file and end of transaction can be sent without data.

Request to Change Direction

The $$RCD format performs a *request to change direction* operation. This format has no fields associated with it.

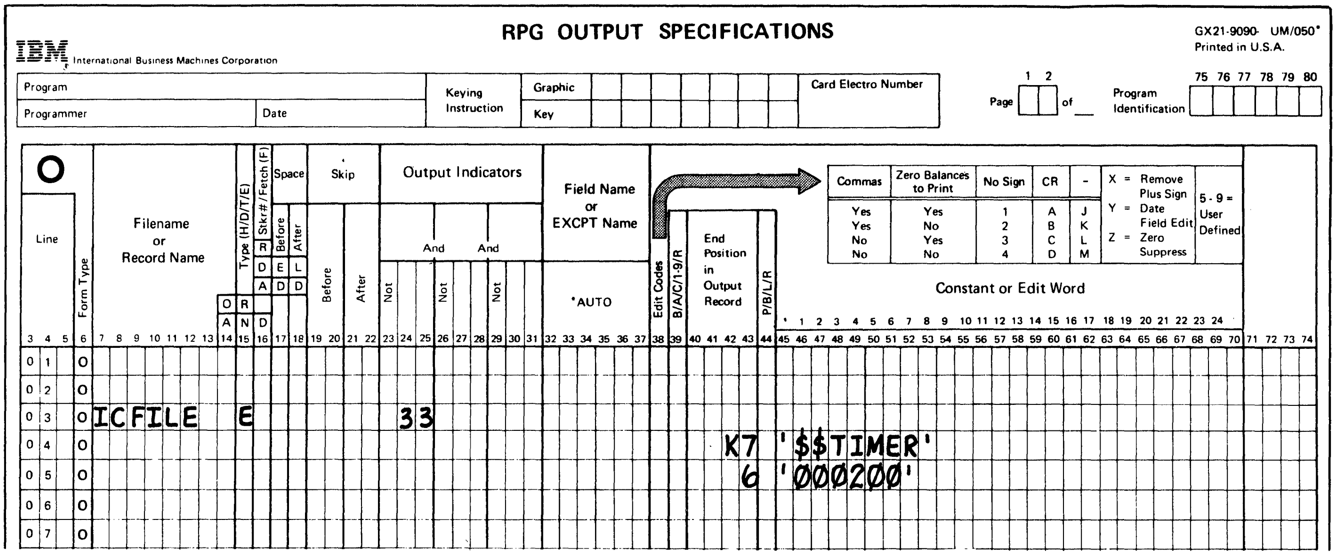
Set Timer Interrupt Value

The $$TIMER format performs a *set timer* operation. The following field is required with this format.

Location	Description
1-6	Interval of time to be set. The time should be specified in hours, minutes, and seconds (hhmmss).

Note: The $$TIMER format cannot be issued unless at least one requester or acquired device 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* and *READ* later in this chapter for more information about these operations).



IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21-9090- UM/050*
Printed in U.S.A.

```
  01 O
  02 O
  03 OICFILE  E        33
  04 O                                   K7 '$$TIMER'
  05 O                                    6 'ØØØ2ØØ'
  06 O
  07 O
```

Send Negative Response

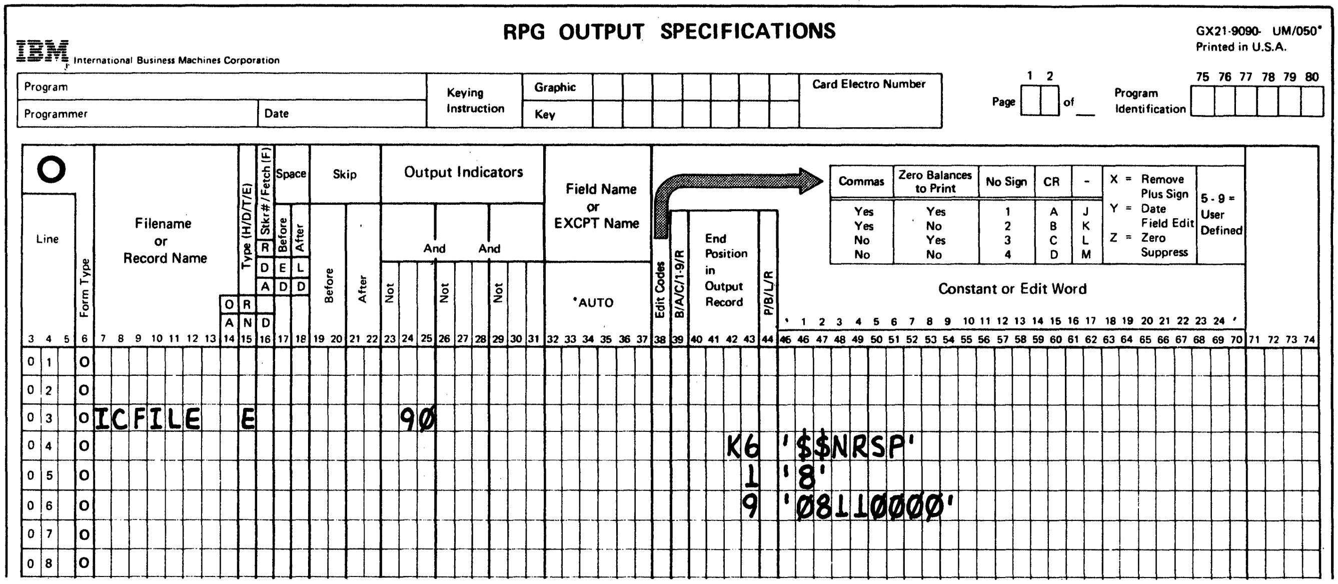
Two formats are provided for sending a negative response:

- $$NRSPNI performs a *negative response* operation.
- $$NRSP performs a *negative response then invite* operation.

The negative response formats can be used only for the Intra or SNUF subsystems. The negative response format can have the following two fields associated with it:

Location	**Description**
1	Length of sense data (must be 0 or blank indicating no sense data, or 8 if sense data is present)
2-9	The sense data to be sent with the negative response

Note: The sense data is user-defined, but the first four characters must be 10xx, 08xx, or 0000.



IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21-9090- UM/050*
Printed in U.S.A.

```
03 OICFILE  E        9Ø
04 O                                   K6 '$$NRSP'
05 O                                    1 '8'
06 O                                    9 'Ø811ØØØØ'
```

Fail

The $$FAIL format is used to issue a *fail* operation. This format has no fields associated with it. The $$FAIL format is valid only for the Intra and Peer subsystems.

Cancel

Two formats are provided for issuing a cancel operation:

- $$CANLNI performs a *cancel* operation.

- $$CANL performs a *cancel then invite* operation.

The cancel formats are valid only for the SNUF and Intra subsystems. These formats have no fields associated with them.

End of Session

The $$EOS format is used to issue an *end of session* operation. This format has no fields associated with it.

Pass-Through Formats

Two formats ($$PTPUT and $$PTINV) are provided for pass-through operations. These formats and a complete description of pass-through support are in Appendix B.

WORKSTN OPERATIONS

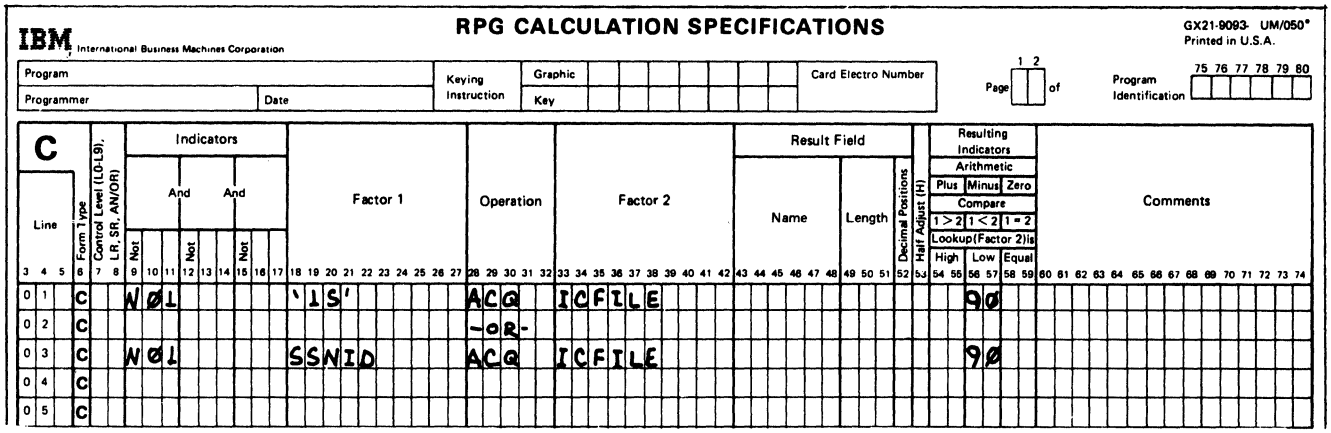
Several of the existing WORKSTN operations are used for interactive communications operations. They are:

- ACQ (acquire)
- REL (release)
- NEXT
- READ
- RPG cycle input

ACQ (Acquire)

The ACQ operation acquires the session specified by the ID (literal or variable) in factor 1. If the session is available, ACQ obtains it for this program. Factor 2 specifies the name of the WORKSTN file from the file description specification.

If the session cannot be acquired, an error occurs. If an indicator is specified in columns 56 and 57 for this operation, the indicator 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* later in this chapter for more details on error handling.



IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

GX21-9093- UM/050*
Printed in U.S.A.

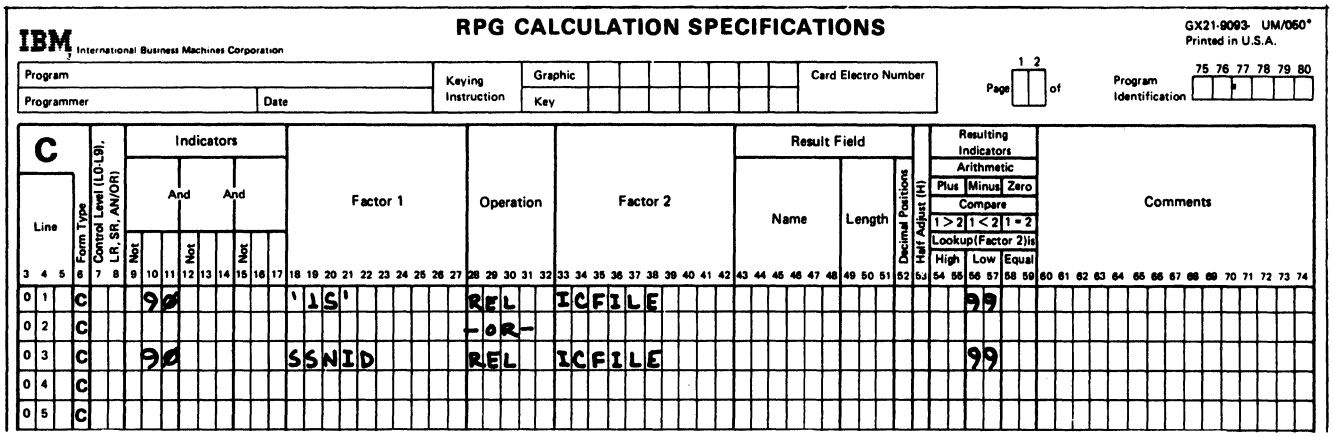
Program | Programmer | Date | Keying Instruction: Graphic, Key | Card Electro Number | Page 1 2 of | Program Identification 75 76 77 78 79 80

Line (3-5)	Form Type (6)	Control Level (L0-L9), LR, SR, AN/OR) (7-8)	Indicators (9-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field Name (43-48)	Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators High (54-55)	Low (56-57)	Equal (58-59)	Comments (60-74)
01	C		NØ1	'IS'	ACQ	ICFILE						9Ø		
02	C				-OR-									
03	C		NØ1	SSNID	ACQ	ICFILE						9Ø		
04	C													
05	C													

REL (Release)

The REL operation issues a release for the session specified in factor 1 (literal or variable). Factor 2 specifies the name of the WORKSTN file from the file description specification.

If an error occurs on the attempt to release the session, the indicator specified in columns 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* later in this chapter for more details on error handling.



IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

GX21-9093- UM/050*
Printed in U.S.A.

Program | Programmer | Date | Keying Instruction: Graphic, Key | Card Electro Number | Page 1 2 of | Program Identification 75 76 77 78 79 80

Line (3-5)	Form Type (6)	Control Level (L0-L9), LR, SR, AN/OR) (7-8)	Indicators Not (9)	Indicators (10-11)	And Not (12)	And (13-14)	And Not (15)	And (16-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field Name (43-48)	Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators High (54-55)	Low (56-57)	Equal (58-59)	Comments (60-74)
01	C			9Ø					'IS'	REL	ICFILE						99		
02	C									-OR-									
03	C			9Ø					SSNID	REL	ICFILE						99		
04	C																		
05	C																		

Resulting Indicators: Arithmetic: Plus, Minus, Zero; Compare: 1 > 2, 1 < 2, 1 = 2; Lookup(Factor 2)is: High, Low, Equal

For more specific information about the release operation, see the appropriate subsystem chapter.

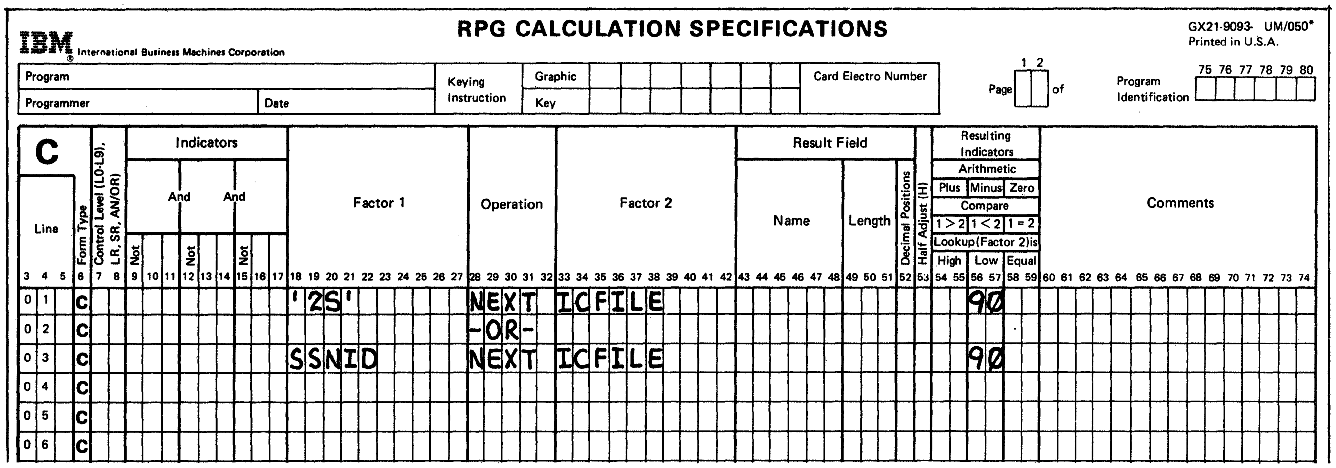
NEXT

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

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

If an error occurs on the NEXT operation, the indicator in columns 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* later in this chapter for more details on error handling.



IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

GX21-9093- UM/050*
Printed in U.S.A.

Program | Keying Instruction | Graphic | Key | Card Electro Number | Page 1 2 of | Program Identification 75 76 77 78 79 80

Programmer | Date

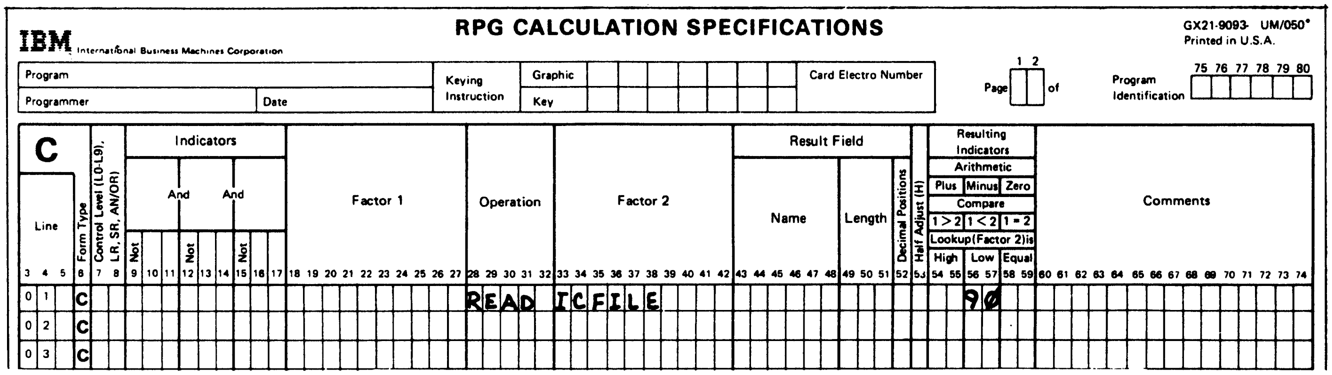
Line (3-5)	Form Type (6)	Control Level (L0-L9, LR, SR, AN/OR) (7-8)	Indicators (9-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field Name (43-48)	Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators High (54-55)	Low (56-57)	Equal (58-59)	Comments (60-74)
01	C			'2S'	NEXT	ICFILE						90		
02	C				-OR-									
03	C			SSNID	NEXT	ICFILE						90		
04	C													
05	C													
06	C													

READ

The READ operation requests input from any display station or session (accept operation) or, when used with the NEXT operation, from a specific display station or session (get operation). If the NUM field on the file description specification is 1, the READ does a get operation. If a NEXT operation has been executed since the last READ, the READ does a get to the session specified by the NEXT operation. If no NEXT has been executed since the last READ, the READ does an accept. Factor 2 contains the name of the file from which a record should be read.

If an error occurs during the READ operation, the indicator in columns 56 and 57 is set on. 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* later in this chapter for more details on error handling.

Columns 58 and 59 can contain an indicator to be set on when the end of file condition occurs. The end of file condition occurs for a session when an accept is issued with no outstanding invites. (See *End of File Considerations* later in this chapter for more information.) The end of file indicator is *not* set on when an end of transaction occurs. The INFDS must be checked to determine the end of a transaction.



IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

GX21-9093- UM/050*
Printed in U.S.A.

Program | Keying Instruction | Graphic | Card Electro Number | Page 1 2 of | Program Identification 75 76 77 78 79 80

Programmer | Date | Key

Line (3-5)	Form Type (6)	Control Level (L0-L9), LR, SR, AN/OR (7-8)	Indicators: Not, And, Not, And, Not (9-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field Name (43-48)	Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Plus / High 1>2 (54-55)	Minus / Low 1<2 (56-57)	Zero / Equal 1=2 (58-59)	Comments (60-74)
01	C				READ	ICFILE						9Ø		
02	C													
03	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 is an accept. The detail of the RPG program cycle is in the *RPG II Reference Manual.* Specifically, a section on WORKSTN file input processing describes the details of the cycle as it affects WORKSTN files.

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 subystem supports the operation. A - indicates that the subsystem does not support the operation.

RPG II Operation	Operation Mnemonic	Communications Subsystem								
		Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
Accept input[1]	READ	x	x	x	x	x	x	x	x	x
Acquire	ACQ	x	x	x	x	x	x	x	x	x
Cancel	$$CANLNI	x	–	–	–	–	–	–	–	x
Cancel then invite	$$CANL	x	–	–	–	–	–	–	–	x
End of session	$$EOS	x	x	x	x	x	x	x	x	x
Evoke	$$EVOKNI	x	x	x	x	x	x[3]	–	x	x
Evoke end of transaction	$$EVOKET	x	x	–	x	x	–	–	x	x
Evoke then invite	$$EVOK	x	x	x	x	x	x[3]	–	x	x
Fail	$$FAIL	x	–	–	–	–	–	–	x	–
Get[1]	READ	x	x	x	x	x	x	x	x	x
Invite[2]		x	x	x	x	x	x	x	x	x
Negative response	$$NRSPNI	x	–	–	–	–	–	–	–	x
Negative response then invite	$$NRSP	x	–	–	–	–	–	–	–	x
Pass-through put then invite	$$PTPUT	x	–	–	–	–	–	–	–	x
Pass-through invite	$$PTINV	x	–	–	–	–	–	–	–	x
Put	$$SENDNI	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
Put FMH	$$SENDNF	x	–	–	–	–	–	x	–	x
Put FMH then invite	$$SENDM	x	–	–	–	–	–	x	–	x
Put then invite	$$SEND	x	x	x	x	x	x	x	x	x
Release	REL	x	x	x	x	x	x	x	x	x
Request to change direction then invite	$$RCD	x	x	x	–	–	–	–	x	x
Set timer	$$TIMER	x	x	x	x	x	x	x	x	x

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

[2] Valid only when used with another operation or by using a $$SEND operation with a record length of 0.

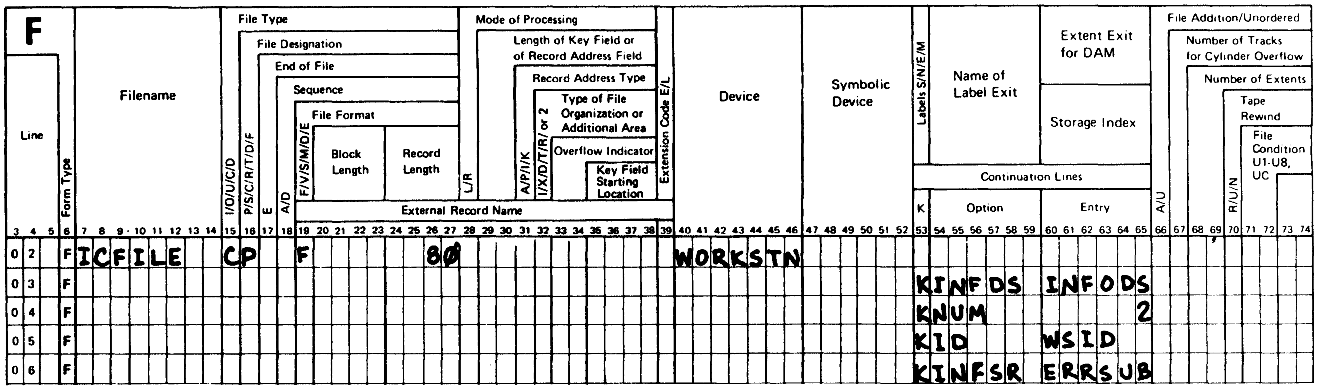
[3] Valid only when the DATAID and FLDLTH parameters are specified on the SESSION OCL statement, and when the HOSTNAME parameter on the SESSION statement is CICS or IMS.

RETURN CODE PROCESSING

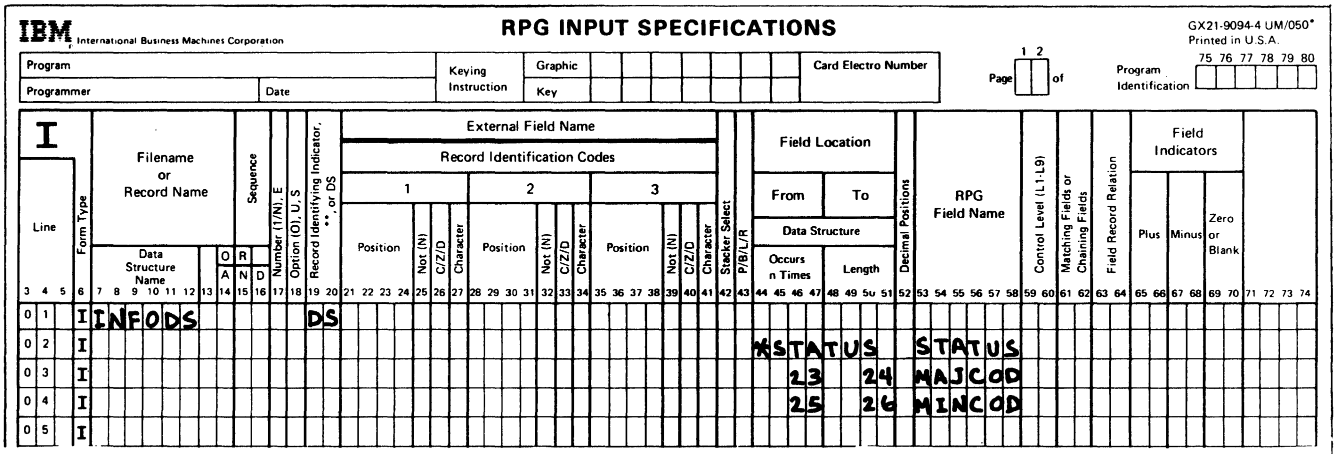
Following each operation, a return code that indicates the results of the operation is issued.

The exception/error processing subroutine (INFSR) and error indicators in columns 56 and 57 of the WORKSTN operation codes (REL, ACQ, NEXT, and READ) allow the programmer to control the program logic if errors occur during WORKSTN file processing. The WORKSTN file information data structure (INFDS) contains status information that the program can check to determine what type of exception or error occurred. Using the information in the INFDS, the program can then determine which conditions to handle in the INFSR subroutine.

File Description Specifications



```
02 FICFILE  CP  F        80            WORKSTN
03 F                                              KINFDS INFODS
04 F                                              KNUM        2
05 F                                              KID    WSID
06 F                                              KINFSR ERRSUB
```



IBM International Business Machines Corporation

RPG INPUT SPECIFICATIONS

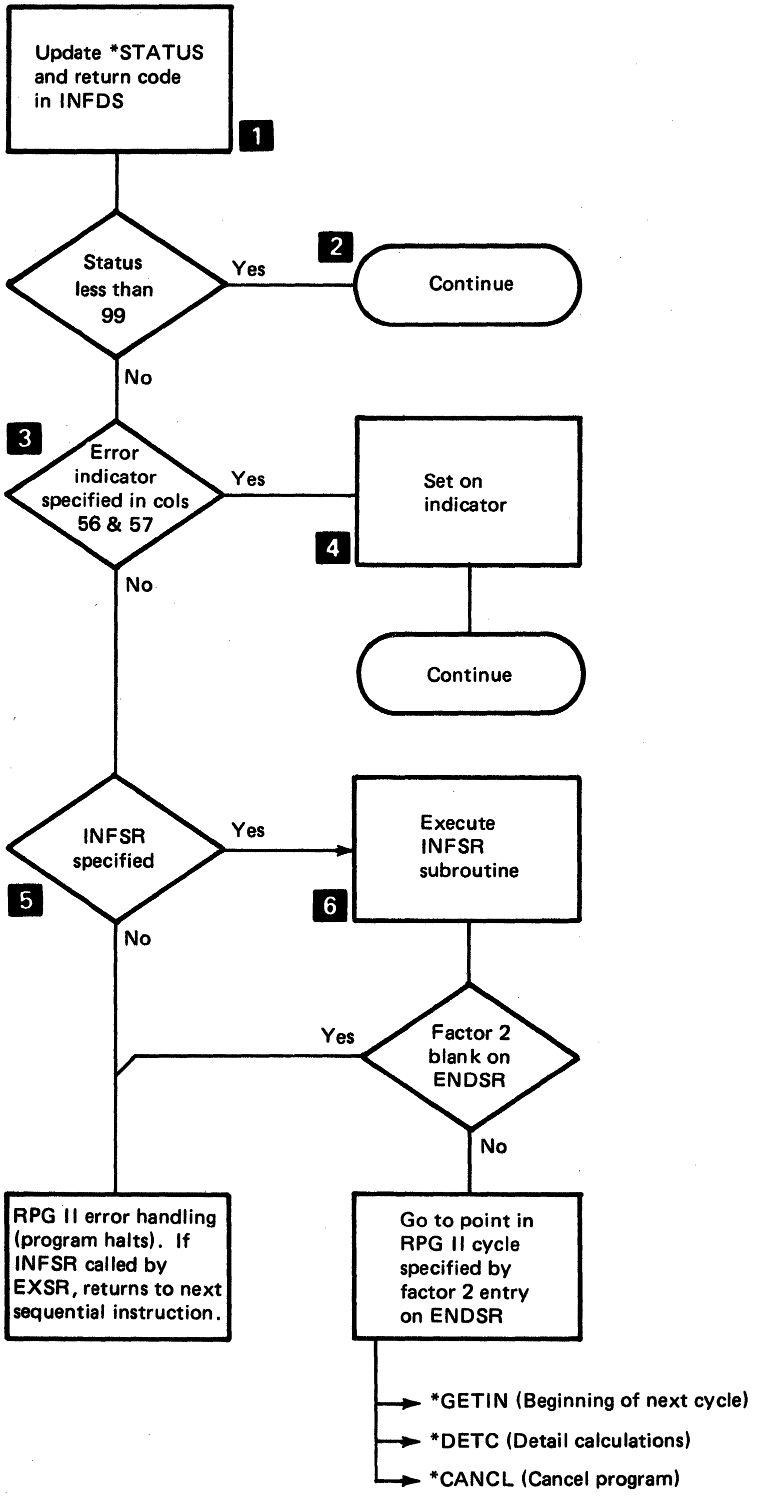
GX21-9094-4 UM/050*
Printed in U.S.A.

```
01 IINFODS      DS
02 I                                         *STATUS  STATUS
03 I                                           23  24 MAJCOD
04 I                                           25  26 MINCOD
05 I
```

If neither the INFSR subroutine nor error indicators are specified, an error is handled by the RPG II error handling routine, which causes a program to halt. The operator must choose the appropriate option.

The INFDS data structure, if specified, contains the return code identifying the exception or error that occurred. The INFDS also contains status information on normal conditions as well as exceptions or errors. The information in the INFDS is updated for each operation. If an exception or error occurs, the programmer can use the INFDS information to determine the cause and to control the resulting program logic.

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


Update *STATUS and return code in INFDS
1
Status less than 99
Yes
2
Continue
No
3
Error indicator specified in cols 56 & 57
Yes
Set on indicator
4
No
Continue
INFSR specified
Yes
Execute INFSR subroutine
5
6
No
Yes
Factor 2 blank on ENDSR
No
RPG II error handling (program halts). If INFSR called by EXSR, returns to next sequential instruction.
Go to point in RPG II cycle specified by factor 2 entry on ENDSR
*GETIN (Beginning of next cycle)
*DETC (Detail calculations)
*CANCL (Cancel program)

1 When an operation is completed the status information (*STATUS and the return code) is updated in INFDS.

2 If the condition is normal, the next instruction in the RPG program is executed.

3 If the condition is an exception or error (*STATUS greater than 99), a check is made to see whether an indicator was specified in columns 56 and 57 of the calculation specification 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 RPG program is executed. In this case, if the INFSR subroutine is to be executed, an EXSR operation can be issued.

5 If no indicator was specified, a check is made to see whether an INFSR was specified. If not or if factor 2 on the ENDSR is blank, RPG issues a halt on the system console.

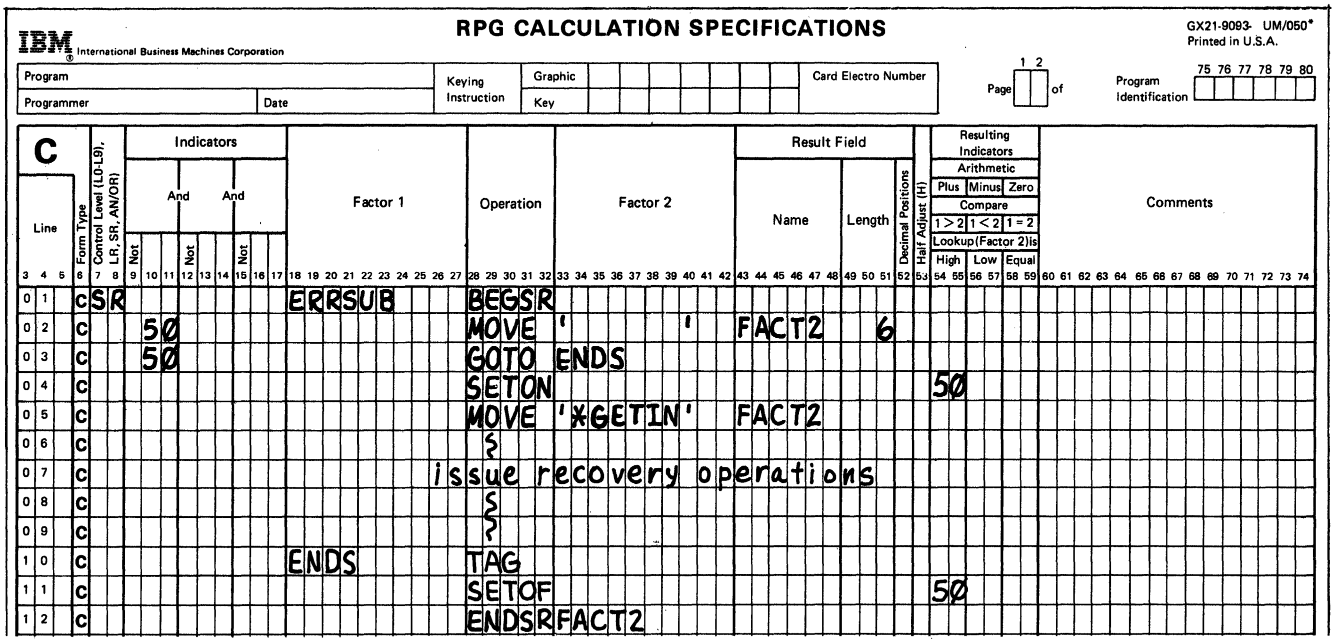
6 If INFSR was specified and factor 2 of the ENDSR is not blank, control is passed to the point specified by factor 2 on the ENDSR. Factor 2 can be *GETIN to go to the beginning of the next input cycle, *DETC to perform detail calculations, *CANCL to cancel the program, or a variable that contains one of these values.

INFSR Coding Considerations

If an INFSR subroutine is coded, return codes 80xx and 81xx should be handled. If any of these codes occur, the INFSR subroutine should issue a release operation to the display station or session. This clears the RPG internal table entry for that display station or session and allows that entry to be used by a subsequent requester. For the session errors mentioned above, an end of session operation ($$EOS) can also be issued.

The return code indicating timer expired (code 0310) causes the INFSR subroutine to be executed. If the set timer operation ($$TIMER) is used, be sure to check for this return code.

When the INFSR subroutine is specified for the WORKSTN file, any exception error encountered for that file causes the INFSR subroutine to be executed. Therefore, if operations are issued from the subroutine to the WORKSTN file that can cause exceptions or errors, be careful to code the subroutine to prevent loops. An advisable technique is shown as follows:



IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

GX21-9093- UM/050* Printed in U.S.A.

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

```
01 CSR       ERRSUB    BEGSR
02 C  5Ø               MOVE ' '       FACT2   6
03 C  5Ø               GOTO ENDS
04 C                   SETON                       5Ø
05 C                   MOVE '*GETIN'  FACT2
06 C                   ⋮
07 C              issue recovery operations
08 C                   ⋮
09 C                   ⋮
10 C         ENDS      TAG
11 C                   SETOF                       5Ø
12 C                   ENDSRFACT2
```

Indicator 50 is set on whenever the INFSR subroutine (ERRSUB) is entered for the first time. If any errors occur in ERRSUB that would cause the subroutine to be reentered, the subroutine exits to the RPG error handler (factor 2 is blanks). The error handler displays the appropriate error message. If the operations issued in the subroutine do not cause exceptions or errors, the subroutine exits to the start of the RPG cycle.

RPG II STATUS VALUE

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 your subsystem chapter. All of the return codes that are valid for your subsystem are described in that chapter. A summary chart in Appendix A shows which codes are valid for each subsystem.

Return Codes		
Major	**Minor**	***STATUS**
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
04	02, 11, 12	01299
08	00	01285
11	00	00011
28	00	00000
34	01	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

When writing interactive communications programs in RPG II, keep the following considerations in mind:

- Continuation lines on the file specification
- SRT and MRT considerations
- End of file considerations
- Release considerations
- Restrictions for WORKSTN files
- Input and output considerations

Information on these and other considerations for WORKSTN file programming is in Chapter 13 of the *RPG II Reference Manual*.

Continuation Lines on the File Specification

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

- NUM
- SAVDS
- IND
- SLN
- ID
- INFSR
- INFDS
- FMTS

NUM

The NUM continuation option is used to specify 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 number of 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 is 4 and the NUM value is 5, only one session can be acquired at a time. The number specified must be right-justified in columns 60 through 65.

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

SAVDS

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 preexecution-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 SAVDS 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 ID of a display station for which an interactive communications request has been made.

IND

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-justified in columns 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 IND available is that of the display station or session from which the last input came.

SLN

The SLN continuation option is used to specify the starting line number for display screen formats. The SLN option does not apply to sessions.

ID

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

INFSR

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

INFDS

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

FMTS

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

SRT and MRT Program Considerations

An SRT 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 specification must reflect the maximum number of concurrently attached sessions and display stations (all those that are acquired plus one requester).

An MRT 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 specification must include the maximum number of requesters plus the number of sessions and display stations that are acquired and that are active simultaneously.

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 invites. (This second condition could occur because no invites were issued or because all display stations and sessions have been released.) If the program has the NEP attribute, the previous two conditions must be satisfied and the system operator must have entered the STOP SYSTEM command.

For 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 columns 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.

Release Considerations

A release can be performed explicitly by using the REL operation (described earlier in this chapter) or by coding an R in column 16 of the output specifications. If a format name is specified in the same specification that contains an R in column 16, the format is displayed or the interactive communications operation is performed before the display station or session is released. If a program terminates before releasing any display stations or sessions, they are automatically released.

If a session was acquired, the release 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 request or a display station requests the program, the release 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:

- WORKSTN file programs cannot be run from the input job queue, and cannot be initiated by an EVOKE OCL statement.

- 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 per program.

- A program cannot contain a KEYBORD, 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 with the request is lost. This is accomplished by placing the requester's ID or blanks in the ID field and performing output as the first operation to the file. (See *Writing Procedures to be Started by Incoming Procedure Requests* in Chapter 2 for other considerations.)

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, the next input operation retrieves a data record.

CODING EXAMPLES

See *How to Write Programs that Use the Intra Subsystem* in Chapter 7 for an example of how to write an RPG II program that uses SSP-ICF and the Intra subsystem. The programming example described in the Intra chapter is also shown in each of the other subsystem chapters with the changes needed to allow communications with that remote system.

Chapter 7. The Intra Subsystem

The Intra subsystem provides distributed data processing support to users of the System/34 SSP by providing an interactive interface between application programs on the same System/34. The Intra subsystem can support multiple application programs communicating concurrently.

The Intra subsystem allows System/34 application programs to initiate procedures on the same system. Some System/34 security options are supported.

The Intra subsystem is useful for several types of applications. Some of these are:

- To test new interactive communications applications without using a communications line. You might have to make some coding changes before actually running the program. In particular, the return code checking might need to be modified.

- To allow the same program to make inquiries into both the local and remote systems.

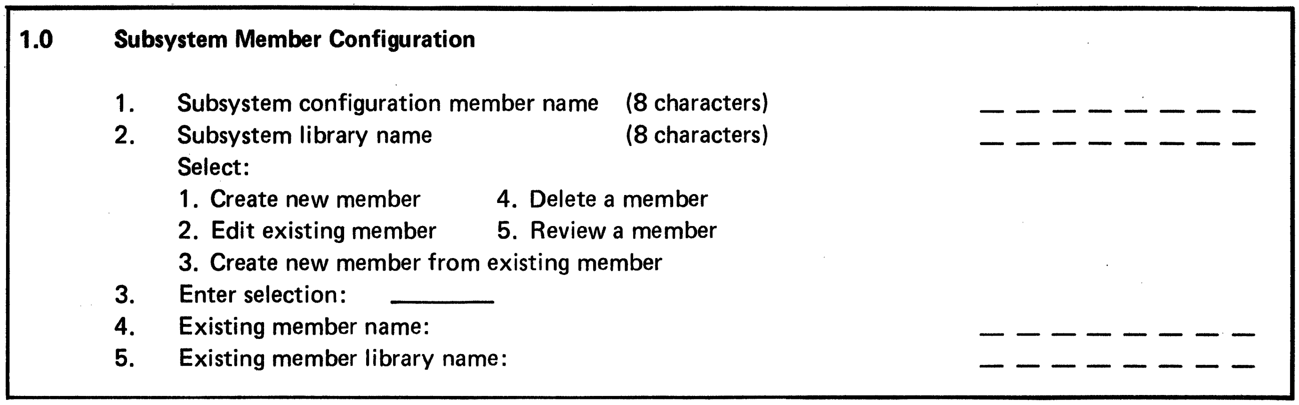
When figuring the number of concurrent sessions, be aware that each Intra session with an active transaction counts as two sessions (one for each program) against the system maximum of 100 sessions.

SETTING UP THE INTRA SUBSYSTEM

The SSP procedures CNFIGSSP and INSTALL are used to include the interactive communications feature and Intra subsystem support on the System/34. The general interactive communications support is included when it is requested on the appropriate CNFIGSSP prompt. The Intra subsystem support is copied to the system library when the appropriate responses to the INSTALL procedure prompts are taken. The CNFIGSSP and INSTALL procedures, with their displays and related responses, are described in the *Installation and Modification Reference Manual.*

After the Intra subsystem has been installed, the CNFIGICF procedure is used to define the subsystem support. The operation of the CNFIGICF procedure is also explained in the *Installation and Modification Reference Manual.* Before running the CNFIGICF procedure, however, you should fill out a planning chart for each subsystem that you want to define. Copies of the planning chart for each subsystem are available in Appendix F of this manual and in the *Installation and Modification Reference Manual.* The following sections explain how to fill out the planning chart for the Intra subsystem.

Display 1.0 Subsystem Member Configuration



```
1.0     Subsystem Member Configuration

        1.  Subsystem configuration member name  (8 characters)     _ _ _ _ _ _ _ _
        2.  Subsystem library name               (8 characters)     _ _ _ _ _ _ _ _
            Select:
            1. Create new member       4. Delete a member
            2. Edit existing member    5. Review a member
            3. Create new member from existing member
        3.  Enter selection: ____
        4.  Existing member name:                                   _ _ _ _ _ _ _ _
        5.  Existing member library name:                           _ _ _ _ _ _ _ _
```

Subsystem configuration member name: Specify a name for this configuration of the subsystem. This name is used to store the member in a library, and is referenced in the ENABLE and DISABLE procedures.

Library name: Specify the name of the library in which the configuration is stored or to be stored. The default is #LIBRARY, however, you should probably store the configuration in a user library.

Enter selection: Specify one of the five options: (1) create a new member, (2) edit an existing member, (3) create a new member from an existing member, (4) delete a member, or (5) review a member without changing it.

Existing member name: This prompt appears if option 3 was selected. Specify the name of the existing subsystem configuration member that is to be used to create the new member. The existing member remains unchanged.

Existing member library name: This prompt appears if option 3 was selected. Specify the library name where the existing member resides.

Display 2.0 Common SSP-ICF Parameters for Each Subsystem

2.0 Common SSP-ICF Parameters for Each Subsystem

1. SSP-ICF common queue space: (2 - 42 K) __ __
2. Define the subsystem type: __ 1

1 Intra	2 BSC IMS/IRSS
3 BSCEL	4 BSC CICS
5 BSC CCP	6 SNA Upline
7 SNA Peer	8 BSC 3270
9 SNA 3270	10 Finance

SSP-ICF common queue space: Specify the size, in multiples of 2 K bytes, of the common queue space. The common queue space requirements for each configuration of the Intra subsystem enabled are 32 bytes.

The common queue space value specified for the first subsystem that is enabled becomes the size of the common queue space. Be sure that the value specified for common queue space size takes into account the requirements of any other subsystem that might be active concurrently.

The size of the common queue space plus the total subsystem queue space of all the enabled Intra subsystem cannot exceed 42 K bytes.

The default common queue space size is 4 K bytes.

Define the subsystem type: Specify a 1 for the Intra subsystem.

Display 3.0 General Subsystem Parameters

3.0 General Subsystem Parameters

1. Location name: (8 characters) _ _ _ _ _ _ _ _
2. Subsystem queue space: (0-40 K) _ _
3. Subsystem support swappable: (0-No 1-Yes) _

Location name: Specify up to 8 characters for the name of the location associated with this configuration. The location name is used in some displayed message texts, and must be coded on the SESSION OCL statement. The location name refers to the name of the location with which communications is to take place. If you do not enter a location name, the system uses the subsystem configuration member name for the location name.

Subsystem queue space: Specify the size, in multiple of 2 K bytes, of the subsystem queue space. The subsystem queue space requirements for each configuration of the Intra subsystem enabled is:

$$S = L_1 + L_2 + \dots + L_n$$

where:
S = number of bytes required for the subsystem queue space
L = maximum record length for each acquired session

The size of the common queue space plus the total subsystems queue space of all the enabled Intra subsystem cannot exceed 42 K bytes.

The default subsystem queue space size is 4 K bytes. If the subsystem queue space is set to 0 K bytes, the common queue space is used. In this case, the subsystem requirements must be added to the common queue space requirements.

Subsystem support swappable: Specify whether you want the subsystem to be swappable. Consider the total system performance, the size of the subsystem, and the amount of user main storage when determining whether you want the subsystem swappable. The Intra subsystem requires 2 K bytes of main storage.

STARTING AND ENDING THE INTRA SUBSYSTEM

The ENABLE procedure is the means of starting the Intra subsystem on the System/34. The ENABLE procedure associates the subsystem with a particular configuration.

The DISABLE procedure stops the subsystem. When a disable is performed, the Intra subsystem no longer handles application program requests.

The formats of the ENABLE and DISABLE procedure commands are in Chapter 2.

STARTING INTRA SUBSYSTEM APPLICATIONS

System/34 Intra subsystem applications can be started by a display station operator entering a procedure command or by a request from another application program. Procedures that are started by a System/34 operator must have a SESSION OCL statement for each session to be started. The following sections describe the SESSION OCL statement and the procedure start requests.

SESSION OCL Statement

The format of the SESSION OCL statement for the Intra subsystem is:

// SESSION LOCATION-name,SYMID-session-id

$$\left[,\text{BATCH-}\left\{\begin{matrix}\text{YES}\\ \underline{\text{NO}}\end{matrix}\right\}\right]$$

LOCATION: Specifies the location name associated with this session. The location name is defined during subsystem configuration, and refers to the name of the location with which communication is to take place.

SYMID: Specifies the symbolic ID of the session with which this OCL statement is associated. The symbolic ID must be two characters, with the first character numeric (0 through 9) and the second character alphabetic (A through Z, #, $, or @). This is the same ID that the application program uses when referring to this session. This ID is the equivalent of the symbolic display station ID as specified on the WORKSTN OCL statement. This parameter has no default.

BATCH: 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.

Note: If the application program is a BASIC program, the SESSION statement is not required unless you need to specify BATCH-YES.

Procedure Requests

For application programs to initiate procedures on the System/34, the program must issue an evoke operation. The subsystem then starts the System/34 application and communications can begin.

OPERATION CONSIDERATIONS

The following sections describe the operations supported by the Intra subsystem. A complete chart of all the interactive communications operations and the subsystems that support them is in each language chapter. The chart also shows the keyword or format name used to code the operation. More information about how an operation is coded is also described in the appropriate programming language chapter.

Whether an operation completes successfully or unsuccessfully, a return code is given to the application program. All of the return codes that are valid for the Intra subsystem are described at the end of this chapter. A summary chart in Appendix A lists all the return codes and the subsystems for which they are valid.

Acquire Operation

The acquire operation establishes a session. Associated with the acquire is a session ID (corresponding to the SYMID parameter on the SESSION OCL statement) that identifies this session. When the acquire operation completes successfully, a session with this ID exists.

Evoke Operations

The evoke operation ($$EVOK, $$EVOKNI, or $$EVOKET) initiates a procedure. For an evoke operation with procedure parameters and data specified, the total length of the procedure name, parameters, and data cannot exceed 120 bytes.

When security is active, the subsystem compares the user ID from the evoke operation with user ID specified at sign on to the display station running the application program. If the IDs are the same, further security checking is bypassed.

The evoke operation with the function management header modifier (assembler only) is valid only if BATCH-YES was specified on the SESSION OCL statement of the program that acquired the session. See Chapter 15 for a description of function management headers.

Put Operations

The put operation ($$SEND, $$SENDNI, $$SENDE, or $$SENDET) sends a record to the other application program. Put operations are valid only during a transaction.

Put function management header is valid only if BATCH-YES was specified on the SESSION OCL statement for the program that acquired the session. Any put function management header 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. See Chapter 15 for a description of function management header operations. (Function management headers have no particular use in the Intra subsystem environment, but are supported for compatability with the SNUF subsystem.)

Input Operations

The input operations for the Intra subsystem are invite, get, and accept. The invite operation can be issued only as a combined operation with a put or evoke operation ($$SEND, $$EVOK) in BASIC, COBOL, and RPG II. Assembler language users can issue an invite operation explicitly. Either a get or invite operation signals the subsystem to obtain data on the session for the application program. A get operation causes the application program to wait for the data to be available. When a program issues an invite operation, it receives the data with the next accept operation. The accept operation allows data from any previously invited session.

Request to Change Direction Operation

The Intra subsystem allows a request to change direction operation ($$RCD) only during a transaction and only when the issuing program is receiving. If the issuing program is receiving data, the operation results in a return code being given to the other application program for the next output operation. If the issuing program is not receiving and not transmitting, the request to change direction operation has no effect. The operation is valid only if BATCH-YES was specified on the SESSION OCL statement for the program that acquired the session.

Negative Response Operation

The negative response operation ($$NRSP or $$NRSPNI) indicates to the other application program that data was not received correctly. Eight bytes of data are passed with the negative response indication. Negative response is only valid while receiving data within a chain or as the first operation after the end of chain, and only if BATCH-YES was specified on the SESSION OCL statement for the program that acquired the session.

The 8 characters of user data should contain user-defined sense information to indicate the reason for the negative response. The Intra subsystem checks to ensure that the first 4 characters are 10xx, 08xx, or 0000.

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 data. The only valid response to a negative response is a cancel.

Cancel Operation

The cancel operation ($$CANL, $$CANLNI) sends a cancel return code to the other application program. The cancel return code indicates to the receiving program to abnormally end this group (chain) of data records and to disregard previous records in this group (all records sent since the previous end of chain). The cancel operation can be issued alone or with an invite or get. The cancel operation should only be issued while transmitting data. A cancel operation does not end a session.

The cancel and negative response operations can be considered as a pair. Cancel is the appropriate response to a negative response. However, if the transmitting program detects an error, cancel can be sent without first receiving a negative response. The cancel operation is valid only if BATCH-YES was specified on the SESSION OCL statement for the program that acquired the session.

Fail Operation

The fail operation ($$FAIL) indicates to the receiving program that an abnormal condition has occurred. The fail operation can be issued while the program is sending or receiving. If a program issues a fail operation while sending, it indicates that the data just sent was in error. All data sent before the fail operation is transmitted to the receiving program, and a return code indicating the fail is given to the receiving program. If a program issues a fail operation while receiving, it indicates that the data received was in error. The subsystem discards all subsequent data until the transmitting subsystem acknowledges receipt of the fail operation. In either case, the program that issued the fail operation must transmit, and the program that receives the fail return code must receive. If both programs issue a fail operation simultaneously, the program that was receiving will be successful and must transmit. The program that was transmitting will receive an unsuccessful return code and must begin receiving. No data can accompany the fail operation.

Release Operation

The release operation is an attempt by the issuing program to terminate the session. Release performs different actions depending on the type of session:

- If the session was acquired by the issuing program, the release operation terminates the session. The same or another session can then be acquired.
- If the session was started by an incoming procedure request and the issuing program is a MRT program, the release operation passes the session to the next step in the procedure. The SSP then executes any further OCL in the procedure.
- If the session was started by an incoming procedure request and the issuing program is a SRT program, the release is delayed until the issuing program terminates. After the issuing program terminates, the session is passed to the next step in the procedure.

A release operation for an acquired session can only be performed if no transaction is active; that is, end of transaction has been successfully sent or received. See Chapter 2 for more information about the release operation.

End of Session Operation

The end of session operation ($$EOS) always results in a normal completion return code. The session is always terminated by the end of session operation. If the session is still communicating when the end of session operation is issued, the transaction is abnormally terminated by the Intra subsystem, and abnormal termination of the other application program could result.

Get Attributes Operation

The get attributes operation (assembler only) can be issued at any time to determine the status of a session. If you are using BASIC, you can use the ATTRIBUTE$ intrinsic function to determine the status of a session.

Set Timer Operation

The set timer operation ($$TIMER) results in a timer expired return code (0310) after a specific time interval in hours, minutes, and seconds has expired.

Pass-Through Operations

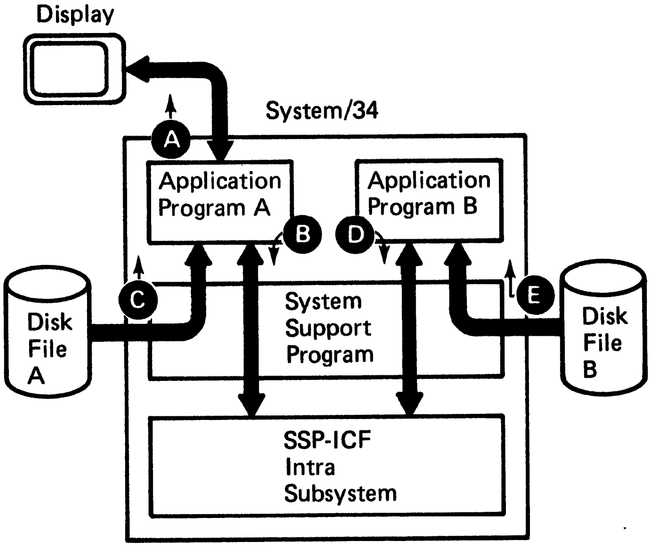
Pass-through operations to be used with the SNUF subsystem are allowed, but not fully supported, by the Intra subsystem. Return codes given by the Intra subsystem may differ from those that the SNUF subsystem issues for the same operation.

How to Write Programs that Use the Intra Subsystem

The following programming examples show you how to write a BASIC, a COBOL, and an RPG II program that uses the Intra subsystem. The configuration parameters and OCL statements used for the programming examples are also shown.

These examples are used for all subsystems except the 3270 subsystem. Each subsystem chapter contains a description of the configuration parameters and the OCL statements for that subsystem. If any changes are required to the programs for that remote system, those changes are shown in that subsystem chapter.

The following inquiry application is used in the programming examples:



1. Application program A displays a prompt asking an operator to enter an item number requesting the stock status for the item Ⓐ.

2. When the operator enters the item number, program A reads the number and searches disk file A (the local file) for the item number Ⓒ.

3. If the item number is in the local file, program A displays the stock status on the screen Ⓐ.

4. If the item is *not* in the local file, program A uses the Intra subystem to send the item number to application program B Ⓑ.

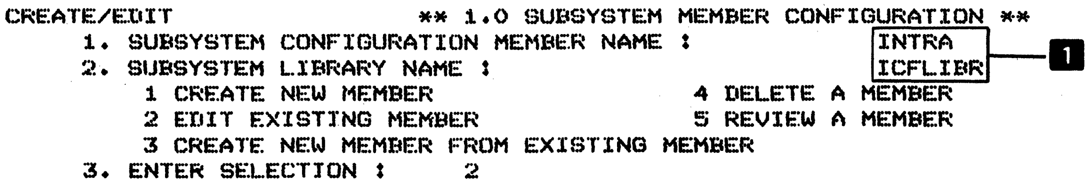
5. Application program B uses the item number to search disk file B (remote file) for the item Ⓔ.

6. If the item is found in the remote file, program B uses the Intra subsystem to send the stock status to program A Ⓓ. If the item is not in the remote file, program B sends the characters *** to program A.

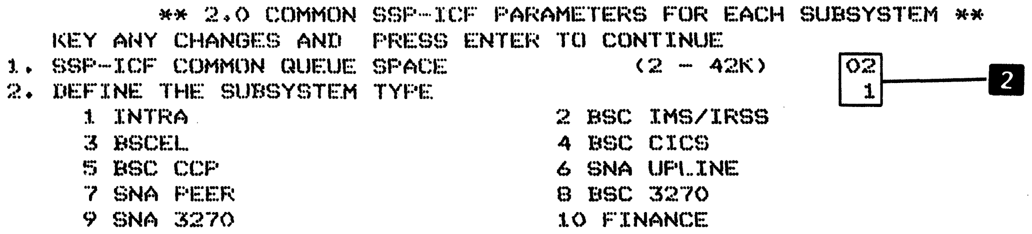
7. If program A receives the stock status, it displays it. If the program receives the characters ***, it displays the message ITEM NOT FOUND Ⓐ.

CONFIGURATION PARAMETERS

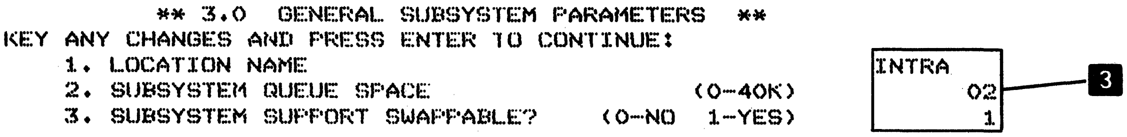
The following configuration parameters are used for this example:



```
CREATE/EDIT                    ** 1.0 SUBSYSTEM MEMBER CONFIGURATION **
     1. SUBSYSTEM CONFIGURATION MEMBER NAME :               INTRA
     2. SUBSYSTEM LIBRARY NAME :                            ICFLIBR
         1 CREATE NEW MEMBER                      4 DELETE A MEMBER
         2 EDIT EXISTING MEMBER                   5 REVIEW A MEMBER
         3 CREATE NEW MEMBER FROM EXISTING MEMBER
     3. ENTER SELECTION :       2
```



```
              ** 2.0 COMMON SSP-ICF PARAMETERS FOR EACH SUBSYSTEM **
     KEY ANY CHANGES AND  PRESS ENTER TO CONTINUE
   1. SSP-ICF COMMON QUEUE SPACE                  (2 - 42K)      02
   2. DEFINE THE SUBSYSTEM TYPE                                   1
        1 INTRA                              2 BSC IMS/IRSS
        3 BSCEL                              4 BSC CICS
        5 BSC CCP                            6 SNA UPLINE
        7 SNA PEER                           8 BSC 3270
        9 SNA 3270                           10 FINANCE
```



```
            ** 3.0  GENERAL SUBSYSTEM PARAMETERS   **
KEY ANY CHANGES AND PRESS ENTER TO CONTINUE:
    1. LOCATION NAME                                        INTRA
    2. SUBSYSTEM QUEUE SPACE                 (0-40K)           02
    3. SUBSYSTEM SUPPORT SWAPPABLE?    (0-NO  1-YES)            1
```

1. On display 1.0, the configuration member name (INTRA) and the library name (ICFLIBR) in which the member is stored are specified.

2. On display 2.0, an SSP-ICF common queue space of 2 K bytes is specified, and the subsystem type selected is the Intra subsystem.

3. On display 3.0, the remote system location name is specified (INTRA), and a subsystem queue space of 2 K bytes is specified. The location name must also be entered on the SESSION OCL statement.

HOW TO WRITE A BASIC PROGRAM THAT USES THE INTRA SUBSYSTEM

The following example shows how to write a BASIC program to use the Intra subsystem for the inquiry application described previously. The example shows two programs (A and B) in the order of execution. The entire programs are not shown; however, listings of the complete programs and the screen format follow the examples. (See *Screen Format and Programs Listings*.) You may want to refer to these listings while you read the example.

OCL Statements

The following procedures and OCL statement are used for the BASIC example:

```
a.   BASICR ITEMABAS,ITEMBAS,30,,BASSESS

b.   // SESSION LOCATION-INTRA,SYMID-1S
```

Procedure and OCL statement for program A. The BASICR procedure includes the SESSION statement.

The remote system location is INTRA (also specified on display 3.0 of the CNFIGICF procedure), and the session identifier (SYMID) is 1S.

```
c.   BASICR ITEMBBAS,ITEMBAS,30
```

Procedure for program B. A SESSION statement is not required for program B.

The procedure for program B will be started by an incoming procedure start request from program A. When you create the procedure for program B, specify PDATA-YES on the copy utility statement for $MAINT or answer *yes* to the prompt for program data in the INCLUDE statement if you use SEU to create the procedure. You must specify PDATA-YES because program B is an SRT program, and program A sends data to program B with the evoke operation. See *Writing Procedures to be Started by Incoming Procedure Start Requests* in Chapter 2 for more information.

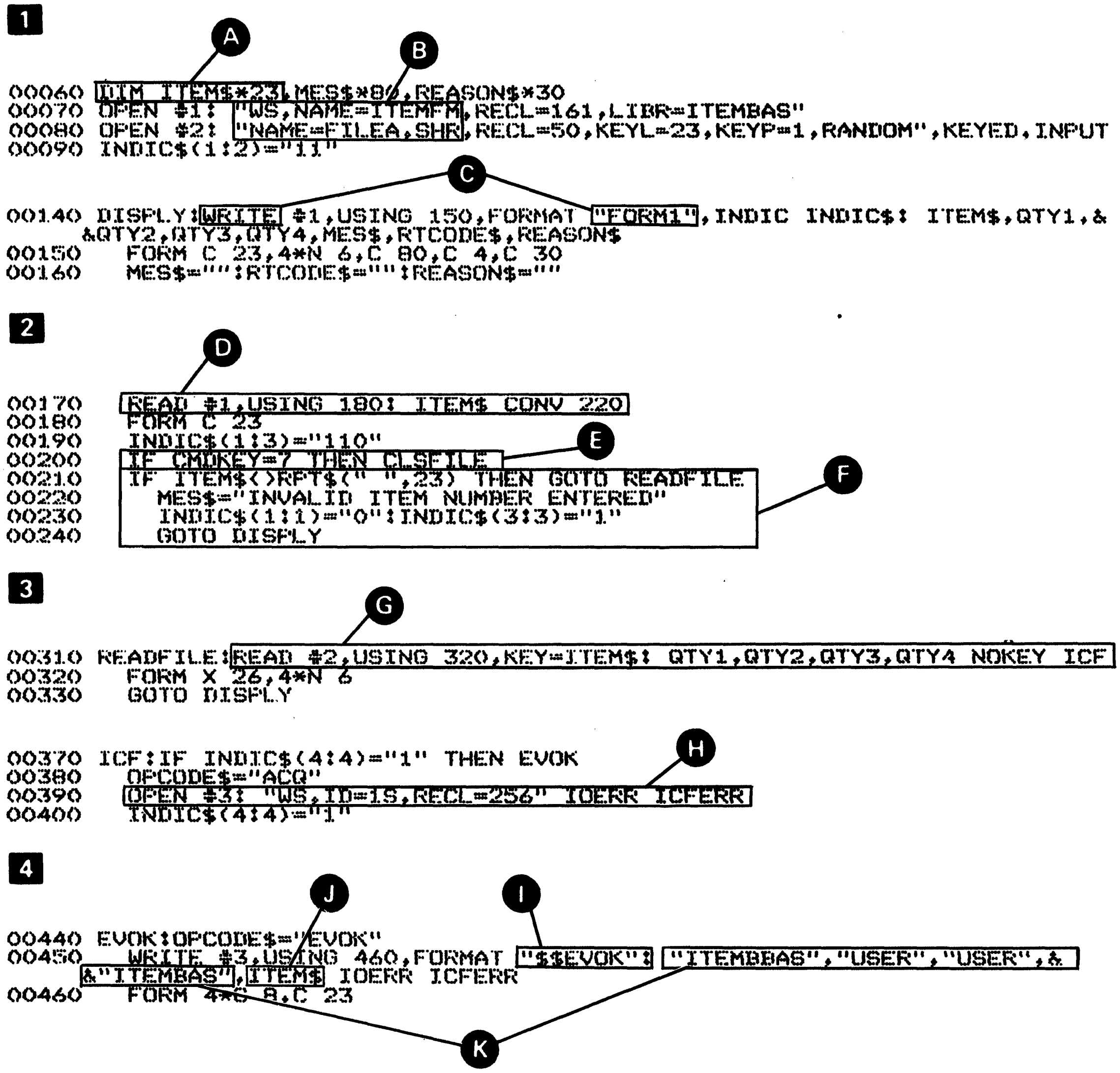
Data Flow and Operations

The following charts and the BASIC program on the facing page show the data flow and the operations that program A and B issue to the Intra subsystem during program execution:

Program A	Intra Subsystem[1]	Program B
1 Set up variables 🅐. Open the work station file and data files 🅑. Display the input prompt using FORM1 🅒.		
2 Read the item number from the display 🅓. If the operator pressed command key 7, go to CLSFILE 🅔. If the item number is zero or blank, set up to display an error message and go back to DISPLY 🅕.		
3 Search the local file for the item 🅖. If the item is there, go to DISPLY and display the item status. If the item is not found in the local file, begin a session. (Open file #3 to begin SSP-ICF session 1S 🅗.)	→ ← Return code	
4 Start program B. Write evoke operation ($$EVOK) 🅘 and send evoke parameters 🅚 with the item number to program B 🅙.	→ ← Return code	

[1] If an error occurs, the program goes to ICFERR to check the return code. This routine is described in the discussion of *How to Check Return Codes with BASIC* later in this example.

Program A


1
A
B
C
2
D
E
F
3
G
H
4
J
I
K

```
00060 DIM ITEM$*23,MES$*80,REASON$*30
00070 OPEN #1: "WS,NAME=ITEMFM,RECL=161,LIBR=ITEMBAS"
00080 OPEN #2: "NAME=FILEA,SHR,RECL=50,KEYL=23,KEYP=1,RANDOM",KEYED,INPUT
00090 INDIC$(1:2)="11"
```

```
00140 DISPLY:WRITE #1,USING 150,FORMAT "FORM1",INDIC INDIC$: ITEM$,QTY1,&
      &QTY2,QTY3,QTY4,MES$,RTCODE$,REASON$
00150    FORM C 23,4*N 6,C 80,C 4,C 30
00160    MES$="":RTCODE$="":REASON$=""
```

```
00170    READ #1,USING 180: ITEM$ CONV 220
00180    FORM C 23
00190    INDIC$(1:3)="110"
00200    IF CMDKEY=7 THEN CLSFILE
00210    IF ITEM$<>RPT$(" ",23) THEN GOTO READFILE
00220      MES$="INVALID ITEM NUMBER ENTERED"
00230      INDIC$(1:1)="0":INDIC$(3:3)="1"
00240      GOTO DISPLY
```

```
00310 READFILE:READ #2,USING 320,KEY=ITEM$: QTY1,QTY2,QTY3,QTY4 NOKEY ICF
00320    FORM X 26,4*N 6
00330    GOTO DISPLY
```

```
00370 ICF:IF INDIC$(4:4)="1" THEN EVOK
00380    OPCODE$="ACQ"
00390    OPEN #3: "WS,ID=1S,RECL=256" IOERR ICFERR
00400    INDIC$(4:4)="1"
```

```
00440 EVOK:OPCODE$="EVOK"
00450    WRITE #3,USING 460,FORMAT "$$EVOK": "ITEMBBAS","USER","USER",&
      &"ITEMBAS",ITEM$ IOERR ICFERR
00460    FORM 4*C 8,C 23
```

Program A	Intra Subsystem[1]	Program B
5	← Return code →	Open files Ⓐ and read the item number sent by program A Ⓑ.
6	Return code →	Search the remote file for the data Ⓒ. If the data is in the remote file, send it to program A Ⓓ. If the data is not found, send the characters *** to program A Ⓔ.
7 Read the data from program B Ⓕ.	→ ← Return code	
8 End the transaction: Send $$SENDET Ⓖ.	→ ← Return code	
9	← Return code →	Read the end of transaction Ⓗ.
10		Close files and end the job Ⓘ.

[1] If an error occurs, the program goes to ICFERR to check the return code. This routine is described in the discussion of *How to Check Return Codes with BASIC* later in this example.

Program B

5

A

```
00060 DIM ITEM$*23
00070 OPEN #1: "NAME=FILEB,SHR,RECL=50,KEYL=23,KEYP=1,RANDOM",KEYED,INPUT
00080 OPEN #2: "WS,RECL=256" IOERR ICFERR
```

B

```
00120 OPCODE$="ACCEPT"
00130 WAITIO IOERR ICFERR
00140 READ #2,USING 150: ITEM$ IOERR ICFERR
00150 FORM C 23
```

6

C

```
00190  READFILE:READ #1,USING 200,KEY=ITEM$: QTY1,QTY2,QTY3,QTY4 NOKEY &
      &ERRORKEY
00200    FORM X 26,4*N 6
```

D

```
00240 SENDDATA:OPCODE$="SEND"
00250    WRITE #2,USING 260,FORMAT "$$SEND": ITEM$,QTY1,QTY2,QTY3,QTY4 &
      $IOERR ICFERR
00260    FORM X 32,C 23,X 145,4*N 6
```

E

```
00400 ERRORKEY:OPCODE$="SEND"
00410   WRITE #2,USING 420,FORMAT "$$SEND": "***",ITEM$ IOERR ICFERR
00420   FORM C 3,X 29,C 192
00430   GOTO 300
```

Program A

7

F

```
00500 OPCODE$="GET"
00510 READ #3,USING 520: DATA$,ITEM$,QTY1,QTY2,QTY3,QTY4 IOERR ICFERR
00520 FORM C 3,X 29,C 23,X 145,4*N 6
```

8

G

```
00560 OPCODE$="SENDET"
00570 WRITE #3,FORMAT "$$SENDET": IOERR ICFERR
```

Program B

9

H

```
00300 OPCODE$="GET"
00310 READ #2: IOERR ICFERR
```

10

I

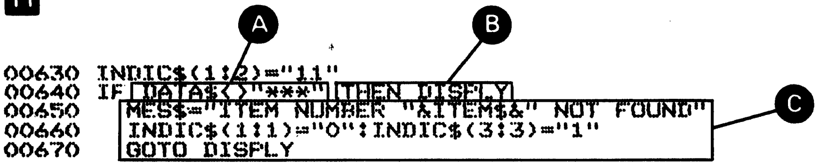
```
00350 CLOSE #1::CLOSE #2:
00360 STOP
```

Program A	Intra Subsystem[1]	Program B
11 Check the record from program B. If the data is not *** Ⓐ, go to DISPLY and display the status from program B Ⓑ. If the data is ***, go to DISPLY and display the message ITEM NUMBER NOT FOUND Ⓒ.		
12 If the operator pressed Command key 7, go to CLSFILE Ⓓ to close files, and end the job. (See step 2 Ⓔ.)		

[1] If an error occurs, the program goes to ICFERR to check the return code. This routine is described in the discussion of *How to Check Return Codes with BASIC* later in this example.

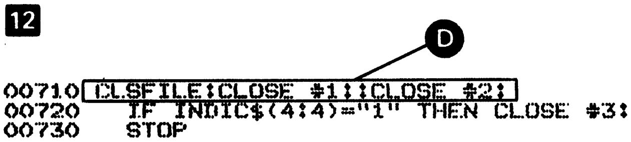
Program A

11



```
00630 INDIC$(1:2)="11"
00640 IF DATA$<>"***" THEN DISPLY
00650    MES$="ITEM NUMBER "&ITEM$&" NOT FOUND"
00660    INDIC$(1:1)="0":INDIC$(3:3)="1"
00670    GOTO DISPLY
```

12



```
00710 CLSFILE:CLOSE #1::CLOSE #2:
00720    IF INDIC$(4:4)="1" THEN CLOSE #3:
00730    STOP
```

How to Check Return Codes with BASIC

The following description and the example on the facing page show how to check SSP-ICF return codes with BASIC.

Program A Return Code Routine

1. The program prints the return code, operation code, and item number as an aid for problem determination Ⓐ.

2. If the return code is greater than or equal to 04xx Ⓑ, the program goes to OUTCHK; otherwise, it goes to SENDEOS to send $$EOS and end the session Ⓗ, and then goes to DISPLY to display the return code.

3. If the return code is greater than 04xx Ⓒ, the program goes to ACQCHK. If not, the return code is 04xx (output exception); data or a system message is ready to be read. The program reads the data or message Ⓓ, goes to subroutine SENDEOS to end the session Ⓗ, and then goes to DISPLY to display the data or message.

4. If the return code is 82xx Ⓔ, the session was not acquired. The program goes to DISPLY to display the message UNABLE TO ACQUIRE Ⓕ. It also displays the return code to show the operator why the session was not acquired.

5. If the return code is greater than 04xx and not 82xx, the program goes to SENDEOS and ends the session, then goes to DISPLY to display the return code Ⓖ.

Program B Return Code Routine

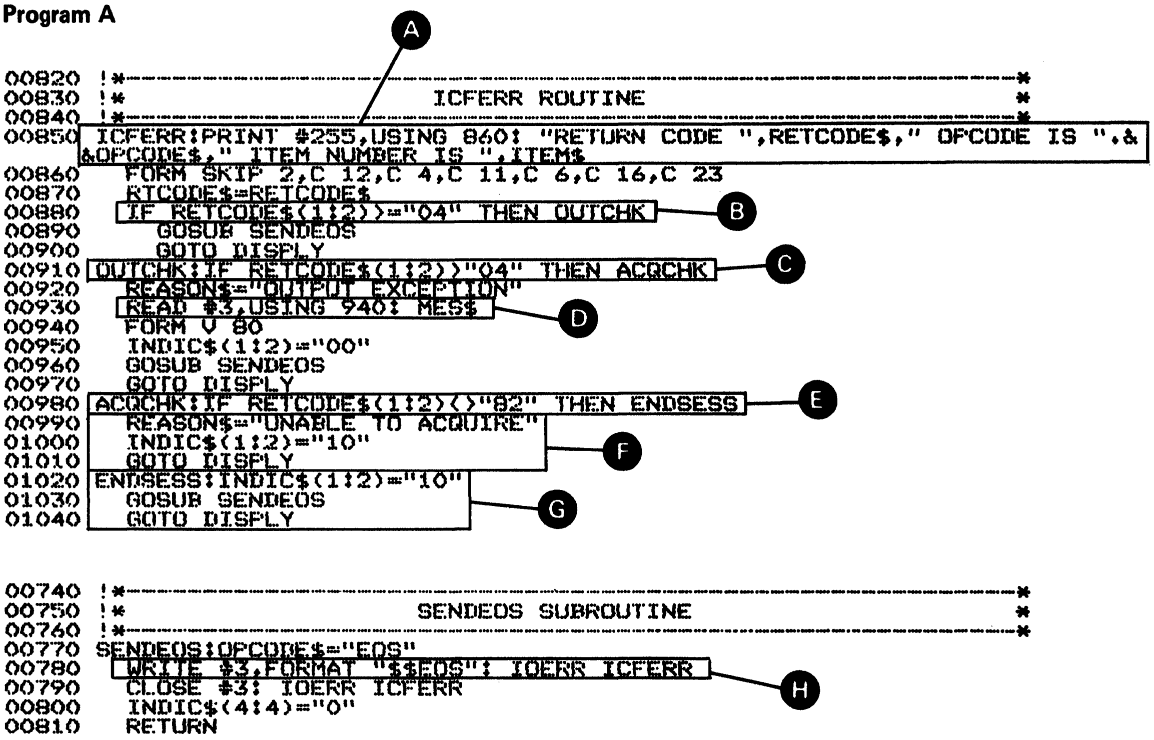
Program B displays and prints any return codes that it receives.

Note About Checking Return Codes

In this example, only the return codes that needed to be checked were checked. All other return codes were displayed or printed. Depending upon your remote system, you may need to check for return codes other than those shown in this example. However, you should display and/or print all return codes as an aid to problem determination. If any codes that are *not* error codes are returned repeatedly, you may want to include these in your return code routine. If an error code is returned, you should, of course, correct the condition causing the error.

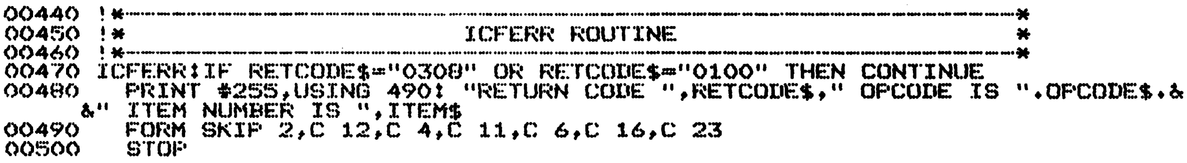
The return codes are described in each subsystem chapter. Only the codes that are valid for that subsystem are described.

Program A



```
00820  !*-----------------------------------------------------------------------*
00830  !*                          ICFERR ROUTINE                               *
00840  !*-----------------------------------------------------------------------*
00850  ICFERR:PRINT #255,USING 860: "RETURN CODE ",RETCODE$," OPCODE IS ".&
&OPCODE$," ITEM NUMBER IS ",ITEM$
00860    FORM SKIP 2,C 12,C 4,C 11,C 6,C 16,C 23
00870    RTCODE$=RETCODE$
00880    IF RETCODE$(1:2)>="04" THEN OUTCHK
00890      GOSUB SENDEOS
00900      GOTO DISPLY
00910  OUTCHK:IF RETCODE$(1:2)>"04" THEN ACQCHK
00920    REASON$="OUTPUT EXCEPTION"
00930    READ #3,USING 940: MES$
00940    FORM V 80
00950    INDIC$(1:2)="00"
00960    GOSUB SENDEOS
00970    GOTO DISPLY
00980  ACQCHK:IF RETCODE$(1:2)<>"82" THEN ENDSESS
00990    REASON$="UNABLE TO ACQUIRE"
01000    INDIC$(1:2)="10"
01010    GOTO DISPLY
01020  ENDSESS:INDIC$(1:2)="10"
01030    GOSUB SENDEOS
01040    GOTO DISPLY

00740  !*-----------------------------------------------------------------------*
00750  !*                        SENDEOS SUBROUTINE                             *
00760  !*-----------------------------------------------------------------------*
00770  SENDEOS:OPCODE$="EOS"
00780    WRITE #3,FORMAT "$$EOS": IOERR ICFERR
00790    CLOSE #3: IOERR ICFERR
00800    INDIC$(4:4)="0"
00810    RETURN
```

Program B



```
00440  !*-----------------------------------------------------------------------*
00450  !*                          ICFERR ROUTINE                               *
00460  !*-----------------------------------------------------------------------*
00470  ICFERR:IF RETCODE$="0308" OR RETCODE$="0100" THEN CONTINUE
00480    PRINT #255,USING 490: "RETURN CODE ",RETCODE$," OPCODE IS ".OPCODE$.&
     &" ITEM NUMBER IS ",ITEM$
00490    FORM SKIP 2,C 12,C 4,C 11,C 6,C 16,C 23
00500    STOP
```

HOW TO WRITE A COBOL PROGRAM THAT USES THE INTRA SUBSYSTEM

The following example shows how to write a COBOL program to use the Intra subsystem for the inquiry application described previously. The example shows two programs (A and B) in the order of execution. The entire programs are not shown; however, listings of the complete programs and the screen format, follow the examples. You may want to refer to these listings while you read the example. (See *Screen Format and Program Listings.*)

OCL Statements

The following OCL statements are used for the COBOL example:

Procedure for Program A

```
// LOAD ITEMAC
// FILE NAME-FILEA
// SESSION LOCATION-INTRA,SYMID-1S  1
// RUN
```

The remote system location is INTRA (also specified on display 3.0 of the CNFIGICF procedure), and the session identifier (SYMID) is 1S 1.

Procedure for Program B

```
// LOAD ITEMBC
// FILE NAME-FILEB
// RUN
```

The procedure for program B will be started by an incoming procedure start request from program A. When you create the procedure for program B, specify PDATA-YES on the copy utility statement for $MAINT or answer *yes* to the prompt for program data in the INCLUDE statement if you use SEU to create the procedure. You must specify PDATA-YES because program B is an SRT program, and program A sends data to program B with the evoke operation. See *Writing Procedures to be Started by Incoming Procedure Start Requests* in Chapter 2 for more information.

This page is intentionally left blank.

1 File Control for Program A and B

- Use the transaction file for ICF sessions and work stations and define the transaction file with the SELECT statement Ⓐ.

- The file status specifications (WS-FS and ICF-FS) Ⓑ are used to test the completion status of transaction file input and output operations.

- Disk data file A (DISK-FILEA) Ⓒ is searched first when the operator enters the item number. Disk file B (DISK-FILEB) Ⓔ is searched next if the item is not found in file A.

- The file PRINT-FILE Ⓓ is used to print file status return codes and error messages.

1 **Program A**

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.                                              A

    SELECT TRANSACTION-FILE
       ASSIGN TO WORKSTATION-ITEMFM-01,
       ORGANIZATION IS TRANSACTION,                          B
       FILE STATUS IS WS-FS, ICF-FS,
       CONTROL-AREA IS WS-CONTROL-AREA.
                                                                C
    SELECT FILEA-FILE ASSIGN TO DISK-FILEA,
       ORGANIZATION IS INDEXED, ACCESS IS RANDOM,
       RECORD KEY IS FILEA-NUMBER.                                  D

    SELECT PRINT-FILE ASSIGN TO PRINTER-PRINTER.
```

Program B

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.

    SELECT TRANSACTION-FILE
       ASSIGN TO WORKSTATION,
       ORGANIZATION IS TRANSACTION,
       FILE STATUS IS WS-FS, ICF-FS,
       CONTROL-AREA IS WS-CONTROL-AREA.                          E

    SELECT FILEB-FILE ASSIGN TO DISK-FILEB,
       ORGANIZATION IS INDEXED, ACCESS IS RANDOM,
       RECORD KEY IS FILEB-NUMBER.

    SELECT PRINT-FILE ASSIGN TO PRINTER-PRINTER.
```

2 Data Division (File Section) for Program A and B

The data division defines the records for the SSP-ICF files Ⓐ, disk files A and B Ⓑ, and the print files Ⓒ.

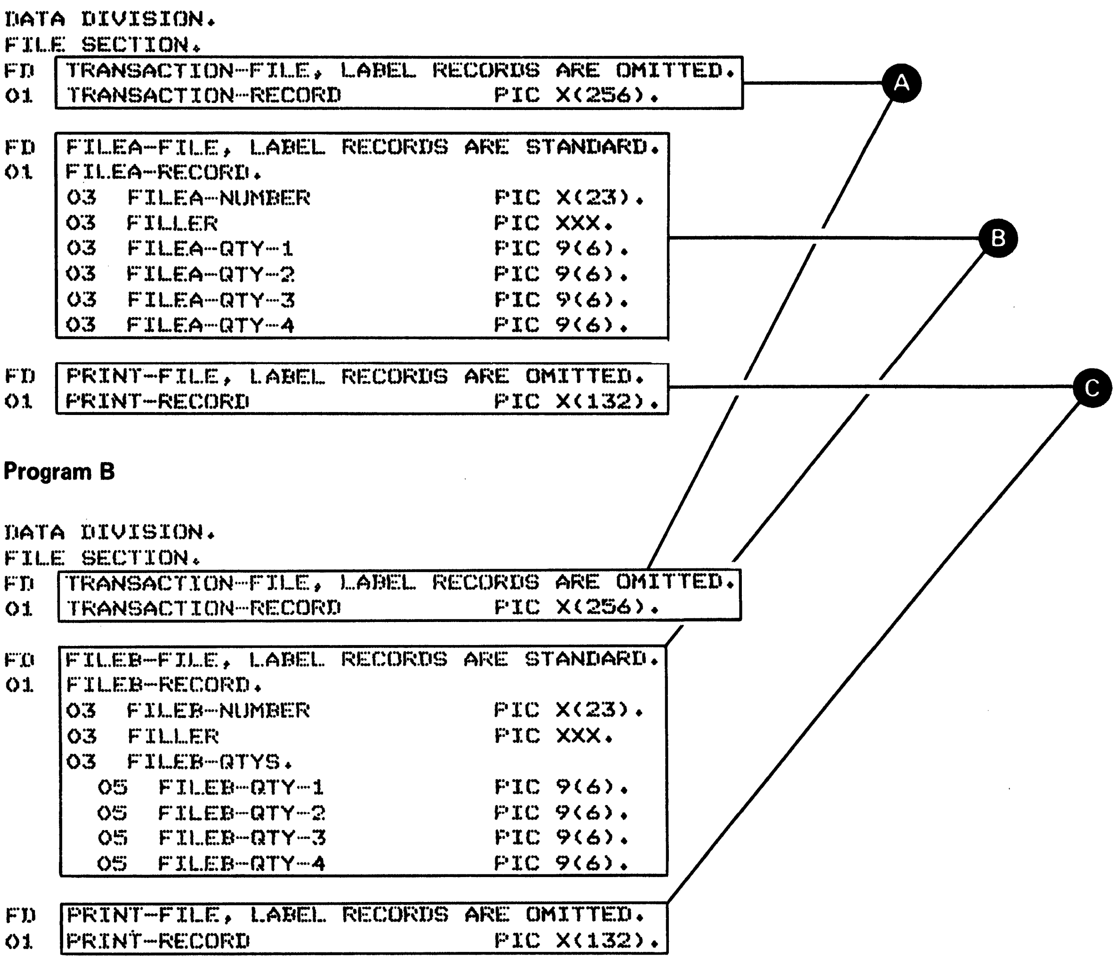
3 Working Storage for Program A

- Your program must know whether input and output is from or to a work station or an SSP-ICF session and, if there is more than one active session, which session is currently in use. (In this example, there is only one session.) The identifier for the SSP-ICF session is 1S Ⓓ. The value in WS-ID Ⓕ is set to the ID of the last session or work station accessed.

- CMD-KEY 7 Ⓔ is used in the program to determine whether the operator pressed command key 7 to request end of job processing.

- Your program must check return codes after each input/output operation. In this example, the return code is returned in ICF-FS Ⓖ and checked by the return code routine, which is described later in this example under *How to Check Return Codes with COBOL.*

2 Program A

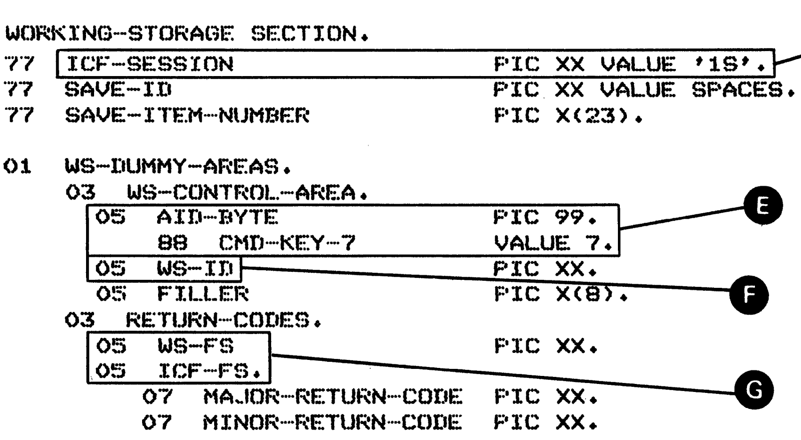


```
DATA DIVISION.
FILE SECTION.
FD  TRANSACTION-FILE, LABEL RECORDS ARE OMITTED.
01  TRANSACTION-RECORD                  PIC X(256).

FD  FILEA-FILE, LABEL RECORDS ARE STANDARD.
01  FILEA-RECORD.
    03  FILEA-NUMBER                    PIC X(23).
    03  FILLER                          PIC XXX.
    03  FILEA-QTY-1                     PIC 9(6).
    03  FILEA-QTY-2                     PIC 9(6).
    03  FILEA-QTY-3                     PIC 9(6).
    03  FILEA-QTY-4                     PIC 9(6).

FD  PRINT-FILE, LABEL RECORDS ARE OMITTED.
01  PRINT-RECORD                        PIC X(132).
```

Program B

```
DATA DIVISION.
FILE SECTION.
FD  TRANSACTION-FILE, LABEL RECORDS ARE OMITTED.
01  TRANSACTION-RECORD                  PIC X(256).

FD  FILEB-FILE, LABEL RECORDS ARE STANDARD.
01  FILEB-RECORD.
    03  FILEB-NUMBER                    PIC X(23).
    03  FILLER                          PIC XXX.
    03  FILEB-QTYS.
      05  FILEB-QTY-1                   PIC 9(6).
      05  FILEB-QTY-2                   PIC 9(6).
      05  FILEB-QTY-3                   PIC 9(6).
      05  FILEB-QTY-4                   PIC 9(6).

FD  PRINT-FILE, LABEL RECORDS ARE OMITTED.
01  PRINT-RECORD                        PIC X(132).
```

3 Program A



```
WORKING-STORAGE SECTION.
77  ICF-SESSION                         PIC XX VALUE '1S'.
77  SAVE-ID                             PIC XX VALUE SPACES.
77  SAVE-ITEM-NUMBER                    PIC X(23).

01  WS-DUMMY-AREAS.
    03  WS-CONTROL-AREA.
      05  AID-BYTE                      PIC 99.
          88  CMD-KEY-7                 VALUE 7.
      05  WS-ID                         PIC XX.
      05  FILLER                        PIC X(8).
    03  RETURN-CODES.
      05  WS-FS                         PIC XX.
      05  ICF-FS.
          07  MAJOR-RETURN-CODE         PIC XX.
          07  MINOR-RETURN-CODE         PIC XX.
```

4 Working Storage for Program B

Program B does not require a work station; therefore, there is no work station identifier. The session identifier (ICF-SESSION) 🅐 is set by SSP-ICF because program B is an evoked program.

5 Display Information for Program A

- The display screen indicators 🅑 are used as follows:
  - If indicator I01 is on (1), the message line is not displayed.
  - If indicator I02 is on, the return code line is not displayed.
  - If indicator I03 is on, the item number is displayed in reverse image.

- The display screen messages 🅒 are explained under *Data Flow and Operations* later in this example.

4 **Program B**

```
WORKING-STORAGE SECTION.

01  ICF-ITEM-NUMBER-IN          PIC X(23) VALUE SPACES.

01  WS-DUMMY-AREAS.
    03  WS-CONTROL-AREA.
      05  AID-BYTE              PIC 99.
      05  ICF-SESSION           PIC XX.                 A
      05  FILLER                PIC X(8).
    03  RETURN-CODES.
      05  WS-FS                 PIC XX.
      05  ICF-FS.
          07  MAJOR-RETURN-CODE PIC XX.
          07  MINOR-RETURN-CODE PIC XX.
```

5 **Program A**

```
01  SCREEN-INDICATORS.
    03  I01                     PIC 1 VALUE ZERO, INDICATOR 01.
    03  I02                     PIC 1 VALUE ZERO, INDICATOR 02.   B
    03  I03                     PIC 1 VALUE ZERO, INDICATOR 03.

01  SCREEN-MESSAGES.
    03  INVALID-ITEM-MSG.
      05  FILLER                PIC X(14) VALUE SPACES.
      05  INVALID-MSG           PIC X(70)
                        VALUE 'INVALID ITEM NUMBER ENTERED.'.    C

    03  ITEM-NOT-FOUND-MSG.
      05  FILLER                PIC X(14) VALUE SPACES.
      05  FILLER                PIC X(12) VALUE 'ITEM NUMBER '.
      05  ITEM-NOT-FOUND        PIC X(23) VALUE SPACES.
      05  FILLER                PIC X(35) VALUE ' NOT FOUND.'.
```

6 Data Records for Program A

- You must define the evoke record that is used to start the remote program or procedure (program B in this example). The record includes the procedure or program name Ⓐ, the password and user ID Ⓑ that the remote system requires before the program or procedure is started, and the name of the library Ⓒ that contains the program or procedure to be started.

- In this example, data (the item number) is sent with the evoke parameters Ⓓ.

- ICF-RECORD-IN describes the data fields for the record from program B. ICF-RECORD-CHECK Ⓔ and ICF-RECORD-OK Ⓕ describe the fields of ICF-RECORD-IN.

- SCREEN-RECORD Ⓖ is the data and constants to be displayed on the screen.

- PRINT-CODES Ⓗ is used when printing return codes and operation codes.

7 Data Records for Program B

- ICF-RECORD-OUT Ⓘ is the item status that program B sends to program A if the item number is found in file B.

- ERROR-RECORD-OUT Ⓙ is the *** message that program B sends to program A if the item is not found in file B.

- PRINT-CODES Ⓚ is used when printing return codes and operation codes.

6 Program A

```
01  EVOKE-RECORD.
    03  PROCEDURE-NAME              PIC X(8) VALUE 'ITEMBCOB'.     A
    03  PASSWORD                    PIC X(8) VALUE 'USER'.
    03  USER-ID                     PIC X(8) VALUE 'USER'.         B
    03  LIBRARY-NAME                PIC X(8) VALUE 'ICFLIBR'.      C
    03  FILLER                      PIC X(20).
    03  DATA-LENGTH                 PIC XXXX VALUE '0023'.
    03  ICF-ITEM-NUMBER-OUT         PIC X(23).                     D

01  ICF-RECORD-IN.
    03  ICF-RECORD-CHECK.
      05  FIRST-3-CHARACTERS        PIC X(3).                      E
      05  REST-OF-DATA              PIC X(253).

    03  ICF-RECORD-OK REDEFINES ICF-RECORD-CHECK.
      05  FILLER                    PIC X(32).
      05  ICF-ITEM-NUMBER-IN        PIC X(23).
      05  FILLER                    PIC X(145).                    F
      05  ICF-QTY-1                 PIC 9(6).
      05  ICF-QTY-2                 PIC 9(6).
      05  ICF-QTY-3                 PIC 9(6).
      05  ICF-QTY-4                 PIC 9(6).
      05  FILLER                    PIC X(32).

01  SCREEN-RECORD.
    03  ITEM-NUMBER                 PIC X(23).
    03  QTY-1                       PIC 9(6).                      G
    03  QTY-2                       PIC 9(6).
    03  QTY-3                       PIC 9(6).
    03  QTY-4                       PIC 9(6).
    03  MSG                         PIC X(80).
    03  RETURN-CODE                 PIC X(4).
    03  REASON-WHY                  PIC X(30).

01  PRINT-CODES.                                                   H
    03  FILLER                      PIC X(14) VALUE 'RETURN CODE '.
    03  PRINT-RETURN-CODE           PIC XXXX.
    03  FILLER                      PIC X(11) VALUE ' OPCODE IS '.
    03  OPCODE                      PIC X(6).
    03  FILLER                      PIC X(11) VALUE ' DATA SENT '.
    03  PRINT-ITEM-NUMBER           PIC X(23).
```

7 Program B

```
01  ICF-RECORD-OUT.
    03  DATA-LENGTH                 PIC X(4) VALUE '0224'.
    03  FILLER                      PIC X(32).
    03  ICF-ITEM-NUMBER-OUT         PIC X(23).                     I
    03  FILLER                      PIC X(145).
    03  ICF-QTYS.
      05  ICF-QTY-1                 PIC 9(6).
      05  ICF-QTY-2                 PIC 9(6).
      05  ICF-QTY-3                 PIC 9(6).
      05  ICF-QTY-4                 PIC 9(6).

01  ERROR-RECORD-OUT.                                              J
    03  DATA-LENGTH                 PIC XXXX VALUE '0003'.
    03  FILLER                      PIC XXX VALUE '***'.

01  PRINT-CODES.                                                   K
    03  FILLER                      PIC X(14) VALUE 'RETURN CODE '.
    03  PRINT-RETURN-CODE           PIC XXXX.
    03  FILLER                      PIC X(11)  VALUE ' OPCODE IS '.
    03  OPCODE                      PIC X(6).
```

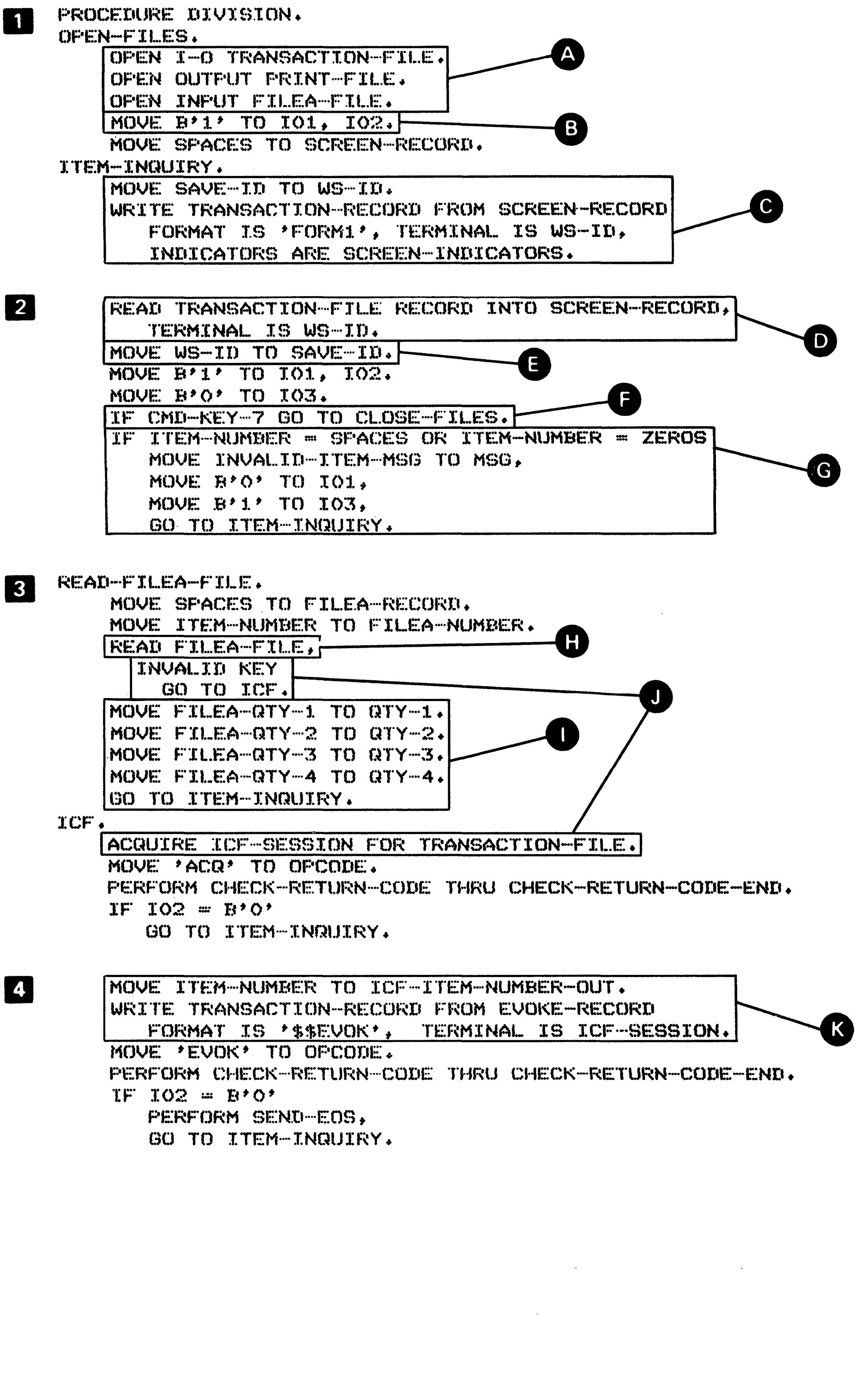
Data Flow and Operations

The following charts and the program on the facing pages show the data flow and the operations that program A and B issue to the Intra subsystem during program execution:

Program A	Intra Subsystem[1]	Program B
1 Open files Ⓐ and set on indicators 01 and 02 Ⓑ. Restore WS-ID and display the input prompt using screen format FORM1 Ⓒ.		
2 Read the item number from the display Ⓓ. Save the WS-ID Ⓔ. If the operator pressed command key 7, go to CLOSE-FILES Ⓕ. If the item number is zero or blank, set up to display an error message Ⓖ (INVALID ITEM NUMBER ENTERED) and go back to ITEM-INQUIRY.		
3 Search the local file for the item Ⓗ. If the item is there, move data to SCREEN-RECORD and go to ITEM-INQUIRY to display the data Ⓘ. If the item is not found in the local file, begin a session → (ACQUIRE) and check the return code Ⓙ. ←	→ Return code	
4 Start program B. → Write evoke operation ($$EVOK) Ⓚ; send evoke parameters with the item number to program B. ← Check the return code.	→ Return code	

[1] Return codes are checked by the return code routine, which is described in the discussion of *How to Check Return Codes with COBOL* later in this example.

Program A



```
1   PROCEDURE DIVISION.
    OPEN-FILES.
        OPEN I-O TRANSACTION-FILE.
        OPEN OUTPUT PRINT-FILE.
        OPEN INPUT FILEA-FILE.
        MOVE B'1' TO IO1, IO2.
        MOVE SPACES TO SCREEN-RECORD.
    ITEM-INQUIRY.
        MOVE SAVE-ID TO WS-ID.
        WRITE TRANSACTION-RECORD FROM SCREEN-RECORD
           FORMAT IS 'FORM1', TERMINAL IS WS-ID,
           INDICATORS ARE SCREEN-INDICATORS.

2       READ TRANSACTION-FILE RECORD INTO SCREEN-RECORD,
           TERMINAL IS WS-ID.
        MOVE WS-ID TO SAVE-ID.
        MOVE B'1' TO IO1, IO2.
        MOVE B'0' TO IO3.
        IF CMD-KEY-7 GO TO CLOSE-FILES.
        IF ITEM-NUMBER = SPACES OR ITEM-NUMBER = ZEROS
           MOVE INVALID-ITEM-MSG TO MSG,
           MOVE B'0' TO IO1,
           MOVE B'1' TO IO3,
           GO TO ITEM-INQUIRY.

3   READ-FILEA-FILE.
        MOVE SPACES TO FILEA-RECORD.
        MOVE ITEM-NUMBER TO FILEA-NUMBER.
        READ FILEA-FILE,
          INVALID KEY
            GO TO ICF.
        MOVE FILEA-QTY-1 TO QTY-1.
        MOVE FILEA-QTY-2 TO QTY-2.
        MOVE FILEA-QTY-3 TO QTY-3.
        MOVE FILEA-QTY-4 TO QTY-4.
        GO TO ITEM-INQUIRY.
    ICF.
        ACQUIRE ICF-SESSION FOR TRANSACTION-FILE.
        MOVE 'ACQ' TO OPCODE.
        PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
        IF IO2 = B'0'
           GO TO ITEM-INQUIRY.

4       MOVE ITEM-NUMBER TO ICF-ITEM-NUMBER-OUT.
        WRITE TRANSACTION-RECORD FROM EVOKE-RECORD
           FORMAT IS '$$EVOK',  TERMINAL IS ICF-SESSION.
        MOVE 'EVOK' TO OPCODE.
        PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
        IF IO2 = B'0'
           PERFORM SEND-EOS,
           GO TO ITEM-INQUIRY.
```

Program A	Intra Subsystem[1]	Program B
5	← Return code →	Open files Ⓐ and read the item number sent by program A Ⓑ. Print the return code and operation code Ⓒ.
6	← Return code →	Search the remote file for the data Ⓓ. If the data is in the remote file, send it to program A Ⓕ and Ⓖ. If the data is not found, send the characters *** to program A Ⓔ and Ⓖ. Print the return code and operation code Ⓗ.
7 Read the data from program B Ⓘ. → ← Return code Check the return code.		
8 End the transaction: Send $$SENDET Ⓙ. → ← Return code Check the return code.		
9	← Return code →	Read the end of transaction Ⓚ. Print the return code and operation code Ⓛ.
10		Go to CLOSE-FILES and end the job Ⓜ.

[1] Return codes are checked by the return code routine, which is described in the discussion of *How to Check Return Codes with COBOL* later in this example.

Program B

```
5  PROCEDURE DIVISION.
   OPEN-FILES.
       OPEN I-O TRANSACTION-FILE.                              A
       OPEN OUTPUT PRINT-FILE.
       OPEN INPUT FILEB-FILE.
       READ TRANSACTION-FILE RECORD INTO ICF-ITEM-NUMBER-IN.   B
       MOVE 'ACCEPT' TO OPCODE.
       PERFORM WRITE-CODES THRU WRITE-CODES-END.               C

6  READ-FILEB-FILE.
       MOVE SPACES TO FILEB-RECORD.
       MOVE ICF-ITEM-NUMBER-IN TO FILEB-NUMBER.
       READ FILEB-FILE,                                        D
         INVALID KEY                                           E
           MOVE ERROR-RECORD-OUT TO TRANSACTION-RECORD,
           GO TO SEND-DATA.
       MOVE FILEB-NUMBER TO ICF-ITEM-NUMBER-OUT.
       MOVE FILEB-QTYS TO ICF-QTYS.                            F
       MOVE ICF-RECORD-OUT TO TRANSACTION-RECORD.
   SEND-DATA.
       WRITE TRANSACTION-RECORD FORMAT IS '$$SEND',            G
          TERMINAL IS ICF-SESSION.
       MOVE 'SEND' TO OPCODE.
       PERFORM WRITE-CODES THRU WRITE-CODES-END.               H
```

Program A

```
7      MOVE SPACES TO ICF-RECORD-IN.
       READ TRANSACTION-FILE RECORD INTO ICF-RECORD-IN,        I
         TERMINAL IS ICF-SESSION.
       MOVE 'GET' TO OPCODE.
       PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
       IF IO2 = B'O'
          PERFORM SEND-EOS,
          GO TO ITEM-INQUIRY.

8      MOVE SPACES TO TRANSACTION-RECORD.
       WRITE TRANSACTION-RECORD                                J
          FORMAT IS '$$SENDET', TERMINAL IS ICF-SESSION.
       MOVE 'SENDET' TO OPCODE.
       PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
       IF IO2 = B'O'
          PERFORM SEND-EOS.
```

Program B

```
                                                               K
9      READ TRANSACTION-FILE RECORD TERMINAL IS ICF-SESSION.
       MOVE 'GET' TO OPCODE.
       PERFORM WRITE-CODES THRU WRITE-CODES-END.               L

10 CLOSE-FILES.
       CLOSE TRANSACTION-FILE, FILEB-FILE, PRINT-FILE.
       STOP RUN.                                               M
```

Program A	Intra Subsystem[1]	Program B
11 Release the session Ⓐ. → Check the return code.	→ ← Return code	
12 Check the record from program B. If the record is *** Ⓑ, set up to display the message ITEM NOT FOUND Ⓒ. If the record is not ***, set up to display the data Ⓓ. Go to ITEM-INQUIRY to display the data or the message Ⓔ.		
13 If the operator pressed command key 7, go to CLOSE-FILES and end the job. (See step 2 Ⓕ.)		

[1] Return codes are checked by the return code routine, which is described in the discussion of *How to Check Return Codes with COBOL* later in this example.

Program A

```
11        IF IO2 = B'1'
             DROP ICF-SESSION FROM TRANSACTION-FILE,                A
             MOVE 'DROP' TO OPCODE,
             PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
             IF IO2 = B'0'
                PERFORM SEND-EOS.

12        MOVE B'1' TO IO1, IO2.
          MOVE ITEM-NUMBER TO SAVE-ITEM-NUMBER.
          MOVE SPACES TO SCREEN-RECORD.
          IF FIRST-3-CHARACTERS = '***'                         B
             MOVE SAVE-ITEM-NUMBER TO ITEM-NOT-FOUND, ITEM-NUMBER,
             MOVE ITEM-NOT-FOUND-MSG TO MSG,                              C
             MOVE B'0' TO IO1, MOVE B'1' TO IO3,
          ELSE
             MOVE ICF-ITEM-NUMBER-IN TO ITEM-NUMBER,
             MOVE ICF-QTY-1 TO QTY-1,
             MOVE ICF-QTY-2 TO QTY-2,                                D
             MOVE ICF-QTY-3 TO QTY-3,
             MOVE ICF-QTY-4 TO QTY-4.
          GO TO ITEM-INQUIRY.                   E

13   CLOSE-FILES.
          CLOSE TRANSACTION-FILE, FILEA-FILE, PRINT-FILE.
          STOP RUN.
```

How to Check Return Codes with COBOL

The following description and the example on the facing page show how to check SSP-ICF return codes with COBOL.

Program A Return Code Routine

1. If the return code is less than 04xx Ⓐ, the return code does not indicate an error and the routine is ended.

2. If the return code is greater than or equal to 04xx, the return code is printed as an aid to problem determination Ⓑ.

3. If the return code is 04xx (output exception) Ⓒ, data or a system message is ready to be read. The program reads the data or message Ⓓ, and sets up fields and indicators Ⓔ so that the data or message and the return code are displayed on the work station screen when the program goes to ITEM-INQUIRY.

4. If the return code is 82xx Ⓕ, the session was not acquired. Move the message UNABLE TO ACQUIRE and set up indicators Ⓖ so that the message and return code are displayed to show the operator why the session was not acquired.

5. If the return code is greater than 04xx and not 82xx, the program sets up indicators Ⓗ to send an end of session ($$EOS) operation Ⓘ and moves the return code to RETURN-CODE to be displayed as an aid to problem determination.

Program B Return Code Routine

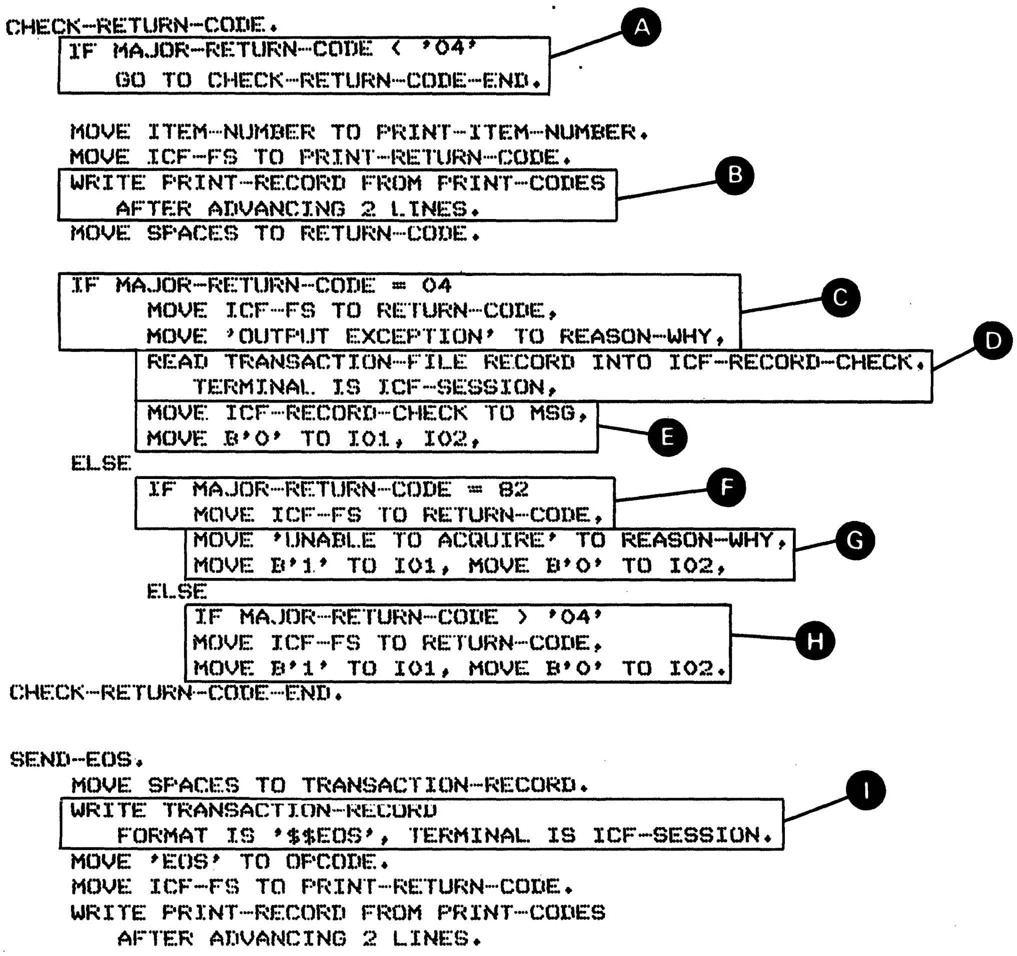
Program B displays and prints any return codes that it receives Ⓙ.

Note About Checking Return Codes

In this example, only the return codes that needed to be checked were checked. All other return codes were displayed or printed. Depending upon your remote system, you may need to check for return codes other than those shown in this example. However, you should display and/or print all return codes as an aid to problem determination. If any codes that are *not* error codes are returned repeatedly, you may want to include these in your return code routine. If an error code is returned, you should, of course, correct the condition causing the error.

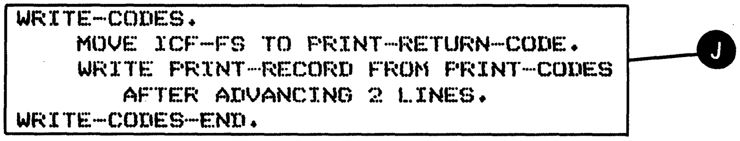
The return codes are described in each subsystem chapter. Only the codes that are valid for that subsystem are described.

Program A



```
CHECK-RETURN-CODE.
    IF MAJOR-RETURN-CODE < '04'
       GO TO CHECK-RETURN-CODE-END.

    MOVE ITEM-NUMBER TO PRINT-ITEM-NUMBER.
    MOVE ICF-FS TO PRINT-RETURN-CODE.
    WRITE PRINT-RECORD FROM PRINT-CODES
       AFTER ADVANCING 2 LINES.
    MOVE SPACES TO RETURN-CODE.

    IF MAJOR-RETURN-CODE = 04
          MOVE ICF-FS TO RETURN-CODE,
          MOVE 'OUTPUT EXCEPTION' TO REASON-WHY,
          READ TRANSACTION-FILE RECORD INTO ICF-RECORD-CHECK.
             TERMINAL IS ICF-SESSION,
          MOVE ICF-RECORD-CHECK TO MSG,
          MOVE B'0' TO I01, I02,
    ELSE
          IF MAJOR-RETURN-CODE = 82
          MOVE ICF-FS TO RETURN-CODE,
             MOVE 'UNABLE TO ACQUIRE' TO REASON-WHY,
             MOVE B'1' TO I01, MOVE B'0' TO I02,
          ELSE
             IF MAJOR-RETURN-CODE > '04'
             MOVE ICF-FS TO RETURN-CODE,
             MOVE B'1' TO I01, MOVE B'0' TO I02.
CHECK-RETURN-CODE-END.

SEND-EOS.
    MOVE SPACES TO TRANSACTION-RECORD.
    WRITE TRANSACTION-RECORD
       FORMAT IS '$$EOS', TERMINAL IS ICF-SESSION.
    MOVE 'EOS' TO OPCODE.
    MOVE ICF-FS TO PRINT-RETURN-CODE.
    WRITE PRINT-RECORD FROM PRINT-CODES
       AFTER ADVANCING 2 LINES.
```

Program B



```
WRITE-CODES.
     MOVE ICF-FS TO PRINT-RETURN-CODE.
     WRITE PRINT-RECORD FROM PRINT-CODES
        AFTER ADVANCING 2 LINES.
WRITE-CODES-END.
```

HOW TO WRITE AN RPG II PROGRAM THAT USES THE INTRA SUBSYSTEM

The following example shows how to write an RPG II program to use the Intra subsystem for the inquiry application described previously. The example shows two programs (A and B) in the order of execution. The entire programs are not shown; however, listings of the complete programs follow the examples. You may want to refer to these listings while you read the example. (See *Screen Format and Program Listings.*)

OCL Statements

The following OCL statements are used for the COBOL example:

Procedure for Program A

```
// LOAD ITEMAR
// FILE NAME-FILEA
// SESSION LOCATION-?1'INTRA'?,SYMID-1S   1
// RUN
```

The remote system location is INTRA (also specified on display 3.0 of the CNFIGICF procedure), and the session identifier (SYMID) is 1S 1.

Procedure for Program B

```
// LOAD ITEMBR
// FILE NAME-FILEB
// RUN
```

The procedure for program B will be started by an incoming procedure start request from program A. When you create the procedure for program B, specify PDATA-YES on the copy utility statement for $MAINT or answer *yes* to the prompt for program data in the INCLUDE statement if you use SEU to create the procedure. You must specify PDATA-YES because program B is an SRT program, and program A sends data to program B with the evoke operation. See *Writing Procedures to be Started by Incoming Procedure Start Requests* in Chapter 2 for more information.

This page is intentionally left blank.

File and Input Specifications

- The file WSFILE is a work station file Ⓐ. On an input operation, data from WSFILE comes from the display screen or from an SSP-ICF session. Output to WSFILE goes to the display screen or to an SSP-ICF session.

- The number of devices and SSP-ICF sessions used by this program Ⓑ.

- ITEMFM is the display screen format used to display prompts and output Ⓒ.

- The ID for both the display station and SSP-ICF session Ⓓ.

- The INFDS and INFSR (Ⓔ in program A; Ⓘ in program B) are used to check return codes.

- The print file (Ⓕ in program A; Ⓙ in program B) is used to print return codes so that you have a record of all return codes returned to the program. You can then use this record to help solve problems if they occur and to decide whether you want to include any of the return codes in the return code checking routine. See *Note About Checking Return Codes* later in this example.

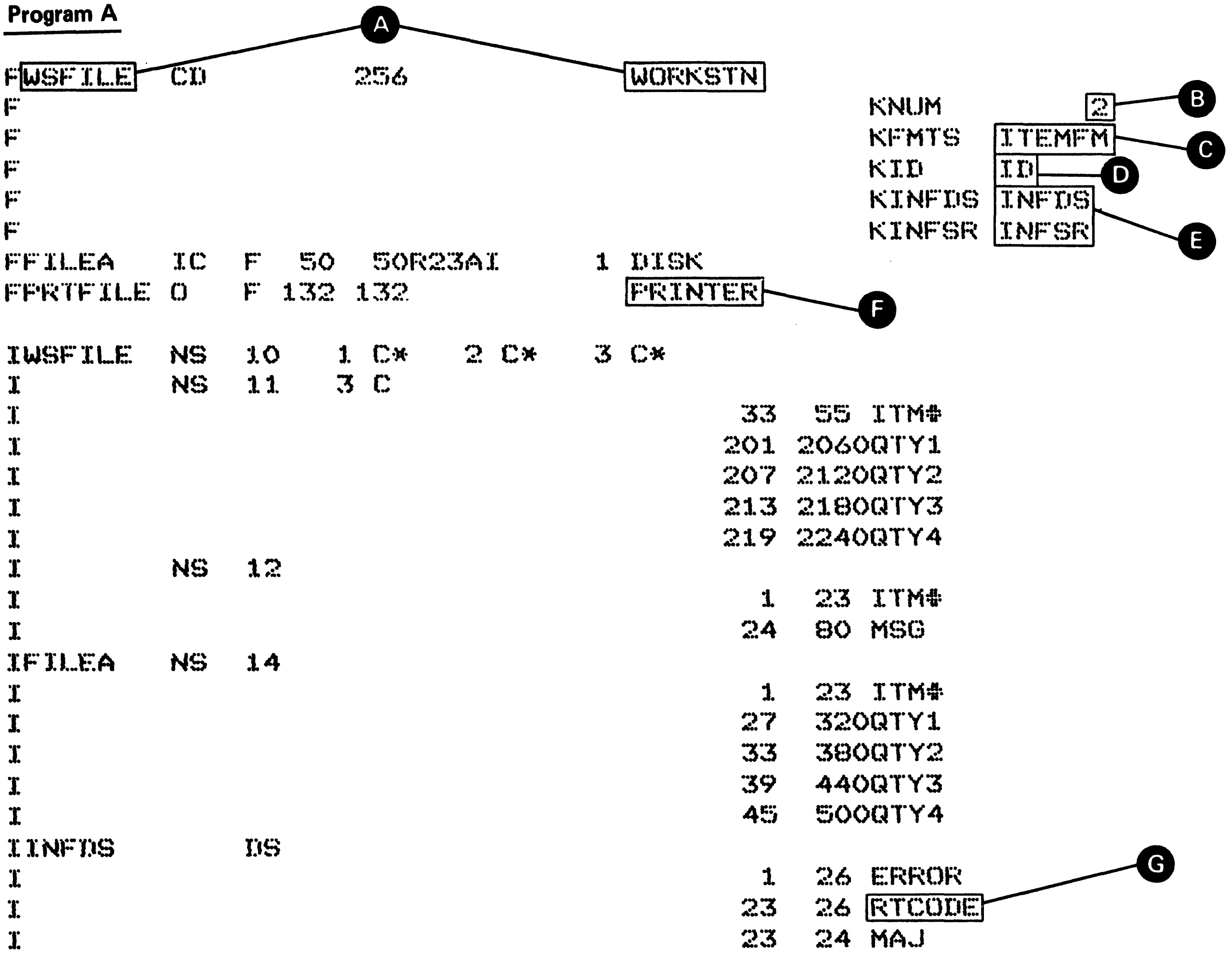
- No screen formats are used by program B Ⓗ.

- The return code field in INFDS is RTCODE Ⓖ.

- Error Ⓚ contains the following fields:

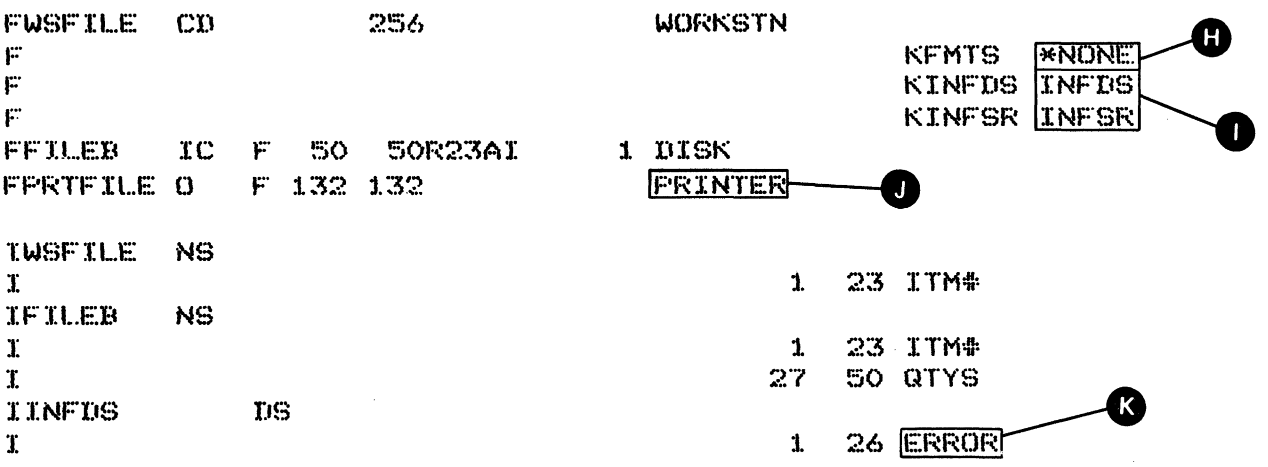
1-5	STATUS
6-10	Operation
11-18	Format
23-24	Major Return Code
25-26	Minor Return Code

Program A



```
FWSFILE  CD        256            WORKSTN
F                                              KNUM        2
F                                              KFMTS  ITEMFM
F                                              KID    ID
F                                              KINFDS INFDS
F                                              KINFSR INFSR
FFILEA   IC  F  50  50R23AI     1 DISK
FPRTFILE O   F 132 132            PRINTER

IWSFILE  NS  10   1 C*   2 C*   3 C*
I        NS  11   3 C
I                                       33  55 ITM#
I                                      201 2060QTY1
I                                      207 2120QTY2
I                                      213 2180QTY3
I                                      219 2240QTY4
I        NS  12
I                                        1  23 ITM#
I                                       24  80 MSG
IFILEA   NS  14
I                                        1  23 ITM#
I                                       27  320QTY1
I                                       33  380QTY2
I                                       39  440QTY3
I                                       45  500QTY4
IINFDS       DS
I                                        1  26 ERROR
I                                       23  26 RTCODE
I                                       23  24 MAJ
```

Program B



```
FWSFILE  CD        256            WORKSTN
F                                              KFMTS  *NONE
F                                              KINFDS INFDS
F                                              KINFSR INFSR
FFILEB   IC  F  50  50R23AI     1 DISK
FPRTFILE O   F 132 132            PRINTER

IWSFILE  NS
I                                        1  23 ITM#
IFILEB   NS
I                                        1  23 ITM#
I                                       27  50 QTYS
IINFDS       DS
I                                        1  26 ERROR
```

Program Indicators

The following indicators are used in the RPG II programming examples. You may want to refer to this list while you read the example. When the indicators are on they mean the following:

Program A Indicators

- *01:* The message line is not displayed.
- *02:* The return code message is not displayed.
- *03:* The item number is displayed in reverse image.
- *04:* The return code is 04xx.
- *05:* The $$EVOK operation is issued.
- *06:* The $$SENDET operation is issued.
- *07:* The $$EOS operation is issued.
- *08:* This is not the first program cycle.
- *09:* The screen format (FORM1 of ITEMFM) is displayed.
- *10:* The data received from program B indicates that the item number was not found in the remote file.
- *11:* The data received from program B is the stock status.
- *12:* Input is from the work station display screen.
- *13:* The item is not in the local file.
- *14:* Input is from the local file.
- *82:* The return code is 82xx.

Program B Indicators

- *03:* The $$SEND operation is issued.
- *98:* The end of transaction was received from program A.
- *99:* The item number was not found in the remote file (file B).

This page is intentionally left blank.

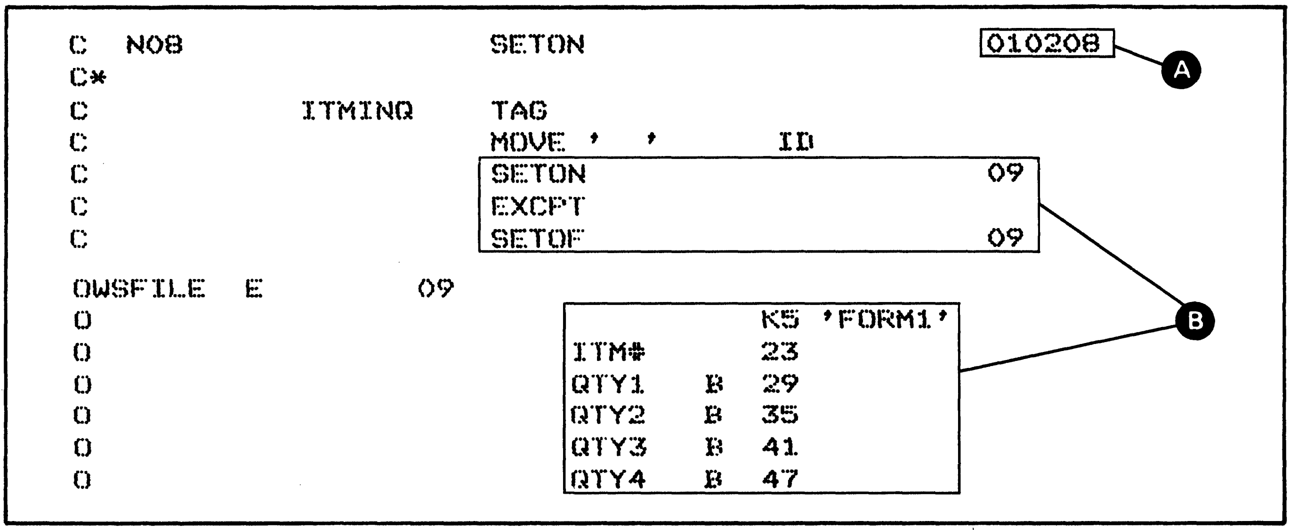
Data Flow and Operations

Program A	Intra Subsystem[1]	Program B
1 Set on indicator 08 Ⓐ to *indicate not first cycle.* Display the input prompt using FORM1 Ⓑ.		
2 Read the item number Ⓒ. If the operator pressed command key 7, go to EOJ Ⓓ. If the item number is zero or blank, set up to display an error message Ⓔ and go back to ITMINQ.		
3 Search the local file for the item Ⓕ. If the item is there, go to ITMINQ and display the data; if not go to step 4.		
4 Begin a session (ACQ) →	→ ← Return code	
5 Start program B → Send evoke parameters ($$EVOK operation) Ⓖ with data (the item number) Ⓗ to program B.	→ ← Return code	

[1] Return codes are checked by the INFSR subroutine, which is described later in this example in the discussion of *How to Check Return Codes with RPG II.*

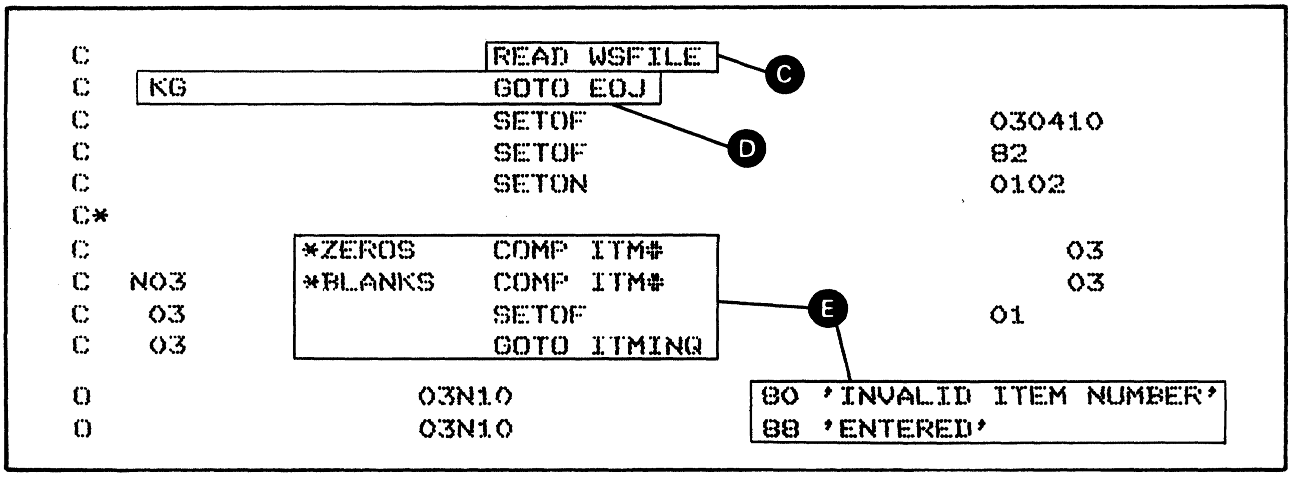
Program A

1



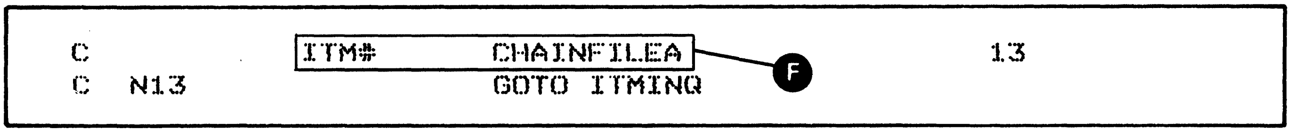
```
C   N08                     SETON                          010208
C*
C                ITMINQ     TAG
C                           MOVE '  '        ID
C                           SETON                          09
C                           EXCPT
C                           SETOF                          09
OWSFILE  E          09
O                                     K5  'FORM1'
O                           ITM#      23
O                           QTY1   B  29
O                           QTY2   B  35
O                           QTY3   B  41
O                           QTY4   B  47
```

2



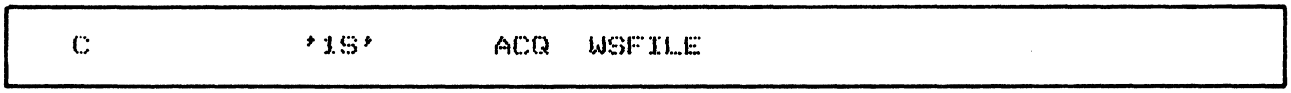
```
C                           READ WSFILE
C    KG                     GOTO EOJ
C                           SETOF                          030410
C                           SETOF                          82
C                           SETON                          0102
C*
C                *ZEROS     COMP ITM#                          03
C   N03          *BLANKS    COMP ITM#                          03
C    03                     SETOF                          01
C    03                     GOTO ITMINQ
O                   03N10               80 'INVALID ITEM NUMBER'
O                   03N10               88 'ENTERED'
```

3



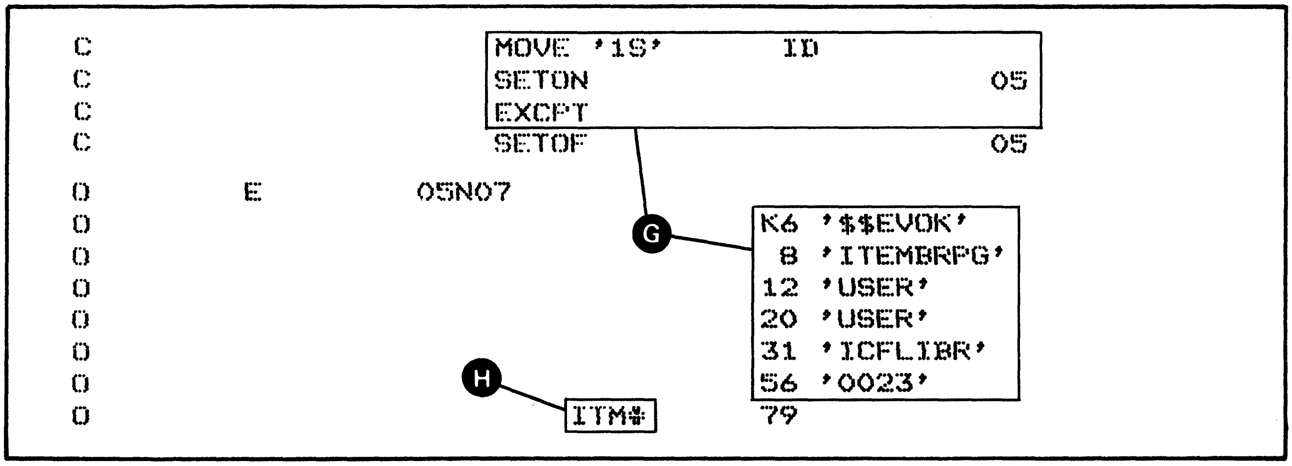
```
C                ITM#       CHAINFILEA                     13
C   N13                     GOTO ITMINQ
```

4



```
C                '1S'       ACQ  WSFILE
```

5



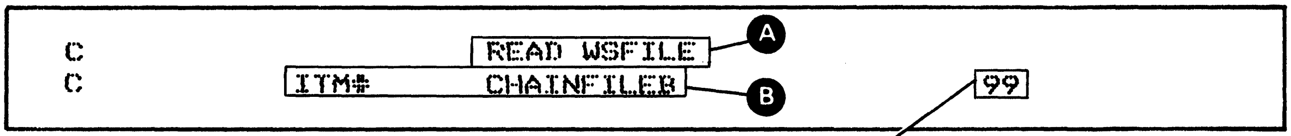
```
C                           MOVE '1S'        ID
C                           SETON                          05
C                           EXCPT
C                           SETOF                          05
O          E        05N07
O                                       K6  '$$EVOK'
O                                        8  'ITEMBRPG'
O                                       12  'USER'
O                                       20  'USER'
O                                       31  'ICFLIBR'
O                                       56  '0023'
O                           ITM#        79
```

Program A	Intra Subsystem[1]	Program B
6	← Return code →	Read the item number from program A Ⓐ and search the remote file for the data Ⓑ.
7	← Return code →	If the data is in the remote file, send it to program A ($$SEND) Ⓒ; otherwise, send a blank record beginning with *** Ⓓ.
8 Read the data from program B. →	← Return code	
9 Send $$SENDET to end the transaction Ⓔ.	← Return code	
10	← Return code → ←	Read the end of transaction Ⓕ. If an error occurs while reading the end of transaction, print the error return code Ⓖ. End of job Ⓗ.
11 Release the session Ⓘ → If indicator 10 is on, set up indicators to display the message ITEM NUMBER NOT FOUND Ⓙ. Go to ITMINQ to display the message or data.		
12 End of job if the operator pressed command key 7 (step 2 Ⓓ).		

[1] Return codes are checked by the INFSR subroutine, which is described later in this example in the discussion of *How to Check Return Codes with RPG II.*

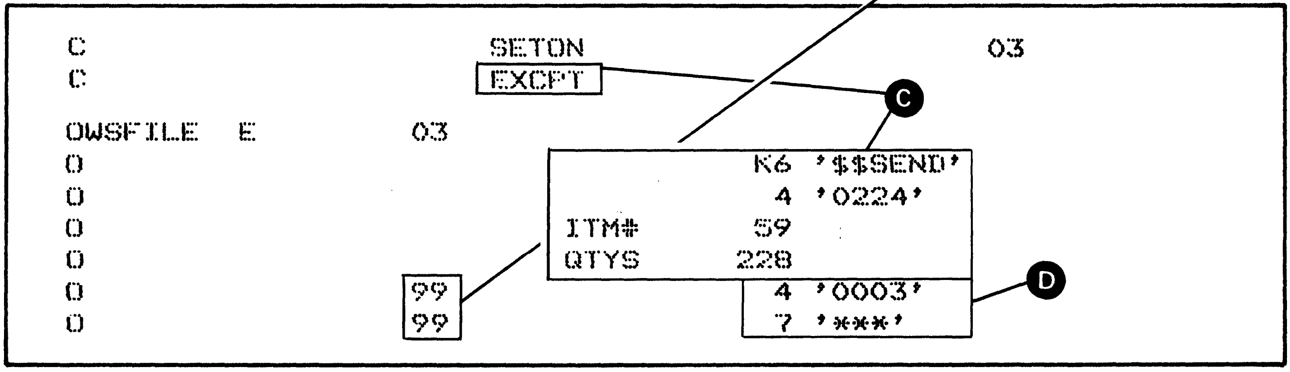
Program B

6



```
C                                 READ WSFILE
C               ITM#              CHAINFILEB                      99
```

7



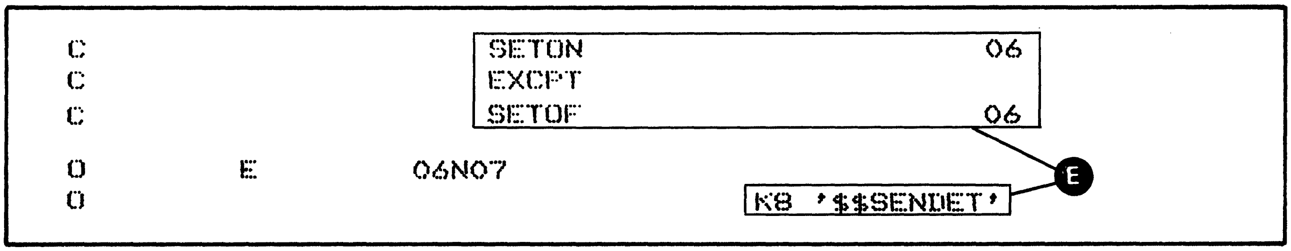
```
C                                 SETON                        03
C                                 EXCPT

OWSFILE   E              03
O                                              K6  '$$SEND'
O                                               4  '0224'
O                                  ITM#        59
O                                  QTYS       228
O                        99                     4  '0003'
O                        99                     7  '***'
```

Program A

8

```
C                  '1S'          NEXT WSFILE
C                                READ WSFILE
```

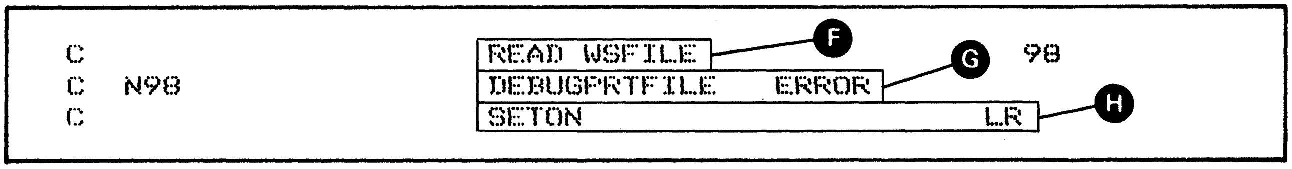
9



```
C                                SETON                          06
C                                EXCPT
C                                SETOF                          06

O         E              06N07
O                                              K8 '$$SENDET'
```

Program B

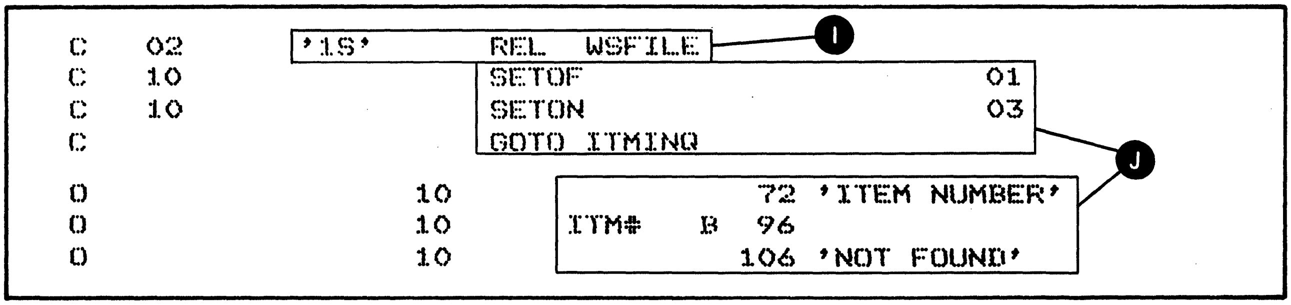
10



```
C                                READ WSFILE                    98
C   N98                          DEBUGPRTFILE      ERROR
C                                SETON                          LR
```

Program A

11



```
C    02            '1S'          REL   WSFILE
C    10                          SETOF                          01
C    10                          SETON                          03
C                                GOTO ITMINQ

O                        10                    72 'ITEM NUMBER'
O                        10        ITM#    B   96
O                        10                   106 'NOT FOUND'
```

12

```
C                  EOJ           TAG
C                                SETON                          LR
```

How to Check Return Codes with RPG II

The following example shows how to check ICF return codes with RPG II.

Program A

- Print the return code Ⓐ. If a problem occurs, you have a printout of what the return code was at the time the problem occurred.

- Save the return code (defined previously as RTCODE) in SAVERT so that it can be displayed later in the program Ⓑ.

- If the return code is 04xx (output exception) Ⓒ, data or a system message has been received from the remote system. The program reads the data or message Ⓓ, sends $$EOS to end the session Ⓕ and Ⓘ, sets up indicators to display the message or data Ⓖ and goes to detail calculations.

- If the return code is 82xx, the session was not acquired Ⓔ. The program sets up indicators to display the message UNABLE TO ACQUIRE Ⓗ and the return code to help the operator determine why the session was not acquired, and goes to detail calculations.

- If any other return codes were received, an error occurred. The program sends $$EOS to end the session Ⓕ and Ⓘ, sets up indicators to display the return code, and goes to detail calculations.

Program B

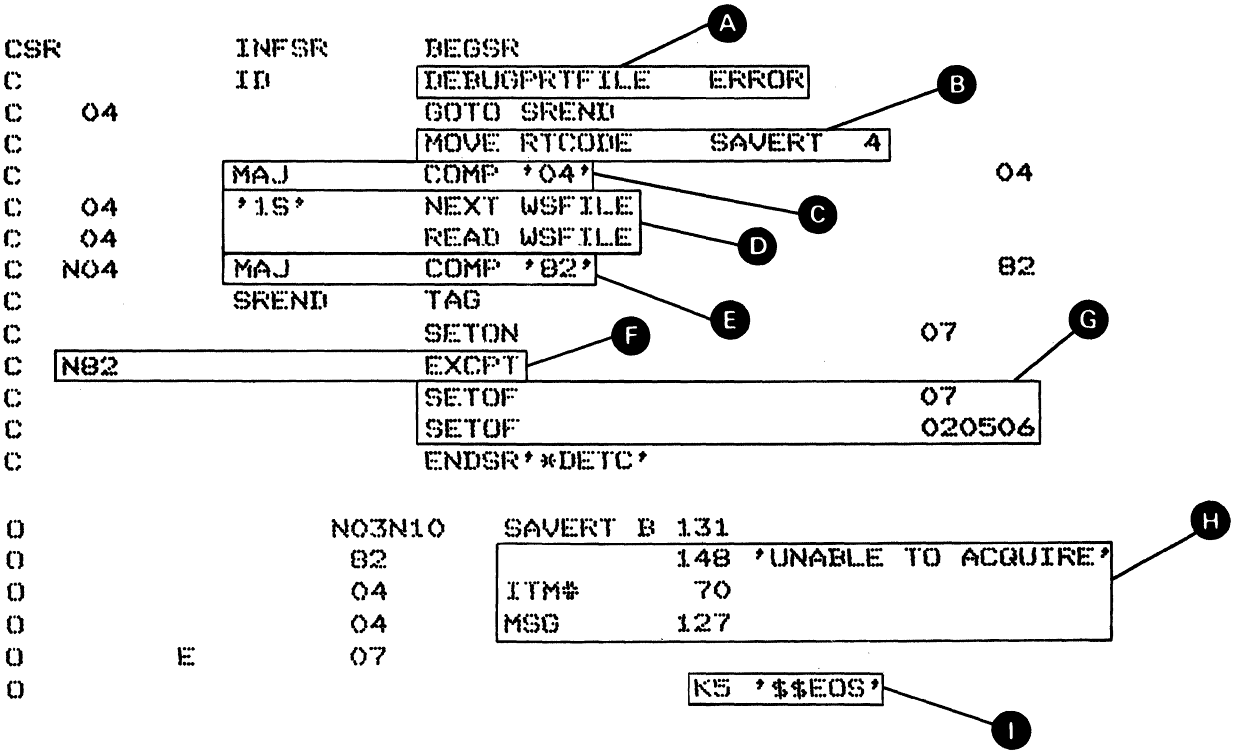
Program B prints all error return codes.

Note About Checking Return Codes

In this example, only the return codes that needed to be checked were checked. All other return codes were displayed or printed. Depending upon your remote system, you may need to check for return codes other than those shown in this example. However, you should display and/or print all return codes as an aid to problem determination. If any codes that are *not* error codes are returned repeatedly, you may want to include these in your return code routine. If an error codes is returned, you should, of course, correct the condition causing the error.

The return codes are described in each subsystem chapter. Only the codes that are valid for that subsystem are described.

Program A



```
CSR         INFSR     BEGSR
C           ID        DEBUGPRTFILE   ERROR
C   04                GOTO SREND
C                     MOVE RTCODE    SAVERT  4
C           MAJ       COMP '04'                    04
C   04      '1S'      NEXT WSFILE
C   04                READ WSFILE
C  N04      MAJ       COMP '82'                    82
C           SREND     TAG
C                     SETON                    07
C  N82                EXCPT
C                     SETOF                    07
C                     SETOF                    020506
C                     ENDSR'*DETC'

O                NO3N10   SAVERT B 131
O                 82                148 'UNABLE TO ACQUIRE'
O                 04      ITM#       70
O                 04      MSG       127
O        E        07
O                                   K5 '$$EOS'
```

Program B

```
CSR         INFSR     BEGSR
C                     DEBUGPRTFILE   ERROR
C                     ENDSR'*CANCL'
```

SCREEN FORMAT AND PROGRAM LISTINGS

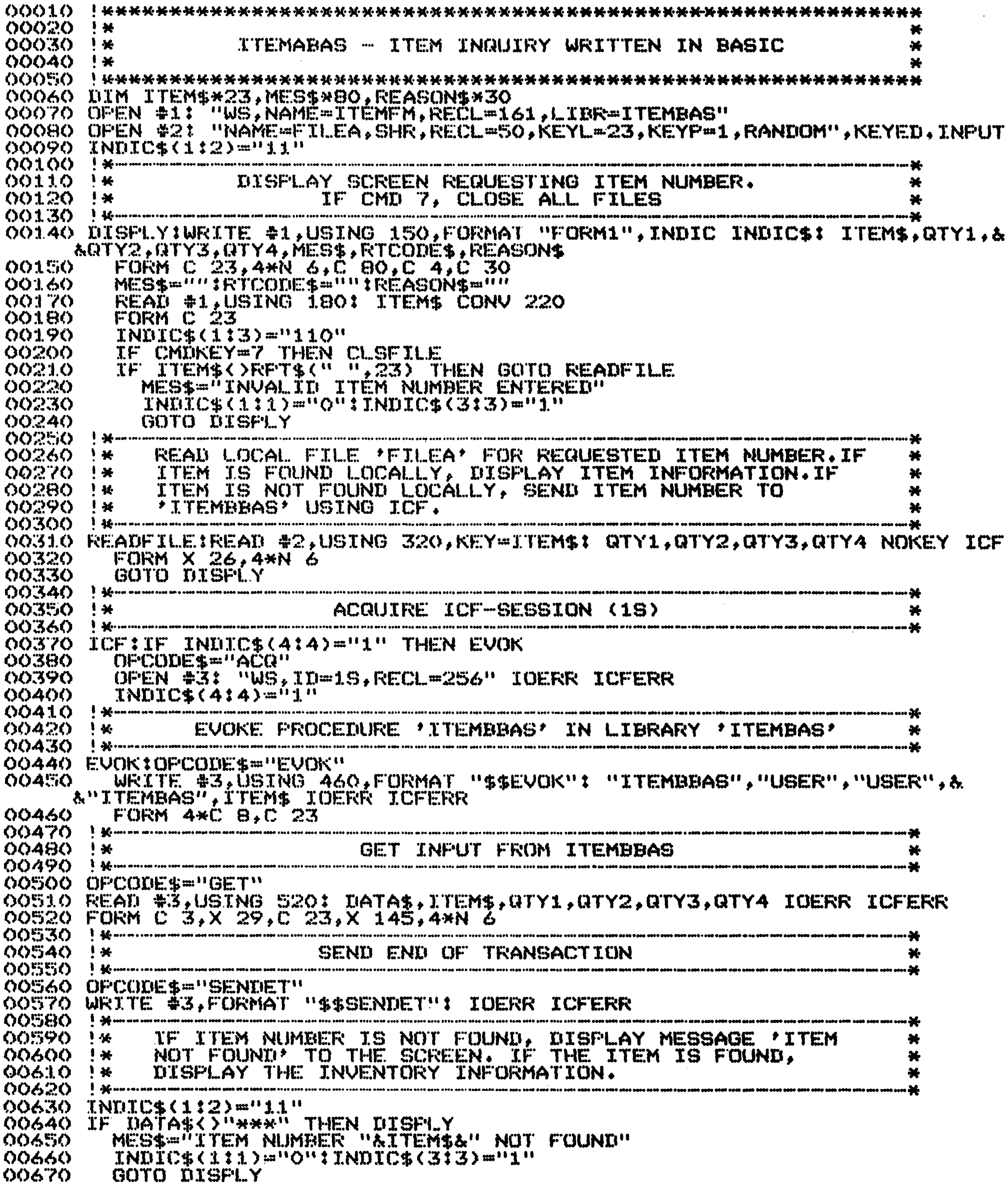
The following listings show the screen format and complete programs for the programming examples described previously.

Screen Format Listing

```
S  ITEMFM     546857/085829      3/0003     120/78       00000000        20/0014

    SFORM1                Y                                  G
    DSSPICF  00200119Y                     Y         C    S S P - I C F
    DITEMINQ 00200219Y                     Y         C     ITEM INQUIRY
    DITEMNUM 00180415Y                               CITEM NUMBER.......
    DITM#    00230434Y  Y    ZY  YY  Y     03
    DWH1     00180615Y                               CWARE HOUSE 1......
    DQTY1    00060634Y               Y
    DWH2     00180715Y                               CWARE HOUSE 2......
    DQTY2    00060734Y               Y
    DWH3     00180815Y                               CWARE HOUSE 3......
    DQTY3    00060834Y               Y
    DWH4     00180915Y                               CWARE HOUSE 4......
    DQTY4    00060934Y               Y
    DMSG     00801101Y               Y   01
    DICF-MSG 00451215Y               Y   02          CCHECK ICF REFERENCE MANX
    DUAL. FOR RETURN CODE OF
    DRTCODE  00041261Y               Y   02
    DREASON  00301315Y               Y   02
    DFL0015  00201346Y               Y   02          CPRESS ENTER TO RETRY
    DFL0016  00201502Y                               CCMD 7: END PROGRAM
```

BASIC Program Listing

Program A



```
00010 !************************************************************
00020 !*                                                          *
00030 !*         ITEMABAS - ITEM INQUIRY WRITTEN IN BASIC         *
00040 !*                                                          *
00050 !************************************************************
00060 DIM ITEM$*23,MES$*80,REASON$*30
00070 OPEN #1: "WS,NAME=ITEMFM,RECL=161,LIBR=ITEMBAS"
00080 OPEN #2: "NAME=FILEA,SHR,RECL=50,KEYL=23,KEYP=1,RANDOM",KEYED,INPUT
00090 INDIC$(1:2)="11"
00100 !*----------------------------------------------------------*
00110 !*         DISPLAY SCREEN REQUESTING ITEM NUMBER.           *
00120 !*               IF CMD 7, CLOSE ALL FILES                  *
00130 !*----------------------------------------------------------*
00140 DISPLY:WRITE #1,USING 150,FORMAT "FORM1",INDIC INDIC$: ITEM$,QTY1,&
     &QTY2,QTY3,QTY4,MES$,RTCODE$,REASON$
00150   FORM C 23,4*N 6,C 80,C 4,C 30
00160   MES$="":RTCODE$="":REASON$=""
00170   READ #1,USING 180: ITEM$ CONV 220
00180   FORM C 23
00190   INDIC$(1:3)="110"
00200   IF CMDKEY=7 THEN CLSFILE
00210   IF ITEM$<>RPT$(" ",23) THEN GOTO READFILE
00220     MES$="INVALID ITEM NUMBER ENTERED"
00230     INDIC$(1:1)="0":INDIC$(3:3)="1"
00240     GOTO DISPLY
00250 !*----------------------------------------------------------*
00260 !*   READ LOCAL FILE 'FILEA' FOR REQUESTED ITEM NUMBER.IF   *
00270 !*   ITEM IS FOUND LOCALLY, DISPLAY ITEM INFORMATION.IF     *
00280 !*   ITEM IS NOT FOUND LOCALLY, SEND ITEM NUMBER TO         *
00290 !*   'ITEMBBAS' USING ICF.                                  *
00300 !*----------------------------------------------------------*
00310 READFILE:READ #2,USING 320,KEY=ITEM$: QTY1,QTY2,QTY3,QTY4 NOKEY ICF
00320   FORM X 26,4*N 6
00330   GOTO DISPLY
00340 !*----------------------------------------------------------*
00350 !*                ACQUIRE ICF-SESSION (1S)                  *
00360 !*----------------------------------------------------------*
00370 ICF:IF INDIC$(4:4)="1" THEN EVOK
00380   OPCODE$="ACQ"
00390   OPEN #3: "WS,ID=1S,RECL=256" IOERR ICFERR
00400   INDIC$(4:4)="1"
00410 !*----------------------------------------------------------*
00420 !*     EVOKE PROCEDURE 'ITEMBBAS' IN LIBRARY 'ITEMBAS'      *
00430 !*----------------------------------------------------------*
00440 EVOK:OPCODE$="EVOK"
00450   WRITE #3,USING 460,FORMAT "$$EVOK": "ITEMBBAS","USER","USER",&
     &"ITEMBAS",ITEM$ IOERR ICFERR
00460   FORM 4*C 8,C 23
00470 !*----------------------------------------------------------*
00480 !*                  GET INPUT FROM ITEMBBAS                 *
00490 !*----------------------------------------------------------*
00500 OPCODE$="GET"
00510 READ #3,USING 520: DATA$,ITEM$,QTY1,QTY2,QTY3,QTY4 IOERR ICFERR
00520 FORM C 3,X 29,C 23,X 145,4*N 6
00530 !*----------------------------------------------------------*
00540 !*               SEND END OF TRANSACTION                    *
00550 !*----------------------------------------------------------*
00560 OPCODE$="SENDET"
00570 WRITE #3,FORMAT "$$SENDET": IOERR ICFERR
00580 !*----------------------------------------------------------*
00590 !*   IF ITEM NUMBER IS NOT FOUND, DISPLAY MESSAGE 'ITEM     *
00600 !*   NOT FOUND' TO THE SCREEN. IF THE ITEM IS FOUND,        *
00610 !*   DISPLAY THE INVENTORY INFORMATION.                     *
00620 !*----------------------------------------------------------*
00630 INDIC$(1:2)="11"
00640 IF DATA$<>"***" THEN DISPLY
00650   MES$="ITEM NUMBER "&ITEM$&" NOT FOUND"
00660   INDIC$(1:1)="0":INDIC$(3:3)="1"
00670   GOTO DISPLY
```

```
00680 !*----------------------------------------------------------*
00690 !*                      CLSFILE ROUTINE                       *
00700 !*----------------------------------------------------------*
00710 CLSFILE:CLOSE #1::CLOSE #2:
00720   IF INDIC$(4:4)="1" THEN CLOSE #3:
00730   STOP
00740 !*----------------------------------------------------------*
00750 !*                     SENDEOS SUBROUTINE                     *
00760 !*----------------------------------------------------------*
00770 SENDEOS:OPCODE$="EOS"
00780   WRITE #3,FORMAT "$$EOS": IOERR ICFERR
00790   CLOSE #3: IOERR ICFERR
00800   INDIC$(4:4)="0"
00810   RETURN
00820 !*----------------------------------------------------------*
00830 !*                      ICFERR ROUTINE                        *
00840 !*----------------------------------------------------------*
00850 ICFERR:PRINT #255,USING 860: "RETURN CODE ",RETCODE$," OPCODE IS ",&
     &OPCODE$," ITEM NUMBER IS ",ITEM$
00860   FORM SKIP 2,C 12,C 4,C 11,C 6,C 16,C 23
00870   RTCODE$=RETCODE$
00880   IF RETCODE$(1:2)>="04" THEN OUTCHK
00890     GOSUB SENDEOS
00900     GOTO DISPLY
00910 OUTCHK:IF RETCODE$(1:2))"04" THEN ACQCHK
00920   REASON$="OUTPUT EXCEPTION"
00930   READ #3,USING 940: MES$
00940   FORM V 80
00950   INDIC$(1:2)="00"
00960   GOSUB SENDEOS
00970   GOTO DISPLY
00980 ACQCHK:IF RETCODE$(1:2)<>"82" THEN ENDSESS
00990   REASON$="UNABLE TO ACQUIRE"
01000   INDIC$(1:2)="10"
01010   GOTO DISPLY
01020 ENDSESS:INDIC$(1:2)="10"
01030   GOSUB SENDEOS
01040   GOTO DISPLY
```

Program B

```
00010 !***********************************************************
00020 !*                                                         *
00030 !*         ITEMBBAS - ITEM INQUIRY WRITTEN IN BASIC        *
00040 !*                                                         *
00050 !***********************************************************
00060 DIM ITEM$*23
00070 OPEN #1: "NAME=FILEB,SHR,RECL=50,KEYL=23,KEYP=1,RANDOM",KEYED,INPUT
00080 OPEN #2: "WS,RECL=256" IOERR ICFERR
00090 !*---------------------------------------------------------*
00100 !*       READ TO ACCEPT ITEM NUMBER SENT BY REQUESTOR      *
00110 !*---------------------------------------------------------*
00120 OPCODE$="ACCEPT"
00130 WAITIO IOERR ICFERR
00140 READ #2,USING 150: ITEM$ IOERR ICFERR
00150 FORM C 23
00160 !*---------------------------------------------------------*
00170 !*      READ FILEB FILE, USING ITEM NUMBER SENT AS KEY     *
00180 !*---------------------------------------------------------*
00190 READFILE:READ #1,USING 200,KEY=ITEM$: QTY1,QTY2,QTY3,QTY4 NOKEY &
     &ERRORKEY
00200   FORM X 26,4*N 6
00210 !*---------------------------------------------------------*
00220 !*             SEND INFORMATION TO REQUESTOR               *
00230 !*---------------------------------------------------------*
00240 SENDDATA:OPCODE$="SEND"
00250   WRITE #2,USING 260,FORMAT "$$SEND": ITEM$,QTY1,QTY2,QTY3,QTY4 &
     $IOERR ICFERR
00260   FORM X 32,C 23,X 145,4*N 6
00270 !*---------------------------------------------------------*
00280 !*             READ SENDET SENT BY REQUESTOR               *
00290 !*---------------------------------------------------------*
00300 OPCODE$="GET"
00310 READ #2: IOERR ICFERR
00320 !*---------------------------------------------------------*
00330 !*                   CLOSE FILES AND END                   *
00340 !*---------------------------------------------------------*
00350 CLOSE #1::CLOSE #2:
00360 STOP
00370 !*---------------------------------------------------------*
00380 !*                     ERRORKEY ROUTINE                    *
00390 !*---------------------------------------------------------*
00400 ERRORKEY:OPCODE$="SEND"
00410   WRITE #2,USING 420,FORMAT "$$SEND": "***",ITEM$ IOERR ICFERR
00420   FORM C 3,X 29,C 192
00430   GOTO 300
00440 !*---------------------------------------------------------*
00450 !*                      ICFERR ROUTINE                     *
00460 !*---------------------------------------------------------*
00470 ICFERR:IF RETCODE$="0308" OR RETCODE$="0100" THEN CONTINUE
00480   PRINT #255,USING 490: "RETURN CODE ",RETCODE$," OPCODE IS ".OPCODE$.&
     &" ITEM NUMBER IS ",ITEM$
00490   FORM SKIP 2,C 12,C 4,C 11,C 6,C 16,C 23
00500   STOP
```

COBOL Program Listing

Program A

```
****************************************************************
*                                                              *
*    ITEMACOB - ITEM INQUIRY WRITTEN IN COBOL                  *
*                                                              *
****************************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. ITEMACOB.

 ENVIRONMENT DIVISION.

 CONFIGURATION SECTION.
 SOURCE-COMPUTER.  IBM-S34.
 OBJECT-COMPUTER.  IBM-S34.

 INPUT-OUTPUT SECTION.
 FILE-CONTROL.

     SELECT TRANSACTION-FILE
        ASSIGN TO WORKSTATION-ITEMFM-01,
        ORGANIZATION IS TRANSACTION,
        FILE STATUS IS WS-FS, ICF-FS,
        CONTROL-AREA IS WS-CONTROL-AREA.

     SELECT FILEA-FILE ASSIGN TO DISK-FILEA,
        ORGANIZATION IS INDEXED, ACCESS IS RANDOM,
        RECORD KEY IS FILEA-NUMBER.

     SELECT PRINT-FILE ASSIGN TO PRINTER-PRINTER.

/
 DATA DIVISION.
 FILE SECTION.
 FD  TRANSACTION-FILE, LABEL RECORDS ARE OMITTED.
 01  TRANSACTION-RECORD           PIC X(256).

 FD  FILEA-FILE, LABEL RECORDS ARE STANDARD.
 01  FILEA-RECORD.
     03  FILEA-NUMBER             PIC X(23).
     03  FILLER                   PIC XXX.
     03  FILEA-QTY-1              PIC 9(6).
     03  FILEA-QTY-2              PIC 9(6).
     03  FILEA-QTY-3              PIC 9(6).
     03  FILEA-QTY-4              PIC 9(6).

 FD  PRINT-FILE, LABEL RECORDS ARE OMITTED.
 01  PRINT-RECORD                 PIC X(132).

/
 WORKING-STORAGE SECTION.
 77  ICF-SESSION                  PIC XX VALUE '1S'.
 77  SAVE-ID                      PIC XX VALUE SPACES.
 77  SAVE-ITEM-NUMBER             PIC X(23).

 01  WS-DUMMY-AREAS.
     03  WS-CONTROL-AREA.
       05  AID-BYTE               PIC 99.
           88  CMD-KEY-7          VALUE 7.
       05  WS-ID                  PIC XX.
       05  FILLER                 PIC X(8).
     03  RETURN-CODES.
       05  WS-FS                  PIC XX.
       05  ICF-FS.
          07  MAJOR-RETURN-CODE  PIC XX.
          07  MINOR-RETURN-CODE  PIC XX.
```

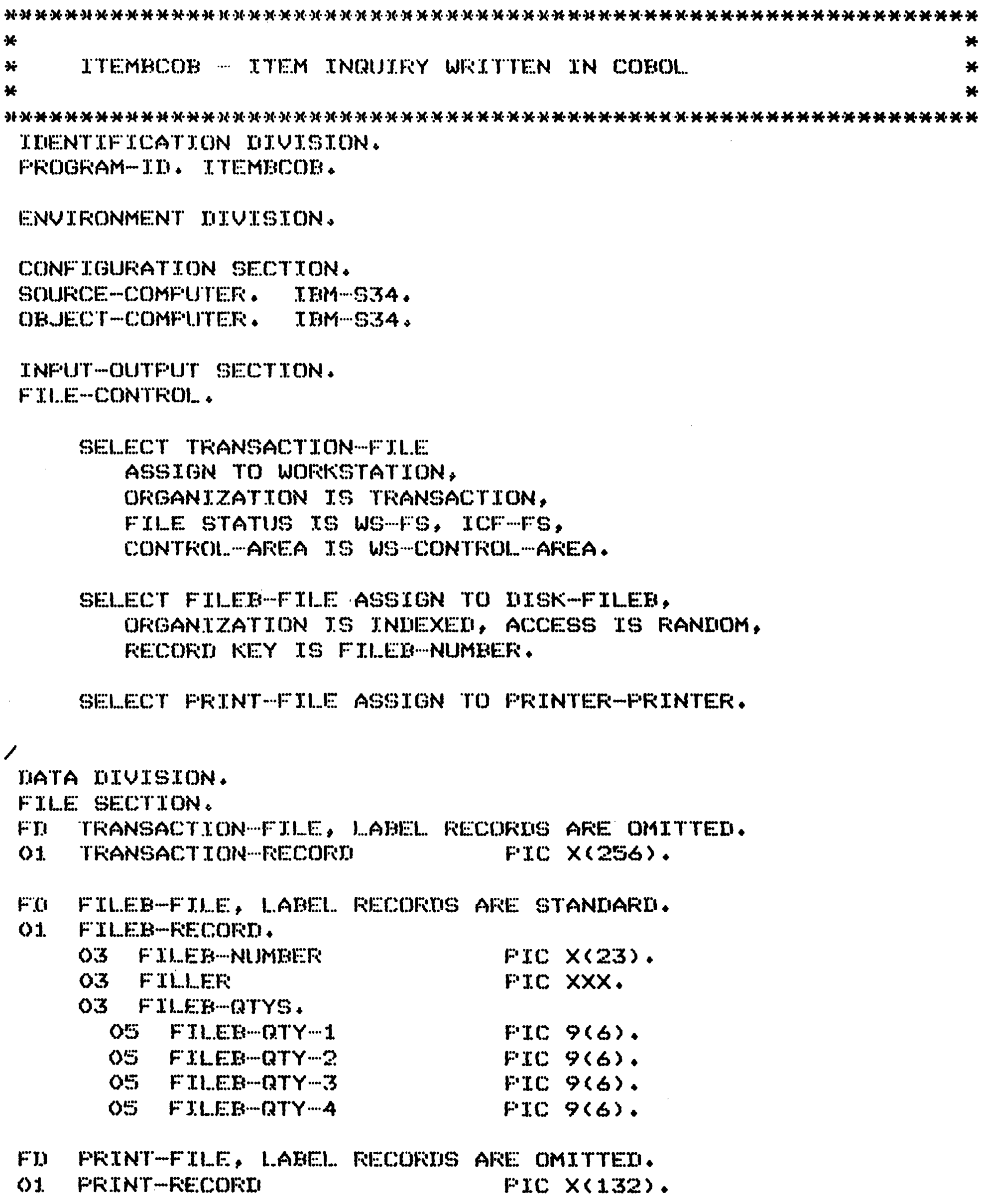
```
*---------------------------------------------------------------*
*    DISPLAY FORMAT INDICATOR DESCRIPTIONS (IF ON):              *
*    01 - NONDISPLAY -- MESSAGE LINE                             *
*    02 - NONDISPLAY -- ERROR LINE - DISPLAY RETURN CODE         *
*    03 - REVERSE IMAGE - ITEM NUMBER                            *
*---------------------------------------------------------------*
 01  SCREEN-INDICATORS.
     03  I01                     PIC 1 VALUE ZERO, INDICATOR 01.
     03  I02                     PIC 1 VALUE ZERO, INDICATOR 02.
     03  I03                     PIC 1 VALUE ZERO, INDICATOR 03.


 01  SCREEN-MESSAGES.
     03  INVALID-ITEM-MSG.
       05  FILLER                PIC X(14) VALUE SPACES.
       05  INVALID-MSG           PIC X(70)
                             VALUE 'INVALID ITEM NUMBER ENTERED.'.

     03  ITEM-NOT-FOUND-MSG.
       05  FILLER                PIC X(14) VALUE SPACES.
       05  FILLER                PIC X(12) VALUE 'ITEM NUMBER '.
       05  ITEM-NOT-FOUND        PIC X(23) VALUE SPACES.
       05  FILLER                PIC X(35) VALUE ' NOT FOUND.'.


/
 01  EVOKE-RECORD.
     03  PROCEDURE-NAME          PIC X(8) VALUE 'ITEMBCOB'.
     03  PASSWORD                PIC X(8) VALUE 'USER'.
     03  USER-ID                 PIC X(8) VALUE 'USER'.
     03  LIBRARY-NAME            PIC X(8) VALUE 'ICFLIBR'.
     03  FILLER                  PIC X(20).
     03  DATA-LENGTH             PIC XXXX VALUE '0023'.
     03  ICF-ITEM-NUMBER-OUT     PIC X(23).

 01  ICF-RECORD-IN.
     03  ICF-RECORD-CHECK.
       05  FIRST-3-CHARACTERS    PIC X(3).
       05  REST-OF-DATA          PIC X(253).

     03  ICF-RECORD-OK REDEFINES ICF-RECORD-CHECK.
       05  FILLER                PIC X(32).
       05  ICF-ITEM-NUMBER-IN    PIC X(23).
       05  FILLER                PIC X(145).
       05  ICF-QTY-1             PIC 9(6).
       05  ICF-QTY-2             PIC 9(6).
       05  ICF-QTY-3             PIC 9(6).
       05  ICF-QTY-4             PIC 9(6).
       05  FILLER                PIC X(32).

 01  SCREEN-RECORD.
     03  ITEM-NUMBER             PIC X(23).
     03  QTY-1                   PIC 9(6).
     03  QTY-2                   PIC 9(6).
     03  QTY-3                   PIC 9(6).
     03  QTY-4                   PIC 9(6).
     03  MSG                     PIC X(80).
     03  RETURN-CODE             PIC X(4).
     03  REASON-WHY              PIC X(30).

 01  PRINT-CODES.
     03  FILLER                  PIC X(14) VALUE 'RETURN CODE '.
     03  PRINT-RETURN-CODE       PIC XXXX.
     03  FILLER                  PIC X(11) VALUE ' OPCODE IS '.
     03  OPCODE                  PIC X(6).
     03  FILLER                  PIC X(11) VALUE ' DATA SENT '.
     03  PRINT-ITEM-NUMBER       PIC X(23).
```

```
/
 PROCEDURE DIVISION.
 OPEN-FILES.
     OPEN I-O TRANSACTION-FILE.
     OPEN OUTPUT PRINT-FILE.
     OPEN INPUT FILEA-FILE.
     MOVE B'1' TO IO1, IO2.
     MOVE SPACES TO SCREEN-RECORD.
*--------------------------------------------------------------*
*    DISPLAY SCREEN REQUESTING ITEM NUMBER.  IF CMD 7, GO TO   *
*    CLOSE FILES.  SET UP INDICATORS TO DISPLAY ERROR IF ITEM  *
*    NUMBER IS SPACES OR ZEROS.                                *
*--------------------------------------------------------------*
 ITEM-INQUIRY.
     MOVE SAVE-ID TO WS-ID.
     WRITE TRANSACTION-RECORD FROM SCREEN-RECORD
        FORMAT IS 'FORM1', TERMINAL IS WS-ID,
        INDICATORS ARE SCREEN-INDICATORS.
     READ TRANSACTION-FILE RECORD INTO SCREEN-RECORD,
        TERMINAL IS WS-ID.
     MOVE WS-ID TO SAVE-ID.
     MOVE B'1' TO IO1, IO2.
     MOVE B'0' TO IO3.
     IF CMD-KEY-7 GO TO CLOSE-FILES.
     IF ITEM-NUMBER = SPACES OR ITEM-NUMBER = ZEROS
        MOVE INVALID-ITEM-MSG TO MSG,
        MOVE B'0' TO IO1,
        MOVE B'1' TO IO3,
        GO TO ITEM-INQUIRY.
*--------------------------------------------------------------*
*    READ LOCAL FILE 'FILEA' FOR REQUESTED ITEM NUMBER.        *
*    IF ITEM IS FOUND LOCALLY, DISPLAY ITEM INFORMATION.       *
*    IF ITEM IS NOT FOUND LOCALLY, INQUIRE OF 'ITEMBCOB'       *
*    USING ICF.                                                *
*--------------------------------------------------------------*
 READ-FILEA-FILE.
     MOVE SPACES TO FILEA-RECORD.
     MOVE ITEM-NUMBER TO FILEA-NUMBER.
     READ FILEA-FILE,
       INVALID KEY
         GO TO ICF.
     MOVE FILEA-QTY-1 TO QTY-1.
     MOVE FILEA-QTY-2 TO QTY-2.
     MOVE FILEA-QTY-3 TO QTY-3.
     MOVE FILEA-QTY-4 TO QTY-4.
     GO TO ITEM-INQUIRY.
*--------------------------------------------------------------*
*    ACQUIRE ICF-SESSION (1S)                                  *
*--------------------------------------------------------------*
 ICF.
     ACQUIRE ICF-SESSION FOR TRANSACTION-FILE.
     MOVE 'ACQ' TO OPCODE.
     PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
     IF IO2 = B'0'
        GO TO ITEM-INQUIRY.
*--------------------------------------------------------------*
*    EVOKE PROCEDURE 'ITEMBCOB' IN LIBRARY 'ICFLIBR'           *
*--------------------------------------------------------------*
     MOVE ITEM-NUMBER TO ICF-ITEM-NUMBER-OUT.
     WRITE TRANSACTION-RECORD FROM EVOKE-RECORD
        FORMAT IS '$$EVOK',  TERMINAL IS ICF-SESSION.
     MOVE 'EVOK' TO OPCODE.
     PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
     IF IO2 = B'0'
        PERFORM SEND-EOS,
        GO TO ITEM-INQUIRY.
```

```
*---------------------------------------------------------------*
*    GET INPUT FROM ITEMBCOB.
*---------------------------------------------------------------*
     MOVE SPACES TO ICF-RECORD-IN.
     READ TRANSACTION-FILE RECORD INTO ICF-RECORD-IN,
       TERMINAL IS ICF-SESSION.
     MOVE 'GET' TO OPCODE.
     PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
     IF IO2 = B'0'
        PERFORM SEND-EOS,
        GO TO ITEM-INQUIRY.
*---------------------------------------------------------------*
*    SEND END OF TRANSACTION.                                   *
*---------------------------------------------------------------*
     MOVE SPACES TO TRANSACTION-RECORD.
     WRITE TRANSACTION-RECORD
        FORMAT IS '$$SENDET', TERMINAL IS ICF-SESSION.
     MOVE 'SENDET' TO OPCODE.
     PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
     IF IO2 = B'0'
        PERFORM SEND-EOS.
*---------------------------------------------------------------*
*    RELEASE SESSION (IF NOT ALREADY ENDED BY $$EOS)            *
*---------------------------------------------------------------*
     IF IO2 = B'1'
        DROP ICF-SESSION FROM TRANSACTION-FILE,
        MOVE 'DROP' TO OPCODE,
        PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END,
        IF IO2 = B'0'
           PERFORM SEND-EOS.
*---------------------------------------------------------------*
*    CHECK FOR ERROR MESSAGE (3 ASTERISKS)                      *
*    IF ITEM NUMBER IS NOT FOUND, DISPLAY MESSAGE 'ITEM NOT     *
*    FOUND' TO THE SCREEN.                                      *
*    IF THE ITEM IS FOUND, DISPLAY THE INVENTORY INFORMATION.   *
*---------------------------------------------------------------*
     MOVE B'1' TO IO1, IO2.
     MOVE ITEM-NUMBER TO SAVE-ITEM-NUMBER.
     MOVE SPACES TO SCREEN-RECORD.
     IF FIRST-3-CHARACTERS = '***'
        MOVE SAVE-ITEM-NUMBER TO ITEM-NOT-FOUND, ITEM-NUMBER,
        MOVE ITEM-NOT-FOUND-MSG TO MSG,
        MOVE B'0' TO IO1, MOVE B'1' TO IO3,
     ELSE
        MOVE ICF-ITEM-NUMBER-IN TO ITEM-NUMBER,
        MOVE ICF-QTY-1 TO QTY-1,
        MOVE ICF-QTY-2 TO QTY-2,
        MOVE ICF-QTY-3 TO QTY-3,
        MOVE ICF-QTY-4 TO QTY-4.
     GO TO ITEM-INQUIRY.
*---------------------------------------------------------------*
*    CLOSE FILES AND END JOB.                                   *
*---------------------------------------------------------------*
 CLOSE-FILES.
     CLOSE TRANSACTION-FILE, FILEA-FILE, PRINT-FILE.
     STOP RUN.
```

```
SEND-EOS.
    MOVE SPACES TO TRANSACTION-RECORD.
    WRITE TRANSACTION-RECORD
       FORMAT IS '$$EOS', TERMINAL IS ICF-SESSION.
    MOVE 'EOS' TO OPCODE.
    MOVE ICF-FS TO PRINT-RETURN-CODE.
    WRITE PRINT-RECORD FROM PRINT-CODES
       AFTER ADVANCING 2 LINES.


CHECK-RETURN-CODE.
    IF MAJOR-RETURN-CODE < '04'
       GO TO CHECK-RETURN-CODE-END.

    MOVE ITEM-NUMBER TO PRINT-ITEM-NUMBER.
    MOVE ICF-FS TO PRINT-RETURN-CODE.
    WRITE PRINT-RECORD FROM PRINT-CODES
       AFTER ADVANCING 2 LINES.
    MOVE SPACES TO RETURN-CODE.

    IF MAJOR-RETURN-CODE = 04
         MOVE ICF-FS TO RETURN-CODE,
         MOVE 'OUTPUT EXCEPTION' TO REASON-WHY,
         READ TRANSACTION-FILE RECORD INTO ICF-RECORD-CHECK.
            TERMINAL IS ICF-SESSION,
         MOVE ICF-RECORD-CHECK TO MSG,
         MOVE B'0' TO I01, I02,
    ELSE
         IF MAJOR-RETURN-CODE = 82
            MOVE ICF-FS TO RETURN-CODE,
            MOVE 'UNABLE TO ACQUIRE' TO REASON-WHY,
            MOVE B'1' TO I01, MOVE B'0' TO I02,
         ELSE
            IF MAJOR-RETURN-CODE > '04'
            MOVE ICF-FS TO RETURN-CODE,
            MOVE B'1' TO I01, MOVE B'0' TO I02.
CHECK-RETURN-CODE-END.
```

Program B



```
******************************************************************
*                                                                *
*    ITEMBCOB - ITEM INQUIRY WRITTEN IN COBOL                    *
*                                                                *
******************************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. ITEMBCOB.

 ENVIRONMENT DIVISION.

 CONFIGURATION SECTION.
 SOURCE-COMPUTER.  IBM-S34.
 OBJECT-COMPUTER.  IBM-S34.

 INPUT-OUTPUT SECTION.
 FILE-CONTROL.

     SELECT TRANSACTION-FILE
        ASSIGN TO WORKSTATION,
        ORGANIZATION IS TRANSACTION,
        FILE STATUS IS WS-FS, ICF-FS,
        CONTROL-AREA IS WS-CONTROL-AREA.

     SELECT FILEB-FILE ASSIGN TO DISK-FILEB,
        ORGANIZATION IS INDEXED, ACCESS IS RANDOM,
        RECORD KEY IS FILEB-NUMBER.

     SELECT PRINT-FILE ASSIGN TO PRINTER-PRINTER.

/
 DATA DIVISION.
 FILE SECTION.
 FD  TRANSACTION-FILE, LABEL RECORDS ARE OMITTED.
 01  TRANSACTION-RECORD          PIC X(256).

 FD  FILEB-FILE, LABEL RECORDS ARE STANDARD.
 01  FILEB-RECORD.
     03  FILEB-NUMBER            PIC X(23).
     03  FILLER                  PIC XXX.
     03  FILEB-QTYS.
       05  FILEB-QTY-1           PIC 9(6).
       05  FILEB-QTY-2           PIC 9(6).
       05  FILEB-QTY-3           PIC 9(6).
       05  FILEB-QTY-4           PIC 9(6).

 FD  PRINT-FILE, LABEL RECORDS ARE OMITTED.
 01  PRINT-RECORD                PIC X(132).
```

```
WORKING-STORAGE SECTION.

01  ICF-ITEM-NUMBER-IN              PIC X(23) VALUE SPACES.

01  WS-DUMMY-AREAS.
    03  WS-CONTROL-AREA.
      05  AID-BYTE                  PIC 99.
      05  ICF-SESSION               PIC XX.
      05  FILLER                    PIC X(8).
    03  RETURN-CODES.
      05  WS-FS                     PIC XX.
      05  ICF-FS.
         07  MAJOR-RETURN-CODE      PIC XX.
         07  MINOR-RETURN-CODE      PIC XX.

01  ICF-RECORD-OUT.
    03  DATA-LENGTH                 PIC X(4) VALUE '0224'.
    03  FILLER                      PIC X(32).
    03  ICF-ITEM-NUMBER-OUT         PIC X(23).
    03  FILLER                      PIC X(145).
    03  ICF-QTYS.
      05  ICF-QTY-1                 PIC 9(6).
      05  ICF-QTY-2                 PIC 9(6).
      05  ICF-QTY-3                 PIC 9(6).
      05  ICF-QTY-4                 PIC 9(6).

01  ERROR-RECORD-OUT.
    03  DATA-LENGTH                 PIC XXXX VALUE '0003'.
    03  FILLER                      PIC XXX VALUE '***'.

01  PRINT-CODES.
    03  FILLER                      PIC X(14) VALUE 'RETURN CODE '.
    03  PRINT-RETURN-CODE           PIC XXXX.
    03  FILLER                      PIC X(11)  VALUE ' OPCODE IS '.
    03  OPCODE                      PIC X(6).
```

```
/
 PROCEDURE DIVISION.
 OPEN-FILES.
     OPEN I-O TRANSACTION-FILE.
     OPEN OUTPUT PRINT-FILE.
     OPEN INPUT FILEB-FILE.
*----------------------------------------------------------------*
*    ACCEPT ITEM NUMBER SENT BY REQUESTOR.                       *
*----------------------------------------------------------------*
     READ TRANSACTION-FILE RECORD INTO ICF-ITEM-NUMBER-IN.
     MOVE 'ACCEPT' TO OPCODE.
     PERFORM WRITE-CODES THRU WRITE-CODES-END.
*----------------------------------------------------------------*
*    READ FILEB FILE, USING ITEM NUMBER SENT AS KEY.             *
*----------------------------------------------------------------*
 READ-FILEB-FILE.
     MOVE SPACES TO FILEB-RECORD.
     MOVE ICF-ITEM-NUMBER-IN TO FILEB-NUMBER.
     READ FILEB-FILE,
       INVALID KEY
         MOVE ERROR-RECORD-OUT TO TRANSACTION-RECORD,
         GO TO SEND-DATA.
     MOVE FILEB-NUMBER TO ICF-ITEM-NUMBER-OUT.
     MOVE FILEB-QTYS TO ICF-QTYS.
     MOVE ICF-RECORD-OUT TO TRANSACTION-RECORD.
*----------------------------------------------------------------*
*    SEND INFORMATION TO REQUESTOR.                              *
*----------------------------------------------------------------*
 SEND-DATA.
     WRITE TRANSACTION-RECORD FORMAT IS '$$SEND',
        TERMINAL IS ICF-SESSION.
     MOVE 'SEND' TO OPCODE.
     PERFORM WRITE-CODES THRU WRITE-CODES-END.
*----------------------------------------------------------------*
*    READ SENDET SENT BY REQUESTOR.                              *
*----------------------------------------------------------------*
     READ TRANSACTION-FILE RECORD TERMINAL IS ICF-SESSION.
     MOVE 'GET' TO OPCODE.
     PERFORM WRITE-CODES THRU WRITE-CODES-END.
*----------------------------------------------------------------*
*    CLOSE FILES AND END.                                        *
*----------------------------------------------------------------*
 CLOSE-FILES.
     CLOSE TRANSACTION-FILE, FILEB-FILE, PRINT-FILE.
     STOP RUN.


 WRITE-CODES.
     MOVE ICF-FS TO PRINT-RETURN-CODE.
     WRITE PRINT-RECORD FROM PRINT-CODES
        AFTER ADVANCING 2 LINES.
 WRITE-CODES-END.
```

RPG II Program Listing

Program A

```
F/EJECT
FWSFILE  CD        256            WORKSTN
F                                              KNUM         2
F                                              KFMTS  ITEMFM
F                                              KID    ID
F                                              KINFDS INFDS
F                                              KINFSR INFSR
FFILEA   IC  F  50  50R23AI     1 DISK
F*----------------------------------------------------------*
F*    PRINTER FILE IS USED FOR DEBUG ONLY.                  *
F*----------------------------------------------------------*
FPRTFILE O   F 132 132            PRINTER
F*
I/EJECT
I*----------------------------------------------------------*
I*    INPUT TO "WSFILE" CAN BE OF THREE TYPES:              *
I*    1) ERROR RECORD RETURNED FROM HOST; FIRST 3           *
I*       CHARACTERS ARE ***, REST OF RECORD IS BLANK        *
I*    2) INFORMATION RECORD RETURNED FROM HOST WITH THE     *
I*       FOLLOWING FORMAT:                                  *
I*                 COLUMNS 1 -  32   BLANKS                 *
I*                        33 -  55   ITEM NUMBER            *
I*                        56 - 200   BLANKS                 *
I*                       201 - 224   QUANTITIES             *
I*    3) ITEM NUMBER READ FROM THE WORKSTAION               *
I*----------------------------------------------------------*
IWSFILE  NS  10   1 C*   2 C*   3 C*
I        NS  11   3 C
I                                       33  55 ITM#
I                                      201 2060QTY1
I                                      207 2120QTY2
I                                      213 2180QTY3
I                                      219 2240QTY4
I        NS  12
I                                        1  23 ITM#
I                                       24  80 MSG
I*
I*----------------------------------------------------------*
I*   FILEA IS THE LOCAL FILE - SEARCHED FIRST               *
I*----------------------------------------------------------*
IFILEA   NS  14
I                                        1  23 ITM#
I                                       27  320QTY1
I                                       33  380QTY2
I                                       39  440QTY3
I                                       45  500QTY4
I*
I*----------------------------------------------------------*
I*   INFORMATION DATA STRUCTURE                             *
I*----------------------------------------------------------*
IINFDS       DS
I                                        1  26 ERROR
I                                       23  26 RTCODE
I                                       23  24 MAJ
I/EJECT
```

```
C**************************************************************
C*                                                            *
C*      I T M I N Q   - I T E M   I N Q U I R Y               *
C*                                                            *
C**************************************************************
C*                                                            *
C*      INITIALIZE INDICATORS ON FIRST CYCLE                  *
C*------------------------------------------------------------*
C  NO8                SETON                     010208
C*
C*------------------------------------------------------------*
C*      SET ID TO REQUESTOR AND DISPLAY SCREEN FORMAT 'FORM1' *
C*------------------------------------------------------------*
C           ITMINQ    TAG
C                     MOVE '  '      ID
C                     SETON                     09
C                     EXCPT
C                     SETOF                     09
C*
C*------------------------------------------------------------*
C*      READ ITEM NUMBER FROM SCREEN.  IF COMMAND KEY 7 WAS   *
C*      PRESSED, GOTO EOJ.  OTHERWISE, RESET INDICATORS.      *
C*------------------------------------------------------------*
C                     READ WSFILE
C   KG                GOTO EOJ
C                     SETOF                     030410
C                     SETOF                     82
C                     SETON                     0102
C*
C*------------------------------------------------------------*
C*      CHECK FOR ITEM NUMBER EQUAL TO ZERO OR BLANK. IF SO   *
C*      ISSUE MESSAGE.                                        *
C*------------------------------------------------------------*
C           *ZEROS    COMP ITM#                     03
C  NO3      *BLANKS   COMP ITM#                     03
C   03                SETOF                     01
C   03                GOTO ITMINQ
C*
C*------------------------------------------------------------*
C*      READ LOCAL ITEM FILE 'FILEA'.  IF ITEM NUMBER IS      *
C*      FOUND, GO TO ITMINQ. IF ITEM NUMBER IS NOT FOUND,     *
C*      CONTINUE AND INQUIRE THROUGH ICF.                     *
C*------------------------------------------------------------*
C           ITM#      CHAINFILEA                13
C  N13                GOTO ITMINQ
C*
C*------------------------------------------------------------*
C*      ACQUIRE ICF SESSION '1S'.                             *
C*------------------------------------------------------------*
C           '1S'      ACQ  WSFILE
C*
C*------------------------------------------------------------*
C*      EVOKE PROCEDURE 'ITEMBRPG'.                           *
```

```
C*-----------------------------------------------------------*
C                     MOVE '1S'      ID
C                     SETON                     05
C                     EXCPT
C                     SETOF                     05
C*
C*-----------------------------------------------------------*
C*      GET INFORMATION SENT BY 'ITEMBRPG'.                  *
C*-----------------------------------------------------------*
C           '1S'      NEXT WSFILE
C                     READ WSFILE
C*
C*-----------------------------------------------------------*
C*      SEND END OF TRANSACTION.                             *
C*-----------------------------------------------------------*
C                     SETON                     06
C                     EXCPT
C                     SETOF                     06
C*
C*-----------------------------------------------------------*
C*      RELEASE ICF SESSION '1S' AND GO TO ITMINQ.           *
C*-----------------------------------------------------------*
C   02      '1S'      REL  WSFILE
C   10                SETOF                     01
C   10                SETON                     03
C                     GOTO ITMINQ
C*
C*-----------------------------------------------------------*
C*      END OF JOB                                           *
C*-----------------------------------------------------------*
C           EOJ       TAG
C                     SETON                     LR
C*
C*************************************************************
C*                                                           *
C*      S U B R O U T I N E    I N F S R                     *
C*                                                           *
C*************************************************************
CSR         INFSR     BEGSR
C           ID        DEBUGPRTFILE   ERROR
C   04                GOTO SREND
C*
C*-----------------------------------------------------------*
C*      SAVE RETURN CODE.                                    *
C*-----------------------------------------------------------*
C                     MOVE RTCODE    SAVERT  4
C*
C*-----------------------------------------------------------*
C*      CHECK FOR MAJOR RETURN CODE 04 - OUTPUT EXCEPTION    *
C*      ISSUE INPUT OPERATION TO GET MESSAGE OR DATA         *
C*-----------------------------------------------------------*
C           MAJ       COMP '04'                     04
C   04      '1S'      NEXT WSFILE
C   04                READ WSFILE
C*
```

```
C*------------------------------------------------------------*
C*      CHECK FOR MAJOR RETURN CODE 82 - ACQUIRE FAILED       *
C*------------------------------------------------------------*
C  N04        MAJ       COMP '82'                    82
C           SREND     TAG
C*
C*------------------------------------------------------------*
C*    SEND END OF SESSION IF NOT ACQUIRE FAILURE, SET UP      *
C*    INDICATORS TO DISPLAY ERROR AND RETURN TO DETAIL CALCS. *
C*------------------------------------------------------------*
C                     SETON                    07
C  N82                EXCPT
C                     SETOF                    07
C                     SETOF                    020506
C                     ENDSR'*DETC'
O/EJECT
O*---------------------------------------------------------*
O*  OUTPUT FORMAT FORM1 OF SOURCE MEMBER ITEMFM            *
O*---------------------------------------------------------*
OWSFILE  E        09
O                                   K5 'FORM1'
O                         ITM#      23
O                         QTY1   B  29
O                         QTY2   B  35
O                         QTY3   B  41
O                         QTY4   B  47
O*---------------------------------------------------------*
O*  INDICATOR 03 ON & 10 NOT ON - INVALID ITEM # MESSAGE   *
O*---------------------------------------------------------*
O                 03N10             80 'INVALID ITEM NUMBER'
O                 03N10             88 'ENTERED'
O*---------------------------------------------------------*
O*  INDICATOR 10 ON - ITEM NOT FOUND MESSAGE               *
O*---------------------------------------------------------*
O                 10                72 'ITEM NUMBER'
O                 10      ITM#   B  96
O                 10               106 'NOT FOUND'
O*---------------------------------------------------------*
O*  IF INDICATORS 03 & 10 NOT ON, OUTPUT ICF INFORMATION   *
O*---------------------------------------------------------*
O                N03N10   SAVERT B 131
O                 82               148 'UNABLE TO ACQUIRE'
O                 04      ITM#      70
O                 04      MSG      127
O*---------------------------------------------------------*
O*   OUTPUT $$EVOK SENDING ITEM NUMBER                     *
O*---------------------------------------------------------*
O        E        05N07
O                                   K6 '$$EVOK'
O                                    8 'ITEMBRPG'
O                                   12 'USER'
O                                   20 'USER'
O                                   31 'ICFLIBR'
O                                   56 '0023'
O                         ITM#      79
```

```
O*---------------------------------------------------------*
O*   OUTPUT $$SENDET - SEND END OF TRANSACTION             *
O*---------------------------------------------------------*
O        E        06N07
O                                   K8 '$$SENDET'
O*---------------------------------------------------------*
O*   OUTPUT $$EOS - END OF SESSION                         *
O*---------------------------------------------------------*
O        E        07
O                                   K5 '$$EOS'
```

Program B

```
F/EJECT
FWSFILE  CD        256           WORKSTN
F                                              KFMTS  *NONE
F                                              KINFDS INFDS
F                                              KINFSR INFSR
FFILEB   IC  F  50  50R23AI     1 DISK
F*----------------------------------------------------------*
F*    PRINTER FILE IS USED FOR DEBUG ONLY.                  *
F*----------------------------------------------------------*
FPRTFILE O   F 132 132            PRINTER
F/SPACE
I*----------------------------------------------------------*
I*  WS INPUT IS FROM ICF                                    *
I*----------------------------------------------------------*
IWSFILE  NS
I                                        1  23 ITM#
IFILEB   NS
I                                        1  23 ITM#
I                                       27  50 QTYS
IINFDS       DS
I                                        1  26 ERROR
I/EJECT
C*----------------------------------------------------*
C*    ACCEPT INPUT SENT                               *
C*----------------------------------------------------*
C                     READ WSFILE
C*
C*----------------------------------------------------*
C*    READ FILE "FILEB"                               *
C*----------------------------------------------------*
C           ITM#      CHAINFILEB                99
C*
C*----------------------------------------------------*
C*    RETURN RESULTS VIA ICF                          *
C*----------------------------------------------------*
C                     SETON                     03
C                     EXCPT
C*
C*----------------------------------------------------*
C*    ACCEPT SENDET                                   *
C*----------------------------------------------------*
C                     READ WSFILE                 98
C  N98                DEBUGPRTFILE   ERROR
C                     SETON                     LR
C*
C******************************************************
C*                                                    *
C*                  I N F S R                         *
C*                                                    *
C******************************************************
CSR         INFSR     BEGSR
C                     DEBUGPRTFILE   ERROR
C                     ENDSR'*CANCL'
C/EJECT
```

```
O*----------------------------------------------------*
O*    OUTPUT $$SENDET. IF ITEM IS NOT FOUND,          *
O*    INDICATOR 99 IS ON, AND OUTPUT ***.             *
O*----------------------------------------------------*
OWSFILE  E        03
O                                   K6 '$$SEND'
O                                    4 '0224'
O                         ITM#      59
O                         QTYS     228
O                 99                 4 '0003'
O                 99                 7 '***'
```

Intra Subsystem Return Codes

This part of Chapter 7 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 four-digit value; the first two digits contain the major code, and the last two 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).

Note: In the return code descriptions, *your program* refers to the local System/34 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/34 with which this program is communicating through SSP-ICF.

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 the System/34.

Input Operations to Your Program	Output Operations from Your Program	Combined Operations in Your Program
Accept input		
	Acquire[1]	
	Cancel	Cancel then get[2] Cancel then invite
	End of session	
	Evoke[3] Evoke end of transaction[3]	Evoke then get[2,3] Evoke then invite[3]
	Fail	
Get Get attributes[4]		
Invite		
	Negative response	Negative response then get[2] Negative response then invite
Pass-through invite	Pass-through put[2]	Pass-through put then invite
	Put Put end of chain Put end of transaction	Put then get[2] Put then invite
	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]	

[1] Normally, 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.

[2] Valid only in assembler language.

[3] Evoke operations in assembler can have OPM-FMH specified with the $WSIO macro.

[4] Valid only in assembler and COBOL languages.

[5] For BASIC and RPG II programs, the set 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.

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):**
Initiated the transaction (evoked the other program)	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.

[1] An evoked program (started by an evoke operation issued by another System/34 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.

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 code of xx00 (a change direction indication, which allows your program to send data), or xx08 (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[1] 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[1] was received with some data on a successful input operation. Your program must continue to receive input until SSP-ICF returns a code of xx00 (a change direction indication) or xx08 (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[1] 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.

[1] The function management header is used in SNA only. However, you can use it with the Intra subsystem to test programs that use the SNUF subsystem. The header describes the format to be used for the data being received (following the header).

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.

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, either issue an end of session operation or terminate.

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 the System/34. 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 indicate 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[1] for the new requester, and issue an output operation or an input operation.

[1] For some situations, no record keeping for the session is necessary. In other situations, you should record the session ID of the new requester. You may also want to keep a table containing the IDs of all active requesters, or to maintain a history log of all requests.

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 code of xx00 (a change direction indication) or xx08 (an end of transaction indication).

Normal Action: Handle any data passed with the request, perform any necessary record keeping[1] 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[1], 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[1] 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[2] were received with some data. Your program must continue to receive input until SSP-ICF returns a code of xx00 (a change direction indication) or xx08 (an end of transaction indication).

Normal Action: Process the function management header, handle the data passed with the request, perform any necessary record keeping[1] for the new requester, and issue an input operation.

[1] For some situations, no record keeping for the session is necessary. In other situations, you should record the session ID of the new requester. You may also want to keep a table containing the IDs of all active requesters, or to maintain a history log of all requests.

[2] The function management header is used in SNA only. However, you can use it with the Intra subsystem to test programs that use the SNUF subsystem. The header describes the format to be used for the data being received (following the header).

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 incoming 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[1] 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[2] 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[1] for the new requester. Then, either issue an end of session operation or terminate your program.

[1] For some situations, no record keeping for the session is necessary. In other situations, you should record the session ID of the new requester. You may also want to keep a table containing the IDs of all active requesters, or to maintain a history log of all requests.

[2] The function management header is used in SNA only. However, you can use it with the Intra subsystem to test programs that use the SNUF subsystem. The header describes the format to be used for the data being received (following the header).

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 incoming 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[1] 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.

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.

[1] For some situations, no record keeping for the session is necessary. In other situations, you should record the session ID of the new requester. You may also want to keep a table containing the IDs of all active requesters, or to maintain a history log of all requests.

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 code of xx00 (a change direction indication) or xx08 (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[1] 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[1] 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 code of xx00 (a change direction indication) or xx08 (an end of transaction indication).

Normal Action: Process the function management header and issue another input operation.

[1] The function management header is used in SNA only. However, you can use it with the Intra subsystem to test programs that use the SNUF subsystem. The header describes the format to be used for the data being received (following the header).

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[1] 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.

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, either issue an end of session operation or terminate.

[1] The function management header is used in SNA only. However, you can use it with the Intra subsystem to test programs that use the SNUF subsystem. The header describes the format to be used for the data being received (following the header).

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, either issue an end of session operation or terminate.

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 code of xx00 (a change direction indication, which allows your program to send data), or xx08 (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.

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. Your program must receive the data before it can perform an output operation.

Normal Action: Issue an input operation to receive the data.

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/34 Messages Guide* 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 (in COBOL and assembler only) a set timer operation, or by using the TIMER intrinsic function (in BASIC only); (2) continue local processing; or (3) terminate.

- 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[1] to reissue the acquire operation, or terminate.

If more than one subsystem was active when the error occurred, all subsystems that were active when the error occurred should be disabled. (Note that all other active Intra subsystems are automatically disabled when the error occurs; all other types of active subsystems must be manually disabled.)

If all subsystems (of all types) on the system are *not* disabled to recover from the processor check, the common queue space used by the failing subsystem *cannot* be freed. And if it is not freed, that space is wasted, and an indication of insufficient common queue space being available can occur. The indication can occur as a message when the failing subsystem is reenabled or when a different subsystem is enabled. The indication can also occur as an 8315 return code for any subsystem that is performing an output operation in a session.

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[1] to reissue the acquire operation, or terminate.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

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/34 Messages Guide* 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 not enough common queue space was available to support a new session). 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.)

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 operation is valid only in an *active* session and cannot, therefore, be issued after an 82xx return code is received. (This restriction does not apply to COBOL and assembler programs, or to the TIMER intrinsic function in BASIC, which also can be used to wait for a specified time interval.)

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, respecify 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[1] to reissue the acquire operation, or terminate.

If more than one subsystem was active when the error occurred, all subsystems that were active when the error occurred should be disabled. (Note that all other active Intra subsystems are automatically disabled when the error occurs; all other types of active subsystems must be manually disabled.)

If all subsystems (of all types) on the system are *not* disabled to recover from the processor check, the common queue space used by the failing subsystem *cannot* be freed. And if it is not freed, that space is wasted, and an indication of insufficient common queue space being available can occur. The indication can occur as a message when the failing subsystem is reenabled or when a different subsystem is enabled. The indication can also occur as an 8315 return code for any subsystem that is performing an output operation in a session.

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[1] to reissue the acquire operation, or terminate.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

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 100 sessions can be active in the System/34 at one time. The session was not started.

Recovery Action: Your program can wait[1] 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 to be enabled is identified by the location parameter in the SESSION statement. That location name must also be specified in the subsystem configuration record (shown on display 3.0 of the subsystem configuration planning charts). The session was not started.

Recovery Action: Verify that the subsystem name was specified correctly on the LOCATION parameter of the SESSION statement. If the correct name was specified, contact the System/34 system operator and request that the specified subsystem be enabled by executing the ENABLE procedure command at the system console. Then reissue the acquire operation. Otherwise, your program can continue local processing, wait[1] 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[1] 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[1] until the subsystem is enabled again, and then reissue the acquire operation. Otherwise, your program can continue local processing, or it can terminate.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

82B4 **Error Indication:** The acquire operation was not successful because all of the resources needed for the session could not be allocated from the assign/free area of the system. All available resources are already being used in the system. The session was not started.

Recovery Action: Wait[1] for the needed resources to become available, then reissue the acquire operation. Otherwise, continue local processing or 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/34 Messages Guide* 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.
   c. 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.)

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

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 an 81xx (session was terminated) or 82xx (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 an 81xx or 82xx return code is handled properly. If you want your program to recover from an improperly handled error condition, issue another acquire operation.

8313 **Error Indication:** On an *output* operation, a queue space error condition was detected. The output operation could not be performed because no *subsystem* queue space was available at the time.

Recovery Action: Your program can issue a set timer operation and wait for a period of time, then reissue the output operation. If an unacceptable number of queue space errors occur, you can disable the subsystem and change the subsystem configuration by specifying a larger subsystem queue space size in the subsystem queue space parameter. After the subsystem is enabled, reissue the acquire operation to restart the session.

8314 **Error Indication:** On an *input* operation, a queue space error condition was detected. The input operation could not be performed because no *subsystem* queue space was available at the time.

Recovery Action: Your program can issue a set timer operation and wait for a period of time, then reissue the input operation. If an unacceptable number of queue space errors occur, you can disable the subsystem and change the subsystem configuration by specifying a larger subsystem queue space size in the subsystem queue space parameter. After the subsystem is enabled, reissue the acquire operation to restart the session.

8315 **Error Indication:** On an evoke operation, a queue space error condition was detected. The evoke operation could not be performed because no *common* queue space was available at the time.

Recovery Action: Your program can issue a set timer operation and wait for a period of time, then reissue the evoke operation. If an unacceptable number of queue space errors occur, you can disable all the subsystems and change the subsystem configuration by specifying a larger common queue space size in the SSP-ICF common queue space parameter. After the subsystem is enabled, reissue the acquire operation to restart the session.

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:** An evoke operation failed to complete successfully, or the evoked program terminated abnormally. A message describing why it failed is waiting in the subsystem input buffer. The evoke operation could have been 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 of 0411 or 0412 (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: You can either change the record length in your program and recompile it, or you can change the value specified for the maximum user record length parameter in the subsystem configuration. 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 erroneously 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 indication). 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 dynamically 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 dynamically 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 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 code of xx00 (a change direction indication) or xx08 (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.

Chapter 8. The BSCEL Subsystem

The BSCEL (BSC equivalence link) subsystem provides distributed data processing support to users of the System/34 SSP in conjunction with another System/34 using the BSCEL subsystem. The BSCEL subsystem provides an interactive interface between application programs on different System/34s. The BSCEL subsystem also allows BSC communications with the following systems:

- System/38
- System/34 with batch BSC
- System/32 with RPG II or assembler
- System/3 with RPG II or MLMP
- System/7 with MSP/7 (as a System/3)
- OS, OS/VS, DOS/VS, or DOS BTAM.
- OS or OS/VS TCAM
- IBM 3741 Model 2 Data Station or Model 4 Programmable Work Station
- IBM 3747 Data Converter
- IBM 5231 Data Collection Controller Model 2 (as a 3741 in transmit mode)
- IBM 3750 Switching System (World Trade only)
- IBM 3705 using NCP EP or PEP
- IBM 5110 (as a 3741)
- Series 1 (as a System/3)
- IBM 5260 Point of Sale Terminal (as a 3741)
- IBM 5280 Distributed Data System (as a 3741)

The BSCEL subsystem treats all these systems identically; any system considerations must be handled by the application program.

The BSCEL subsystem supports up to four active lines. Each line must be one of three types:

- Point-to-point nonswitched

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

- Multipoint tributary

The BSCEL subsystem supports one session on each line. One application program can, however, conduct four concurrent sessions, provided they are on four different lines and four different BSCEL configurations are enabled. Also, one application program can conduct multiple sessions consecutively on one line.

The BSCEL subsystem provides EBCDIC/ASCII translation. The application program must process all data in EBCDIC. If ASCII is used, the subsystem translates output data to ASCII before transmitting it, and translates input data to EBCDIC before passing it to the application program.

The BSCEL subsystem also supports the following items, which are currently supported by batch BSC, when communicating with the same remote systems:

- 3740 multiple files with null records separating the files

- 3780 format blank compression and truncation

- Record separator mode

- Intermediate block check mode

- Blocked records

The batch BSC support is described in the *Data Communications Reference Manual.*

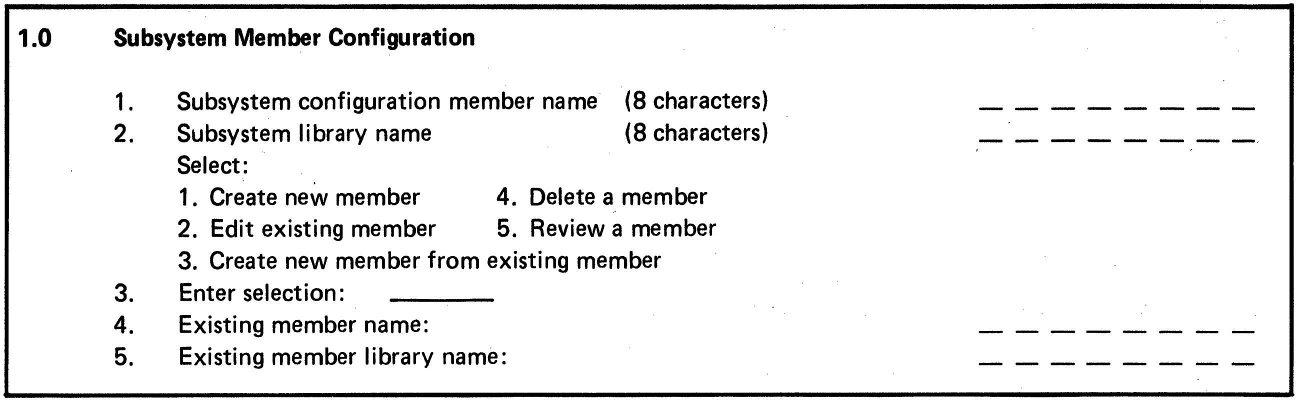
SETTING UP THE BSCEL SUBSYSTEM

The SSP procedures CNFIGSSP and INSTALL are used to include the interactive communications feature and BSCEL subsystem support on the System/34. The general interactive communications and line control support is included when it is requested on the appropriate CNFIGSSP prompt. The BSCEL subsystem support is copied to the system library when the appropriate responses to the INSTALL procedure prompts are taken. The CNFIGSSP and INSTALL procedures, with their displays and related responses, are described in the *Installation and Modification Reference Manual.*

After the BSCEL subsystem has been installed, the CNFIGICF procedure is used to tailor the subsystem support to an existing or proposed network. The operation of the CNFIGICF procedure is also explained in the *Installation and Modification Reference Manual.* Before running the CNFIGICF procedure, however, you should fill out a planning chart for each subsystem that you want to define.

Copies of the planning chart for each subsystem are available in Appendix F of this manual and in the *Installation and Modification Reference Manual.* The following sections explain how to fill out the planning chart for the BSCEL subsystem.

Display 1.0 Subsystem Member Configuration



```
1.0     Subsystem Member Configuration

        1.   Subsystem configuration member name  (8 characters)     _ _ _ _ _ _ _ _
        2.   Subsystem library name               (8 characters)     _ _ _ _ _ _ _ _
             Select:
             1. Create new member       4. Delete a member
             2. Edit existing member    5. Review a member
             3. Create new member from existing member
        3.   Enter selection: ______
        4.   Existing member name:                                   _ _ _ _ _ _ _ _
        5.   Existing member library name:                           _ _ _ _ _ _ _ _
```

Subsystem configuration member name: Specify a name for this configuration of the subsystem. This name is used to store the member in a library, and is referenced in the ENABLE and DISABLE procedures.

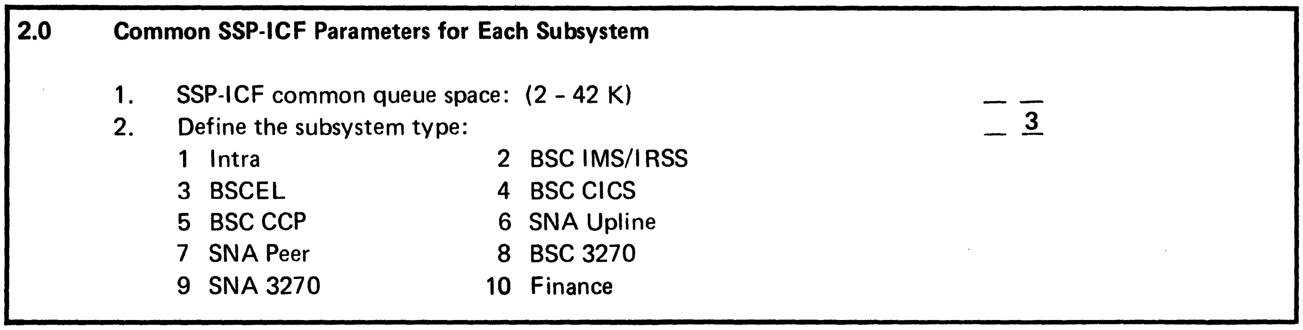
Library name: Specify the name of the library in which the configuration is stored or to be stored. The default is #LIBRARY, however, you should probably store the configuration in a user library.

Enter selection: Specify one of the five options: (1) create a new member, (2) edit an existing member, (3) create a new member from an existing member, (4) delete a member, or (5) review a member without changing it.

Existing member name: This prompt appears if option 3 was selected. Specify the name of the existing subsystem configuration member that is to be used to create the new member. The existing member remains unchanged.

Existing member library name: This prompt appears if option 3 was selected. Specify the library name where the existing member resides.

Display 2.0 Common SSP-ICF Parameters for Each Subsystem



```
2.0     Common SSP-ICF Parameters for Each Subsystem

        1.   SSP-ICF common queue space: (2 - 42 K)                    _ _
        2.   Define the subsystem type:                                _ 3
             1  Intra                2  BSC IMS/IRSS
             3  BSCEL                4  BSC CICS
             5  BSC CCP              6  SNA Upline
             7  SNA Peer             8  BSC 3270
             9  SNA 3270            10  Finance
```

SSP-ICF common queue space: Specify the size, in multiples of 2 K bytes, of the common queue space. The common queue space requirements for each configuration of the BSCEL subystem with an active session are:

$$C = 16N + 558$$

where:

C = number of bytes required for common queue space
N = number of remote switched line IDs (0 if a nonswitched line is specified or if only 1 remote ID is specified)

If ASCII is selected, add 768 bytes to the common queue space requirements.

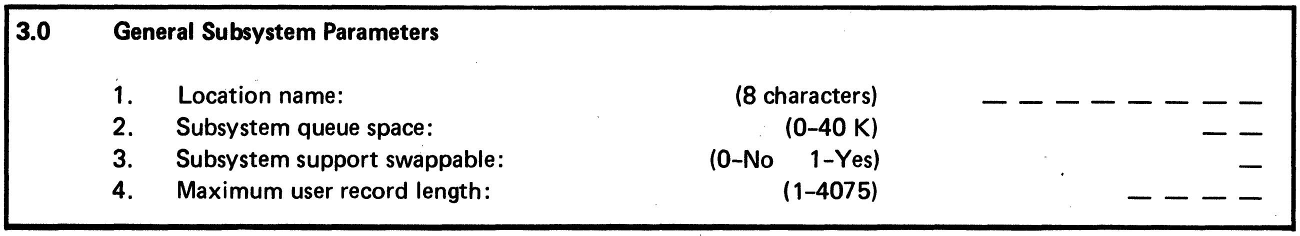
The common queue space value specified for the first subsystem that is enabled becomes the size of the common queue space. Be sure that the value specified for common queue space size takes into account the requirements of any other subsystem that might be active concurrently.

The size of the common queue space plus the total subsystem queue space of all the enabled BSCEL subsystems cannot exceed 42 K bytes.

The default common queue space size is 4 K bytes.

Define the subsystem type: Specify a 3 for the BSCEL subsystem.

Display 3.0 General Subsystem Parameters



```
3.0    General Subsystem Parameters

       1.  Location name:                        (8 characters)     _ _ _ _ _ _ _ _
       2.  Subsystem queue space:                      (0-40 K)                 _ _
       3.  Subsystem support swappable:          (0-No   1-Yes)                   _
       4.  Maximum user record length:                (1-4075)             _ _ _ _
```

Location name: Specify up to 8 characters for the name of the location associated with this configuration. The location name is used in some of the displayed message texts, and must be coded on the SESSION OCL statement. The default is the subsystem configuration member name. The location name refers to the name of the location with which communications is to take place.

Subsystem queue space: Specify the size, in multiples of 2 K bytes, of the subsystem queue space. The subsystem queue space requirements for each configuration of the BSCEL subsystem with an active session is:

$$S = 2R + 271,$$

where:

S = The number of bytes required for subsystem queue space
R = (The maximum user record length x F) + N

where:

F = The number of records per block (1 if no blocking is used)
N = The number of bytes needed for ITB or record separator characters as follows:

N = F if record separators are used
N = (F-1) if nontransparent ITB mode is used
N = (F-1) * 3 if transparent ITB mode is used
N = 0 if no blocking is used

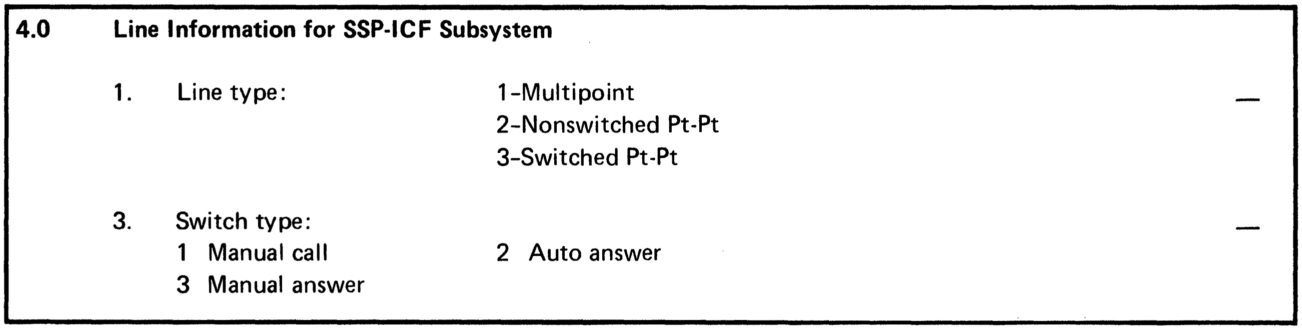
The size of the common queue space plus the total subsystem queue space of all the enabled BSCEL subsystems cannot exceed 42 K bytes. The subsystem queue space can be specified as 0 K bytes if these requirements are added to the common queue space.

The default subsystem queue space size is 4 K bytes.

Subsystem support swappable: Specify whether you want the subsystem to be swappable. Consider the total system performance, the size of the subsystem, and the amount of user main storage when determining whether you want the subsystem to be swappable. The BSCEL subsystem requires 10 K bytes of main storage.

Maximum user record length: Specify the maximum record length (1 through 4075 bytes) to be sent or received by any System/34 application program using this subsystem configuration. For sessions using partner-norm, the maximum user record length must be at least 90 bytes. The default is 1024 bytes.

Display 4.0 Line Information for SSP-ICF Subsystem

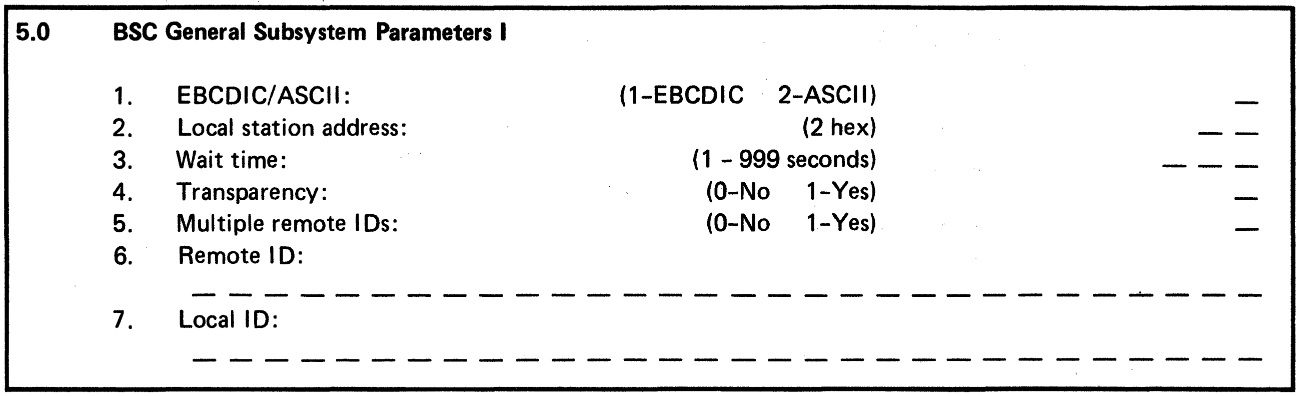


```
4.0     Line Information for SSP-ICF Subsystem

        1.   Line type:            1-Multipoint                          _
                                   2-Nonswitched Pt-Pt
                                   3-Switched Pt-Pt

        3.   Switch type:                                                _
             1  Manual call        2  Auto answer
             3  Manual answer
```

Line type: Specify the line type that is suitable to your communications environment. The default is 3 (switched point-to-point).

Switch type: This prompt is displayed only if a switched point-to-point line type was selected. Specify the switch type you want for the line. The default is 1 (manual call). If you are using autocall, you should specify 1 (manual call). If you are using the switched X.21 feature, specify 2 (auto answer). See *Using Switched Lines* in this section for additional information.

Display 5.0 BSC General Subsystem Parameters I



```
5.0     BSC General Subsystem Parameters I

        1.   EBCDIC/ASCII:                (1-EBCDIC  2-ASCII)        _
        2.   Local station address:                (2 hex)          _ _
        3.   Wait time:                       (1 - 999 seconds)    _ _ _
        4.   Transparency:                      (0-No   1-Yes)       _
        5.   Multiple remote IDs:               (0-No   1-Yes)       _
        6.   Remote ID:
             _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
        7.   Local ID:
             _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
```

EBCDIC/ASCII: Specify which code you require. The code you select must be compatible with the remote system. ASCII cannot be selected if you specify transparency. The default is 1 (EBCDIC).

Local station address: Specify, in hexadecimal, the System/34 multipoint line address identified for this configuration. You can specify B (C2) through R (D9) for EBCDIC, or A (41) through Z (5A) for ASCII. This prompt is displayed only if a multipoint line was selected. Be sure to enter the hexadecimal equivalent in the code (EBCDIC or ASCII) used.

Wait time: Specify the number of seconds to wait without line activity before a permanent error is indicated. The default is 999, which indicates an infinite wait time.

Transparency: Specify whether the EBCDIC data will be transmitted in transparent mode. If you specify 1 (yes) with ASCII data, blank compression, or record separators, an error message is displayed. If you specify 1 (yes) for ITB mode, the application program can only receive. The System/34 cannot transmit ITB characters in transparent mode. The default is 0 (no).

Multiple remote IDs: Specify whether more than one remote switched line ID is used. The default is 0 (no). If 1 (yes) is specified, the DEFINEID procedure is executed after normal completion of the CNFIGICF procedure. This prompt is displayed only if a switched line was selected.

Remote ID: Specify from 0 to 30 hexadecimal characters to be used for identification of a remote device. The number of hexadecimal characters must be a multiple of 2. The field is left-justified with all unused positions filled with zeros. The hexadecimal characters cannot include any BSC control characters. Therefore, the following hexadecimal characters cannot be used:

ASCII (hex)	**EBCDIC (hex)**
01	01
02	02
03	03
04	10
05	1F
10	26
15	2D
16	32
17	37
1F	3D
80 through FF	

If multiple remote IDs are selected, a value for remote ID need not be specified. Multiple remote IDs are specified via the DEFINEID procedure. This ID must be the same as the local ID of the remote BSCEL system. This prompt is displayed only if a switched line was selected.

Local ID: Specify from 0 to 30 hexadecimal characters to be used for the local switched line identification. The number of hexadecimal characters must be a multiple of 2. The field is left-justified with all unused positions filled with zeros. The hexadecimal characters cannot include any BSC control characters. Therefore, the characters shown in the chart in the *Remote ID* description cannot be used. The local ID must be the same as the remote ID of the remote BSCEL system. This prompt is displayed only if a switched line was selected.

Display 5.1 BSC General Subsystem Parameters II

In the following prompt, parameters 1 and 2 are displayed only if you specified a line type of *switched pt-pt* on display 4.0.

```
5.1      BSC General Subsystem Parameters II

         1.   Phone list name:                                                   _ _ _ _ _ _ _ _
         2.   Refresh:                              (0-No   1-Yes)                             _
         3.   Block length:                             (0–4075)                         _ _ _ _
         4.   Record separator:                     (Hexadecimal)                            _ _
         5.   ITB mode:                             (0-No   1-Yes)                             _
         6.   Blank:              (0-No, 1-Compression, 2-Truncation)                          _
         7.   3740 Multiple files:                  (0-No   1-Yes)                             _
```

Phone list name: Specify the name of the load member that contains the list of phone numbers for the autocall feature or the list of numbers for the public data network (X.21 feature) to be called by the System/34. The load member must be in either #LIBRARY or in the same library as the configuration member.

Use the *refresh* parameter to define how you want the list of numbers processed.

Refresh: Specify 1 (yes) if you want the list reinitialized (calling to begin with the first number on the list) after a successful call. If you enter a list name and do not enter a value for refresh, refresh-yes is assumed.

Specify 0 (no) if you do not want the list reinitialized after a successful call.

Note: See *Automatic Calling on Switched Lines* in this chapter for more information about the phone list and refresh parameters.

Block length: The length of the block of data records that is transmitted or received. If you specify a block length, it must be at least as long as one record. If you do not want blocking, enter a 0. The default is 0.

The block length and the maximum user record length are used to calculate the size of the BSCEL subsystem input/output buffers. The buffer size cannot exceed 4096 bytes or the capacity of the remote system. Use the following formula to determine the buffer size:

Buffer size = (maximum user record length * F) + N + 21

where:
F = The number of records per block (1 if blocking is not used)
N = The number of bytes needed for ITB or record separator characters as follows:

N = F if record separators are used
N = F-1 if ITB characters are used in nontransparent mode
N = (F-1) * 3 if ITB characters are used in transparent mode
N = 0 if blocking is not used

Record separator: The record separator character (in hexadecimal) used to separate records when the records are blocked. You can enter any EBCDIC or ASCII character except the following:

ASCII (hex)	**EBCDIC (hex)**
01	01
02	02
03	03
04	10
05	1F
10	26
15	2D
16	32
17	37
1F	3D
80 through FF	

If you do not want a record separator character, enter hexadecimal 00. The default is hexadecimal 00.

Notes About Record Separators:

- If the data records have variable lengths, you should specify a record separator character.
- If you specify blank compression or blank truncation and you do not specify a record separator character, a hexadecimal 1E record separator character is used.
- If you specify record separators, you must also specify a block length.
- Record separators *cannot* be used with ITB or transparency mode.

ITB mode: Specify 1 (yes) if you want an ITB character inserted after each record in a block. The ITB character causes error checking at the end of each record instead of at the end of the block.

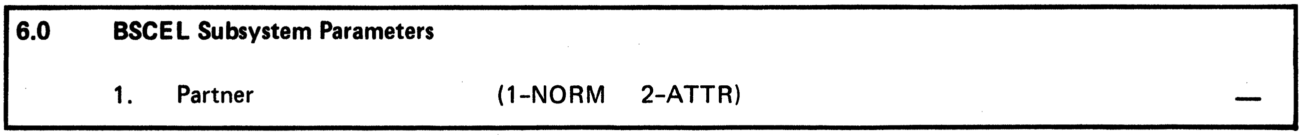
Do not specify ITB mode if you use blank truncation, blank compression, record separators, or 3740 multiple files. You cannot transmit data in transparent mode if you select ITB mode. A block length must be specified if you select ITB mode.

Blank: Specify whether or not you want blank truncation or blank compression. Specify 2 (truncation) if you want trailing blanks truncated from each data record. Specify 1 (compression) if you want each series of three or more blanks deleted from each record. The blanks are replaced when the data is received.

Do not specify compression with transparency or ITB mode. Do not specify truncation with ITB mode. If you specify either compression or truncation, you must also specify a block length. The default is 0 (no truncation or compression).

3740 Multiple Files: Specify 1 (yes) if you are going to transmit and/or receive 3740 multiple files. See *3740 Multiple Files* in this section for additional information. Do not specify ITB mode if you specify 1 (yes) for 3740 multiple files. The default is 0 (no).

Display 6.0 BSCEL Subsystem Parameters



```
6.0     BSCEL Subsystem Parameters

        1.   Partner                  (1-NORM   2-ATTR)                    _
```

Partner: Specify the type of session you want with the remote system. A partner attribute of NORM indicates that the remote system can recognize the BSCEL transaction initiation commands, transaction termination commands, and online messages. In most cases, NORM indicates that the remote system is another System/34 with the BSCEL subsystem. A partner attribute of ATTR indicates that the remote system cannot recognize the NORM commands or messages. In most cases, ATTR is used when communicating with remote systems such as a 3741 Data Entry Station or a 5230 Data Collection System.

STARTING AND ENDING THE BSCEL SUBSYSTEM

The ENABLE procedure is the means of starting the BSCEL subsystem on the System/34. The ENABLE procedure associates the subsystem with a particular BSC line. After being enabled, the subsystem monitors the line and waits for application program requests.

The DISABLE procedure stops the subsystem. When a disable is performed, the association between the subsystem and the BSC line is broken, and no further activity occurs on the line. If the subsystem was active on a multipoint line using the multiline communications adapter, the adapter will continue to respond negatively if polled or selected after the disable has been completed.

The formats of the ENABLE and DISABLE procedure commands are in Chapter 2.

STARTING BSCEL SUBSYSTEM APPLICATIONS

System/34 BSCEL subsystem applications can be started by a display station operator entering a procedure command or by a request from the remote system. Procedures that are started by a System/34 operator must have a SESSION statement for each session to be started. Requests from the remote system to start a procedure must be in a special format. The following sections describe the SESSION statement and the incoming procedure start requests.

SESSION OCL Statement

The format of the SESSION statement for the BSCEL subsystem is:

$$\text{// SESSION LOCATION-name},\text{SYMID-session-id}$$

$$\left[,\text{RECL-nnnn}\right] \left[,\text{SWTYP-}\left\{\begin{matrix}\text{AA}\\ \text{MA}\\ \text{MC}\end{matrix}\right\}\right]$$

$$\left[,\text{TRANSP-}\left\{\begin{matrix}\text{YES}\\ \text{NO}\end{matrix}\right\}\right] \left[,\text{PARTNER-}\left\{\begin{matrix}\text{ATTR}\\ \text{NORM}\end{matrix}\right\}\right] \left[,\text{RECSEP-character}\right]$$

$$\left[,\text{BLANK-}\left\{\begin{matrix}\text{C}\\ \text{T}\\ \text{N}\end{matrix}\right\}\right] \left[,\text{PHONE-name}\right] \left[,\text{REFRESH-}\left\{\begin{matrix}\underline{\text{YES}}\\ \text{NO}\end{matrix}\right\}\right]$$

$$\left[,\text{RESTORE-}\left\{\begin{matrix}\text{YES}\\ \underline{\text{NO}}\end{matrix}\right\}\right] \left[,\text{BLKL-nnnn}\right] \left[,\text{ITB-}\left\{\begin{matrix}\text{YES}\\ \text{NO}\end{matrix}\right\}\right]$$

LOCATION: Specifies the location name associated with this session. The location name is defined during subsystem configuration, and refers to the name of the location with which communication is to take place.

SYMID: Specifies the symbolic ID of the session with which this SESSION statement is associated. The symbolic ID must be two characters, with the first character numeric (0 through 9) and the second character alphabetic (A through Z, #, $, or @). This is the same ID that the application program uses when referring to this session. This ID is the equivalent of the symbolic display station ID as specified on the WORKSTN OCL statement. This parameter has no default.

RECL: Specifies the maximum record length that will be transmitted or received for this session. The record length can be any value from 1 through 4075 bytes, however, if PARTNER-NORM is specified, RECL must be at least 90 bytes to accommodate online messages.

SWTYP: Specifies the method of making a connection on a switched line. MC indicates that the operator will initiate the call manually or that autocall is being used. If the system is configured or enabled for autocall and a phone list is specified, the system will dial the remote system automatically. See *Autocall Considerations* in this section for additional information. AA indicates that the System/34 will answer the call automatically. MA indicates that the operator will answer the call manually.

TRANSP: Specifies whether data will be transmitted in transparent mode. If you specify YES when using ASCII, blank compression, or record separators, an error message is displayed. If you specify YES for ITB mode, the application program can receive only. The System/34 cannot transmit ITB characters in transparent mode.

PARTNER: Specifies the partner attribute for this session. NORM indicates that the BSCEL messages and commands can be recognized by the remote system (usually another BSCEL system). ATTR indicates that these commands and messages cannot be handled by the remote system (usually a device such as a 3741).

RECSEP: Specifies the record separator character (in hexadecimal) used to separate data records when the records are blocked. You can enter any ASCII or EBCDIC character except the following:

ASCII (hex)	**EBCDIC (hex)**
01	01
02	02
03	03
04	10
05	1F
10	26
15	2D
16	32
17	37
1F	3D
7F through FF	

BLANK: Specifies blank truncation or blank compression.

Specify T if you want trailing blanks truncated from each data record. Specify C if you want each series of three or more blanks deleted from each record. The blanks are replaced when the data is received. Specify N if you want neither blank truncation nor blank compression.

PHONE: Specifies the name of the load member that contains the list of phone numbers for the autocall feature or the list of numbers for the public data network (X.21 feature) to be called by the System/34. The load member must be in either the current user library or in #LIBRARY. If you do not specify a list name, the name specified during subsystem configuration or the name specified when the subsystem is enabled is used.

Use the *REFRESH* parameter and the *RESTORE* parameter to define how you want the list of numbers processed.

REFRESH: Specifies whether you want the list reinitialized (calling to begin with the first number on the list) after a successful call. If you enter a list name and do not enter a value for REFRESH, REFRESH-YES is assumed.

Specify NO if you do not want the list reinitialized after a successful call.

RESTORE: Specifies whether you want the list used in a previous job step reinitialized before executing the current job step.

Specify NO if you do not want the list used in a previous job step reinitialized.

If you specify a list and do not specify RESTORE, RESTORE-NO is assumed.

Note: See *Automatic Calling on Switched Lines* in this chapter for more information about the PHONE, REFRESH, and RESTORE parameters.

BLKL: Specifies the length (1-4075 bytes) of the block of data records to be transmitted or received. If you specify 0, data records are not blocked.

ITB: Specifies ITB mode. Specify YES if you want ITB characters to be inserted between data records when the data is blocked. The ITB character causes error checking after each record instead of at the end of each block.

Note: See *Data Formats* in this section for additional information about record separators, blank truncation, blank compression, and ITB characters.

If a value for RECL, SWTYP, TRANSP, PARTNER, BLKL, RECSEP, ITB, or BLANK is not specified, the value defined during subsystem configuration or enable is used.

Incoming Procedure Start Requests

For remote systems to initiate programs on the System/34, the remote system must transmit an *EXEC, *EXEX, *EXNC, or *EXNX procedure start request. The format of these requests is in Chapter 2. If the remote system is another System/34 with BSCEL, the BSCEL subsystem formats and transmits the appropriate procedure start request when an evoke operation is issued.

A session started from the remote system via a procedure start request does not have a partner attribute. The format of the procedure start request defines whether BSCEL commands and online messages are transmitted. *EXEC and *EXEX implies that the remote system can recognize BSCEL commands and online messages; *EXNC and *EXNX implies that the remote system cannot recognize BSCEL commands and online messages. BSCEL commands and online messages are not issued for *EXNC or *EXNX procedure start requests.

If one of the following parameters: maximum user record length, block length, record separator, ITB mode, blank, and/or transparency is specified with the *EXEC or *EXNC procedure start request, the parameter with the procedure start request is used instead of the parameter specified during subsystem configuration or the enable procedure.

The BSCEL subsystem can conduct only one session per line. If a procedure start request is received from the remote system while the session is active, the procedure cannot be started. The BSCEL subsystem abnormally terminates the transmission and displays SYS-8131 on the system console. The active session cannot receive or send any more data. If the application program terminates the session normally, the BSCEL subsystem will terminate the session normally. If the application program attempts to transmit or receive more data, the BSCEL subsystem will abnormally terminate the session and issue a return code (819D) to the application program to indicate that unexpected data was received.

OPERATION CONSIDERATIONS

The following sections describe the operations supported by the BSCEL subsystem. A complete chart of all interactive communications operations and the subsystems that support them is in each language chapter. The chart also shows the keyword or format name used to code the operation. More information about how an operation is coded is in the appropriate programming language chapter.

Whether an operation completes successfully or unsuccessfully, a return code is given to the application program. All of the return codes that are valid for this subsystem are described at the end of this chapter. A summary chart in Appendix A lists all the return codes and the subsystems for which they are valid.

Acquire Operation

The acquire operation is issued by the application program to establish a session. Acquire performs the following functions:

- Verifies that the requested line is available.

- Reserves a line between the System/34 and the remote system. No other program can acquire the line until the session terminates. If PARTNER-NORM is specified and the remote system is a System/34 using BSCEL, the line is also reserved on the remote system. No remote program can acquire the line until the session terminates.

- Initializes the BSCEL subsystem with the parameters specified on the SESSION OCL statement. Any parameter specified on the SESSION statement overrides any corresponding parameter defined during subsystem configuration or the ENABLE procedure.

- Allocates resources for use during the session; these include control blocks and line buffers.

Evoke Operation

The evoke ($$EVOKNI), evoke then invite ($$EVOK), and evoke then get (assembler only) operations initiate a transaction. The output length of any evoke operation cannot exceed 120 bytes. If the evoke includes a procedure name, the actual output length is the length of the data plus the length of the procedure name plus 1. Depending on the partner attribute, the evoke operation is interpreted in one of two ways.

If the partner attribute is NORM, the evoke operation sends a 160-byte procedure start request to the remote system. This procedure start request contains the parameters to initialize the procedure on the remote system and to define the data format to be used for this session, such as the maximum user record length and the block length. If the remote system is another System/34 that is also using the BSCEL subsystem, the procedure start request parameters that define the data format override the corresponding configuration parameters at the remote System/34. Thus, the remote application program uses the same data format as the local application program.

The remote system processes the procedure start request and initiates an application program. The BSCEL subsystem waits for a message response to the procedure start request. If a successful program start message is received, the remote application is considered to be running, and the transaction is initiated. If an error message is received in response to the procedure start request, a return code (831A) is passed to the application program indicating that the evoke operation failed and a message is waiting. The application program can optionally issue an input operation to obtain the message. Following the input operation, another evoke operation must be issued before the transaction can be established.

If the partner attribute is ATTR, the evoke operation initiates the transaction. This form of the evoke assumes that the remote application is already active; that is, ready to exchange data with the local program. No line transmission occurs as a result of this evoke operation, therefore, no data should accompany the evoke. The next input or output operation following this evoke initiates the transfer of data related to this transaction.

The evoke end of transaction operation ($$EVOKET) initiates a remote procedure that does not communicate with the System/34 application program. When this operation is issued, an *EXEX procedure start request is transmitted to the remote system. The evoke end of transaction is not completed until a response is received from the remote system. If a successful program start message is received, a normal return code is given to the application program. If an error message is received, the application program receives a return code (831A) indicating that the evoke failed and a message is waiting. The application program can optionally issue an input operation to receive the message. Evoke end of transaction performs no function for sessions with a partner attribute of ATTR.

Put Operations

The put operation is used to pass data from the System/34 application program to the remote system. The put operation can be issued alone ($$SENDNI) or combined with an input operation ($$SEND).

If the put operation is combined with an input operation, the data followed by an EOT (end of transmission), is sent. (The data is followed by a null record when transmitting 3740 multiple files.) Then data is received by the subsystem. In the case of a BASIC, COBOL, or RPG II program, the input data is made available to the application program on the next accept (READ) operation. Assembler users can choose to wait for the input data by using a put then get operation. If the put then get operation fails, the minor part of the return code indicates whether the error occurred while the system was receiving or transmitting.

If the put operation is not combined with an input operation, and *blocking is not used*, one data record is sent for each put operation. If the put operation is not combined with an input operation and *blocking is used*, data records are blocked before being transmitted. When multiple put operations ($$SENDNI) are issued sequentially, the application program is normally two or more puts ahead of the communications line. If a communications line failure occurs while transmitting, the application program is notified by a return code on its current put operation, which is two or more puts ahead of the failing operation. Therefore, if a line failure occurs, the application program can not determine if all data was transmitted before the error occurred. The put end of file operation ($$SENDE) is used to terminate a series of put operations that do not request input. The associated data record is sent followed by an EOT, or a null record, if 3740 multiple files are being transmitted.

Also available is the put end of transaction operation ($$SENDET). The put end of transaction operation is issued by the locally initiated application program to terminate the transaction or by the remotely initiated program to terminate the session. Because only one transaction can be active at a time, the active transaction must end before a new transaction can be established. A transaction is considered terminated when an end of transaction is successfully issued. If 3740 multiple files are being transmitted, see *3740 Multiple Files* in this section for additional information.

Input Operations

The input operations for the BSCEL subsystem are invite, get, and accept. The invite operation can be issued only as a combined operation with a put, request change direction, or evoke operation ($$SEND, $$RCD, $$EVOK) in BASIC, COBOL, or RPG II. Assembler language users can issue an invite operation explicitly. A get or invite operation signals the subsystem to obtain data from a particular session for the application program. A get operation causes the application program to wait for the data to be available. When a program issues an invite operation, it receives the data with the next accept operation. The accept operation allows input from any previously invited session.

The end of file indication (EOT) is not received by the BSCEL subsystem with the input data. Therefore, the return code that indicates end of file is 0300 (no data received and EOT received). If 3740 multiple files are being received, see *3740 Multiple Files* in this section for additional information.

The end of transaction indicator is received by the BSCEL subsystem with or without input data. If the end of transaction is received with data, the return code is 0008 (data and end of transaction). If the end of transaction is received with a message, the return code is 0028 (message and end of transaction). If the end of transaction is received without data, the return code is 0308 (no data and end of transaction).

Request to Change Direction Operation

The BSCEL subsystem allows a request to change direction operation ($$RCD) only during a transaction and only when the issuing program is not transmitting data. The request to change direction operation results in a reverse interrupt (RVI) being sent to the remote system as the response to the next data record received. Only one RVI is sent to the remote system before the change direction occurs; therefore only the first request to change direction operation before the change direction occurs has any effect. If the issuing program is not receiving and not transmitting, the operation has no effect.

Request to change direction is always accompanied by a get or invite operation. Regardless of whether the RVI is sent, the invite or get operation is performed.

Release Operation

The release operation is issued by the application program to terminate a session it acquired or by a remotely initiated MRT program to pass the session on to the next job step.

When issued by the program that acquired the session, the release operation performs the following functions:

- Terminates the session; the line can then be acquired for another session.
- If the partner attribute is NORM and the remote system is using the BSCEL subsystem, releases the line on the remote system. This allows a remote application program to acquire a session on the line.
- If this line is a switched line, disconnects the line.
- Frees the resources that were used during the session.

End of Session Operation

The end of session operation ($$EOS) always results in a normal completion return code. The session is always terminated by the end of session operation. If the session is still communicating when the end of session operation is issued, the transmission is abnormally terminated by the BSCEL subsystem, and abnormal termination of the BSCEL application program could result.

Get Attributes Operation

The get attributes operation (assembler only) can be issued at any time to determine the status of a session.

Set Timer Operation

The set timer operation ($$TIMER) results in a timer expired return code (0310) after a specific time interval in hours, minutes, and seconds has expired.

PROGRAMMING CONSIDERATIONS

Online Messages

The BSCEL subsystem supports the receipt and optional transmission of online messages. The online messages inform the subsystem and application program of key events.

The BSCEL subsystem transmits an online message after receiving an *EXEC or *EXEX procedure start request. The message informs the remote system of the success or failure of the procedure start request.

If the partner attribute of the session is NORM, or if the session was started by an *EXEC request, the BSCEL subsystem transmits an online message when either of the following occurs:

- The application program abnormally terminates. The message informs the remote system of the abnormal termination.

- A DISABLE command abnormally terminates the session. The message informs the remote system of the disable.

An online message generated by the BSCEL subsystem is 90 bytes long and in the following format:

ICFxƀƀSYS-nnnnƀmessage-text

where *x* is an M (for an informational message) or an E (for an error message). The *nnnn* is the message identification code of the message that was issued on the local system. Note that if the message was not a system message, the SYS would be replaced by the appropriate program product prefix, such as RPG or CBL.

The BSCEL subsystem checks any incoming data record for an ICFM or ICFE in the first four bytes of the record. The first 14 characters of each of these messages is logged to the history file as SYS-8170. If a session is active when the message is received, the application program must issue an input operation to receive the message. If the next operation is not an input operation, the operation is rejected with a message waiting return code. If no session is active when the message is received, the first 14 characters are displayed at the system console as SYS-8170.

The ICFM message (SYS-8190) received as a result of the evoke operation (in response to a procedure start request) is not logged to the history file and cannot be received by the application program. This message is used by the BSCEL subsystem to indicate that the procedure start request was successful.

An ICFE message received as a result of an evoke operation (in response to a procedure start request) can optionally be received by the application program. The application program receives a return code (831A) indicating that the evoke operation failed and a message is waiting. The application program can issue an input operation to receive the message, issue another evoke operation, or terminate the session.

If an ICFE message is received while a transaction is active, the application program receives a return code (0028) indicating that a message and end of transaction were received. If the session is an acquired session, another evoke is necessary to begin the transaction, or the session can be terminated. If the session was started by an incoming procedure start request, the next application program operation must be an end of session operation, or the program can terminate.

The BSCEL subsystem also recognizes a status message from the remote system without an active session. This message is normally received from a device, such as a 3741, after a session has been abnormally terminated. The message has the following format:

```
S       S   E
O % x   T y T
H       X   X
```

SOH is the start of header BSC control character.

% is the character representation of the percent sign.

x is a single character identifying the type of the remote system.

STX is the start of text BSC control character.

y is one or more characters identifying the status of the remote system.

ETX is the end of text BSC control character.

The status message is not analyzed by the BSCEL subsystem, but is displayed on the system console as SYS-8170. The displayed message contains the text %xƀy.

Data Formats

The BSCEL subsystem supports the following data formats:

- Unblocked data records in either transparent or nontransparent mode.
- Blocked data records in either transparent or nontransparent mode.
- Blocked data records with record separators in nontransparent mode.
- Blocked data records with ITB characters in nontransparent mode. (ITB characters can be received in transparent mode, but not transmitted.)
- Blocked data records with blank compression in nontransparent mode.
- Blocked data records with blank truncation in either transparent or nontransparent mode.
- 3740 multiple files.

You select the above formats during subsystem configuration, with the ENABLE procedure, with the SESSION statement, and/or with the procedure start request. Both the transmitting and the receiving station must use the same data format. Following is a description of each of the data formats.

Unblocked Data Records

If you do not specify a block length (entry of 0 for the block length), the BSCEL subsystem transmits and receives unblocked data records. That is, each put operation causes one record to be transmitted, and each get operation causes one record to be received. Use this format for interactive applications that transmit and receive fixed- or variable-length data records.

Blocked Data Records

If you specify a block length, the BSCEL subsystem places data records in blocks before they are transmitted and removes data records from blocks when they are received. As many records as possible are placed in a block depending upon the following:

- Block length
- Maximum user record length
- Whether blank truncation or blank compression is specified
- Whether the records are fixed or variable lengths

If the records are fixed-length, you should specify a maximum user record length that is equal to the fixed length of a record and a block length that is a multiple of the record length. For example, if the record length is 128 bytes, a block length of 512 bytes would allow four records to be placed in a block before the block is transmitted.

If the records are variable-length, you should specify a maximum user record length that is equal to or greater than the longest record you expect to transmit or receive. Also, specify a block length that is large enough to contain the minimum number of records you want to transmit or expect to receive in one block. For example, if the longest record is 128 bytes, a block length of 512 would allow a minimum of four records to be placed in a block. Variable-length records are separated in a block by record separator characters called IRS (interrecord separator) characters or by ITB (intermediate text block) characters. The character used depends upon which of the following modes you specify:

- Record separators
- ITB mode
- Blank compression
- Blank truncation

If data is transmitted or received without record separators, ITB characters, blank compression, or blank truncation, the data records must be fixed-length.

Blocked Record Formats

Fixed-Length Records: The following example shows how data is blocked with fixed-length records for both nontransparent and transparent modes:

Nontransparent Mode

```
STX Record 1 Record 2 Record 3 . . . Record n ETB
```

OR

Transparent Mode

```
DLE STX Record 1 Record 2 Record 3 . . . Record n DLE ETB
```

Use this format for batch applications that transmit and receive fixed-length data records. To configure the BSCEL subsystem to block fixed-length records, specify the following:

- A block length large enough to contain the number of records you expect to be transmitted or received in one block
- A record separator of hexadecimal 00
- No for ITB mode
- No for blank truncation or compression

Blocked Data with Record Separators: When data is transmitted or received with record separators, a record separator character (IRS) is used to separate records in a block as shown in the following example:

```
STX Record 1 IRS Record 2 IRS Record 3 IRS ... Record n IRS ETB
```

Use this format for batch applications that transmit and receive large numbers of fixed or variable length records. To configure the BSCEL subsystem for this format, specify the following:

- A maximum user record length equal to or greater than the longest record to be transmitted or received
- A block length large enough to contain the number of records you want to transmit or expect to receive in one block
- A record separator character other than hexadecimal 00
- No for ITB mode
- No for blank truncation or compression

Blocked Data Records with ITB Mode: When data is transmitted or received with ITB mode, an ITB character is used to separate records in a block as shown in the following example:

Nontransparent Data

```
STX Record 1 ITB Record 2 ITB Record 3 ITB ... Record n ETB
```

OR

Transparent Data (receive only)

```
DLE STX Record 1 DLE ITB DLE STX Record 2 DLE ITB DLE STX ... Record n DLE ETB
```

Normally, data is checked for errors after each data block is received. However, when you use ITB mode, the ITB character causes the data to be checked at the end of each record rather than at the end of the block. Therefore, you can use this format to transmit or receive large numbers of fixed- or variable-length records when additional error checking is required. To configure the BSCEL subsystem for this format, specify the following:

- A maximum user record length that is equal to or greater than the longest record you expect to transmit or receive
- A block length large enough to contain the number of records you want to transmit or expect to receive in one block
- A record separator character of hexadecimal 00
- Yes for ITB mode
- No for blank truncation or blank compression

Blocked Data with Blank Compression: When transmitting data with blank compression, any series of three or more blanks within the data is replaced by an IGS (interchange group separator) character followed by a blank-count byte. (The System/34 uses the same blank compression format as the IBM 3780.) The blank count byte indicates the number of blanks deleted (compressed). Up to 63 blanks can be deleted by one IGS character and one blank-count byte. If more than 63 bytes are deleted, two or more IGS and blank-count byte pairs are inserted. When data with IGS characters is received, the subsystem replaces the blanks that were removed at the transmitting station as indicated by the IGS character and blank-count byte. The following example shows the format of data records with blank compression specified. The character (B) is the blank-count byte:

```
STX Record 1 IGS(B) Record 1 IGS(B) Record 1 IRS Record 2 IGS(B)
IGS(B) Record 2 IRS . . . Record n IGS(B) Record n IGS(B) IRS ETB
```

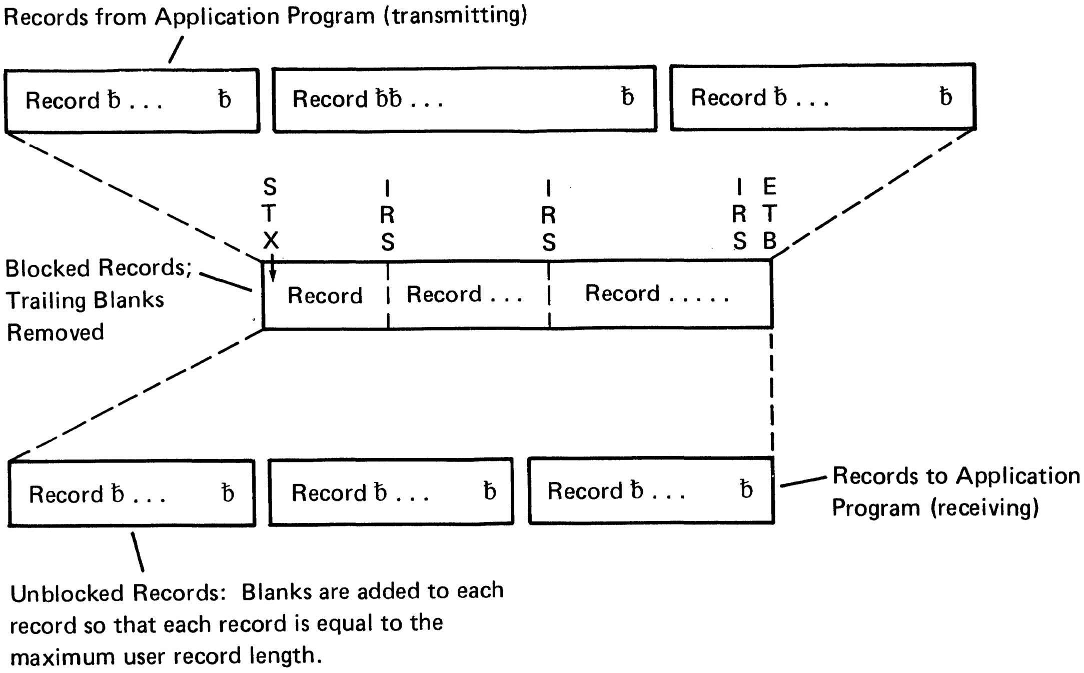
The record separator character (IRS) is used to separate records in the block. If you did not specify a record separator character, a default separator of hexadecimal 1E is used.

You can use this format for batch applications that transmit and receive fixed- or variable-length data records that contain many blanks. To configure the BSCEL subsystem for this format, specify the following:

- A record length that is equal to or greater than the longest record you expect to transmit or receive
- A block length large enough to contain the number of records you want to transmit or receive in one block
- No for ITB mode
- Yes for blank compression

Blocked Data Records with Blank Truncation: When data is transmitted with blank truncation, trailing blanks are removed from each record. The record separator character you specified or the default record separator (hexadecimal 1E) is inserted after each record. When data records are received, blanks are inserted at the end of each record. The number of blanks inserted depends upon the maximum user record length. That is, the length of a record passed to the application program is equal to the maximum user record length you specified. If blank truncation is specified for transparent mode, blanks are not truncated and record separators are not inserted. The data format used to transmit the records is the same format that is used for blocked fixed-length records in transparent mode.

The following example shows how blanks are removed and inserted:


Records from Application Program (transmitting)
Record ƀ . . . ƀ
Record ƀƀ . . . ƀ
Record ƀ . . . ƀ
STX
IRS
IRS
IRS
ETB
Blocked Records; Trailing Blanks Removed
Record
Record . . .
Record . . . . .
Record ƀ . . . ƀ
Record ƀ . . . ƀ
Record ƀ . . . ƀ
Records to Application Program (receiving)
Unblocked Records: Blanks are added to each record so that each record is equal to the maximum user record length.

Use blank truncation for batch applications that transmit and receive data records that contain many trailing blanks. To configure the BSCEL subsystem for blank truncation, specify the following:

- A maximum user record length equal to or greater than the longest record you expect to be transmitted or received
- A block length large enough to contain the number of records you expect to transmit or receive in one block
- No for ITB mode
- Yes for blank truncation

Receiving Null Records

When your program receives null records (STX ETX), return code 0301 (no data and continue to receive) indicates that a null record has been received from the remote system. Your program must issue an input operation for each null record received.

3740 Multiple Files

When you specify 3740 multiple files, the BSCEL subsystem can transmit and receive multiple files in 3740 format. In 3740 format, multiple files are transmitted during one transaction by transmitting a null record (STX ETX) after each file to indicate the end of one file and the beginning of the next file.

In an application program, to indicate the end of a file (cause the BSCEL subsystem to transmit a null record), use the *put end of file* operation. Then begin transmitting the next file. To indicate the end of the last file and begin receiving data, issue an *input* operation. This causes the subsystem to transmit a null record followed by an EOT, which allows the remote system to begin transmitting. To indicate the end of the last file and terminate the session, issue a *put end of transaction* operation.

To configure the BSCEL subsystem for 3740 multiple files, select YES on the prompt during subsystem configuration or specify YES in the ENABLE procedure.

When 3740 multiple files are received, the return code 0301 (no data and continue to receive) indicates the end of the logical file. Return code 0301 is returned each time a null record is received. Return code 0300 (no data and an EOT was received) indicates that all of the files have been received.

Using Switched Lines

To connect the System/34 to a remote location on switched lines, you can do the following:

- Call a remote location manually
- Answer a call from a remote location manually
- Call a remote location automatically (if you have the autocall or switched X.21 feature)
- Answer a call from a remote location automatically (if you have the auto answer or switched X.21 feature)

Note: If you have the switched X.21 feature, the System/34 calls and answers automatically. You cannot call or answer manually with the switched X.21 feature.

After the operator enables the BSCEL subsystem and before a session is started, the System/34 monitors the switched line for incoming calls. If you specified a partner attribute of NORM on the SESSION OCL statement or on display 6.0 of the CNFIGICF procedure, the line is connected (or a message is displayed asking the operator to connect the line) when the program issues the acquire operation. If you specified a partner attribute of ATTR, the line is connected (or the operator is asked to connect the line) when your program first sends or receives.

If the System/34 called the remote system, the switched line is disconnected when your program ends the session or when the remote system sends disconnect. If the remote system called the System/34, the line is disconnected when the remote system sends disconnect.

Manual Calling and Manual Answering on Switched Lines

If you want the operator to call a remote location manually, specify 1 (manual call) on display 4.0 of the CNFIGICF procedure and do not specify a list name for the *phone list name* parameter on display 5.1 or for the *PHONE* parameter on the SESSION statement. The message OPERATOR DIAL REQUIRED is displayed on the system console when the operator is to make the call.

If you want the operator to answer a call from a remote location manually, specify a 3 (manual answer) on display 4.0 and do not specify a list name for the *phone list name* parameter on display 5.1 or for the *PHONE* parameter on the SESSION statement. The message OPERATOR ANSWER REQUIRED is displayed on the system console when the operator is to answer the call.

Automatic Calling on Switched Lines

The System/34 can call remote locations automatically on switched lines using either the autocall or X.21 feature. To call a remote location automatically, you must do the following:

- Specify the correct configuration parameters.
- Create a list of phone numbers for the autocall unit or a list of numbers for the public data network.
- Enter the name of the list when you configure the subsystem, when you enable the subsystem, or when you write the SESSION statement.
- Enable the subsystem on an autocall or switched X.21 line.

Configuring the Subsystem for the Autocall Feature: When you use the autocall feature, the System/34 calls the remote location(s) automatically regardless of the switch type parameter you specified during subsystem configuration or enable. However, you should specify 1 (manual call), which allows the operator to enable the subsysystem on a manual call line if an autocall line is not available. A message is then displayed to ask the operator to dial the remote location manually.

Configuring the Subsystem for the X.21 Feature: When you use the switched X.21 feature, specify 2 (auto answer) for the switch type parameter on display 4.0 of the CNFIGICF procedure. The System/34 calls a location(s) automatically and answers calls automatically when you specify auto answer. You cannot call a location manually or answer a call manually with the X.21 feature.

Creating a Phone List for Autocall or a List of Numbers for the Public Data Network: The phone list or list for the public data network contains the number or numbers you want the System/34 to call. You create the list and store it in a load member using the DEFINEPN or DEFINX21 procedure. These procedures are described in the *System Support Reference Manual.*

If the list name is specified during subsystem configuration or when the subystem is enabled, the load member containing the list must be in either #LIBRARY or in the same library as the configuration member. If the list name is specified on the SESSION statement, the load member must be in either #LIBRARY or the current user library.

Following is an example of how the System/34 handles a list. Although the example shows a list for the autocall feature, the System/34 handles a list for a public data network (X.21 feature) in the same mannner except where noted.

Phone List Example

The System/34 calls the numbers in the order listed. When a number is called, the call might be unsuccessful. A retry count specified during the DEFINEPN procedure is associated with each number in the list. When a call is unsuccessful, the retry count is decremented by 1, and the next number on the list is called.

Note: If you are using the switched X.21 feature, the retry count is decremented depending upon the reason for the unsuccessful call. A call progress signal (CPS) gives the reason for the unsuccessful call. The call progress signal is displayed on the system console with the unsuccessfull call message. If the call progress signal begins with a 4, 5, or 7, the number is attempted only once, the retry count is set to 0, and the number is marked as unsuccessfully called.

Phone Number	**Retry Count**	
8672906	5 -1 = 4	(The retry count is decremented
6286500	1	and the next number is called.)
6280363	3	

This number will not be called again until the other numbers on the list have been called, or until the phone list is reinitialized.

If the retry count reaches 0, a message is displayed on the system console indicating that the call was unsuccessful, and the number is marked as unsuccessfully called.

Phone Number	**Retry Count**	
8672906	5 -1 = 4	
6286500	1 -1 = 0	(Unsuccessfully called)
6280363	3	

When a number is marked as unsuccessfully called, it is not called again until the phone list is reintialized.

If a number is successfully called, an operator message is displayed on the system console, the line connection is established, and the number is marked as successfully called.

Phone Number	**Retry Count**	
8672906	5 -1 = 4	
6286500	1 -1 = 0	(Unsuccessfully called)
6280363	3	(Successfully called)

The number is not called again until the list is reinitialized.

In the previous example, the first number in the list has not been marked as called. Therefore, if an application program uses the phone list without reinitializing it, the first number is the only number that can be called. If the call is successful, the number is marked as successfully called.

Phone Number	Retry Count
8672906	4 (Successfully called)
6286500	0 (Unsuccessfully called)
6280363	3 (Successfully called)

The phone list must now be reinitialized to be used again. If the list is not reinitialized and an application program attempts to use the list, a message PHONE LIST EXHAUSTED is displayed on the system console, and a return code of xx86 is returned to the application program.

In the previous example, if the call to the first number in the list is unsuccessful and the retry count reaches 0, the number is marked as unsuccessfully called, a message NO NUMBERS REACHED is displayed on the system console, and a return code of xx85 is returned to the application program. The phone list must be reinitialized to be used again.

Phone Number	Retry Count
8672906	0 (Unsuccessfully called)
6286500	0 (Unsuccessfully called)
6280363	3 (Successfully called)

When the list is reinitialized, all retry counts are set to the counts specified during the DEFINEPN procedure and calling begins with the first number in the list when the list is used again.

The specification of the REFRESH and RESTORE parameters indicates how the list is reinitialized. The REFRESH parameter can be specified during configuration, during ENABLE, or on the SESSION statement. The RESTORE parameter can be specified on the SESSION statement only. Following is a description of the REFRESH and RESTORE parameters.

REFRESH Parameter

If you specify REFRESH-YES or do not specify REFRESH (the default is YES), the list is reinitialized after the first successful call or after all numbers in the list have been marked as unsuccessfully called.

If you specify REFRESH-NO, the list is not reinitialized after a successful call. The list is reinitialized as follows:

- After the PHONE LIST EXHAUSTED message is displayed
- After the NO NUMBERS REACHED message is displayed
- As specified by the RESTORE parameter

RESTORE Parameter

If you specify RESTORE-YES on the SESSION statement, the phone list specified is reinitialized prior to executing the current step in the procedure. The current user library is searched first for the phone list. If the list is not found, the system library is searched.

If you specify RESTORE-NO, the list is not reinitialized prior to executing the current step in the procedure. The default is NO.

IF LISTDONE Conditional Expression

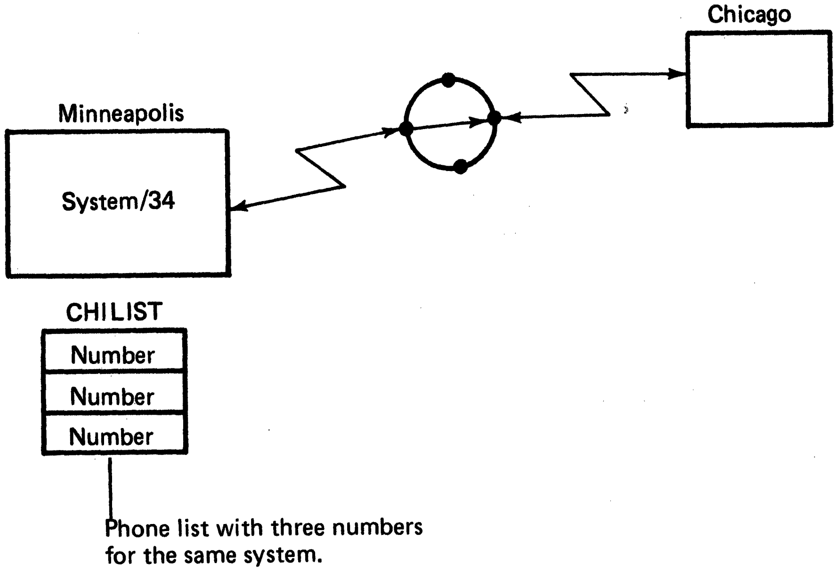
You can use the IF LISTDONE conditional expression to determine if all of the numbers in a phone list used by a previous job step have been called. If all numbers in the list are marked as successfully called or unsuccessfully called (retry counts are 0), the expression is true. The expression is false if REFRESH-YES was specified on the SESSION OCL statement and any number in the list was successfully called. The expression can only be used to test lists that have been specified on a SESSION statement. If the name of the list tested was not specified on the SESSION statement, the expression is false.

Note: In some countries a delay is required before a call can be placed when you use the autocall feature. If this delay is required in your country, you specify the delay when you create the phone list using the DEFINEPN procedure. See the *System Support Reference Manual* for information about the DEFINEPN procedure and the DELAY parameter.

The following examples show how you can use the *refresh* parameter, the *restore* parameter, and the *IF LISTDONE* statement.

Example of Calling One Location

In this example, the System/34 calls one location many times. A list name was entered for the *phone list name* parameter on display 5.1 of the CNFIGICF procedure. The list in this example contains three numbers for the same remote system in Chicago. When the remote system is called, it is not important which number is called successfully, but we want the System/34 to begin calling with the first number in the list each time the program uses the list. To do this, 1 (yes) was specified for the refresh parameter on display 5.1.



The list and setup in this example can be used for either batch or interactive communications with the remote system.

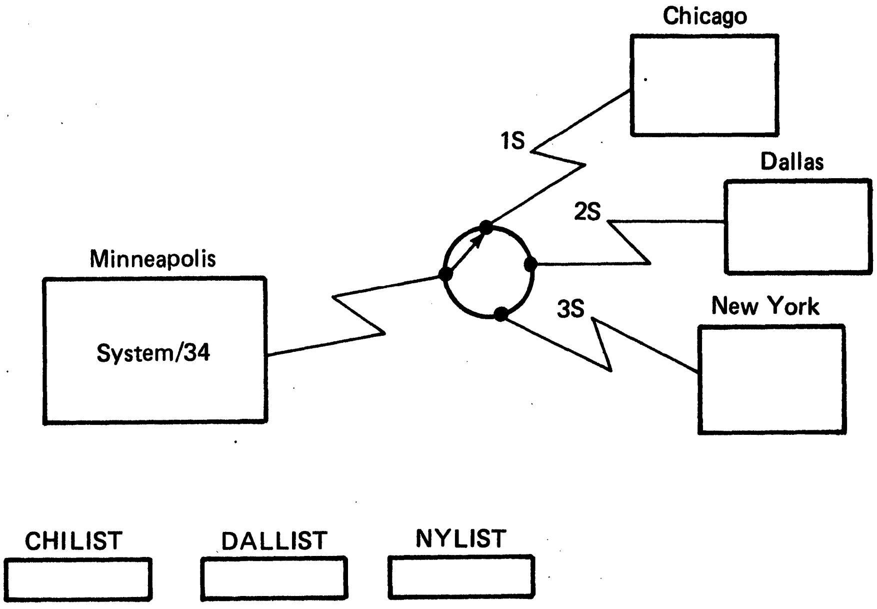
Example of Calling Multiple Locations (Refresh-Yes)

In this example, an MRT program is used for interactive communications with three remote systems.

One list was created for each remote system with one number in each list. The list names and refresh parameters are specified on three SESSION OCL statements as follows:

```
// LOAD MRTPROG
// SESSION LOCATION-CHICAGO,SYMID-1S,PHONE-CHILIST
// SESSION LOCATION-DALLAS,SYMID-2S,PHONE-DALLIST
// SESSION LOCATION-NEWYORK,SYMID-3S,PHONE-NYLIST
// RUN
```

For example, the program calls the system in Chicago (session 1S) using the list CHILIST. When the System/34 program processes the data from Chicago, it determines that additional data is needed. The program then calls Dallas (session 2S) using the list DALLIST to find the additional data. The data was not at the Dallas location, so the program calls New York (session 3S) using list NYLIST and gets the data it needs. After processing the data, the program calls Chicago again and sends the results to the Chicago system.

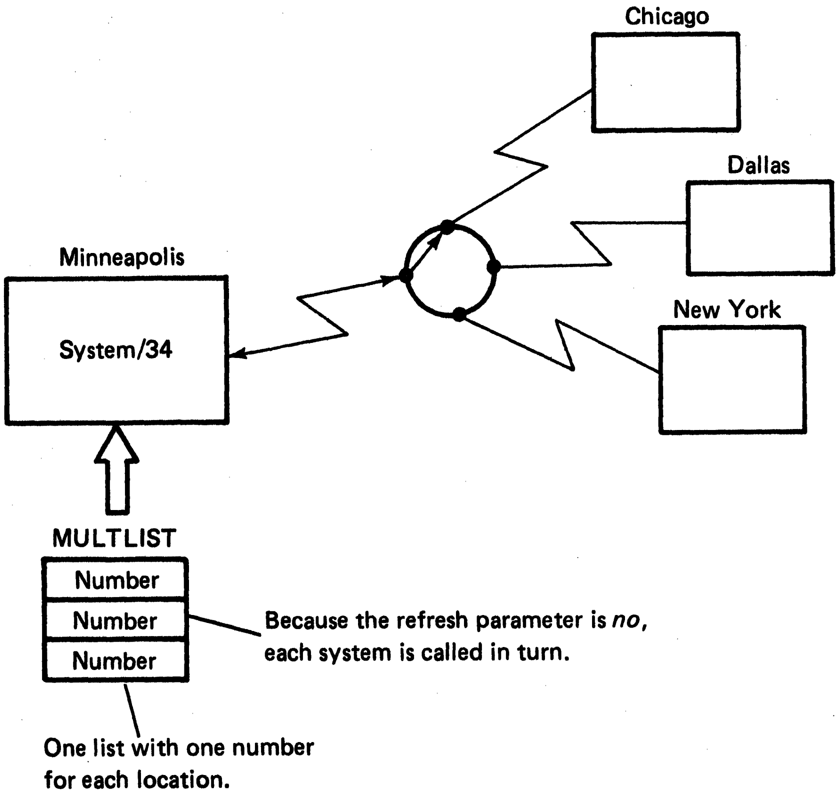


One list for each location.

Example of Calling Multiple Locations (Refresh-No)

In the two previous examples, the refresh parameter was set to *yes*. In this example, the parameter is set to *no*. One list is used, which was specified on display 5.1 of the CNFIGICF procedure. The refresh parameter was also set to 0 (no) on this display.

The program in this example loops through the list calling each system in turn to send one or more files of data to each system. The program checks for return code 8285 (NO NUMBERS REACHED) or 8286 (PHONE LIST EXHAUSTED) to determine when all numbers on the list have been called or tried.



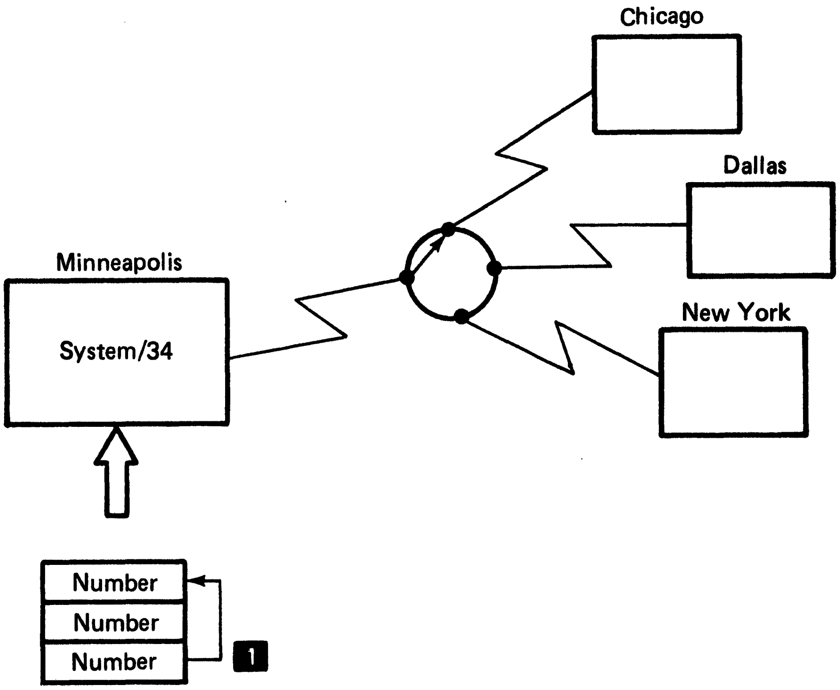
Because the refresh parameter is *no*, each system is called in turn.

One list with one number for each location.

Example of Calling Multiple Locations (RESTORE-YES)

In this example, the list is reinitialized by entering RESTORE-YES on the SESSION statement. Two programs (A and B) in one procedure use the same list to call the same remote system. If program A ends before all numbers have been called, the list is not reinitialized. To ensure that the list is reinitialized before program B uses the list, RESTORE-YES is entered on the SESSION statement. This ensures that the first number on the list is called when program B uses the list. For example:

```
// LOAD PROGA
// SESSION LOCATION-CHICAGO,SYMID-1S,
// PHONE-MULTLIST,REFRESH-NO
// RUN
// LOAD PROGB
// SESSION LOCATION-CHICAGO,SYMID-1S,
// PHONE-MULTLIST,REFRESH-NO,RESTORE-YES
// RUN                                 |
                                       1
```



The list is reinitialized before the next program uses it.

Example of Using the IF LISTDONE OCL Statement

You can also use the IF LISTDONE OCL statement to test the status of the list if your program does not test for a list done return code. This example shows how you can use the IF LISTDONE statement to check the list status.

Each time the program is loaded, it calls the next system, transmits or receives one or more files of data, and ends. If the list is done, the procedure also ends; otherwise, the program is loaded again and the next system is called.

```
// TAG TOP
// LOAD PROGA      1                             2           3
// SESSION LOCATION-MULTIPLE,SYMID-1S,PHONE-MULTLIST,REFRESH-NO
// RUN
// IFF LISTDONE-MULTLIST GOTO TOP

                4
```

1 Each location is called, one at a time.

2 The list contains one number for each location.

3 The parameter REFRESH-NO allows each number to be called.

4 If all numbers have not been called, go to TOP.

3740 Data Entry System Considerations

A procedure start request must be transmitted in unblocked records; therefore, to start a System/34 procedure from a 3741 with the expanded communications buffer feature, the first two files must contain one record in each file. The files following the procedure start request are data files. For example:

- File 1 on the 3741 contains the first record of the continuation procedure start request.

- File 2 contains the second record of the continuation procedure start request.

- Files 3 through n contain the data records to be transmitted (if any) to the System/34.

- Files n+1 to x are set up to receive data (if any) from the System/34.

BSCEL COMMANDS

The BSCEL subsystem transmits several commands as a result of various operations or conditions. The commands are transmitted only if the partner attribute is NORM or if the session was started by an *EXEC procedure start request. The System/34 application program that uses the BSCEL subsystem will not see these commands; however, the BSCEL subsystem checks all incoming data for these commands. The application programs should ensure that they do not send data that looks like one of these commands.

*ACQ Command

The *ACQ command is transmitted as a result of an acquire operation. If the BSCEL subsystem receives the *ACQ command, it indicates that the remote system has acquired the line. The format of this command is the four characters, *ACQ.

*REL Command

The *REL command is transmitted as a result of a release operation. If the BSCEL subsystem receives an *REL command, it indicates that the remote system has released the line. The format of this command is the four characters, *REL.

Procedure Start Request

The *EXEC procedure start request is transmitted as a result of an evoke operation. The *EXEX procedure start request is transmitted as a result of an evoke end of transaction operation. The BSCEL subsystem can receive the *EXEC and *EXEX requests as well as *EXNC and *EXNX requests. The formats of these procedure start requests is in Chapter 2.

*EOX Command

The *EOX command is transmitted as a result of a put end of transaction operation. If the BSCEL subsystem receives the *EOX command, it indicates that the remote system has terminated the transaction. If data is not included with the end of transaction, the format of the command is the four characters, *EOX.

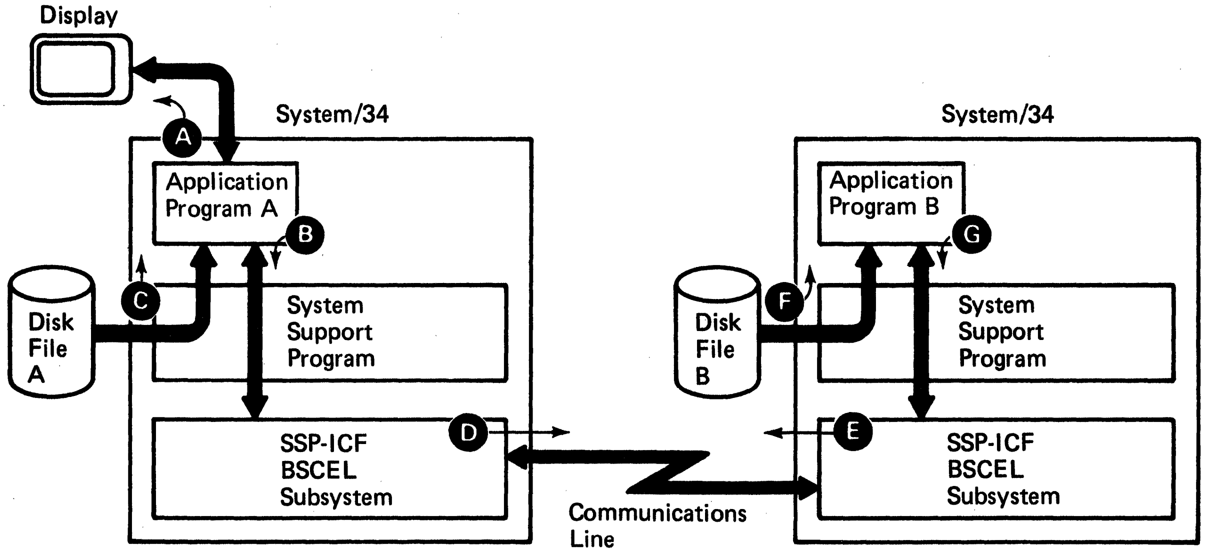
If data is included, the format depends on whether you are using blocking or no blocking as follows:

- If you are using blocking, the following occurs:
  - The data record is placed in the block.
  - The block is transmitted.
  - The *EOX command is transmitted.

- If you are not using blocking, the *EOX command is placed at the beginning of the data record before the record is transmitted.

How to Write Programs that Use the BSCEL Subsystem

The following example shows the inquiry application used in the programming examples:



1. Application program A (in the local System/34) displays a prompt asking an operator to enter an item number requesting the stock status for the item Ⓐ.

2. When the operator enters the item number, program A reads the number and searches file A (the local file) for the item Ⓒ.

3. If the item is found in the local file, program A displays the stock status on the screen Ⓐ.

4. If the item is *not* in the local file, program A uses the BSCEL subsystem to send the item number to the remote System/34 Ⓑ and Ⓓ.

5. Program B (in the remote system) uses the item number to search the remote file for the item Ⓕ.

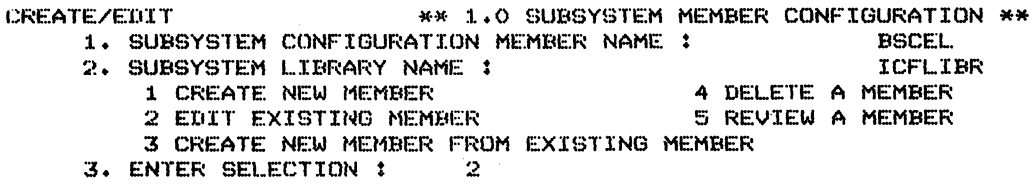
6. If the item is in the remote file, program B sends the stock status to program A Ⓖ and Ⓔ. If the item is not in the remote file, program B sends the characters ***.

7. If program A receives the stock status, it displays it. If it receives the characters ***, it displays the message ITEM NOT FOUND Ⓐ.

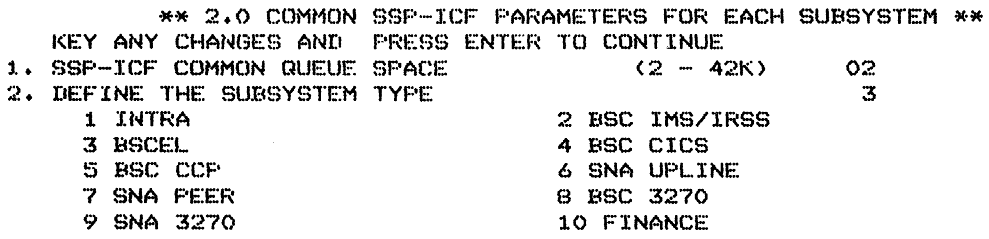
Programs A and B in this example are the programs described in Chapter 7 for the Intra subsystem. If you have not read the description of these programs in Chapter 7, see *How to Write Programs that Use the Intra Subsystem* in that chapter. The configuration and OCL examples in Chapter 7 are for the Intra subsystem only; you do not need to read those. Following are the configuration parameters and OCL statements for the BSCEL subsystem.

Configuration Parameters

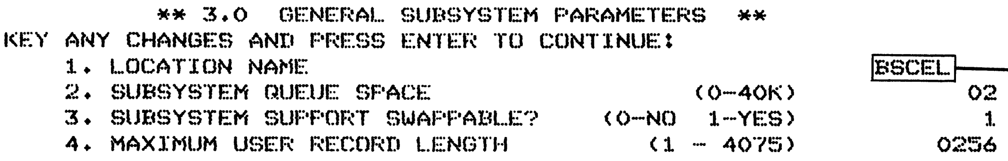
The following configuration parameters are used for this example. For a description of the configuration parameters, see *Setting up the BSCEL Subsystem* at the beginning of this chapter.



```
CREATE/EDIT                   ** 1.0 SUBSYSTEM MEMBER CONFIGURATION **
     1. SUBSYSTEM CONFIGURATION MEMBER NAME :              BSCEL
     2. SUBSYSTEM LIBRARY NAME :                           ICFLIBR
         1 CREATE NEW MEMBER                    4 DELETE A MEMBER
         2 EDIT EXISTING MEMBER                 5 REVIEW A MEMBER
         3 CREATE NEW MEMBER FROM EXISTING MEMBER
     3. ENTER SELECTION :      2
```

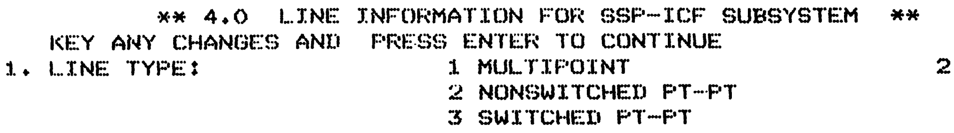


```
          ** 2.0 COMMON SSP-ICF PARAMETERS FOR EACH SUBSYSTEM **
   KEY ANY CHANGES AND  PRESS ENTER TO CONTINUE
1. SSP-ICF COMMON QUEUE SPACE              (2 - 42K)      02
2. DEFINE THE SUBSYSTEM TYPE                               3
     1 INTRA                         2 BSC IMS/IRSS
     3 BSCEL                         4 BSC CICS
     5 BSC CCP                       6 SNA UPLINE
     7 SNA PEER                      8 BSC 3270
     9 SNA 3270                      10 FINANCE
```



```
          ** 3.0   GENERAL SUBSYSTEM PARAMETERS   **
KEY ANY CHANGES AND PRESS ENTER TO CONTINUE:
    1. LOCATION NAME                                     BSCEL
    2. SUBSYSTEM QUEUE SPACE              (0-40K)           02
    3. SUBSYSTEM SUPPORT SWAPPABLE?    (0-NO  1-YES)         1
    4. MAXIMUM USER RECORD LENGTH         (1 - 4075)      0256
```

Also specified on the SESSION OCL statement.



```
          ** 4.0   LINE INFORMATION FOR SSP-ICF SUBSYSTEM   **
   KEY ANY CHANGES AND  PRESS ENTER TO CONTINUE
1. LINE TYPE:                1 MULTIPOINT                  2
                             2 NONSWITCHED PT-PT
                             3 SWITCHED PT-PT
```

Configuration Parameters (continued)

```
             ** 5.0  BSC GENERAL SUBSYSTEM PARAMETERS   I   **
   KEY ANY CHANGES AND PRESS ENTER TO CONTINUE:
1. EBCDIC/ASCII               (1-EBCDIC  2-ASCII)              1

3. WAIT TIME                    (1 - 999 SECONDS)            999
4. TRANSPARENCY ?                   (0-NO  1-YES)              0


             ** 5.1  BSC GENERAL SUBSYSTEM PARAMETERS  II   **
   KEY ANY CHANGES AND PRESS ENTER TO CONTINUE:


3. BLOCK LENGTH                          (0 - 4075)          0000
4. RECORD SEPARATOR                   (HEXADECIMAL)            00
5. ITB MODE                           (0-NO  1-YES)             0
6. BLANK        (0-NO, 1-COMPRESSION, 2-TRUNCATION)             0
7. 3740 MULTIPLE FILES                (0-NO  1-YES)             0


             ** 6.0  BSCEL SUBSYSTEM PARAMETERS  **                   W1
   KEY ANY CHANGES AND PRESS ENTER TO CONTINUE:

1. PARTNER                    (1-NORM   2-ATTR)                1
```

OCL Statements

The following procedure and OCL statement are used for the BASIC example:

```
BASICR ITEMABAS,ITEMBAS,30,,BSCSESS

// SESSION LOCATION=BSCEL,SYMID=1S
```

The BASICR procedure includes the SESSION statement.

The location name (BSCEL) is also specified on display 3.0 of the CNFIGICF procedure.

The SYMID (session ID) is explained in the description of the program.

The following OCL statements are used for the COBOL example:

```
// LOAD ITEMAC
// FILE NAME=FILEA
// SESSION LOCATION=BSCEL,SYMID=1S
// RUN
```

The location name (BSCEL) is also specified on display 3.0 of the CNFIGICF procedure.

The SYMID (session ID) is explained in the description of the program.

The following OCL statements are used for the RPG II example:

```
// LOAD ITEMAR
// FILE NAME=FILEA
// SESSION LOCATION=BSCEL,SYMID=1S
// RUN
```

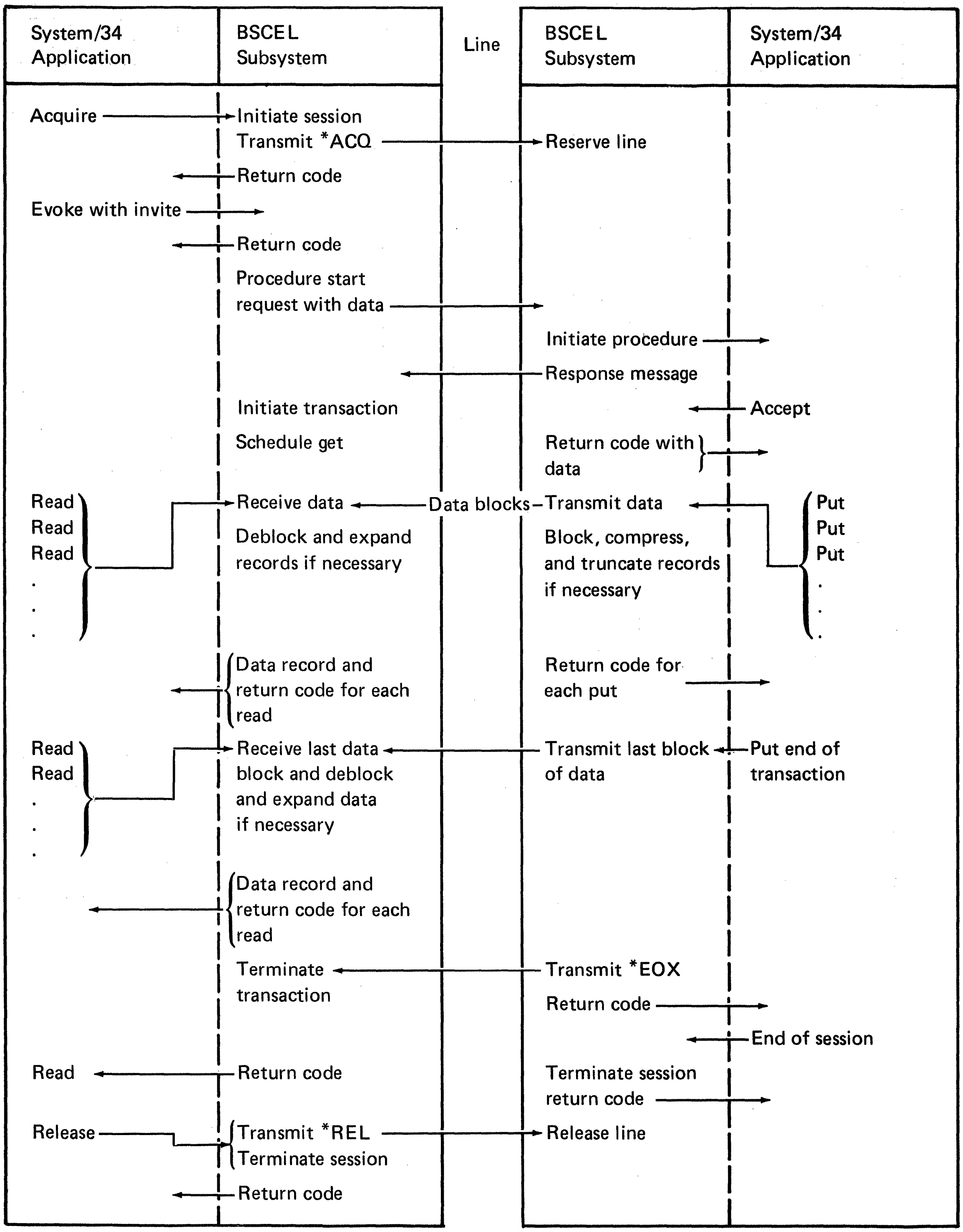
The location name (BSCEL) is also specified on display 3.0 of the CNFIGICF procedure.

The SYMID (session ID) is explained in the description of the program.

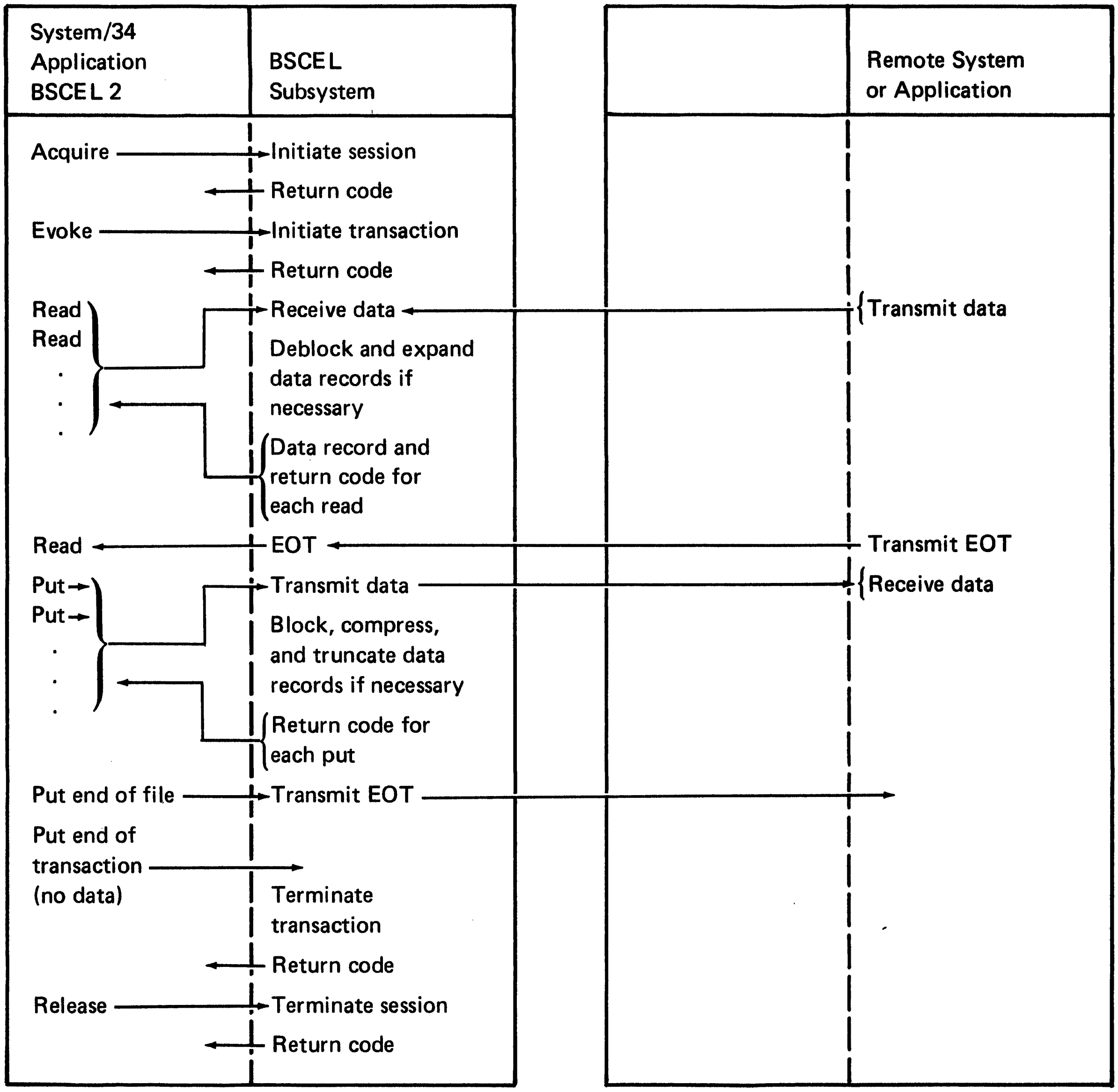
Other Applications for BSCEL

The following are application examples for the BSCEL subsystem. These examples are in addition to the example in Chapter 7; however, there are no coded programming examples for these applications.

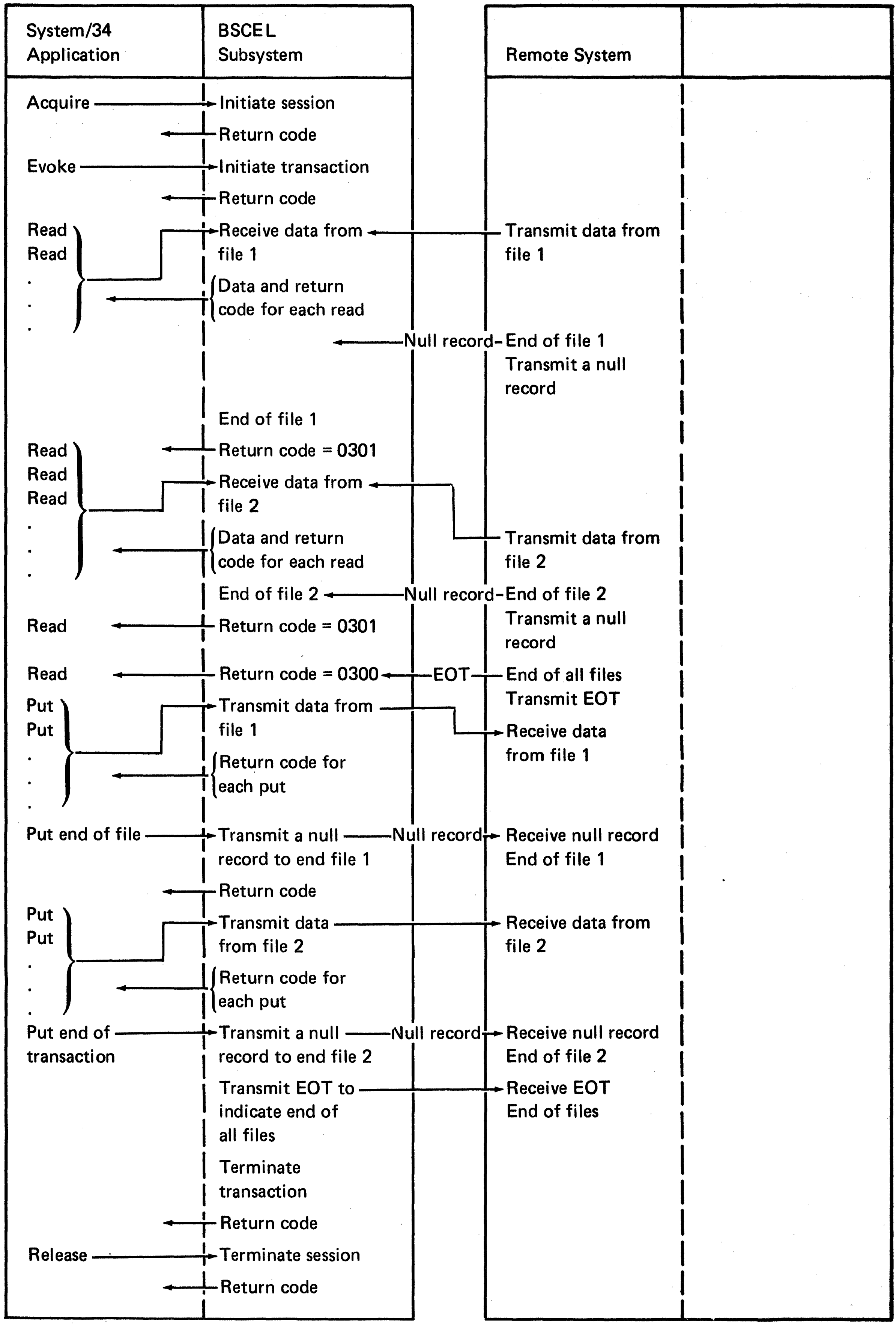
The following sample application flow shows communication between two System/34 application programs. Each program is using the BSCEL subsystem (PARTNER-NORM, and data records are blocked).



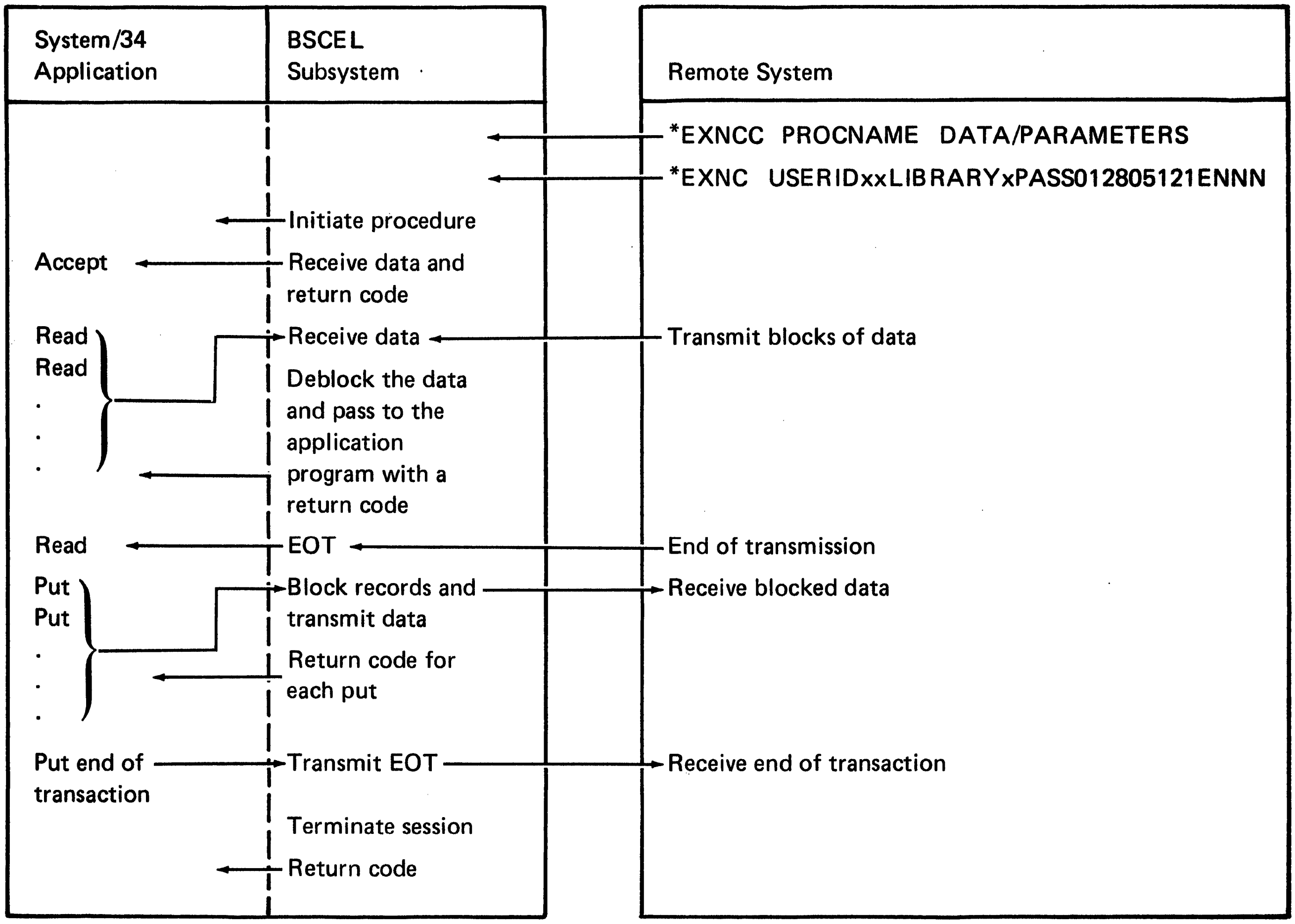
The following sample application flow shows communication between a System/34 application program and a remote device (PARTNER-ATTR).



The following sample application flow shows communication between a System/34 application program and a remote system using 3740 multiple files (PARTNER-ATTR).



The following sample application flow shows the initiation of a System/34 procedure from a remote system and the communication between the System/34 and,the remote system following the procedure initiation. The maximum user record length is 128 bytes, the block length is 512 bytes, and the record separator character is hexadecimal 1E.



BSCEL Subsystem Return Codes

This part of Chapter 8 describes all the return codes that are valid for the BSCEL 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 four-digit value; the first two digits contain the major code, and the last two 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).

Note: In the return code descriptions, *your program* refers to the local System/34 application program that initiates the operation and receives the return code from the subsystem. The *remote program* refers to the application program in the remote (or host) system with which the System/34 application program is communicating through SSP-ICF.

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 BSCEL subsystem. Although all the operations shown are valid for BSCEL, their validity also depends on the logical sequence of communications events occurring between the System/34 and the remote system.

Input Operations to Your Program	Output Operations from Your Program	Combined Operations in Your Program
Accept input		
	Acquire[1]	
	End of session	
	Evoke Evoke end of transaction	Evoke then get[2] Evoke then invite
Get Get attributes[3]		
Invite		
	Put Put end of file Put end of transaction	Put then get[2] Put then invite
	Release	
		Request to change direction then get[2] Request to change direction then invite
	Set timer[4]	

[1] Normally, 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.

[2] Valid only in assembler language.

[3] Valid only in assembler and COBOL languages.

[4] For BASIC and RPG II programs, the set 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.

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, or it received a message from the remote system.

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:** The *output* operation just performed by your program was completed successfully; your program can continue to send data.

Normal Action: 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 This Output Operation Was:	Then (in This Session):
Initiated the transaction (evoked the remote program[1])	Acquire operation	Issue an evoke 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[2] (by a remote procedure start request)	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.

[1] A remote program is evoked only if PARTNER-NORM is specified in the SESSION statement or in the subsystem configuration. If PARTNER-ATTR is specified, the transaction is initiated without evoking a program.

[2] An evoked program (started by a procedure start request) 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.

0001 **Normal Indication:** Your program has received some data on a successful input operation. It can continue to receive input until SSP-ICF returns a code of 0300 (an end of transmission indication, which allows your program to send data), or xx08 (an end of transaction indication).

Normal Action: Issue another input operation. If your program can detect something equivalent to an end of file condition, indicating that the last of the data was just received, it can issue an output operation.

0008 **Normal Indication:** An end of transaction indication was received with the last of the data on a successful input operation. Communications have ended with the remote *program*, but the session with the remote *system* is still active.

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 remote program on a successful *output* operation for your program; the remote program wants to send data as soon as possible. (If the remote system is another System/34, it has issued a *request to change direction then get* operation or a *request to change direction then invite* operation.) You should allow the remote program to send its data.

Normal Action: Issue an input operation as soon as possible.

0020 **Normal Indication:** A message from the remote system and an end of transmission indication were received on a successful input operation. The message, now in your program's input buffer, was received from the remote system as a result of your program's previous evoke operation that was unsuccessful. (That unsuccessful operation caused return code 831A to be returned to your program.)

Normal Action: Handle the message in the input buffer (possibly display it). Your program now has control of the session; issue another evoke operation (to start another program), issue a release operation (to either perform local processing or to start another session), or terminate.

0021 **Normal Indication:** A message was received from the remote system on a successful input operation. (The message is now in your program's input buffer.) Your program can continue to receive input.

Normal Action: Handle the message in the input buffer (possibly display it), and issue another input operation. If your program can detect something equivalent to an end of file condition, indicating that the last of the data was just received, it can issue an output operation.

0028 **Normal Indication:** An end of transaction indication was received with a remote 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. Communications have ended with the remote *program*, but the session with the remote *system* is still active.

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.

0030 **Normal Indication:** A truncated message from the remote system and an end of transmission indication were received on a successful input operation. (The message was truncated when it was placed in your program's input buffer, because it was too long for the buffer.) The message was received from the remote system as a result of your program's previous evoke operation that was unsuccessful. (That unsuccessful operation caused return code 831A to be returned to your program.)

Normal Action: Handle the truncated message (possibly display it) in the input buffer; then issue another evoke operation (to start another program), issue a release operation (to either perform local processing or to start another session), or terminate.

0031 **Normal Indication:** A truncated message was received from the remote system on a successful input operation. (The message was truncated because it was too long for your program's input buffer.) Your program can continue to receive input.

Normal Action: Handle the truncated message (possibly display it) in your program's input buffer, and issue another input operation. If your program can detect something equivalent to an end of file condition, indicating that the last of the data was just received, it can issue an output operation.

0038 **Normal Indication:** An end of transaction indication was received with a *truncated* remote 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. Communications have ended with the remote *program*, but the session with the remote *system* is still active.

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, either issue an end of session operation or terminate.

Major Code 01 – Successful operation with a new requester.

The new requester is a program on a remote system that initiated a session with your program by sending to the local system a procedure start request. 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 procedure start request, initiated by the remote program, was sent by the remote system in the form of an *EXEC, *EXEX, *EXNC, or *EXNX procedure start statement. 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 that the *output* operation issued by your program in response to a new requester completed successfully.

If the operation was an *input* operation, your program may have received some data from the requester. If any data was received on the input operation, it was included by the remote system in the incoming procedure start request statement.

If your program is an SRT program that was evoked by an incoming procedure start request 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 procedure start request statement 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 was received, and some data may have been received with the request. The remote program may now want to continue to send data, or it may want to receive data; your program should issue the appropriate operation. 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[1] 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 can continue to receive input until SSP-ICF returns a code of 0300 (an end of transmission indication) or xx08 (an end of transaction indication).

Normal Action: Handle any data passed with the request, perform any necessary record keeping[1] for the new requester, and issue another input operation. If your program can detect something equivalent to an end of file condition, indicating that the last of the data was just received, it can issue an output 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 remote program. Its communications have ended with your program; however, the session is still active between the local and remote systems.)

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 incoming 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[1] for the new requester of the transaction that has ended. Then, either issue an end of session operation or terminate your program.

[1] For some situations, no record keeping for the session is necessary. In other situations, you should record the session ID of the new requester. You may also want to keep a table containing the IDs of all active requesters, or to maintain a history log of all requests.

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 remote 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 incoming 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[1] for the new requester of the transaction that has ended. Then, because your program was evoked, issue an end of session 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, or it received a message from the remote system. 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.

[1] For some situations, no record keeping for the session is necessary. In other situations, you should record the session ID of the new requester. You may also want to keep a table containing the IDs of all active requesters, or to maintain a history log of all requests.

Code **Indication/Action**

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 can continue to receive input until SSP-ICF returns a code of 0300 (an end of transmission indication) or xx08 (an end of transaction indication).

Normal Action: Issue another input operation. If your program can detect something equivalent to an end of file condition, indicating that the last of the data was just received, it can issue an output operation.

0208 **Normal Indication:** An end of transaction indication was received with the last of the data on a successful input operation. Although communications have ended with the remote program, the session with the remote system is still active. 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 (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.

0220 **Normal Indication:** A message from the remote system and an end of transmission indication were received on a successful input operation. The message, now in your program's input buffer, was received from the remote system as a result of your program's previous evoke operation that was unsuccessful. (That unsuccessful operation caused return code 831A to be returned to your program.) 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 (possibly display it); then issue another evoke operation (to start another program), issue a release operation (to perform local processing), or terminate.

0221 **Normal Indication:** A message was received from the remote system on a successful input operation. (The message is now in your program's input buffer.) Also, a stop system request or a disable subsystem request is pending; no new sessions using the subsystem can be initiated. Your program can continue to receive input.

Normal Action: Handle the message in the input buffer (possibly display it), and issue another input operation. If your program can detect something equivalent to an end of file condition, indicating that the last of the data was just received, it can issue an output operation.

0228 **Normal Indication:** An end of transaction indication was received with a remote 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. Although communications have ended with the remote program, the session with the remote system is still active. 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 your program's input buffer (display it, for example). If 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 your program was evoked, either issue an end of session operation or terminate.

0230 **Normal Indication:** A truncated message from the remote system and an end of transmission indication were received on a successful input operation. (The message was truncated when it was placed in your program's input buffer, because it was too long for the buffer.) The message was received from the remote system as a result of your program's previous evoke operation that was unsuccessful. (That unsuccessful operation caused return code 831A to be returned to your program.)

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 (possibly display it) in your program's input buffer; then issue another evoke operation (to start another program), issue a release operation (to perform local processing), or terminate.

0231 **Normal Indication:** A truncated message was received from the remote system on a successful input operation. (The message was truncated because it was too long for your program's input buffer.) Also, a stop system request or a disable subsystem request is pending; no new sessions using the subsystem can be initiated. Your program can continue to receive input.

Normal Action: Handle the truncated message (possibly display it) in your program's input buffer, and issue another input operation. If your program can detect something equivalent to an end of file condition, indicating that the last of the data was just received, it can issue an output operation.

0238 **Normal Indication:** An end of transaction indication was received with a *truncated* remote 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. Although communications have ended with the remote program, the session with the remote system is still active. 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 (to start another program), it can issue a release operation (to perform local processing), or it can terminate. If your program was evoked, either issue an end of session operation or terminate.

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:** An end of transmission indication with *no* data was received on a successful input operation. The last record in the file has been received. If *yes* was specified for the 3740 multiple files parameter in the subsystem configuration, this return code indicates that the last *file* has been received. Communications with the remote program have ended; however, the session is still active between the local and remote systems.)

Normal Action: Issue another input operation, issue an output operation, or terminate the transaction with a put end of transaction operation.

0301 **Normal Indication:** On a successful input operation, *no* data (a null record) was received. If *yes* was specified for the 3740 multiple files parameter in the subsystem configuration, this return code indicates that the last record in the file has been received. Your program can continue to receive input until SSP-ICF returns a code of 0300 (an end of transmission indication, which allows your program to send data), or xx08 (an end of transaction indication).

Normal Action: Issue another input operation.

0308 **Normal Indication:** An end of transaction indication was received *without* data on a successful input operation. Although communications have ended with the remote program, the session with the remote system is still active.

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.

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 remote program. Your output, associated with this output operation and any previous output operations for this file, was sent to the remote system. Your program can attempt to send its output later.

Note: If your program issues another output operation, an error return code of 831C will be received.

General Recovery Actions: Issue an input operation to receive data or a message from the remote system.

Code **Indication/Action**

0411 **Normal Indication:** The remote 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 remote 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. Your program must receive the data before it can perform an output operation.

Normal Action: Issue an input operation to receive the data.

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 a new requester) 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 requesting program have been completed.

3401 **Error Indication:** This input operation was rejected because the record length of the data sent by the remote 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/34 Messages Guide* 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 (in COBOL and assembler only) a set timer operation, or by using the TIMER intrinsic function (in BASIC only); (2) continue local processing; or (3) terminate.
- If the session should be started again after the subsystem is enabled, it must be reacquired by your program or restarted by the remote 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 either in this subsystem or in the interrupt handler.

Recovery Action: This subsystem has been disabled; it must be enabled again before communications can resume. Your program can continue local processing, wait[1] to reissue the acquire operation, or terminate.

If more than one subsystem was active when the error occurred, all subsystems that were active when the error occurred should be disabled. (Note that all other active BSCEL subsystems are automatically disabled when the error occurs; all other types of active subsystems must be manually disabled.)

If all subsystems (of all types) on the system are *not* disabled to recover from the processor check, the common queue space used by the failing subsystem *cannot* be freed. And if it is not freed, that space is wasted, and an indication of insufficient common queue space being available can occur. The indication can occur as a message when the failing subsystem is reenabled or when a different subsystem is enabled. The indication can also occur as a return code to your program for any subsystem that is starting a *new* session (code 8215) or performing an output operation in any *existing* session (code 8315).

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 remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, wait[1] to reissue the acquire operation, or terminate.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

Major Code 81 – Permanent (nonrecoverable) session error.

Error Description: A nonrecoverable error has occurred in the session; the session cannot be continued and 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/34 Messages Guide* 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 session must be acquired again before communications can resume.

General Recovery Actions: The following general actions can be taken for each 81xx return code. Other specific actions are given in each return code description.

- Several return codes indicate that an error condition must be corrected by changing a value in the subsystem configuration record or in the SESSION statement for your program.
  - To change a parameter value in the subsystem configuration being used by your program, you must disable the subsystem before making the change in the subsystem's configuration record, and enable the subsystem again to make the change effective.
  - To change a parameter value in the SESSION statement associated with your program, you must terminate your program only.

  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.

- If the session should be started again, it must be reacquired by your program or restarted by the remote program before communications can resume.

- An end of session (EOS or $$EOS) operation should be issued for the session that has terminated. Your program can also continue local processing, or it can terminate.

Note: If the session is started again, it starts from the beginning, not at the point where the session error occurred.

Code **Indication/Action**

8136 **Error Indication:** On the first *output* operation requiring that a switched line connection be established for the session, an invalid remote identifier for the remote system (or device) was received from the remote system; the session has been terminated. The received remote identifier must match the remote identifier specified for this subsystem configuration.

Recovery Action: Verify that the remote identifier specified for this subsystem configuration was specified correctly either by the remote ID parameter in the CNFIGICF procedure or by the SSP DEFINEID procedure (when multiple remote identifiers are specified). If the remote identifier was specified correctly, call the remote location to correct the error at the remote end of the switched line.

8137 **Error Indication:** On the first *input* operation requiring that a switched line connection be established for the session, an invalid remote identifier for the remote system (or device) was received from the remote system; the session has been terminated. The received remote identifier must match the remote identifier specified for this subsystem configuration.

Recovery Action: Verify that the remote identifier specified for this subsystem configuration was specified correctly either by the remote ID parameter in the CNFIGICF procedure or by the SSP DEFINEID procedure (when multiple remote identifiers are specified). If the remote identifier was specified correctly, call the remote location to correct the error at the remote end of the switched line.

8183 **Error Indication:** An MLCA (multiline communications adapter) controller check occurred on an *output* operation; data may have been lost. The session has been terminated.

Recovery Action: If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

8184 **Error Indication:** An MLCA (multiline communications adapter) controller check occurred on an *input* operation; data may have been lost. The session has been terminated.

Recovery Action: If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

8185 **Error Indication**: An attempt by this subsystem to automatically call one or more remote locations using the autocall or X.21 feature was not successful. All available numbers in the list of phone numbers or in the list of public data network numbers were called, but no connection was established. The session has been terminated.

Recovery Action: The list has been reinitialized. If the session should be started again, reissue the acquire operation; calling will begin with the first number in the list. Otherwise, your program can continue local processing or terminate.

8186 **Error Indication**: An attempt by this subsystem to automatically call one or more remote locations using the autocall or X.21 feature was not successful. All numbers in the list of phone numbers or in the list of public data network numbers were already marked as called. The message *PHONE LIST EXHAUSTED* (SYS-8607) has been displayed on the system console, and the session has been terminated.

Recovery Action: The list has been reinitialized. If the session should be started again, reissue the acquire operation; calling will begin with the first number in the list. Otherwise, your program can continue local processing or terminate.

8191 **Error Indication**: A permanent line error occurred on an *output* operation, and the system operator has taken a recovery option in response to a permanent line error message. (You can find out what type of line error occurred by asking the system operator.) The session has been terminated.

Recovery Action: If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

8192 **Error Indication**: A permanent line error occurred on an *input* operation, and the system operator has taken a recovery option in response to a permanent line error message. (You can find out what type of line error occurred by asking the system operator.) The session has been terminated.

Recovery Action: If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

8193 **Error Indication**: A disconnect indication (for switched lines only) was received on an *output* operation. A disconnect time-out in the remote system was exceeded, the line was unexpectedly disconnected, or your program may have sent some invalid data. The session has been terminated.

Recovery Action: Verify that your program did not cause a time-out and that it did not send data that was invalid. Also, verify that it did not try to send data after the transaction had ended. If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

8194 **Error Indication**: A disconnect indication (for switched lines only) was received on an *input* operation. A disconnect time-out in the remote system was exceeded, or the line was unexpectedly disconnected. The session has been terminated.

Recovery Action: Verify that your program did not cause a time-out. Also, verify that it did not try to receive data after it had received an end of transaction indication. If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

8197 **Error Indication**: An abort transmission sequence was received from the remote system on an *output* operation; the remote system is terminating the line transmission abnormally because it could not or did not want to continue the session. The session has been terminated.

Recovery Action: If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

8198 **Error Indication**: An abort transmission sequence was received from the remote system on an *input* operation; the remote system is terminating the line transmission abnormally because it could not or did not want to continue the session. The session has been terminated.

Recovery Action: If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

8199 **Error Indication:** On an *output* operation, the time between successive data blocks being sent to the remote system exceeded the value specified for the wait time parameter in the subsystem configuration.

Recovery Action: Check the wait time parameter value to ensure that it is large enough for your program. Also check your program for excessive delays between output operations. If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

819A **Error Indication:** On an *input* operation, the time between successive data blocks being received from the remote system exceeded the value specified for the wait time parameter in the subsystem configuration.

Recovery Action: Check the wait time parameter value in the subsystem configuration to ensure that it is large enough for your program. Also check your program for excessive delays between input operations. If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

819B **Error Indication:** On an *output* operation, in a put-versus-put situation, the subsystem detected a block size error before it detected that both your program and the remote system were attempting to send data at the same time. The remote system sent data, but the length of the data block exceeded the length of the subsystem's line buffer. The session has been terminated.

Recovery Action: Check the maximum user record length (RECL) parameter and block length (BLKL) parameter that are specified in the subsystem configuration record and in the SESSION statement for your program; then correct them if necessary. If the parameters are correct, notify the remote system programmer and verify that the record length or block length is correct. Then, if your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

819C **Error Indication**: On an *input* operation, the length of the data block sent by the remote system exceeded the length of the subsystem line buffer. The session has been terminated.

Recovery Action: Check the maximum user record length (RECL) parameter and block length (BLKL) parameter that are specified in the subsystem configuration record and in the SESSION statement for your program; then correct them if necessary. If the parameters are correct, notify the remote system programmer and verify that the record length or block length is correct. Then, if your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

819D **Error Indication:** On an input or output operation, unexpected data was received from the remote system either after an end of transaction indication was received or before an evoke operation was issued by your program. The session has been terminated.

Recovery Action: Check that your program did not issue an end of transaction operation before the transaction was completed. Also check to see if the remote system sent a procedure start request while your session was still active. If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

81B8 **Error Indication**: On an *output* operation, in a put-versus-put situation, the subsystem detected that a data record received from the remote system is too long. (The subsystem detected the length error before it detected that both your program and the remote system were attempting to send data at the same time.) The record exceeds the maximum user record length specified for this session. The session has been terminated.

Recovery Action: Check that the maximum user record length parameter (RECL) in the subsystem configuration or in the SESSION statement is specified with a value that is large enough for the longest record to be sent or received. If the value was specified correctly, notify the remote system programmer and have the record length changed at the remote end. If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

81B9 **Error Indication:** On an *input* operation, the subsystem detected that a data record is too long for your program. The record, received from the remote program, exceeds the maximum user record length specified for the subsystem configuration. The session has been terminated.

Recovery Action: Check that the maximum user record length parameter (RECL) in the subsystem configuration or in the SESSION statement is specified with a value that is large enough for the longest record to be sent or received. If the value was specified correctly, notify the remote system programmer and have the record length changed at the remote end. If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

81BC **Error Indication:** An attempt by this subsystem to automatically call a remote location using the autocall or X.21 feature was not successful because the wrong type of list was used to make the call. Either a list of public data network numbers was used to make the call on an autocall line, or a list of phone numbers was used to make the call on an X.21 line. The session has been terminated.

Recovery Action: Change the name specified in the *phone list name* parameter in the subsystem configuration, or in the PHONE parameter of the SESSION statement for this program. Then reissue the acquire operation to restart the session.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

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/34 Messages Guide* 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, (and depending on what the return code description says):
   a. Notify the remote location that a change is required on that end to correct the error received.
   b. Simply reissue the acquire operation. It could be successful if the error occurred because there was not enough common queue space available to support a new session, because an isolated line error occurred, or because the remote system was not active temporarily.
   c. If the acquire operation is again unsuccessful, retry it only a limited number of times. (The limit for retries should be specified in your program.)
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 operation is valid only in an *active* session and cannot, therefore, be issued after an 82xx return code is received. (This restriction does not apply to COBOL and assembler programs, or to the TIMER intrinsic function in BASIC, which also can be used to wait for a specified time interval.)

Code **Indication/Action**

820A **Error Indication:** On an unsuccessful acquire operation, an invalid combination of data attributes was detected. ASCII code was specified during subsystem configuration, but TRANSP-YES was specified on the SESSION statement. The two values are not valid together.

Recovery Action: Change either the configuration of the BSCEL subsystem or the TRANSP (transparency) parameter on the SESSION statement, and reissue the acquire operation.

8213 **Error Indication:** On an unsuccessful acquire operation, a queue space error condition was detected. The session could not be started because no *subsystem* queue space was available at the time.

Recovery Action: Your program can wait[1] for a period of time, then reissue the acquire operation. If an unacceptable number of queue space errors occur, you can disable the subsystem and change the subsystem configuration by specifying a larger subsystem queue space size in the subsystem queue space parameter. After the subsystem is enabled, reissue the acquire operation to start the session.

8215 **Error Indication:** On an unsuccessful acquire operation, a queue space error condition was detected. The session cannot be started because the size of the *common* queue space, specified during subsystem configuration, is too small.

Recovery Action: Your program can wait[1] for a period of time, then reissue the acquire operation. If an unacceptable number of queue space errors occur, you can disable all the subsystems that are active in the system, and change the subsystem configuration by specifying a larger common queue space size in the SSP-ICF common queue space parameter. After the subsystem is enabled, reissue the acquire operation to start the session.

821E **Error Indication:** The acquire operation attempted by your program (BASIC programs only) was unsuccessful because there was no SESSION statement specified between the LOAD and RUN statements for your program. The method used to issue the acquire operation is not supported by the BSCEL subsystem. The session was not started.

Recovery Action: Issue a SESSION statement that specifies, in the SYMID parameter, the identifier of the session to be acquired. The same identifier must be specified in the ID parameter of the OPEN statement.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

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, respecify 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.

8236 **Error Indication:** On an unsuccessful acquire operation (of a session over switched lines only), an invalid remote identifier was received from the remote system or device with which the session is being acquired. The received remote identifier must match the remote identifier specified for this subsystem configuration.

Recovery Action: Verify that the remote identifier specified for this subsystem configuration was specified correctly either by the remote ID parameter in the CNFIGICF procedure or by the SSP DEFINEID procedure (when multiple remote identifiers are specified). If the remote identifier was specified correctly, call the remote location to correct the error at the remote end of the switched line.

8281 **Error Indication:** On an unsuccessful acquire operation, an SSP-ICF error condition was detected. The error caused a processor check either in this subsystem or in the interrupt handler.

Recovery Action: This subsystem has been disabled; it must be enabled again before communications can resume. Your program can continue local processing, wait[1] to reissue the acquire operation, or terminate.

If more than one subsystem was active when the error occurred, all subsystems that were active when the error occurred should be disabled. (Note that all other active BSCEL subsystems are automatically disabled when the error occurs; all other types of active subsystems must be manually disabled.)

If all subsystems (of all types) on the system are *not* disabled to recover from the processor check, the common queue space used by the failing subsystem *cannot* be freed. And if it is not freed, that space is wasted, and an indication of insufficient common queue space being available can occur. The indication can occur as a message when the failing subsystem is reenabled or when a different subsystem is enabled. The indication can also occur as a return code to your program for any subsystem that is starting a *new* session (code 8215) or performing an output operation in any *existing* session (code 8315).

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

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 remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, wait[1] to reissue the acquire operation, or terminate.

8283 **Error Indication:** On an unsuccessful acquire operation, an MLCA (multiline communications adapter) controller check occurred. The session was not started.

Recovery Action: Your program can reissue the acquire operation, continue local processing, or terminate.

8285 **Error Indication:** An acquire operation, for which the subsystem used the autocall or X.21 feature to automatically call one or more remote locations, was not successful. All available numbers in the list of phone numbers or in the list of public data network numbers were called, but no connection was established. The session was not started.

Recovery Action: The list has been reinitialized. Your program can reissue the acquire operation, continue local processing, or terminate. (Calling will begin with the first number in the list.)

8286 **Error Indication:** An acquire operation, for which the subsystem used the autocall or X.21 feature to automatically call one or more remote locations, was not successful. All the numbers in the list of phone numbers or in the list of public data network numbers were already marked as called. The message *PHONE LIST EXHAUSTED* (SYS-8607) has been displayed on the system console, and the session was not started.

Recovery Action: The list has been reinitialized. Your program can reissue the acquire operation, continue local processing, or terminate. (Calling will begin with the first number in the list.)

8289 **Error Indication:** On an unsuccessful acquire operation, an invalid combination of attributes was detected. Both a *record separator* was specified (either during subsystem configuration or in the RECSEP parameter of the SESSION statement) and *transparent mode* was specified (either during configuration or in the TRANSP (transparency) parameter of the SESSION statement). A record separator and transparent mode cannot be specified together.

Recovery Action: Change the configuration of the BSCEL subsystem, or change the value of the RECSEP or TRANSP parameter on the SESSION statement; then reissue the acquire operation.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

828A **Error Indication**: On an unsuccessful acquire operation, an invalid combination of attributes was detected. Both a *record separator* was specified (either during subsystem configuration or in the RECSEP parameter of the SESSION statement) and *ITB* (intermediate text block) mode was specified (either during configuration or in the ITB parameter of the SESSION statement). A record separator and ITB mode cannot be specified together.

Recovery Action: Change the configuration of the BSCEL subsystem, or change the value of the RECSEP or ITB parameter on the SESSION statement; then reissue the acquire operation.

828B **Error Indication**: On an unsuccessful acquire operation, an invalid combination of *record length* and *block length* attributes was detected. The maximum user record length was specified (either during subsystem configuration or in the RECL parameter of the SESSION statement) to be greater than the block length (either during configuration or in the BLKL parameter of the SESSION statement). The block length must be greater than or equal to the maximum user record length.

Recovery Action: Change the configuration of the BSCEL subsystem, or change the value of the RECL or BLKL parameter on the SESSION statement; then reissue the acquire operation.

828C **Error Indication**: On an unsuccessful acquire operation, an invalid combination of *ITB mode* and *3740 file* attributes was detected. During subsystem configuration, the 3740 multiple files parameter was specified as *1* (yes), and then ITB (intermediate text block) mode was also specified in the ITB parameter of the SESSION statement used for the session being acquired. The two values are not valid together.

Recovery Action: Change the configuration of the BSCEL subsystem, or change the value of the ITB parameter on the SESSION statement; then reissue the acquire operation.

828D **Error Indication**: On an unsuccessful acquire operation, an invalid combination of data attributes was detected. Both *blank compression* was specified (either during subsystem configuration or in the BLANK parameter of the SESSION statement) and *ITB* (intermediate text block) mode was specified (either during configuration or in the ITB parameter of the SESSION statement). Blank compression and ITB mode cannot be specified together.

Recovery Action: Change the configuration of the BSCEL subsystem, or change the value of the BLANK or ITB parameter on the SESSION statement; then reissue the acquire operation.

828E **Error Indication:** On an unsuccessful acquire operation, an invalid combination of data attributes was detected. Both *blank truncation* was specified (either during subsystem configuration or in the BLANK parameter of the SESSION statement) and *ITB* (intermediate text block) mode was specified (either during configuration or in the ITB parameter of the SESSION statement). Blank truncation and ITB mode cannot be specified together.

Recovery Action: Change the configuration of the BSCEL subsystem, or change the value of the BLANK or ITB parameter on the SESSION statement; then reissue the acquire operation.

828F **Error Indication:** On an unsuccessful acquire operation, an invalid *block length* was detected. The block length was specified as 0 (during subsystem configuration or in the BLKL parameter of the SESSION statement); 0 indicates no blocking. A block length greater than the maximum user record length *must* be specified if any of the following attributes are also specified for the subsystem: record separator, ITB mode, blank compression, or blank truncation.

Recovery Action: Change the configuration of the subsystem, or change the value(s) of the BLKL, RECSEP, ITB, or BLANK parameters on the SESSION statement to specify either no blocking, or blocking with at least one of the other attributes.

8290 **Error Indication:** On an unsuccessful acquire operation, an invalid combination of attributes was detected. Both *blank compression* was specified (either during subsystem configuration or in the BLANK parameter of the SESSION statement) and *transparent mode* was specified (either during configuration or in the TRANSP parameter of the SESSION statement). The two values are not valid together.

Recovery Action: Change the configuration of the BSCEL subsystem, or change the value of the BLANK or TRANSP parameter on the SESSION statement, then reissue the acquire operation.

8291 **Error Indication:** A permanent line error occurred on an unsuccessful acquire operation, and the system operator has taken a recovery option in response to a permanent line error message. (You can find out what type of line error occurred by asking the system operator.) The session was not started.

Recovery Action: Your program can reissue the acquire operation, continue local processing, or terminate.

8293 **Error Indication:** A disconnect indication (for switched lines only) was received on an unsuccessful acquire operation. The line was unexpectedly disconnected. The session was not started.

Recovery Action: Check with your remote system location to determine why the remote system sent a disconnect. Your program can reissue the acquire operation, continue local processing, or terminate.

8297 **Error Indication:** An abort transmission sequence was received from the remote system on an unsuccessful acquire operation; the remote system is terminating the line abnormally because it could not continue the communications. The session was not started.

Recovery Action: Your program can issue another acquire operation, continue local processing, or terminate.

82A0 **Error Indication:** On an unsuccessful acquire operation, an invalid record separator character was detected. The invalid record separator character was specified in the RECSEP parameter of the SESSION statement. The session was not started.

Recovery Action: Change the value of the record separator character in the RECSEP parameter on the SESSION statement. Then reissue the acquire operation to start the session.

82A7 **Error Indication:** The acquire operation was unsuccessful because the specified communications line was already in use. The session was not started.

Recovery Action: Your program can wait[1] for the line to become available, then reissue the acquire operation. Otherwise, it can continue local processing 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 100 sessions can be active in the System/34 at one time. The session was not started.

Recovery Action: Your program can wait[1] 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 to be enabled is identified by the location parameter in the SESSION statement. That location name must also be specified in the subsystem configuration record (shown on display 3.0 of the subsystem configuration planning charts). The session was not started.

Recovery Action: Verify that the subsystem name was specified correctly on the LOCATION parameter of the SESSION statement. If the correct name was specified, contact the System/34 system operator and request that the specified subsystem be enabled by executing the ENABLE procedure command at the system console. Then reissue the acquire operation. Otherwise, your program can continue local processing, wait[1] to reissue the acquire operation, or terminate.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

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[1] 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[1] until the subsystem is enabled again, and then reissue the acquire operation. Otherwise, your program can continue local processing, or it can terminate.

82B4 **Error Indication**: The acquire operation was not successful because all of the resources needed for the session could not be allocated from the assign/free area of the system. All available resources are already being used in the system. The session was not started.

Recovery Action: Wait[1] for the needed resources to become available, then reissue the acquire operation. Otherwise, continue local processing or terminate.

82BC **Error Indication**: On an unsuccessful acquire operation, an invalid type of list was detected by the subsystem when it attempted to automatically call a remote location using the autocall or X.21 feature. Either a list of public data network numbers was used to make the call on an autocall line, or a list of phone numbers was used to make the call on an X.21 line. The session was not started.

Recovery Action: Specify the name of the list that is of the correct type for the line to be used. Change the name specified in the *phone list name* parameter in the subsystem configuration, or in the PHONE parameter of the SESSION statement for this program; then reissue the acquire operation.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

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/34 Messages Guide* 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.

   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.

   Note: If the session is started again, it starts from the beginning, not at the point where the session error occurred.
3. If no change is needed in your program or in the subsystem, (and depending on what the return code description says):
   a. Retry the operation, if possible. It could be successful if the error occurred because there was not enough common queue space available at the time or because of an isolated line failure.
   b. If another operation is not successful, retry it only a limited number of times. (The limit for retries should be specified in your program.)
   c. Issue a set timer operation in your program so it can wait for a specified time interval before reissuing the operation.

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 an 81xx (session was terminated) or 82xx (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 an 81xx or 82xx return code is handled properly. If you want your program to recover from an improperly handled error condition, issue another acquire operation.

8315 **Error Indication:** On an evoke operation, a queue space error condition was detected. The evoke operation could not be performed because no *common* queue space was available at the time.

Recovery Action: Your program can issue a set timer operation and wait for a period of time, and then reissue the evoke operation. If an unacceptable number of queue space errors occur, you can disable all the subsystems and change the subsystem configuration by specifying a larger common queue space size in the SSP-ICF common queue space parameter. After the subsystem is enabled, reissue the acquire operation to restart the session.

831A **Error Indication:** An evoke operation failed to complete successfully, or the evoked program terminated abnormally. A message from the remote system describing why it failed is waiting in the subsystem input buffer. The evoke operation could have been 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.

831C **Error Indication:** The output operation issued before this output operation received a return code of 0411 or 0412 (indicating that the remote 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. The session is still active, however.

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 before attempting to communicate with the remote 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. If this operation was an evoke operation, the length of your remote procedure name (if any) plus the length of your data exceeds 120 bytes. The session is still active, however.

Recovery Action: If you want your program to recover dynamically, reissue the output operation with a smaller output length. Otherwise, you can either change the record length in your program and recompile it, or you can change the value specified for the maximum user record length parameter in the subsystem configuration or in the SESSION statement. 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 erroneously followed by a *request to change direction then get* operation or a *request to change direction then invite* operation. Neither of these operations is valid while your program is in the send state. The session, however, 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 remote 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 remote 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 remote program. The session is still active.

Recovery Action: If you want your program to dynamically 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 incoming procedure start request and cannot, therefore, issue any evoke operations in this session.

Recovery Action: If you want your program to dynamically 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.

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 a release operation. If a coding error in your program caused the error, correct your program.

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.

8334 **Error Indication:** On an evoke operation issued by your program, no procedure name was included with the operation.

Recovery Action: Correct the evoke operation statement by supplying the correct procedure name, and reissue the operation.

833C **Error Indication:** On an unsuccessful output operation, an invalid combination of data modes was detected. Both *transparency* mode was specified (either during subsystem configuration or in the TRANSP parameter of the SESSION statement) and *ITB (intermediate text block)* mode was specified (either during configuration or in the ITB parameter of the SESSION statement). The two values are not valid together on an output operation.

Recovery Action: Change either the configuration of the BSCEL subsystem, or the TRANSP or ITB parameter of the SESSION statement, then issue another output operation. Otherwise, do not issue any more operations that send data.

Chapter 9. The BSC CCP Subsystem

The BSC CCP subsystem provides distributed data processing support to users of the System/34 SSP in conjunction with a System/3 Model 15 using CCP. The BSC CCP subsystem provides an interactive interface between System/34 application programs and CCP application programs. The CCP subsystem can support multiple application programs concurrently communicating with CCP.

The CCP subsystem allows System/34 application programs to initiate application programs on the CCP system. System/34 procedures can be initiated from CCP application programs. System/34 security options as well as CCP security options are supported.

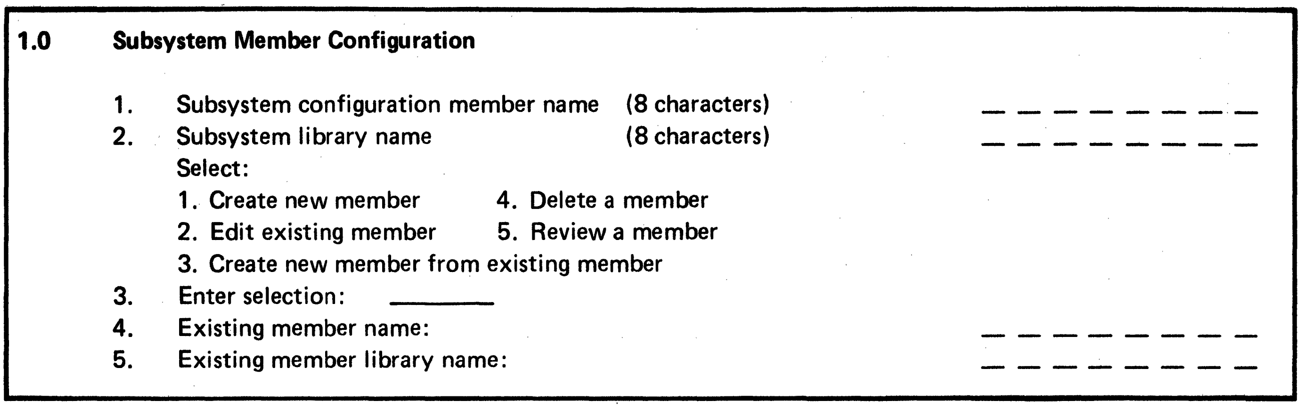
The CCP subsystem can operate on a point-to-point nonswitched line, a point-to-point switched line, or a multipoint line (System/34 as a tributary station). For a point-to-point switched line, the System/34 must be the caller, unless the CCP application starts a procedure on the System/34. The CCP subsystem transmits and receives transparent EBCDIC, nontransparent EBCDIC, or ASCII data. All System/34 application programs must use only EBCDIC data; if translation to or from ASCII is required, the subsystem performs the translation.

SETTING UP THE CCP SUBSYSTEM

The SSP procedures CNFIGSSP and INSTALL are used to include the interactive communications feature and CCP subsystem support on the System/34. The general interactive communications and line control support is included when it is requested on the appropriate CNFIGSSP prompt. The CCP subsystem support is then copied to the system library when the appropriate responses to the INSTALL procedure prompts are taken. The CNFIGSSP and INSTALL procedures, with their displays and related responses, are described in the *Installation and Modification Reference Manual.*

After the CCP subsystem has been installed, the CNFIGICF procedure is used to tailor the subsystem support to an existing or proposed network. The operation of the CNFIGICF procedure is also explained in the *Installation and Modification Reference Manual.* Before running the CNFIGICF procedure, however, you should fill out a planning chart for each subsystem that you want to define. Copies of the planning chart for each subsystem are available in Appendix F of this manual and in the *Installation and Modification Reference Manual.* The following sections explain how to fill out the planning chart for the CCP subsystem.

Display 1.0 Subsystem Member Configuration



```
1.0    Subsystem Member Configuration

       1.   Subsystem configuration member name  (8 characters)        _ _ _ _ _ _ _ _
       2.   Subsystem library name               (8 characters)        _ _ _ _ _ _ _ _
            Select:
            1. Create new member         4. Delete a member
            2. Edit existing member      5. Review a member
            3. Create new member from existing member
       3.   Enter selection:  ______
       4.   Existing member name:                                      _ _ _ _ _ _ _ _
       5.   Existing member library name:                              _ _ _ _ _ _ _ _
```

Subsystem configuration member name: Specify a name for this configuration of the subsystem. This name is used to store the member in a library and is referenced in the ENABLE and DISABLE procedures.

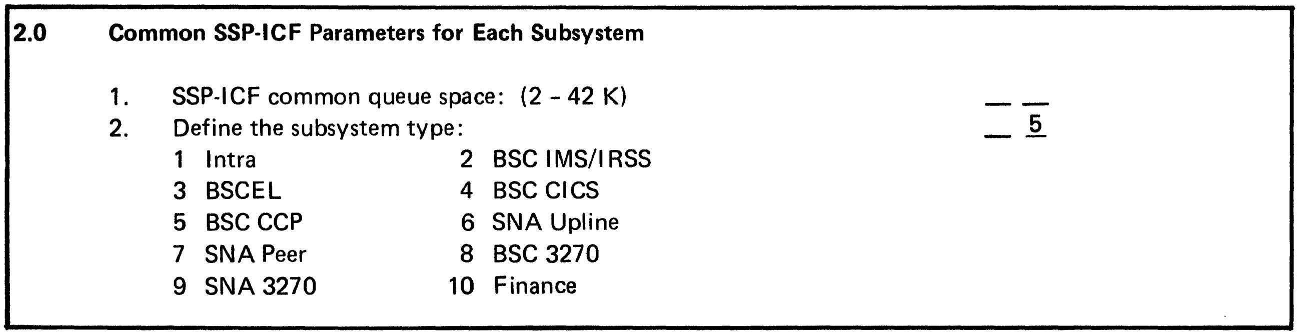
Library name: Specify the name of the library in which the configuration member is stored or is to be stored. The default is #LIBRARY; however, you should probably store the configuration in a user library.

Enter selection: Specify one of the five options: (1) create a new member, (2) edit an existing member, (3) create a new member from an existing member, (4) delete a member, or (5) review a member without changing it.

Existing member name: This prompt appears if option 3 was selected. Specify the name of the existing subsystem configuration member that is to be used to create the new member. The existing member remains unchanged.

Existing member library name: This prompt appears if option 3 was selected. Specify the library name where the existing member resides.

Display 2.0 Common SSP-ICF Parameters for Each Subsystem



2.0 Common SSP-ICF Parameters for Each Subsystem

1. SSP-ICF common queue space: (2 - 42 K) __ __
2. Define the subsystem type: __ 5

1 Intra	2 BSC IMS/IRSS
3 BSCEL	4 BSC CICS
5 BSC CCP	6 SNA Upline
7 SNA Peer	8 BSC 3270
9 SNA 3270	10 Finance

SSP-ICF common queue space: Specify the size, in multiples of 2 K bytes, of the common queue space. The common queue space requirements for each configuration of the CCP subsystem enabled are:

C = 81D + 82

where:
C = number bytes required for the common queue space
D = number defined session address for all CCP lines enabled

If ASCII is selected on any line, add 768 bytes to the common queue space requirements.

The common queue space value specified for the first subsystem that is enabled becomes the size of the common queue space. Be sure that the value specified for common queue space size takes into account the requirements of any other subsystem that might be active concurrently.

The size of the common queue space plus the total subsystem queue space sizes of all the enabled CCP subsystems cannot exceed 42 K bytes.

The default common queue space size is 4 K bytes.

Define the subsystem type: Specify a 5 for the BSC CCP subsystem.

Display 3.0 General Subsystem Parameters

```
3.0    General Subsystem Parameters

       1.  Location name:                          (8 characters)   _ _ _ _ _ _ _ _
       2.  Subsystem queue space:                       (0-40 K)                _ _
       3.  Subsystem support swappable:           (0-No   1-Yes)                  _
       4.  Maximum user record length:                 (1-4075)            _ _ _ _
```

Location name: Specify up to 8 characters for the name of the location associated with this configuration. The location name is used in some of the displayed message texts, and must be coded on the SESSION OCL statement. The default is the subsystem configuration member name. The location name refers to the name of the location with which communications is to take place.

Subsystem queue space: Specify the size, in multiples of 2 K bytes, of the subsystem queue space. The subsystem queue space requirements for each configuration of the CCP subsystem enabled is:

$$S = 3A(R + 21) + R + 86$$

where:

S = number bytes required for the subsystem queue space
A = number of sessions that will be active concurrently (including remote procedure starts)
R = maximum record length

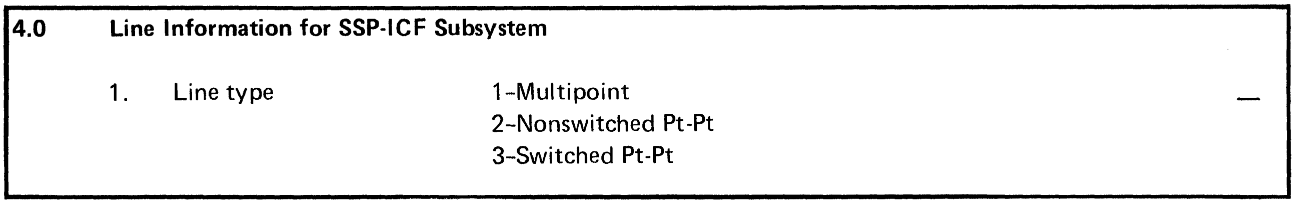
The size of the common queue space plus the total subsystem queue space sizes of all the enabled CCP subsystem configurations cannot exceed 42 K bytes.

The default subsystem queue space size is 4 K bytes.

Subsystem support swappable: Specify whether you want the subsystem to be swappable. Consider the total system performance, the size of the subsystem, and the amount of user main storage when determining whether you want the subsystem swappable. The CCP subsystem requires 8 K bytes of main storage.

Maximum user record length: Specify the maximum record length (1 through 4075 bytes) to be sent or received by any System/34 application program using this subsystem configuration. This length corresponds to the BLKL parameter on the TERMATTR statement in the CCP assignment set. The default is 1024 bytes.

Display 4.0 Line Information for SSP-ICF Subsystem



```
4.0     Line Information for SSP-ICF Subsystem

        1.   Line type              1-Multipoint                                    —
                                    2-Nonswitched Pt-Pt
                                    3-Switched Pt-Pt
```

Line type: Specify the line type that is suitable to your communications environment. There is no default line type. If you are using switched lines, see *Using Switched Lines* in this chapter for more information.

Display 5.0 BSC General Subsystem Parameters I

```
5.0     BSC General Subsystem Parameters I

        1.   EBCDIC/ASCII:                     (1-EBCDIC   2-ASCII)          _
        2.   Local station address:                        (2 hex)          _ _
        3.   Wait time:                            (1 - 999 seconds)      _ _ _
        4.   Transparency:                            (0-No   1-Yes)          _

        6.   Remote ID:
             _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
```

EBCDIC/ASCII: Specify the transmission code you require. The code you select must be compatible with the CCP system. The default is 1 (EBCDIC).

Local station address: Specify, in hexadecimal, the System/34 multipoint line address identified for this configuration. You can specify B (C2) through R (D9) for EBCDIC, or A (41) through Z (5A) for ASCII. This prompt is displayed only for multipoint line. Be sure to enter the hexadecimal equivalent in the code used.

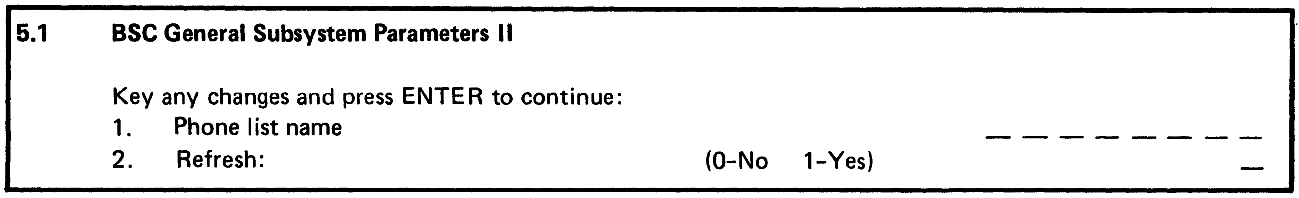
Wait time: Specify the number of seconds to wait for an input or output operation before a permanent error is indicated. The default is 999, which indicates an infinite wait time.

Transparency: Specify whether the EBCDIC data will be transmitted in transparent mode. If you specify 1 (yes) with ASCII data, an error message is displayed. This parameter should be consistent with the TRANSP parameter on the TERMATTR statement. The default is 0 (no).

Remote ID: Specify from 0 to 30 hexadecimal characters to be used for identification of the remote system on a switched line when the connection is established. The number of hexadecimal characters must be a multiple of 2. This field is left-justified with all unused positions filled with zeros. The hexadecimal characters cannot include any BSC control characters. Therefore, the following hexadecimal characters cannot be used: 01, 02, 03, 10, 1F, 26, 2D, 32, and 3D. This parameter must correspond to the IDEXSEND parameter on the BSCALINE statement.

Display 5.1 BSC General Subsystem Parameters II

The following prompt is displayed only if you specified a line type of *switched pt-pt* on display 4.0.



```
5.1      BSC General Subsystem Parameters II

         Key any changes and press ENTER to continue:
         1.   Phone list name                                        _ _ _ _ _ _ _ _
         2.   Refresh:                        (0-No   1-Yes)                       _
```

Phone list name: Specify the name of the load member that contains the list of phone numbers for the autocall feature or the list of numbers for the public data network (X.21 feature) to be called by the System/34. The load member must be in either #LIBRARY or in the same library as the configuration member.

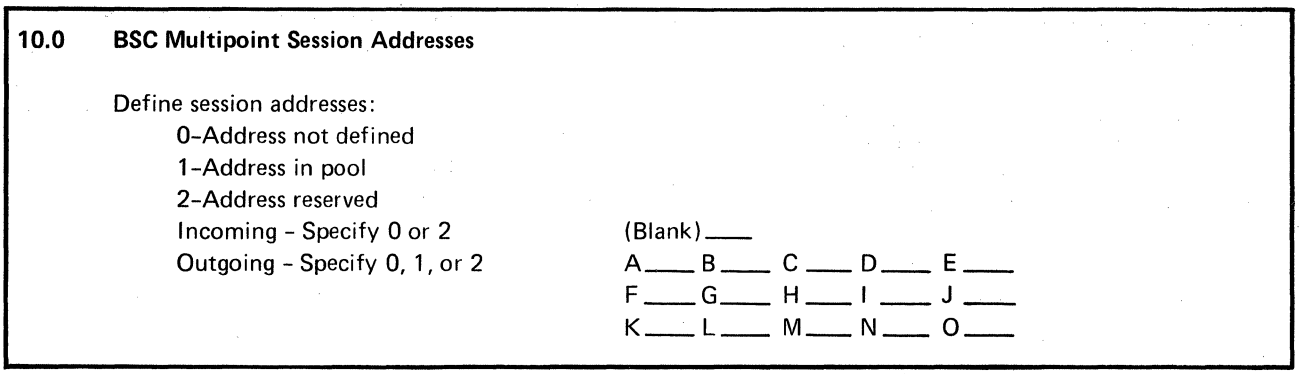
Use the *refresh* parameter to define how you want the list of numbers processed.

Refresh: Specify 1 (yes) if you want the list reinitialized (calling to begin with the first number on the list) after a successful call. If you enter a list name and do not enter a value for refresh, refresh-yes is assumed.

Specify 0 (no) if you do not want the list reinitialized after a successful call.

Note: See *Automatic Calling on Switched Lines* in this chapter for more information about how to use the phone list and refresh parameter.

Display 10.0 BSC Multipoint Session Addresses

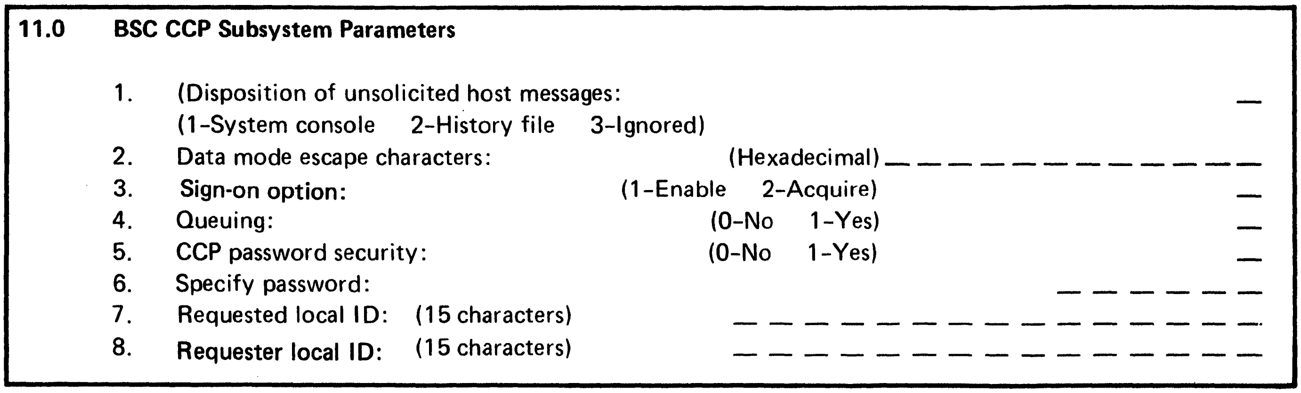


10.0 BSC Multipoint Session Addresses

Define session addresses:
0-Address not defined
1-Address in pool
2-Address reserved
Incoming - Specify 0 or 2 (Blank)____
Outgoing - Specify 0, 1, or 2 A____ B____ C____ D____ E____
F____ G____ H____ I____ J____
K____ L____ M____ N____ O____

Define session addresses: Specify the type of session address for each session on a multipoint line. Each address can be defined as a 0 (not used), 1 (in the address pool), or 2 (reserved). Reserved addresses must have the specific session address coded on the SESSION OCL statement (SESNADDR parameter). Up to 15 outgoing session addresses and one incoming address can be defined. The default for an outgoing session is 1 (in the pool); the default for the incoming session is 2 (reserved). The relation of session addresses to the CCP assignment set is shown in *CCP Assignment Set Considerations* later in this chapter.

Display 11.0 BSC CCP Subsystem Parameters



```
11.0    BSC CCP Subsystem Parameters

        1.   (Disposition of unsolicited host messages:                         _
             (1-System console   2-History file   3-Ignored)
        2.   Data mode escape characters:                (Hexadecimal)_ _ _ _ _ _ _ _ _ _ _ _
        3.   Sign-on option:                        (1-Enable   2-Acquire)      _
        4.   Queuing:                                  (0-No   1-Yes)           _
        5.   CCP password security:                    (0-No   1-Yes)           _
        6.   Specify password:                                          _ _ _ _ _ _
        7.   Requested local ID:   (15 characters)     _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
        8.   Requester local ID:   (15 characters)     _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
```

Disposition of unsolicited host messages: Specify how unexpected messages from the CCP system are handled.

1–System console: The messages are displayed at the system console and written to the history file.

2–History file: The messages are written to the history file, but are not displayed.

3–Ignored: The messages are discarded.

The default is 1 (system console).

Data mode escape characters: Specify in hexadecimal the six data mode escape characters (12 hex digits) to be used. These characters are used to cause CCP to accept the next input on this session as a command and not as data. The characters must be a sequence of characters that will not be sent as data, and must match the CCP generation of data mode escape characters (ESCAPE parameter on $EFAC macro). The default value is a string of six slashes (//////) specified in hexadecimal as 616161616161. The hexadecimal characters cannot include any BSC control characters. Therefore, the following hexadecimal characters are invalid: 01, 02, 03, 10, 1F, 26, 2D, 32, 37, and 3D.

Sign-on option: Specify when you want the CCP subsystem to send the sign-on command (/ON) to CCP. You can specify that the sign-on be sent after the subsystem is enabled when the S07 message is received from CCP, (thus saving line time when the acquire operation is issued by the application program), or you can specify that the sign-on be sent when the acquire is issued. If the sign-on is sent with the acquire operation, sign-off is performed with the release operation. If the sign-on is sent when the S07 message is received, the sign-off is performed when the CCP subsystem is disabled. If you are using a switched line, signing on during the acquire operation is required.

Queuing: Specify whether you want a System/34 application program to have its transaction request queued by CCP if the resources needed by the program are temporarily unavailable. This function is the same as if an operator using CCP had entered the /Q command. If you do not specify queuing, and the requested program is unavailable, the operation following the evoke operation will fail. If you do specify queuing and the program is unavailable, the request is queued, and the operation following the evoke does not complete until the program becomes available.

CCP password security: Specify 1 (yes) if CCP has password security active. If CCP does not have password security, specify 0 (no). The default is 0 (no).

Specify CCP password: This prompt is displayed if you specified 1 (yes) to CCP password security. You must specify the same password as that specified in the PASSWORD parameter of the SYSTEM statement in the CCP assignment set on the System/3.

Requested local ID: This prompt is displayed only if you specified a line type of switched point-to-point. Specify from 0 to 30 hexadecimal digits to be used for switched line identification with an incoming procedure request. The number of hexadecimal digits must be a multiple of 2. This field is left-justified with all unused positions filled with zeros. The hexadecimal characters cannot include any BSC control characters. Therefore, the following hexadecimal characters are invalid: 01, 02, 03, 10, 1F, 26, 2D, 32, 37, and 3D. This parameter corresponds to the IDEXRCV parameter in the BSCATERM statement for a COMMAND-NO terminal.

Requester local ID: This prompt is displayed only if you specified a line type of switched point-to-point. Specify from 0 to 30 hexadecimal digits to be used for switched line identification with an outgoing program request. The number of hexadecimal digits must be a multiple of 2. This field is left-justified with all unused positions filled with zeros. The hexadecimal characters cannot include any BSC control characters. Therefore, the following hexadecimal characters are invalid: 01, 02, 03, 10, 1F, 26, 2D, 32, 37, and 3D. This parameter corresponds to the IDEXRCV parameter in the BSCATERM statement for a COMMAND-YES terminal.

CCP GENERATION

The CPUMSG operand on the $EFAC statement must be YES to allow CCP to send S group messages.

The ESCAPE operand on the $EFAC statement must be specified to allow a remote release.

To facilitate dynamic assignment of outgoing session subaddresses within the subsystem, the SECURE operand on the $SEC statement must *not* be specified as USER.

The TYPE operand on the $EBSD statement must include CPU as a terminal type.

For information about CCP generation, see the *CCP System Reference Manual.*

CCP Assignment Set Considerations

1 The TRANSP (transparency) parameter of the TERMATTR statement must match the transparency value specified during CNFIGICF on the System/34.

2 The OUTPOLL parameter of the BSCALINE statement must be NO if the System/34 will transmit records longer than 256 bytes. (OUTPOLL-NO is assumed if you do not enter the parameter.)

3 The COMMAND parameter of the BSCATERM statement must be NO for sessions that will be started by CCP application programs, and YES for sessions started by System/34 application programs.

4 The ONLINE parameter of the BSCATERM statement must be specified with a value of NO. After CCP has been started, the desired session(s) should be varied on. This results in an S07 message being transmitted to the System/34 for each session that has been varied on.

5 The ICF parameter of the BSCATERM statement must be specified as Y, indicating to CCP that this terminal is a System/34 with the interactive communications feature.

6 The ENDMSG parameter of the PROGRAM statement should be specified with a value of YES for any program that may be started from the System/34. This results in the following messages being transmitted to the System/34 when the CCP program terminates or releases the session:

```
S01 PROG REL-PROCEED
S02 PROG REL-SHUTDOWN
S03 PROG END-PROCEED
S04 PROG END-SHUTDOWN
```

These messages are required by the CCP subsystem to maintain session status. Thus, if the CCP application program requires ENDMSG-NO, the CCP application program must send these messages as data to maintain valid session status.

7 The PGMDATA parameter of the PROGRAM statement must be YES if the program is to be evoked with data, and NO if it is to be evoked without data.

The following is a sample CCP assignment set with the above parameter highlighted to show how they are coded.

SYSTEM/3 MODEL 15D
SAMPLE ASSIGNMENT SET FOR USE WITH SYSTEM/34 SSP-ICF CCP SUBSYSTEM

```
// SET ID-A,ACTION-REPLACE,DFLTEXEC-YES
// SYSTEM MINUPA-60K,MINTPBUF-12240,PQMREQL-80,DFFPACK-PROGRAM,
          MAXCHAIN-15,DFFINDX-YES,PRINTER-YES,COMMANDL-80
// TERMATTR ATTRID-3,TRANSLATE-NO,UPCASE-NO,DATAFORM-RECORD,
// DFF3270-NO,BLKL-800,RECL-800,TRANSP-NO ——— 1
// BSCALINE TYPE-CS,OUTPOLL-NO,LINENUM-3,POLLLIST-'10,11,12,13,16,
//                    2                       17,18,19,20,21'
                                                              3
// BSCATERM TERMID-10,TYPE-CPU,ATTRID-3,COMMAND-NO,ONLINE-NO,
//    ADDRCHAR-*E2E24040*,POLLCHAR-*C2C24040*,ICF-Y    3
// BSCATERM TERMID-11,TYPE-CPU,ATTRID-3,COMMAND-YES,ONLINE-NO,
//    ADDRCHAR-*E2E2C1C1*,POLLCHAR-*C2C2C1C1*,ICF-Y          4
// BSCATERM TERMID-12,TYPE-CPU,ATTRID-3,COMMAND-YES,ONLINE-NO,
//    ADDRCHAR-*E2E2C2C2*,POLLCHAR-*C2C2C2C2*,ICF-Y
// BSCATERM TERMID-13,TYPE-CPU,ATTRID-3,COMMAND-YES,ONLINE-NO,
//    ADDRCHAR-*E2E2C3C3*,POLLCHAR-*C2C2C3C3*,ICF-Y — 5
// BSCATERM TERMID-16,TYPE-CPU,ATTRID-3,COMMAND-YES,ONLINE-NO,
//    ADDRCHAR-*E2E2C6C6*,POLLCHAR-*C2C2C6C6*,ICF-Y
// BSCATERM TERMID-17,TYPE-CPU,ATTRID-3,COMMAND-YES,ONLINE-NO,
//    ADDRCHAR-*E2E2C7C7*,POLLCHAR-*C2C2C7C7*,ICF-Y
// BSCATERM TERMID-18,TYPE-CPU,ATTRID-3,COMMAND-YES,ONLINE-NO,
//    ADDRCHAR-*E2E2C8C8*,POLLCHAR-*C2C2C8C8*,ICF-Y
// BSCATERM TERMID-19,TYPE-CPU,ATTRID-3,COMMAND-YES,ONLINE-NO,
//    ADDRCHAR-*E2E2C9C9*,POLLCHAR-*C2C2C9C9*,ICF-Y
// BSCATERM TERMID-20,TYPE-CPU,ATTRID-3,COMMAND-YES,ONLINE-NO,
//    ADDRCHAR-*E2E2D1D1*,POLLCHAR-*C2C2D1D1*,ICF-Y
// BSCATERM TERMID-21,TYPE-CPU,ATTRID-3,COMMAND-YES,ONLINE-NO,
//    ADDRCHAR-*E2E2D2D2*,POLLCHAR-*C2C2D2D2*,ICF-Y

// TERMNAME NAME-ICF,TERMID-10
// TERMNAME NAME-ICFA,TERMID-11
// TERMNAME NAME-ICFB,TERMID-12
// TERMNAME NAME-ICFC,TERMID-13
// TERMNAME NAME-ICFF,TERMID-16
// TERMNAME NAME-ICFG,TERMID-17
// TERMNAME NAME-ICFH,TERMID-18
// TERMNAME NAME-ICFI,TERMID-19
// TERMNAME NAME-ICFJ,TERMID-20
// TERMNAME NAME-ICFK,TERMID-21

// DISKFILE NAME-ITEMICF,ORG-I,RECL-256,KEYL-23,KEYPOS-33
// DISKFILE NAME-ORDRICF,ORG-I,RECL-760,KEYL-6,KEYPOS-37
                          6
// PROGRAM NAME-ICII,ENDMSG-YES,PGMDATA-YES,EXECFIND-YES,
//          FILES-'ITEMICF/IR'             7
// PROGRAM NAME-ICOI,ENDMSG-YES,PGMDATA-YES,EXECFIND-YES,
//          FILES-'ORDRICF/IRA/SHR'          7
// PROGRAM NAME-ICOP,ENDMSG-YES,PGMDATA-NO,EXECFIND-YES,
//          FILES-'ITEMICF/IRU,ORDRICF/IRUA/SHR'
```

STARTING AND ENDING THE CCP SUBSYSTEM

The ENABLE procedure is used to start the CCP subsystem on a particular BSC line. After being enabled, the subsystem monitors the line and waits for application program requests. If specified during configuration, the sign-on command (/ON) and the queue command (/Q) are sent to the CCP following the enable.

The DISABLE procedure stops the subsystem. When a disable is performed, the association between the subsystem and the BSC line is broken, and no further activity occurs on the line. If the subsystem was active on a multipoint line using the multiline communications adapter, the adapter will continue to respond to polling after the disable has been completed.

The formats of the ENABLE and DISABLE procedure commands are in Chapter 2.

STARTING CCP SUBSYSTEM APPLICATIONS

System/34 CCP subsystem applications can be started by a display station operator entering a procedure command or by a request from the remote system. Procedures that are started by the System/34 operator must contain a SESSION OCL statement for each session to be started. Requests from the CCP system to start a procedure must be in a special format. The following sections describe the SESSION statement and the incoming procedure start requests.

SESSION OCL Statement

The format of the SESSION OCL statement for the CCP subsystem is:

$$\text{// SESSION LOCATION-name, SYMID-session-id } \left[\text{, SESNADDR-x}\right]$$

$$\left[\text{,PHONE-name}\right] \left[\text{,REFRESH}\left\{\begin{array}{c}\underline{\text{YES}}\\ \text{NO}\end{array}\right\}\right] \left[\text{,RESTORE-}\left\{\begin{array}{c}\text{YES}\\ \underline{\text{NO}}\end{array}\right\}\right]$$

Note: If the application program is a BASIC program, the SESSION statement might not be required. See Chapter 4 for more information.

LOCATION: Specifies the location name associated with this session. The location name is defined during subsystem configuration, and refers to the name of the location with which communication is to take place.

SYMID: Specifies the symbolic ID of the session with which this OCL statement is associated. The symbolic ID must be two characters, with the first character numeric (0 through 9) and the second character alphabetic (A through Z, #, $, or @). This is the same ID that the application program uses when referring to this session. This ID is the equivalent of the symbolic display station ID as specified on the WORKSTN OCL statement. This parameter has no default.

SESNADDR: Specifies the address to be used by this session on a multipoint line. This address must be an alphabetic character from A through O, and must have been configured during subsystem configuration (CNFIGICF). If you do not specify this parameter, the CCP subsystem assigns an address from the address pool.

PHONE: Specifies the name of the load member that contains the list of phone numbers for the autocall feature or the list of numbers for the public data network (X.21 feature) to be called by the System/34. The load member must be in either the current user library or in #LIBRARY. If you do not specify a list name, the name specified during subsystem configuration or the name specified when the subsystem is enabled is used.

Use the REFRESH parameter and the RESTORE parameter to define how you want the list of numbers processed.

REFRESH: Specifies if you want the list reinitialized (calling to begin with the first number on the list) after a successful call. If you enter a list name and do not enter a value for REFRESH, REFRESH-YES is assumed.

Specify NO if you do not want the list reinitialized after a successful call.

RESTORE: Specifies if you want the list used in a previous job step reinitialized before executing the current job step.

Specify NO if you do not want the list used in a previous job step reinitialized.

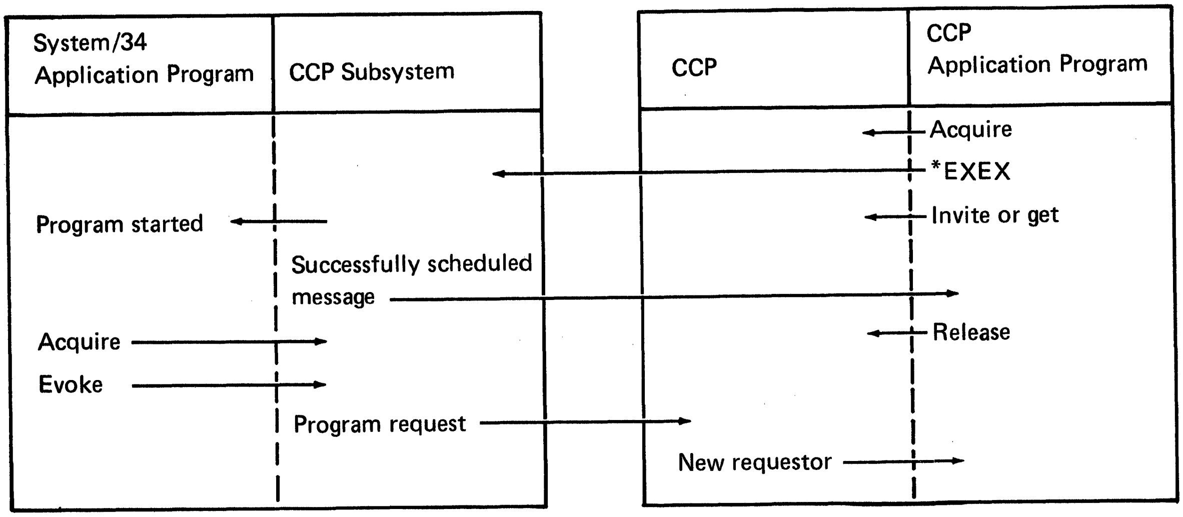
If you specify a list and do not specify RESTORE, RESTORE-NO is assumed.

Note: See *Automatic Calling on Switched Lines* in this chapter for more information about how to use the PHONE, REFRESH, and RESTORE parameters.

Incoming Procedure Start Requests

For CCP applications to initiate programs on System/34, the CCP application must transmit an *EXEC or an *EXEX request. The *EXEC request is only supported on a switched line. The format of the *EXEX and *EXEC requests is in Chapter 2.

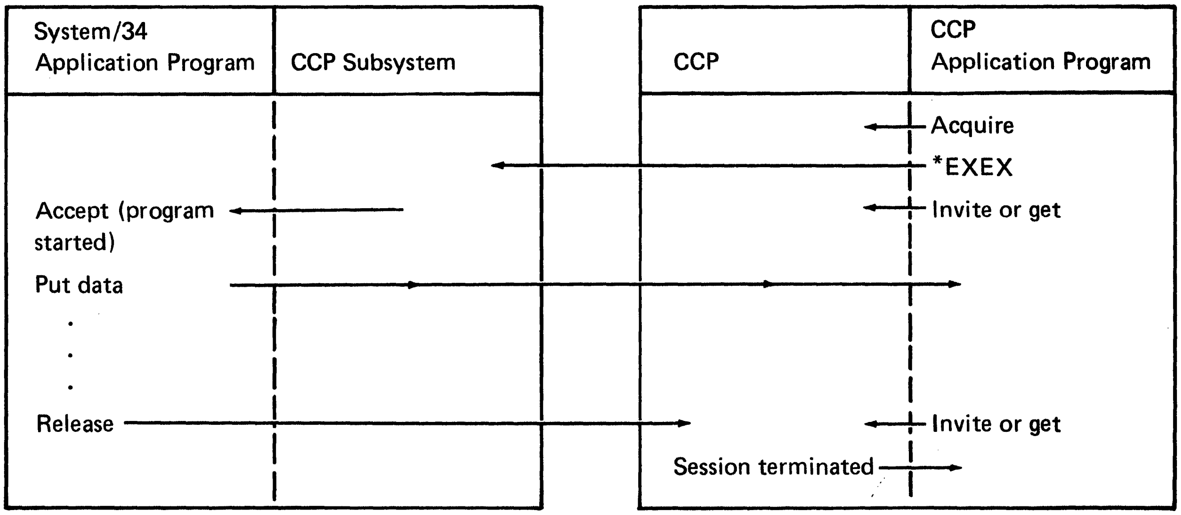
On a multipoint line, the procedure request must be transmitted to the subaddress defined to the CCP subsystem as the incoming address. The CCP application program first acquires the subaddress and then transmits the procedure request. The CCP application must then do an invite or get to this address. If the procedure is successfully scheduled to be initiated, the CCP subsystem transmits a successfully scheduled message (identified by SYS-8190 beginning in position 16), which satisfies the CCP get or invite. If the procedure cannot be successfully scheduled, a failure message is transmitted to the CCP application. No data except that with the procedure request can be transmitted to the System/34. For the two programs to communicate further, the CCP application must release the session, and the System/34 program must acquire another session and issue an evoke. This evoke can attach to the program that originally sent the request to the System/34. This process is illustrated as follows:



When using the *EXEC form of procedure request, the following must be taken into consideration:

- The CCP application program must do an invite or get operation as the next operation to the session after transmitting the request. The CCP application receives data or a failure message as a result of the invite or get.
- The CCP application must not send a new *EXEC or *EXEX request before the session ends.
- The System/34 application program must send data after its initial accept operation to satisfy the invite issued by the CCP application program.

This process is illustrated as follows:



OPERATION CONSIDERATIONS

The following sections describe the operations supported by the CCP subsystem. A complete chart of all interactive communications operations and the subsystems that support them is in each language chapter. The chart also shows the keyword or format name used to code each operation. More information about how an operation is coded is in the appropriate programming language chapter.

Whether an operation completes successfully or unsuccessfully, a return code is given to the application program. All of the return codes that are valid for this subsystem are described at the end of this chapter. A summary chart in Appendix A lists all the return codes and the subsystems for which they are valid.

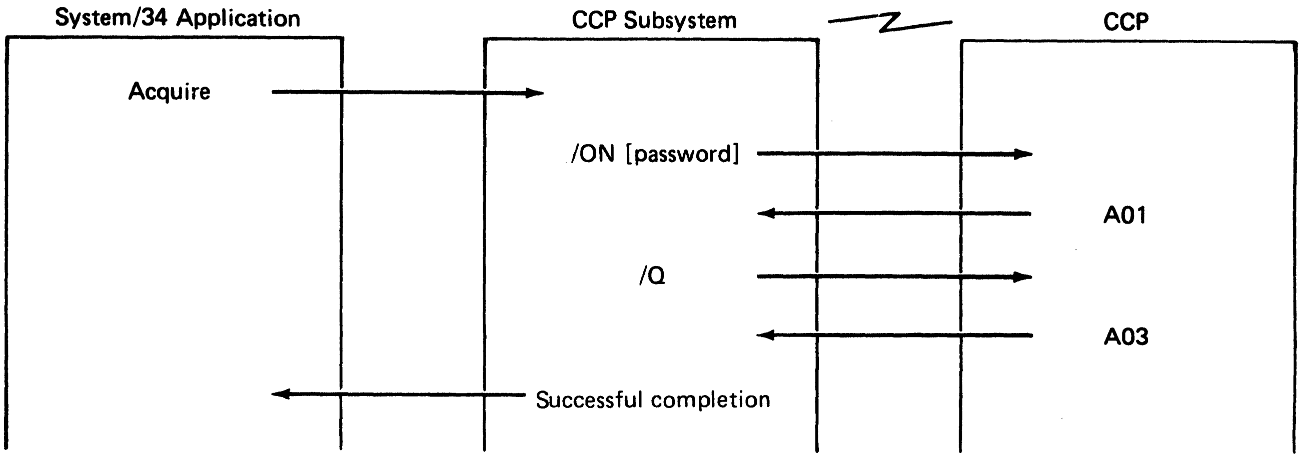
Acquire Operation

The acquire operation is used to start a session. When the acquire operation is issued by an application program on a multipoint line, a session address (CCP device address) must be assigned. The CCP subsystem either acquires the session address that was specified on the SESSION OCL statement (SESNADDR parameter) or assigns an address from the address pool. No session address is assigned on a point-to-point line.

The sign-on command (/ON) must be transmitted to sign-on to CCP. The sign-on command can be transmitted automatically when the S07 message is received from CCP, if so specified during configuration. If the sign-on command was not transmitted automatically, the subsystem transmits it when the acquire operation is issued.

When CCP receives the *sign-on* command, CCP sends a response. The CCP subsystem determines whether this response is an A01 message indicating that the sign-on was successful. If the response is not A01, the subsystem returns control to the application program with an acquire failed return code. If the response is A01, and queuing was specified during subsystem configuration, the subsystem transmits a /Q command to CCP. If the response is A01 and queuing was not specified, the subsystem returns control to the application program with a successful return code.

When CCP receives the /Q command, CCP transmits a response. The subsystem determines whether this response is A03 (request successfully queued). If the response is not A03, the subsystem returns control to the application program with an acquire failed return code. If the response is A03, the subsystem returns control to the application program with a successful return code.



Evoke Operation

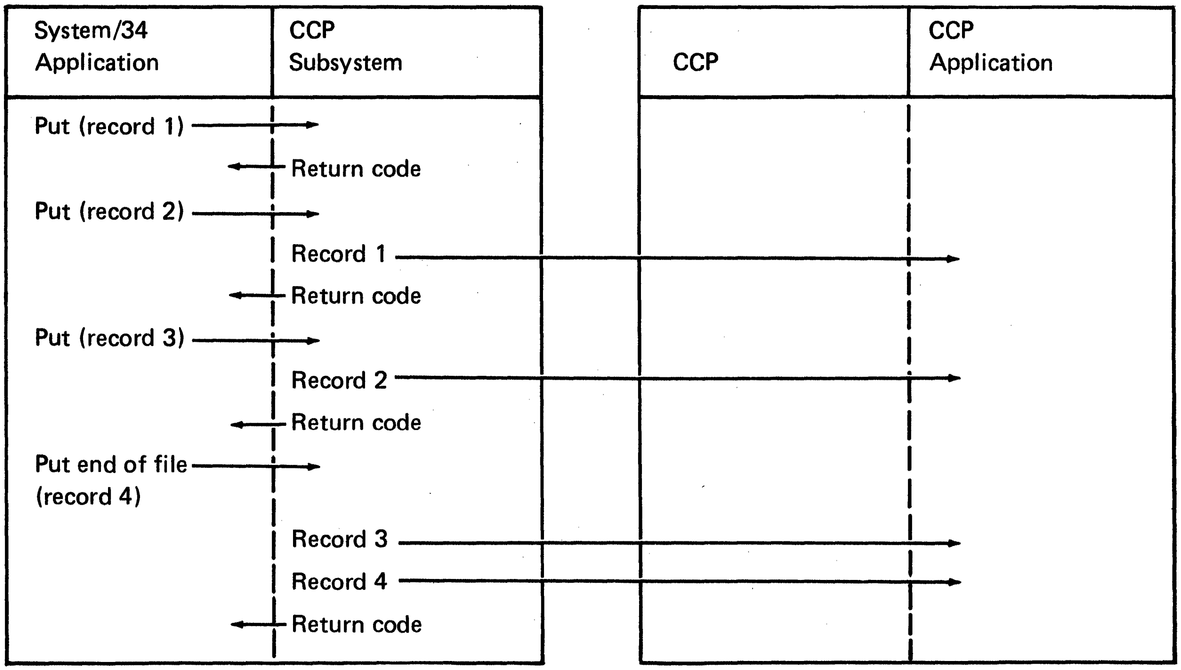
An evoke operation is used to initiate a task on the CCP system. When the evoke operation ($$EVOK or $$EVOKNI) is issued by the application program, the CCP subsystem retrieves the program name contained in the evoke parameter list, and generates the program request to be transmitted to CCP. The CCP subsystem then transmits the request, optionally with data. When the CCP subsystem successfully transmits the request, the subsystem returns control to the application program.

The evoke then get operation (assembler only) is identical to the evoke operation except that the application program does not receive control until the CCP subsystem receives the first data record from the CCP application or a failure message.

Put Operation

The put operation is used to pass data from the System/34 application program to the CCP application program. The put operation can be issued alone ($$SENDNI) or combined with an invite operation ($$SEND). If the put operation is combined with an invite operation, the record, followed by an EOT, is sent. Then a record is received by the subsystem. For a BASIC, COBOL, or RPG II program, the input data is made available to the application program on the next accept (READ) operation. Assembler users can choose to wait for the input data by using a put then get operation.

If the put operation is not combined with an input operation, the current put operation is not started until the previous put operation is complete. If the previous put operation failed, the current put is not performed, and the application program is informed via the appropriate return code. The put end of file ($$SENDE) operation is used to terminate a series of put operations that do not request input.



Input Operations

The input operations for the CCP subsystem are invite, get, and accept. The invite operation can be issued only as a combined operation with a put or evoke operation ($$SEND, $$EVOK) in BASIC, COBOL, or RPG II. Assembler language users can issue an invite operation explicitly. Either a get or invite operation signals the subsystem to obtain data on the session for the application program. A get operation causes the application program to wait for the data to be available. When a program issues an invite operation, it receives the data with the next accept operation. The accept operation allows input from any previously invited session.

The end of file indication (EOT) is not received by the CCP subsystem with the input data. Therefore, the return code that indicates end of file is 0300 (no data received and EOT received).

Another return code that can be received for an input operation is 0308. This return code indicates that no data was received and that communications with the CCP application program has ended. Depending on the CCP application, the System/34 application program could issue input operations until the 0308 return code is received.

Request to Change Direction Operation

The request to change direction operation ($$RCD) indicates that the System/34 application program wants to transmit data. The operation can be issued only when the program is receiving. After issuing the request to change direction operation, the System/34 application program should continue to receive data until it receives a return code indicating that the CCP application program is ready to receive data. The System/34 application program can then begin sending data.

Release Operation

The release operation breaks the logical connection between the application program and the CCP subsystem. When the release operation is issued, the CCP subsystem ends communication with the application program and makes the session address available for another session. Even if the application program running under CCP has already ended (normally) before the release is issued, the release is considered successful.

If the System/34 application program issues a release operation before the CCP application program has ended, the CCP subsystem sends a data mode escape sequence to the System/3. The CCP application program then receives a return code indicating that the /RELEASE command was received and that the session address is no longer available for data transfer.

If /ON was sent during the acquire operation, the subsystem transmits the /OFF command to CCP following the transmission of the /RELEASE command. If the /ON was sent when the S07 message was received, another session can be acquired without sending /ON, thereby saving time on the communications line.

End of Session Operation

The end of session operation ($$EOS) always results in a normal return code. The session is always terminated by the end of session operation. If the session is still communicating when the end of session operation is issued, the transmission is abnormally terminated by the CCP subsystem, and abnormal termination of the CCP application program could result.

Get Attributes Operation

The get attributes operation for assembler users can be issued at any time to determine the status of a session.

Set Timer Operation

The set timer operation ($$TIMER) results in a timer expired return code (0310) after a time interval specified in hours, minutes, and seconds has elapsed.

PROGRAMMING CONSIDERATIONS

CCP Commands

The following CCP commands are generated internally by the CCP subsystem.

- /ON [password] — As the result of receiving the S07 message or an acquire operation
- /Q — Optionally as the result of receiving the A01 message
- Program request — As the result of an evoke operation
- Data mode escape — As the result of a release operation or the System/34 program ending
- /RELEASE — As the result of a release operation or the System/34 program ending
- /OFF — As the result of the DISABLE procedure or a release operation

Messages From CCP

The CCP subsystem monitors incoming data for CCP messages. If a CCP message is received in response to a CCP command that was generated by the CCP subsystem and if the response is as expected, the subsystem continues normally. If the response is not as expected, the subsystem returns control to the application program with the appropriate error return code.

If a CCP message other than PROG END PROCEED, PROG CANCELLED PROCEED, or CCP SHUTDOWN is received and the message is not a response to a CCP command, the subsystem returns control to the program with a message waiting return code (831A). The application program error subroutine must issue another input operation, check the message, and take appropriate action. The messages that CCP can send are described in the *System/3 CCP Message Manual.*

CCP can send all A, E, R, and S group messages. Therefore, the first 4 bytes of user data sent by the CCP application must not be:

A numeric numeric blank
E numeric numeric blank
R numeric numeric blank
S numeric numeric blank

For example, a data record of *A03 other data* must not be sent.

System operator messages sent by the /MSG command to the System/34 are directed to the destination specified during subsystem configuration for unsolicited host messages.

Using Switched Lines

On switched lines, you can call a remote location manually or call a remote location automatically (if you have the autocall feature).

Note: If you have the switched X.21 feature, the System/34 calls automatically. You cannot call manually with the X.21 feature.

Manual Calling on Switched Lines

If you want the operator to call a remote location manually, do not specify a phone list name. The message OPERATOR DIAL REQUIRED is displayed on the system console when the operator is to make the call.

Automatic Calling on Switched Lines

The System/34 can call remote locations automatically on switched lines using either the autocall or switched X.21 feature. To call a remote location(s) automatically, do the following:

- Specify the correct configuration parameters.
- Create a list of phone numbers for the autocall unit or a list of numbers for the public data network.
- Enter the name of the list when you configure the subsystem, when you enable the subsystem, or when you write the SESSION OCL statement.
- Enable the subsystem on an autocall or switched X.21 line.

Configuring the Subsystem for the Autocall Feature: When you use the autocall feature and specify a phone list name, the System/34 calls the remote location(s) automatically. However, if the operator enables the subsystem on a manual call line, for example, if an autocall line is not available, a message is displayed asking the operator to dial the remote location manually.

Configuring the Subsystem for the X.21 Feature: When you use the switched X.21 feature, the System/34 calls the location automatically when you enter the name of the list that contains the numbers for the public data network with the phone list parameter.

Creating a Phone List for Autocall or a List of Numbers for the Public Data Network: The list for autocall or the list for the public data network contains the number or numbers you want the System/34 to call. You create the list and store it in a load member using the DEFINEPN or DEFINX21 procedure. These procedures are described in the *System Support Reference Manual.*

If the list name is specified during subsystem configuration or when the subsystem is enabled, the load member containing the list must be in either #LIBRARY or in the same library as the configuration member. If the list name is specified on the SESSION statement, the load member must be in either #LIBRARY or the current user library.

Following is an example of how the System/34 handles a list. Although the example shows a list for the autocall feature, the System/34 handles a list for the public data network (X.21 feature) in the same manner except where noted.

Phone List Example: The System/34 calls the numbers in the order listed. When a number is called, the call might be unsuccessful. A retry count specified during the DEFINEPN procedure is associated with each number in the list. When a call is unsuccessful, the retry count is decremented by 1, and the next number on the list is called as shown in the following example:

Note: If you are using the X.21 feature, the retry count is decremented depending upon the reason for the unsuccessful call. A call progress signal (CPS) gives the reason for the unsuccessful call. The call progress signal is displayed on the system console with the unsuccessful call message. If the call progress signal begins with a 4, 5, or 7, the number is attempted only once, the retry count is set to 0, and the number is marked as unsuccessfully called.

Phone Number	**Retry Count**	
8672906	5 -1 = 4	(The retry count is decremented,
6286500	1	and the next number is called.)
6280363	3	

This number will not be called again until the other numbers in the list have been called or until the phone list is reinitialized.

If the retry count reaches 0, a message is displayed on the system console indicating that the call was unsuccessful, and the number is marked as unsuccessfully called.

Phone Number	**Retry Count**	
8672906	5 -1 = 4	
6286500	1 -1 = 0	(Unsuccessfully called)
6280363	3	

When a number is marked as unsuccessfully called, it is not called again until the phone list is reintialized.

If a number is successfully called, an operator message is displayed on the system console, the line connection is established, and the number is marked as successfully called.

Phone Number	Retry Count	
8672906	5 -1 = 4	
6286500	1 -1 = 0	(Unsuccessfully called)
6280363	3	(Successfully called)

The number is not called again until the list is reinitialized.

In the previous example, the first number in the list has not been marked as called. Therefore, if an application program uses the phone list without reinitializing it, the first number is the only number that can be called. If the call is successful, the number is marked as successfully called.

Phone Number	Retry Count
8672906	4 (Successfully called)
6286500	0 (Unsuccessfully called)
6280363	3 (Successfully called)

The phone list must now be reinitialized to be used again. If the list is not reinitialized and an application program attempts to use the list, a message PHONE LIST EXHAUSTED is displayed on the system console and a return code of xx86 is returned to the application program.

In the previous example, if the call to the first number in the list is unsuccessful and the retry count reaches 0, the number is marked as unsuccessfully called, a message NO NUMBERS REACHED is displayed on the system console, and a return code of xx85 is returned to the application program. The phone list must be reinitialized to be used again.

Phone Number	Retry Count
8672906	0 (Unsuccessfully called)
6286500	0 (Unsuccessfully called)
6280363	3 (Successfully called)

When the list is reinitialized, all retry counts are set to the counts specified during the DEFINEPN procedure and calling begins with the first number in the list when the list is used again.

The specification of the REFRESH and RESTORE parameters indicates how the list is reinitialized. The REFRESH parameter can be specified during configuration, during ENABLE, or on the SESSION statement. The RESTORE parameter can be specified on the SESSION statement only. Following is a description of the REFRESH and RESTORE parameters.

REFRESH Parameter

If you specify REFRESH-YES or do not specify REFRESH (the default is YES), the list is reinitialized after the first successful call or after all numbers in the list have been marked as unsuccessfully called.

If you specify REFRESH-NO, the list is not reinitialized after a successful call. The list is reinitialized as follows:

- After the PHONE LIST EXHAUSTED message is displayed

- After the NO NUMBERS REACHED message is displayed

- As specified by the RESTORE parameter

RESTORE Parameter

If you specify RESTORE-YES on the SESSION OCL statement, the list specified is reinitialized prior to executing the current step in the procedure. The current user library is searched first for the list. If the list is not found, the system library is searched.

If you specify RESTORE-NO, the list is not reinitialized prior to executing the current step in the procedure. The default is NO.

IF LISTDONE Conditional Expression

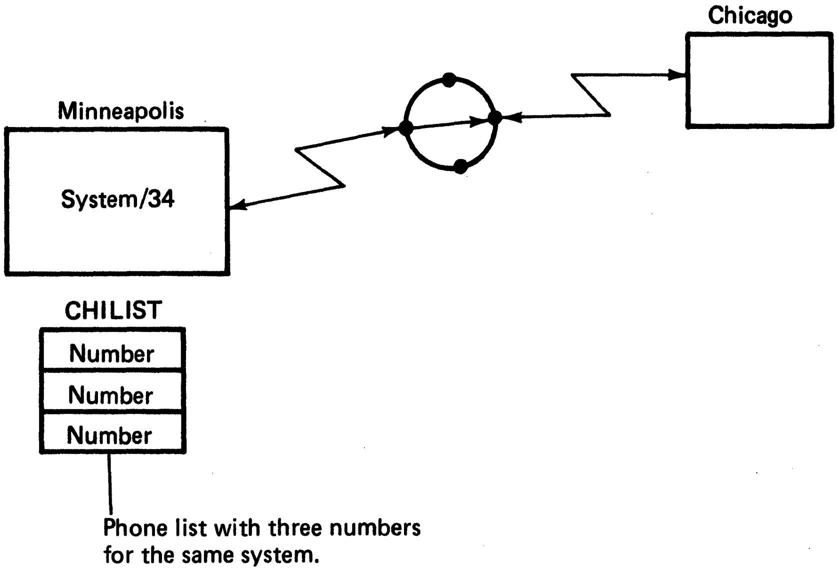
You can use the IF LISTDONE conditional expression to determine if all of the numbers on a list used by a previous job step have been called. If all numbers in the list are marked as successfully called or unsuccessfully called (retry counts are 0), the expression is true. The expression is false if REFRESH-YES was specified on the SESSION OCL statement and any number in the list was successfully called. The expression can only be used to test lists that have been specified on a SESSION statement. If the name of the list tested was not specified on the SESSION statement, the expression is false.

Note: In some countries a delay is required before a call can be placed when you use the autocall feature. If this delay is required in your country, specify the delay when you create the phone list using the DEFINEPN procedure. See the *System Support Reference Manual* for information about the DEFINEPN procedure and the delay parameter.

The following examples show how you can use the *refresh* parameter, the *restore* parameter, and the *IF LISTDONE* statement.

Example of Calling One Location

In this first example, the System/34 calls one location many times. A list name was entered on display 5.1 of the CNFIGICF procedure. The phone list in this example contains three numbers for the same remote system in Chicago. When the remote system is called, it is not important which number is called successfully, but we want the System/34 to begin calling with the first number in the list each time the program uses the list. To do this, 1 (yes) was specified for the refresh parameter on display 5.1.



The list and setup in this example can be used for either batch or interactive communications with the remote system.

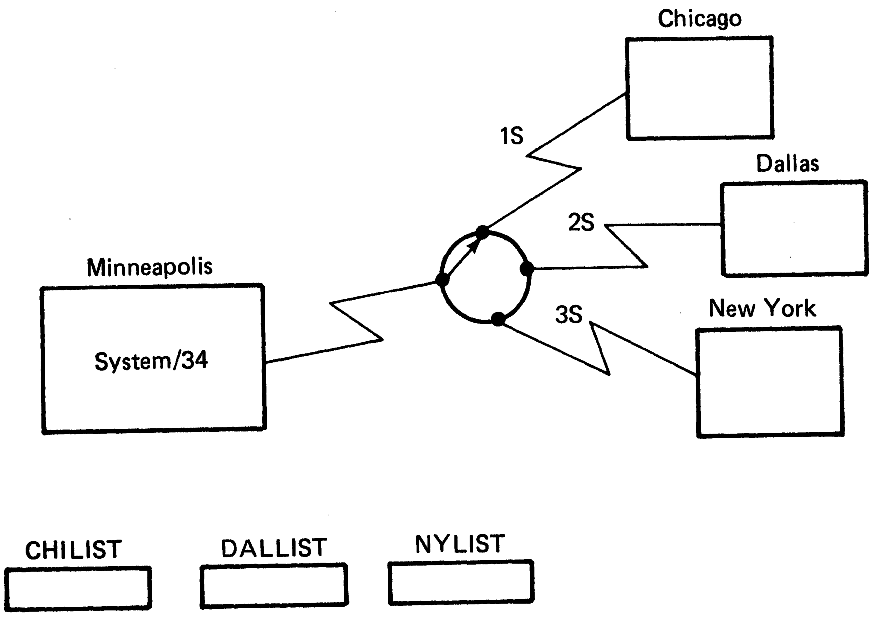
Example of Calling Multiple Locations (Refresh-Yes)

In this example, an MRT program is used for interactive communications with three remote systems.

One phone list was created for each remote system with one number in each list. The phone list names and refresh parameters are specified on the SESSION OCL statements as follows:

```
// LOAD MRTPROG
// SESSION LOCATION-MULTLOC,SYMID-1S,PHONE-CHILIST
// SESSION LOCATION-MULTLOC,SYMID-2S,PHONE-DALLIST
// SESSION LOCATION-MULTLOC,SYMID-3S,PHONE-NYLIST
// RUN
```

For example, the program calls the system in Chicago (session 1S) using the list CHILIST. When the System/34 program processes the data from Chicago, it determines that additional data is needed. The program then calls Dallas (session 2S) using the phone list DALLIST to find the additional data. The data was not at the Dallas location, so the program calls New York (session 3S) using phone list NYLIST and gets the data it needs. After processing the data, the program calls Chicago again and sends the results to the Chicago system.

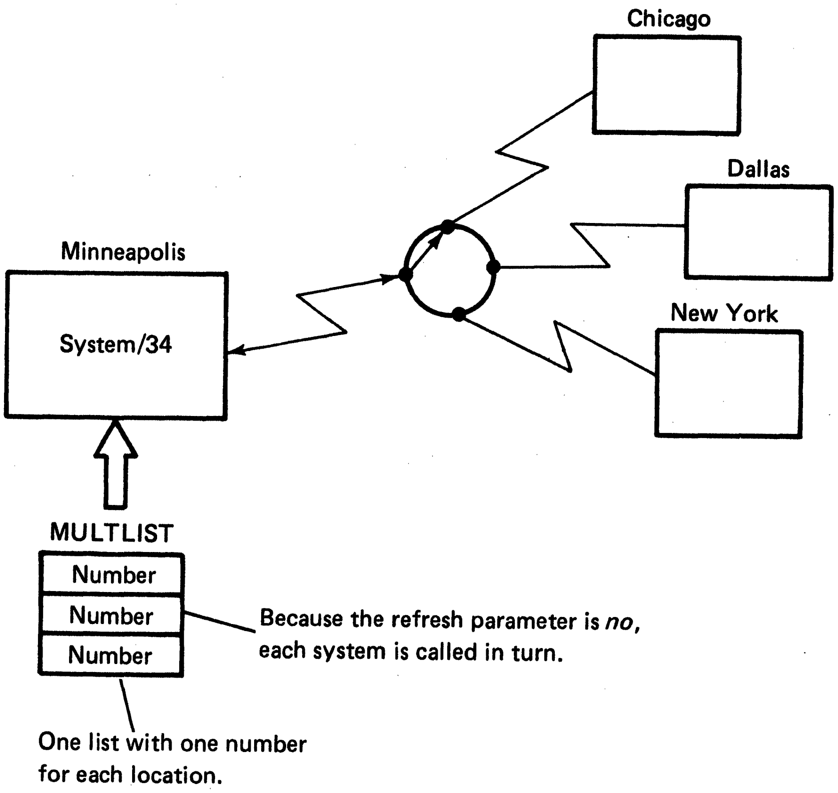


One list for each location.

Example of Calling Multiple Locations (Refresh-No)

In the two previous examples, the refresh parameter was set to *yes*. In this example, the parameter is set to *no*. One phone list is used, which was specified on display 5.1 of the CNFIGICF procedure. The refresh parameter was also set to 0 (no) on this display.

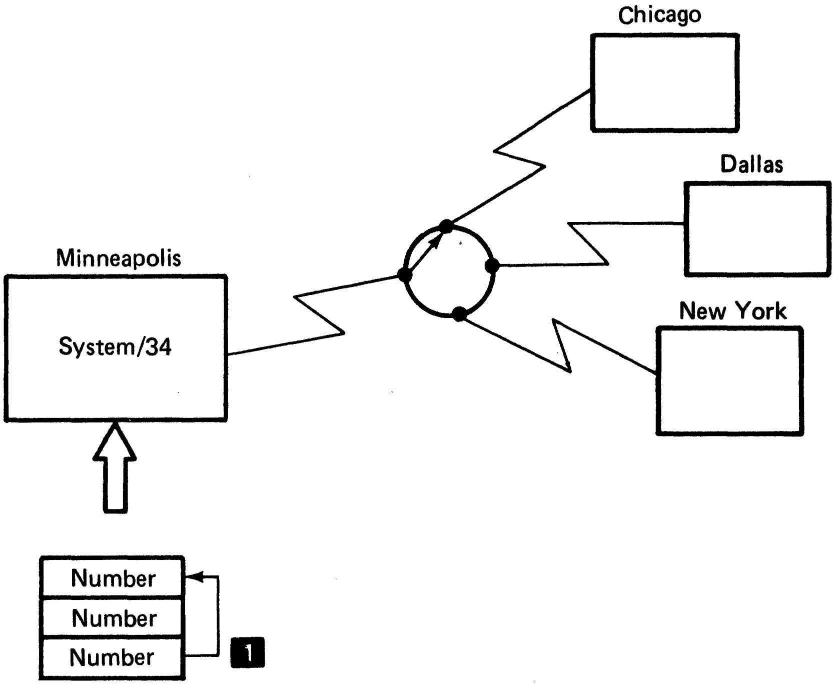
The program in this example loops through the list calling each system in turn to send one or more files of data to each system. The program checks for return code 8285 (NO NUMBERS REACHED) or 8286 (PHONE LIST EXHAUSTED) to determine when all numbers on the list have been called or tried.


Chicago
Dallas
Minneapolis
System/34
New York
MULTLIST
Number
Number
Number
Because the refresh parameter is *no*, each system is called in turn.
One list with one number for each location.

Example of Calling Multiple Locations (RESTORE-YES)

In this example, the phone list is reinitialized by entering RESTORE-YES on the SESSION OCL statement. Two programs (A and B) in one procedure use the same list to call the same remote system. If program A ends before all numbers have been called, the list is not reinitialized. To ensure that the list is reinitialized before program B uses the list, RESTORE-YES is entered on the SESSION statement. This ensures that the first number on the list is called when program B uses the list. For example:

```
// LOAD PROGA
// SESSION LOCATION-MULTLOC,SYMID-1S,
// PHONE-MULTLIST,REFRESH-NO
// RUN
// LOAD PROGB
// SESSION LOCATION-MULTLOC,SYMID-1S,
// PHONE-MULTLIST,REFRESH-NO,RESTORE-YES
// RUN                                      |
                                            1
```



The list is reinitialized before the next program uses it.

Example of Using the IF LISTDONE OCL Statement

You can also use the IF LISTDONE OCL statement to test the status of the list if your program does not test for a list done return code. This example shows how you can use the IF LISTDONE statement to check the list status.

Each time the program is loaded, it calls the next system, transmits or receives one or more files of data, and ends. If the list is done, the procedure also ends; otherwise, the program is loaded again and the next system is called.

```
// TAG TOP
// LOAD PROGA     1                       2             3
// SESSION LOCATION-MULTIPLE,SYMID-1S,PHONE-MULTLIST,REFRESH-NO
// RUN
// IFF LISTDONE-MULTLIST GOTO TOP  4
```

1 Each location is called, one at a time.

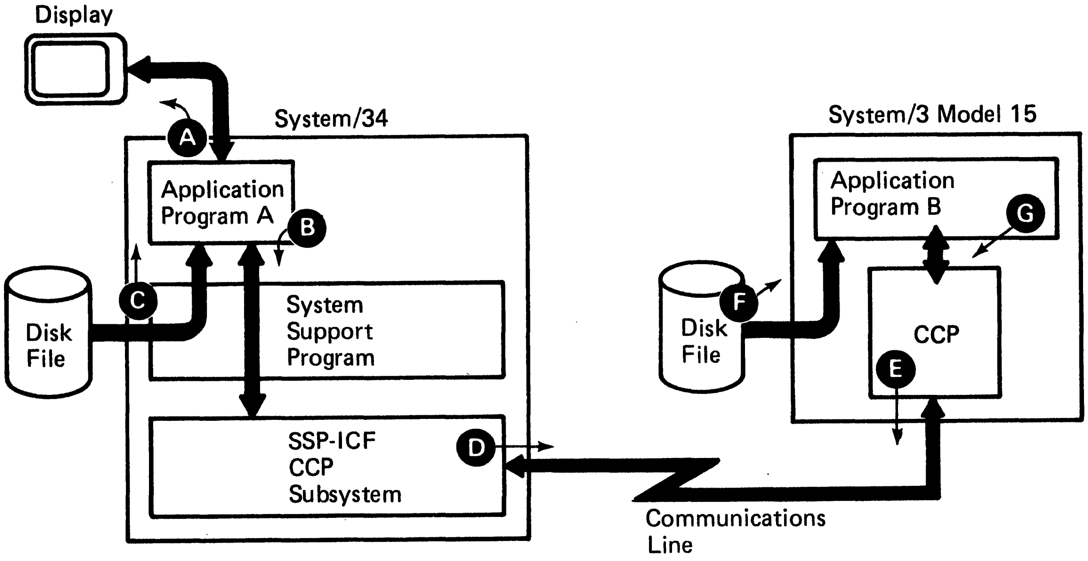
2 The list contains one number for each location.

3 The parameter REFRESH-NO allows each number to be called.

4 If all numbers have not been called, go to TOP.

How to Write Programs that Use the CCP Subsystem

The following inquiry application is used in the programming examples:



1. Application program A (in the System/34) displays a prompt asking an operator to enter an item number requesting the stock status for the item Ⓐ.

2. When the operator enters the item number, program A reads the number and searches file A (the local file) for the item Ⓒ.

3. If the item is found in the local file, program A displays the stock status on the screen Ⓐ.

4. If the item is *not* in the local file, program A uses the BSC-CCP subsystem to send the item number to the System/3 Ⓑ and Ⓓ.

5. Program B (in the remote system) uses the item number to search the remote file for the item Ⓕ.

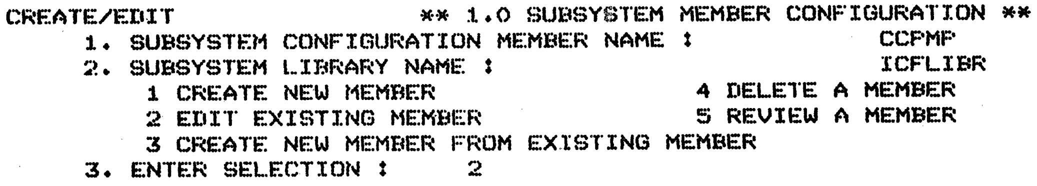
6. If the item is in the remote file, program B sends the stock status to program A Ⓖ and Ⓔ. If the item is not in the remote file, program B sends the characters ***.

7. If program A receives the stock status, it displays it. If it receives the characters ***, it displays the message ITEM NOT FOUND Ⓐ.

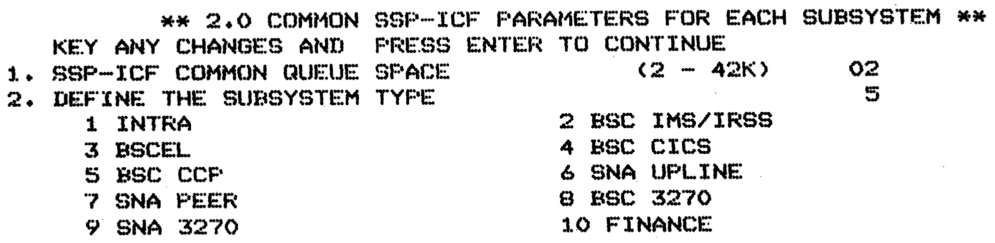
Except for a few minor changes, program A in this example is the program described in Chapter 7 for the Intra subsystem. The changes to the program required for CCP are shown in this chapter following the Configuration and OCL examples. If you have not read the description of program A in Chapter 7, see *How to Write Programs that Use the Intra Subsystem* in Chapter 7. The application, configuration, and OCL examples in Chapter 7 are for the Intra subsystem only; you do not need to read those. Following are the configuration parameters and OCL statements for the CCP subsystem.

Configuration Parameters

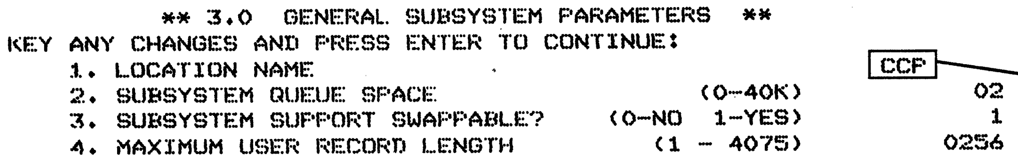
The following configuration parameters are used for this example. For a description of the configuration parameters, see *Setting up the CCP Subsystem* at the beginning of this chapter.



```
CREATE/EDIT                     ** 1.0 SUBSYSTEM MEMBER CONFIGURATION **
     1. SUBSYSTEM CONFIGURATION MEMBER NAME :              CCPMF
     2. SUBSYSTEM LIBRARY NAME :                           ICFLIBR
         1 CREATE NEW MEMBER                    4 DELETE A MEMBER
         2 EDIT EXISTING MEMBER                 5 REVIEW A MEMBER
         3 CREATE NEW MEMBER FROM EXISTING MEMBER
     3. ENTER SELECTION :      2
```

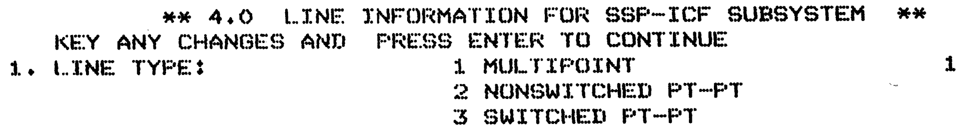


```
          ** 2.0 COMMON SSP-ICF PARAMETERS FOR EACH SUBSYSTEM **
   KEY ANY CHANGES AND  PRESS ENTER TO CONTINUE
1. SSP-ICF COMMON QUEUE SPACE                 (2 - 42K)     02
2. DEFINE THE SUBSYSTEM TYPE                                 5
     1 INTRA                          2 BSC IMS/IRSS
     3 BSCEL                          4 BSC CICS
     5 BSC CCP                        6 SNA UPLINE
     7 SNA PEER                       8 BSC 3270
     9 SNA 3270                       10 FINANCE
```



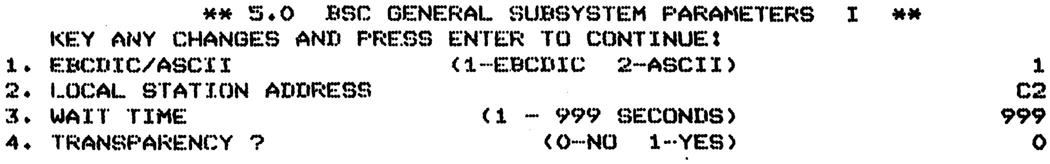
```
          ** 3.0  GENERAL SUBSYSTEM PARAMETERS   **
KEY ANY CHANGES AND PRESS ENTER TO CONTINUE:
    1. LOCATION NAME                                    CCP
    2. SUBSYSTEM QUEUE SPACE                (0-40K)          02
    3. SUBSYSTEM SUPPORT SWAPPABLE?    (0-NO  1-YES)          1
    4. MAXIMUM USER RECORD LENGTH         (1 - 4075)        0256
```

The location name is also specified on the SESSION statement.

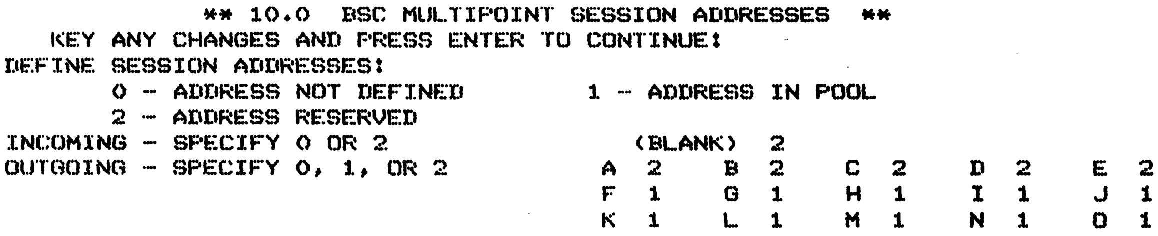


```
          ** 4.0  LINE INFORMATION FOR SSP-ICF SUBSYSTEM  **
   KEY ANY CHANGES AND  PRESS ENTER TO CONTINUE
1. LINE TYPE:                 1 MULTIPOINT                     1
                              2 NONSWITCHED PT-PT
                              3 SWITCHED PT-PT
```

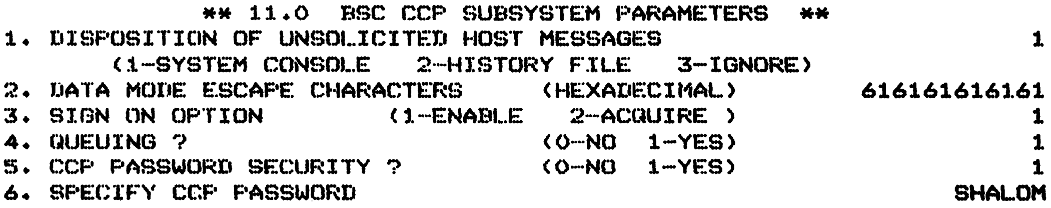
Configuration Parameters (continued)



```
              ** 5.0  BSC GENERAL SUBSYSTEM PARAMETERS  I  **
   KEY ANY CHANGES AND PRESS ENTER TO CONTINUE:
1. EBCDIC/ASCII                (1-EBCDIC  2-ASCII)                 1
2. LOCAL STATION ADDRESS                                          C2
3. WAIT TIME                     (1 - 999 SECONDS)               999
4. TRANSPARENCY ?                   (0-NO  1-YES)                  0
```



```
              ** 10.0  BSC MULTIPOINT SESSION ADDRESSES  **
   KEY ANY CHANGES AND PRESS ENTER TO CONTINUE:
DEFINE SESSION ADDRESSES:
       0 - ADDRESS NOT DEFINED        1 - ADDRESS IN POOL
       2 - ADDRESS RESERVED
INCOMING - SPECIFY 0 OR 2                (BLANK)  2
OUTGOING - SPECIFY 0, 1, OR 2          A  2    B  2    C  2    D  2    E  2
                                       F  1    G  1    H  1    I  1    J  1
                                       K  1    L  1    M  1    N  1    O  1
```



```
              ** 11.0  BSC CCP SUBSYSTEM PARAMETERS  **
1. DISPOSITION OF UNSOLICITED HOST MESSAGES                        1
       (1-SYSTEM CONSOLE   2-HISTORY FILE   3-IGNORE)
2. DATA MODE ESCAPE CHARACTERS      (HEXADECIMAL)       616161616161
3. SIGN ON OPTION          (1-ENABLE   2-ACQUIRE )                 1
4. QUEUING ?                        (0-NO  1-YES)                  1
5. CCP PASSWORD SECURITY ?          (0-NO  1-YES)                  1
6. SPECIFY CCP PASSWORD                                       SHALOM
```

OCL Statements

The following procedure and OCL statement are used for the BASIC example:

```
BASICR CCPBAS,ITEMBAS,30,,CCPSESS

// SESSION LOCATION-CCP,SYMID-1S
```

The BASICR procedure includes the SESSION statement.

The location name (CCP) is also specified on display 3.0 of the CNFIGICF procedure.

The SYMID (session ID) is explained in the description of the program.

The following OCL statements are used for the COBOL example:

```
// LOAD COBCCP
// FILE NAME-FILEA
// SESSION LOCATION-CCP,SYMID-1S
// RUN
```

The location name (CCP) is also specified on display 3.0 of the CNFIGICF procedure.

The SYMID (session ID) is explained in the description of the program.

The following OCL statements are used for the RPG II example:

```
// LOAD RPGCCP
// FILE NAME-FILEA
// SESSION LOCATION-CCP,SYMID-1S
// RUN
```

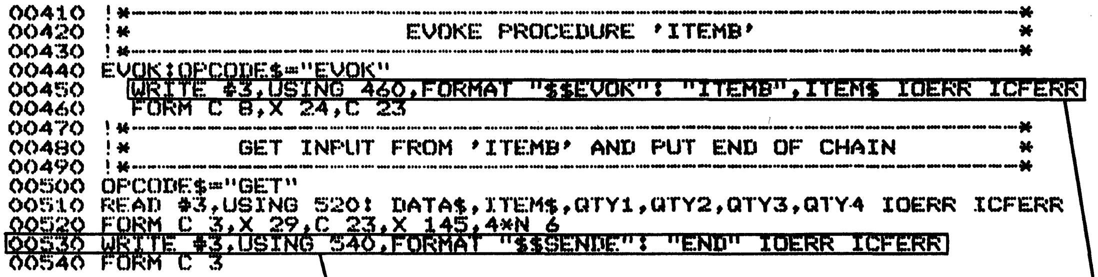
The location name (CCP) is also specified on display 3.0 of the CNFIGICF procedure.

The SYMID (session ID) is explained in the description of the program.

Changes for the Programming Example

The following examples show the changes required for program A, described in Chapter 7, to permit communications with CCP.

Changes for BASIC



```
00410 !*-------------------------------------------------------------------*
00420 !*                      EVOKE PROCEDURE 'ITEMB'                      *
00430 !*-------------------------------------------------------------------*
00440 EVOK:OPCODE$="EVOK"
00450   WRITE #3,USING 460,FORMAT "$$EVOK": "ITEMB",ITEM$ IOERR ICFERR
00460   FORM C 8,X 24,C 23
00470 !*-------------------------------------------------------------------*
00480 !*        GET INPUT FROM 'ITEMB' AND PUT END OF CHAIN                *
00490 !*-------------------------------------------------------------------*
00500 OPCODE$="GET"
00510 READ #3,USING 520: DATA$,ITEM$,QTY1,QTY2,QTY3,QTY4 IOERR ICFERR
00520 FORM C 3,X 29,C 23,X 145,4*N 6
00530 WRITE #3,USING 540,FORMAT "$$SENDE": "END" IOERR ICFERR
00540 FORM C 3
```

The program sends an end of file and processes the data received.

The program ITEMB is evoked on the System/3. The password, user ID, and library name are not required with CCP.

```
01   EVOKE-RECORD.
     03  PROCEDURE-NAME             PIC X(8) VALUE 'ITEMB'.
     03  PASSWORD                   PIC X(8).
     03  USER-ID                    PIC X(8).
     03  LIBRARY-NAME               PIC X(8).
     03  FILLER                     PIC X(20).
     03  DATA-LENGTH                PIC XXXX VALUE '0023'.
     03  ICF-ITEM-NUMBER-OUT        PIC X(23).
```

Program ITEMB is evoked at the System/3. The password, user ID, and library name are not required with CCP.

```
*-------------------------------------------------------------------------*
*     EVOKE 'ITEMB' AT HOST                                               *
*-------------------------------------------------------------------------*
      MOVE ITEM-NUMBER TO ICF-ITEM-NUMBER-OUT.
      WRITE TRANSACTION-RECORD FROM EVOKE-RECORD
         FORMAT IS '$$EVOK',  TERMINAL IS ICF-SESSION.
      MOVE 'EVOK' TO OPCODE.
      PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
      IF IO2 = B'0'
         PERFORM SEND-EOS,
         GO TO ITEM-INQUIRY.
*-------------------------------------------------------------------------*
*     GET INPUT FROM HOST
*-------------------------------------------------------------------------*
      MOVE SPACES TO ICF-RECORD-IN.
      READ TRANSACTION-FILE RECORD INTO ICF-RECORD-IN,
        TERMINAL IS ICF-SESSION.
      MOVE 'GET' TO OPCODE.
      PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
      IF IO2 = B'0'
         PERFORM SEND-EOS,
         GO TO ITEM-INQUIRY.
*-------------------------------------------------------------------------*
*     RELEASE SESSION                                                     *
*-------------------------------------------------------------------------*
      DROP ICF-SESSION FROM TRANSACTION-FILE.
      MOVE 'DROP' TO OPCODE.
      PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
      IF IO2 = B'0'
        PERFORM SEND-EOS.
```

The program does not send an end of transaction. Releasing the session sends the CCP command /RELEASE to end the transaction.

Changes for RPG II

```
O          E          05N07
O                                                 K6 '$$EVOK'
O                                                  5 'ITEMB'
O                                                 56 '0023'
O                                ITM#             79
```

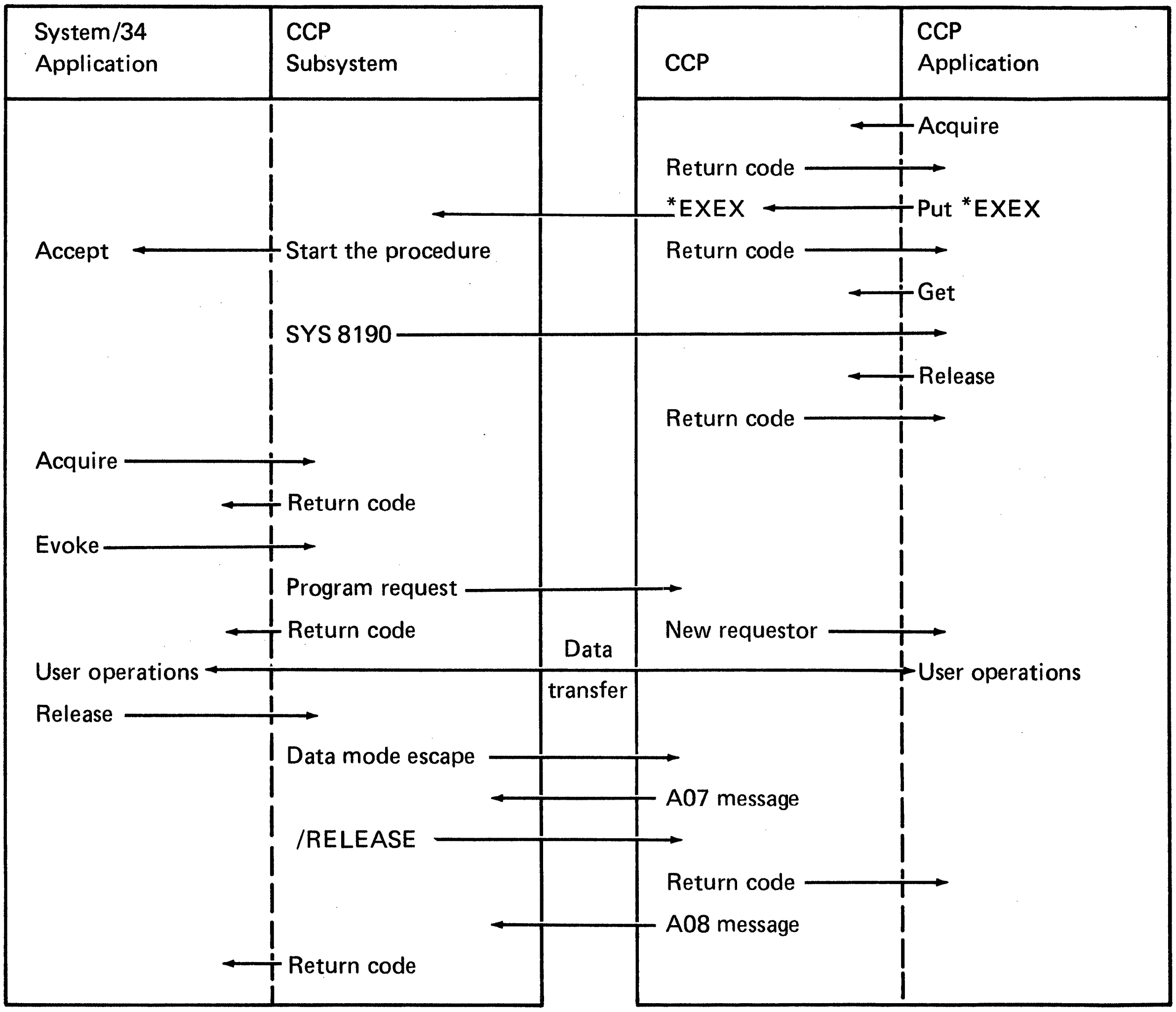
Program ITEMB is evoked at the System/3. The password, user ID, and library name are not required with CCP.

```
C*-----------------------------------------------------------------*
C*       EVOKE 'ITEMB' AT HOST                                     *
C*-----------------------------------------------------------------*
C                          MOVE '1S'      ID
C                          SETON                        05
C                          EXCPT
C                          SETOF                        05
C*
C*-----------------------------------------------------------------*
C*       GET INFORMATION SENT BY 'ITEMB'.                          *
C*-----------------------------------------------------------------*
C             '1S'         NEXT WSFILE
C                          READ WSFILE
C*
C*-----------------------------------------------------------------*
C*       RELEASE ICF SESSION '1S' AND GO TO ITMINQ.                *
C*-----------------------------------------------------------------*
C             '1S'         REL  WSFILE
C   10                     SETOF                        01
C   10                     SETON                        03
C                          GOTO ITMINQ
```

The program does not send an end of transaction. Releasing the session sends the CCP command /RELEASE to end the transaction.

Remote Procedure Start Request Example

The following sample application flow shows a System/3 CCP application program starting a System/34 procedure, and the System/34 procedure, and the System/34 program attaching to the CCP application program that originally performed the procedure request.



CCP Subsystem Return Codes

This part of Chapter 9 describes all the return codes that are valid for the CCP 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 four-digit value; the first two digits contain the major code, and the last two 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).

Note: In the return code descriptions, *your program* refers to the local System/34 application program that initiates the operation and receives the return code from the subsystem. The *remote program* refers to the application program in the remote (or host) system with which the System/34 application program is communicating through SSP-ICF.

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 CCP subsystem. Although all the operations shown are valid for CCP, their validity also depends on the logical sequence of communications events occurring between the System/34 and the remote system.

Input Operations to Your Program	Output Operations from Your Program	Combined Operations in Your Program
Accept input		
	Acquire[1]	
	End of session	
	Evoke	Evoke then get[2] Evoke then invite
Get Get attributes[3]		
Invite		
	Put Put end of file	Put then get[2] Put then invite
	Release	
		Request to change direction then get[2] Request to change direction then invite
	Set timer[4]	

[1] Normally, 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.

[2] Valid only in assembler language.

[3] Valid only in assembler and COBOL languages.

[4] For BASIC and RPG II programs, the set 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.

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, or it received a message from the remote system.

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:** The *output* operation just performed by your program was completed successfully; your program can continue to send data.

Normal Action: 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 This Output Operation Was:	Then (in This Session):
Initiated the transaction (evoked the remote program)	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 a remote procedure start request)	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.

[1] An evoked program (started by a procedure start request) 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.

0001 **Normal Indication:** Your program has received some data on a successful input operation. It can continue to receive input until SSP-ICF returns a code of xx00 (an end of transmission indication, which allows your program to send data), or xx08 (an end of transaction indication).

Normal Action: Issue another input operation. If your program can detect something equivalent to an end of file condition, indicating that the last of the data was just received, it can issue an output operation.

0010 **Normal Indication:** A request to change direction was received from the remote program on a successful *output* operation for your program; the remote program wants to send data as soon as possible. You should allow the remote program to send its data.

Normal Action: Issue an input operation as soon as possible.

0020 **Normal Indication:** A message from the remote system and an end of transmission indication were received on a successful input operation. The message, now in your program's input buffer, describes why the previous operation was rejected. The remote system may now want to receive some data.

Normal Action: Handle the message in the input buffer (possibly display it). Your program now has control of the session; issue an output operation or another input operation.

Major Code 01 – Successful operation with a new requester.

The new requester is a program on a remote system that initiated a session with your program by sending to the local system a procedure start request. 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 procedure start request, initiated by the remote program, was sent by the remote system in the form of an *EXEC or *EXEX procedure start statement. The request may have included, for your program, data from the remote *program* or a system message from the remote *system*.

Normal Description: Each of the following return codes indicates either that the *input* operation issued by your program and responded to by a new requester completed successfully, or that the *output* operation issued by your program in response to a new requester completed successfully.

If the operation was an *input* operation, your program received some data, no data, or a system message from the requester. If any data was received, it was included in the incoming procedure start request statement. If a system message was received, the procedure that was started was the default destination procedure; the message (now in the subsystem input buffer) will be passed to your program when the next input operation is issued.

If your program is an SRT program that was evoked by an incoming procedure start request 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 procedure start request statement 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 or message 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, either data or a message 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 was received, and some data may have been received with the request. The remote program may now want to continue to send data, or it may want to receive data; your program should issue the appropriate operation.

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[1] 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 can continue to receive input until SSP-ICF returns a code of xx00 (an end of transmission indication) or xx08 (an end of transaction indication).

Normal Action: Handle any data passed with the request, perform any necessary record keeping[1] for the new requester, and issue another input operation. If your program can detect something equivalent to an end of file condition, indicating that the last of the data was just received, it can issue an output operation.

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, or it received a message from the remote system. 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.

[1] For some situations, no record keeping for the session is necessary. In other situations, you should record the session ID of the new requester. You may also want to keep a table containing the IDs of all active requesters, or to maintain a history log of all requests.

Code **Indication/Action**

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 can continue to receive input until SSP-ICF returns a code of xx00 (an end of transmission indication) or xx08 (an end of transaction indication).

Normal Action: Issue another input operation. If your program can detect something equivalent to an end of file condition, indicating that the last of the data was just received, it can issue an output operation.

0220 **Normal Indication:** A message from the remote system and an end of transmission indication were received on a successful input operation. (The message, now in your program's input buffer, describes why the previous operation was rejected.) The remote system may now want 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: Handle the message in your program's input buffer (possibly display it), and issue an output operation or another input operation.

0230 **Normal Indication:** A truncated message from the remote system and an end of transmission indication were received on a successful input operation. The remote system may now want to receive some data. (The message, truncated because it was too long for your program's input buffer, describes why the previous operation was rejected.) 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 (possibly display it) in your program's input buffer, and issue an output operation.

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:** An end of transmission indication with *no* data was received on a successful input operation. The last record in the file has been received. Although communications have ended with the remote program, the session with the remote system is still active.

Normal Action: Issue another input operation, issue an output operation, or terminate the transaction with a put end of transaction operation.

0308 **Normal Indication:** An end of transaction indication was received *without* data on a successful input operation. Although communications have ended with the remote program, the session with the remote system is still active.

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.

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 remote 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 will be received.

General Recovery Actions: Issue an input operation to receive data or a message from the remote system.

Code Indication/Action

0411 **Normal Indication:** The remote 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 remote 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. Your program must receive the data before it can perform an output operation.

Normal Action: Issue an input operation to receive the data.

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 a new requester) 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 requesting program have been completed.

3401 **Error Indication:** This input operation was rejected because the record length of the data sent by the remote 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/34 Messages Guide* 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 (in COBOL and assembler only) a set timer operation, or by using the TIMER intrinsic function (in BASIC only); (2) continue local processing; or (3) terminate.
- If the session should be started again after the subsystem is enabled, it must be reacquired by your program or restarted by the remote 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 either in this subsystem or in the interrupt handler.

Recovery Action: This subsystem has been disabled; it must be enabled again before communications can resume. Your program can continue local processing, wait[1] to reissue the acquire operation, or terminate.

If more than one subsystem was active when the error occurred, all subsystems that were active when the error occurred should be disabled. (Note that all other active CCP subsystems are automatically disabled when the error occurs; all other types of active subsystems must be manually disabled.)

If all subsystems (of all types) on the system are *not* disabled to recover from the processor check, the common queue space used by the failing subsystem *cannot* be freed. And if it is not freed, that space is wasted, and an indication of insufficient common queue space being available can occur. The indication can occur as a message when the failing subsystem is reenabled or when a different subsystem is enabled. The indication can also occur as a return code to your program for any subsystem that is starting a *new* session (code 8215) or performing an output operation in any *existing* session (code 8315).

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 remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, it can wait[1] until the subsystem has been reenabled to reissue the acquire operation, or it can terminate.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

8083 **Error Indication:** An MLCA (multiline communications adapter) controller check occurred on an *output* operation. The subsystem has been disabled.

Recovery Action: Communications with the remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, wait[1] until the subsystem is enabled again, or terminate.

8084 **Error Indication:** An MLCA (multiline communications adapter) controller check occurred on an *input* operation. The subsystem has been disabled.

Recovery Action: Communications with the remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, wait[1] until the subsystem is enabled again, or terminate.

80BD **Error Indication:** Your program attempted to make a connection on an X.21 line, but the X.21 connect task was not active. The X.21 task (which was activated when the IPL process was used to start the system) has terminated abnormally for some reason. The subsystem has been disabled.

Recovery Action: Your program can continue local processing or terminate. Before your program can communicate over an X.21 line: the IPL process must be performed again to reactivate the X.21 task, the subsystem must be reenabled, and the session must be reacquired.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

Major Code 81 – Permanent (nonrecoverable) session error.

Error Description: A nonrecoverable error has occurred in the session; the session cannot bı continued and has been terminated. The error indication has been sent as a message tc the display station or to the system console; the operator can refer to the *System/34 Messages Guide* 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 session must be acquired again before communications can resume.

General Recovery Actions: The following general actions can be taken for each 81xx return code. Other specific actions are given in each return code description.

- Several return codes indicate that an error condition must be corrected by changing a value in the subsystem configuration record or in the SESSION statement for your program.
  - To change a parameter value in the subsystem configuration being used by your program, disable the subsystem before making the change in the subsystem's configuration record, and enable the subsystem again to make the change effective
  - To change a parameter value in the SESSION statement associated with your program, terminate your program only.

  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.

- If the session should be started again, it must be reacquired by your program or restarted by the remote program before communications can resume.

- An end of session (EOS or $$EOS) operation should be issued for the session that has terminated. Your program can also continue local processing, or it can terminate.

  Note: If the session is started again, it starts from the beginning, not at the point wher the session error occurred.

Code	Indication/Action
8183	**Error Indication:** An MLCA (multiline communications adapter) controller check occurred on an *output* operation; data may have been lost. The session has been terminated. **Recovery Action:** If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.
8184	**Error Indication:** An MLCA (multiline communications adapter) controller check occurred on an *input* operation; data may have been lost. The session has been terminated. **Recovery Action:** If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.
8185	**Error Indication:** An attempt by this subsystem to automatically call one or more remote locations using the autocall or X.21 feature was not successful. All available numbers in the list of phone numbers or in the list of public data network numbers were called, but no connection was established. The session has been terminated. **Recovery Action:** The list has been reinitialized. If the session should be started again, reissue the acquire operation; calling will begin with the first number in the list. Otherwise, your program can continue local processing or terminate.
8186	**Error Indication:** An attempt by this subsystem to automatically call one or more remote locations using the autocall or X.21 feature was not successful. All numbers in the list of phone numbers or in the list of public data network numbers were already marked as called. The message *PHONE LIST EXHAUSTED* (SYS-8607) has been displayed on the system console, and the session has been terminated. **Recovery Action:** The list has been reinitialized. If the session should be started again, reissue the acquire operation; calling will begin with the first number in the list. Otherwise, your program can continue local processing or terminate.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

8191 **Error Indication:** A permanent line error occurred on an *output* operation, and the system operator has taken a recovery option in response to the line error message. (You can find out what type of line error occurred by asking the system operator.) The session has been terminated.

Recovery Action: If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

8192 **Error Indication:** A permanent line error occurred on an *input* operation, and the system operator has taken a recovery option in response to the line error message. (You can find out what type of line error occurred by asking the system operator.) The session has been terminated.

Recovery Action: If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

8193 **Error Indication:** A disconnect indication (for switched lines only) was received on an *output* operation. A disconnect time-out in the remote system was exceeded, the line was unexpectedly disconnected, or your program may have sent some invalid data. The session has been terminated.

Recovery Action: Verify that your program did not cause a time-out and that it did not send data that was invalid. Also, verify that it did not try to send data after the transaction had ended. If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

8194 **Error Indication:** A disconnect indication (for switched lines only) was received on an *input* operation. A disconnect time-out in the remote system was exceeded, or the line was unexpectedly disconnected. The session has been terminated.

Recovery Action: Verify that your program did not cause a time-out. Also, verify that it did not try to receive data after it had received an end of transaction indication. If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

8199 **Error Indication:** On an *output* operation, the time between successive data blocks being sent to the remote system exceeded the value specified for the wait time parameter in the subsystem configuration.

Recovery Action: Check the wait time parameter value to ensure that it is large enough for your program. Also check your program for excessive delays between output operations. If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

819A **Error Indication:** On an *input* operation, the time between successive data blocks being received from the remote system exceeded the value specified for the wait time parameter in the subsystem configuration.

Recovery Action: Check the wait time parameter value in the subsystem configuration to ensure that it is large enough for your program. Also check your program for excessive delays between input operations. If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

819B **Error Indication:** On an *output* operation, in a put-versus-put situation, the subsystem detected a block size error before it detected that both your program and the remote system were attempting to send data at the same time. The remote system sent data, but the length of the data block exceeded the length of the subsystem's line buffer. The session has been terminated.

Recovery Action: Check that the maximum user record length is correct in the subsystem configuration record. If the parameter is correct, notify the remote system programmer and verify that the record length is correct. Then, if your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

819C **Error Indication:** On an *input* operation, the length of the data block sent by the remote system exceeded the length of the subsystem line buffer. The session has been terminated.

Recovery Action: Check that the maximum user record length is correct in the subsystem configuration record. If the parameter is correct, notify the remote system programmer and verify that the record length is correct. Then, if your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

819E **Error Indication:** An *abnormal* shutdown indication was received from the CCP host system; some kind of error was detected by the host system and it is shutting down. The session has been terminated.

Recovery Action: If your program started the session, issue a set timer operation to wait for the host system to be started again; then reissue the acquire operation. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

819F **Error Indication:** A *normal* shutdown indication was received from the CCP host system; CCP was being shut down while this session was still active. The session has been terminated.

Recovery Action: If your program started the session, it can issue a set timer operation to wait for CCP to be restarted, then reissue the acquire operation; or it can continue local processing or terminate. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

81BC **Error Indication:** An attempt by this subsystem to automatically call a remote location using the autocall or X.21 feature was not successful because the wrong type of list was used to make the call. Either a list of public data network numbers was used to make the call on an autocall line, or a list of phone numbers was used to make the call on an X.21 line. The session has been terminated.

Recovery Action: Change the name specified in the *phone list name* parameter in the subsystem configuration, or in the PHONE parameter of the SESSION statement for this program. Then reissue the acquire operation to restart the session.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

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/34 Messages Guide* 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, first disable the subsystem, make the change in the subsystem's configuration record, and 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, (and depending on what the return code description says):
   a. Notify the remote location that a change is required on that end to correct the error received.
   b. Simply reissue the acquire operation. It could be successful if the error occurred because there was not enough common queue space available to support a new session, because an isolated line error occurred, or because the remote system was not active temporarily.
   c. If the acquire operation is again unsuccessful, retry it only a limited number of times. (The limit for retries should be specified in your program.)
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 operation is valid only in an *active* session and cannot, therefore, be issued after an 82xx return code is received. (This restriction does not apply to COBOL and assembler programs, or to the TIMER intrinsic function in BASIC, which also can be used to wait for a specified time interval.)

Code Indication/Action

8213 **Error Indication:** On an unsuccessful acquire operation, a queue space error condition was detected. The session could not be started because no *subsystem* queue space was available at the time.

Recovery Action: Your program can wait[1] for a period of time, then reissue the acquire operation. If an unacceptable number of queue space errors occur, you can disable the subsystem and change the subsystem configuration by specifying a larger subsystem queue space size in the subsystem queue space parameter. After the subsystem is enabled, reissue the acquire operation to start the session.

8215 **Error Indication:** On an unsuccessful acquire operation, a queue space error condition was detected. The session cannot be started because the size of the *common* queue space, specified during subsystem configuration, is too small.

Recovery Action: Your program can wait[1] for a period of time, then reissue the acquire operation. If an unacceptable number of queue space errors occur, you can disable all the subsystems that are active in the system, and change the subsystem configuration by specifying a larger common queue space size in the SSP-ICF common queue space parameter. After the subsystem is enabled, reissue the acquire operation to start the session.

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, respecify 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.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

8236 **Error Indication:** On an unsuccessful acquire operation (of a session over switched lines only), an invalid remote identifier was received from the remote system or device with which the session is being acquired. The received remote identifier must match the remote identifier specified for this subsystem configuration.

Recovery Action: Verify that the remote identifier specified for this subsystem configuration was specified correctly either by the remote ID parameter in the CNFIGICF procedure or by the SSP DEFINEID procedure (when multiple remote identifiers are specified). If the remote identifier was specified correctly, call the remote location to correct the error at the remote end of the switched line.

8281 **Error Indication:** On an unsuccessful acquire operation, an SSP-ICF error condition was detected. The error caused a processor check either in this subsystem or in the interrupt handler.

Recovery Action: This subsystem has been disabled; it must be enabled again before communications can resume. Your program can continue local processing, wait[1] to reissue the acquire operation, or terminate.

If more than one subsystem was active when the error occurred, all subsystems that were active when the error occurred should be disabled. (Note that all other active CCP subsystems are automatically disabled when the error occurs; all other types of active subsystems must be manually disabled.)

If all subsystems (of all types) on the system are *not* disabled to recover from the processor check, the common queue space used by the failing subsystem *cannot* be freed. And if it is not freed, that space is wasted, and an indication of insufficient common queue space being available can occur. The indication can occur as a message when the failing subsystem is reenabled or when a different subsystem is enabled. The indication can also occur as a return code to your program for any subsystem that is starting a *new* session (code 8215) or performing an output operation in any *existing* session (code 8315).

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 remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, wait[1] to reissue the acquire operation, or terminate.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

8283 **Error Indication:** On an unsuccessful acquire operation, an MLCA (multiline communications adapter) controller check occurred. The session was not started.

Recovery Action: Your program can reissue the acquire operation, continue local processing, or terminate.

8285 **Error Indication:** An acquire operation, for which the subsystem used the autocall or X.21 feature to automatically call one or more remote locations, was not successful. All available numbers in the list of phone numbers or in the list of public data network numbers were called, but no connection was established. The session was not started.

Recovery Action: The list has been reinitialized. Your program can reissue the acquire operation, continue local processing, or terminate. (Calling will begin with the first number in the list.)

8286 **Error Indication:** An acquire operation, for which the subsystem used the autocall or X.21 feature to automatically call one or more remote locations, was not successful. All the numbers in the list of phone numbers or in the list of public data network numbers were already marked as called. The message *PHONE LIST EXHAUSTED* (SYS-8607) has been displayed on the system console, and the session was not started.

Recovery Action: The list has been reinitialized. Your program can reissue the acquire operation, continue local processing, or terminate. (Calling will begin with the first number in the list.)

8291 **Error Indication:** A permanent line error occurred on an unsuccessful acquire operation, and the system operator has taken a recovery option in response to the line error message. (You can find out what type of line error occurred by asking the system operator.) The session was not started.

Recovery Action: Your program can reissue the acquire operation, continue local processing, or terminate.

8293 **Error Indication:** A disconnect indication (for switched lines only) was received on an unsuccessful acquire operation. The line was unexpectedly disconnected. The session was not started.

Recovery Action: Check with your remote system location to determine why it unexpectedly sent a disconnect. Your program can reissue the acquire operation, continue local processing, or terminate.

829B **Error Indication:** On an unsuccessful acquire operation, a block size error was detected. The length of the output data, consisting of a /ON or /Q CCP command, from your program exceeded the length of the subsystem line buffer. The session was not started.

Recovery Action: Disable the subsystem and change the value specified for the maximum user record length parameter in the subsystem configuration. The maximum user record length must be large enough to handle the longest record to be sent or received. Enable the subsystem and reissue the acquire operation.

829F **Error Indication:** On an unsuccessful acquire operation, a normal shutdown indication was received from the CCP host system; CCP was being shut down at the time the acquire operation was issued. The session was not started.

Recovery Action: Your program can wait[1] for the CCP system to be started again, then reissue the acquire operation to start the session. Or, it can continue local processing or terminate.

82A2 **Error Indication:** On an unsuccessful acquire operation, the sign-on portion of the session failed. The password used to sign on to CCP was invalid. The session was not started.

Recovery Action: Disable the CCP subsystem and change the value of the CCP password security parameter in the subsystem configuration. After the subsystem has been enabled, reissue the acquire operation.

82A7 **Error Indication:** The acquire operation was unsuccessful because the specified communications line was already in use. The session was not started.

Recovery Action: Your program can wait[1] for the line to become available, then reissue the acquire operation. Otherwise, it can continue local processing 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 100 sessions can be active in the System/34 at one time. The session was not started.

Recovery Action: Your program can wait[1] for another session to end and then reissue the acquire operation. Otherwise, your program can continue local processing or terminate.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

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 to be enabled is identified by the location parameter in the SESSION statement. That location name must also be specified in the subsystem configuration record (shown on display 3.0 of the subsystem configuration planning charts). The session was not started.

Recovery Action: Verify that the subsystem name was specified correctly on the LOCATION parameter of the SESSION statement. If the correct name was specified, contact the System/34 system operator and request that the specified subsystem be enabled by executing the ENABLE procedure command at the system console. Then reissue the acquire operation. Otherwise, your program can continue local processing, wait[1] 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[1] until the subsystem has been enabled, and 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[1] until the subsystem is enabled again, and then reissue the acquire operation. Otherwise, your program can continue local processing, or it can terminate.

82B1 **Error Indication:** On an unsuccessful acquire operation, the session specified by the session address on the SESSION statement was identified as already being in use. (The session address is specified in the SESNADDR parameter of the SESSION statement. The address matches one of the addresses in the list of addresses that were specified during subsystem configuration.) The session was not started.

Recovery Action: (1) Your program can wait[1] for the specified session to become available and reissue the acquire operation. (2) You can remove the SESNADDR parameter value from the SESSION statement and reissue the acquire operation; the subsystem will then assign a session address from those available in the subsystem configuration. (3) Or, your program can continue local processing or terminate.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

82B2 **Error Indication:** The acquire operation was not successful because all of the sessions in the address pool are already in use. (The session address was not specified in the SESNADDR parameter of the SESSION statement; any available session in the pool was to be used.) The session was not started.

Recovery Action: Wait[1] for one of the sessions in the pool to become available, then reissue the acquire operation. Otherwise, continue local processing or terminate.

82B4 **Error Indication:** The acquire operation was not successful because all of the resources needed for the session could not be allocated from the assign/free area of the system. All available resources are already being used in the system. The session was not started.

Recovery Action: Wait[1] for the needed resources to become available, then reissue the acquire operation. Otherwise, continue local processing or terminate.

82BC **Error Indication:** On an unsuccessful acquire operation, an invalid type of list was detected by the subsystem when it attempted to automatically call a remote location using the autocall or X.21 feature. Either a list of public data network numbers was used to make the call on an autocall line, or a list of phone numbers was used to make the call on an X.21 line. The session was not started.

Recovery Action: Specify the name of the list that is of the correct type for the line to be used. Change the name specified in the *phone list name* parameter in the subsystem configuration, or in the PHONE parameter of the SESSION statement for this program; then reissue the acquire operation.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

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/34 Messages Guide* 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, first disable the subsystem, make the change in the subsystem's configuration record, and 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.

   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.

   Note: If the session is started again, it starts from the beginning, not at the point where the session error occurred.

3. If no change is needed in your program or in the subsystem, (and depending on what the return code description says):
   a. Retry the operation, if possible. It could be successful if the error occurred because there was not enough common queue space available at the time or because an isolated line error occurred.
   b. If another operation is not successful, retry it only a limited number of times. (The limit for retries should be specified in your program.)
   c. Issue a set timer operation in your program so it can wait for a specified time interval before reissuing the operation.

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 an 81xx (session was terminated) or 82xx (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 an 81xx or 82xx return code is handled properly. If you want your program to recover from an improperly handled error condition, issue another acquire operation.

8315 **Error Indication:** On an evoke operation, a queue space error condition was detected. The evoke operation could not be performed because no *common* queue space was available at the time.

Recovery Action: Your program can issue a set timer operation and wait for a period of time, and then reissue the evoke operation. If an unacceptable number of queue space errors occur, you can disable all the subsystems and change the subsystem configuration by specifying a larger common queue space size in the SSP-ICF common queue space parameter. After the subsystem is enabled, reissue the acquire operation to restart the session.

831A **Error Indication:** An evoke operation failed to complete successfully, or the remote program terminated abnormally. A message from the remote system describing why it failed is waiting in the subsystem input buffer. The evoke operation could have been 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.

831C **Error Indication:** The output operation issued before this output operation received a return code of 0411 or 0412 (indicating that the remote 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. 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 before attempting to communicate with the remote 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: If you want your program to recover dynamically, reissue the output operation with a smaller output length. Otherwise, you can either change the record length in your program and recompile it, or you can change the value specified for the maximum user record length parameter in the subsystem configuration. 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 erroneously followed by a *request to change direction then get* operation, a *request to change direction then invite* operation, or a release 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 other program.

8329 **Error Indication:** An invalid evoke operation was detected in this session. Your program was evoked by an incoming procedure start request and cannot, therefore, issue any evoke operations in this session.

Recovery Action: If you want your program to dynamically 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.

832B **Error Indication:** On the first output operation, a record length of 0 was detected by the subsystem. The output operation was not performed.

Recovery Action: If a coding error in your program caused the error, correct your program. If the data record is in error, correct it. Then issue another output operation.

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.

832E **Error Indication:** A program cancel indication was received from the remote CCP system when no transactions were active (but the session was active).

Recovery Action: If your program started the session, issue a release or end of session operation. If your program was evoked, it can wait to be evoked again (MRT programs only), continue local processing, or terminate.

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 a release operation. If a coding error in your program caused the error, correct your program.

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.

8383 **Error Indication:** An MLCA (multiline communications adapter) controller check occurred on an *output* operation; data may have been lost. The session is still active.

Recovery Action: Issue another output operation to send the same data again. If the same error indication continues to occur, issue an end of session operation to end the session. Then your program can continue local processing, or it can terminate.

8384 **Error Indication:** An MLCA (multiline communications adapter) controller check occurred on an *input* operation; data may have been lost. The session is still active.

Recovery Action: Issue another input operation to receive the same data again. If the same error indication continues to occur, issue an end of session operation to end the session. Then your program can continue local processing, or it can terminate.

83A7 **Error Indication:** The operation was not performed because the specified communications line was already in use. The session is still active.

Recovery Action: Your program can wait for the line to become available by using the set timer operation, then reissue the operation that was not successful. Otherwise, it can continue local processing or terminate.

Chapter 10. The BSC CICS Subsystem

The BSC CICS subsystem provides distributed data processing support to users of the System/34 SSP in conjunction with a System/370 using CICS/VS. The BSC CICS subsystem provides an interactive interface between System/34 application programs and programs on a CICS/VS (OS/VS or DOS/VS) system with BTAM. The subsystem allows a System/34 application program to interactively communicate with a CICS/VS application program. The CICS subsystem can support multiple application programs concurrently communicating with CICS/VS.

The CICS subsystem allows System/34 application programs to initiate tasks on the CICS/VS system. System/34 procedures can be initiated from tasks on the CICS/VS system. System/34 security options as well as CICS/VS security options are supported.

Any BSC line configuration supported by BTAM and CICS/VS is supported by the BSC CICS subsystem.

One or more System/34s can exist in a CICS/VS network. Each System/34 on a multipoint line should be defined as one or more (up to 16) System/3 terminals in the CICS/VS network. (See *CICS/VS Configuration Considerations* later in this chapter for more information.)

A System/34 can also exist in a point-to-point CICS/VS network. The System/34 reacts with all the characteristics of a single System/3 terminal. The CICS subsystem also supports incoming procedure requests and interactive communications.

The user can select either ASCII or EBCDIC transmission for each line. The System/34 application program must process all data in EBCDIC. If ASCII is used, the subsystem translates output to ASCII before transmitting it, and translates input data to EBCDIC before passing it to the application program.

EBCDIC output can optionally be sent as transparent data. Transparent input data is detected and handled automatically by the System/34 BSC interrupt handler. Data transparency is not available with ASCII. All data is sent or received in unblocked logical records of varying length up to a user-specified maximum, which cannot exceed 4075 characters.

SETTING UP THE BSC CICS SUBSYSTEM

The SSP procedures CNFIGSSP and INSTALL are used to include the interactive communications feature and CICS subsystem support on the System/34. The general interactive communications and line control support is included when it is requested on the appropriate CNFIGSSP prompt. The CICS subsystem support is then copied to the system library when the appropriate responses to the INSTALL procedure prompts are taken.

After the CICS subsystem has been installed, the CNFIGICF procedure is used to tailor the subsystem support to an existing or proposed network. The CNFIGSSP and INSTALL procedures, with their displays and related responses, are explained in detail in the *Installation and Modification Reference Manual.*

The operation of the CNFIGICF procedure is also explained in the *Installation and Modification Reference Manual.* Before running the CNFIGICF procedure, however, you should fill out a planning chart for each subsystem that you want to define. Copies of the planning chart are available in Appendix F of this manual and in the *Installation and Modification Reference Manual.* The following sections explain how to fill out the planning chart for the CICS subsystem.

Display 1.0 Subsystem Member Configuration

1.0 Subsystem Member Configuration

1. Subsystem configuration member name (8 characters) _ _ _ _ _ _ _ _
2. Subsystem library name (8 characters) _ _ _ _ _ _ _ _
   Select:
   1. Create new member 4. Delete a member
   2. Edit existing member 5. Review a member
   3. Create new member from existing member
3. Enter selection: ______
4. Existing member name: _ _ _ _ _ _ _ _
5. Existing member library name: _ _ _ _ _ _ _ _

Subsystem configuration member name: Specify name for this configuration of the subsystem. This name is the configuration name used in the ENABLE and DISABLE procedures.

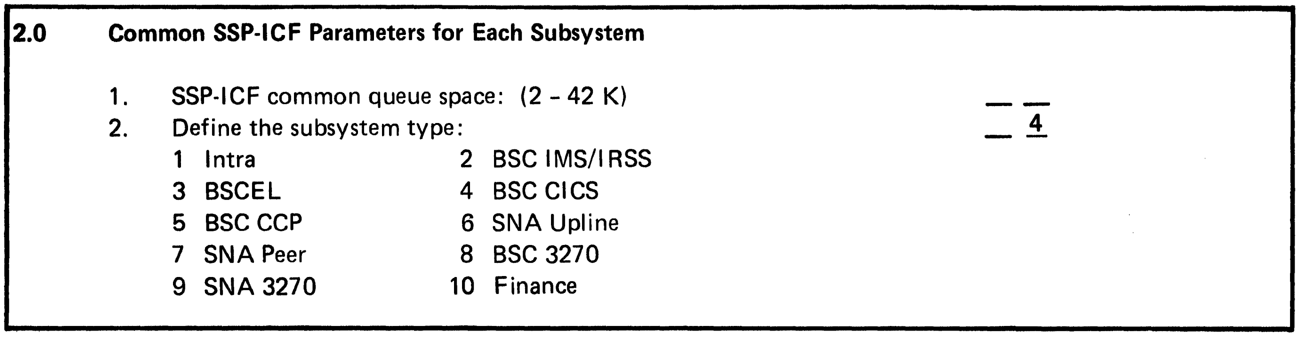
Library name: Specify the name of the library in which the configuration is stored or is to be stored. The default is #LIBRARY.

Enter selection: Specify one of the five options: (1) create a new member, (2) edit an existing member, (3) create a new member from an existing member, (4) delete a member, or (5) review a member without changing it.

Existing member name: This prompt appears if option 3 was selected. Specify the name of the existing subsystem configuration member that is to be used to create the new member. The existing member remains unchanged.

Exiting library member name: This prompt appears if option 3 was selected. Specify the library name where the existing member resides.

Display 2.0 Common SSP-ICF Parameters for Each Subsystem



```
2.0     Common SSP-ICF Parameters for Each Subsystem

        1.   SSP-ICF common queue space: (2 - 42 K)                __ __
        2.   Define the subsystem type:                            __ 4
             1  Intra           2  BSC IMS/IRSS
             3  BSCEL           4  BSC CICS
             5  BSC CCP         6  SNA Upline
             7  SNA Peer        8  BSC 3270
             9  SNA 3270       10  Finance
```

SSP-ICF common queue space: Specify the size, in mulitiples of 2 K bytes, of the common queue space. The common queue space requirements for each configuration of the CICS subsystem enabled are:

C = 84D + 32

where:
C = number of bytes required for the common queue space
D = number of defined session addresses for each line enabled

If ASCII is selected, add 768 bytes to the common queue space requirements.

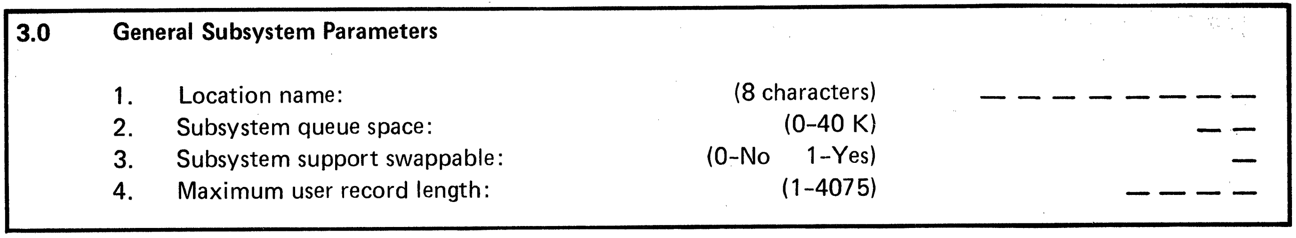
The common queue space value specified for the first subsystem that is enabled becomes the size of the common queue space. Be sure that the value specified for common queue space size takes into account the requirements of any other subsystem that might be enabled concurrently.

The common queue space cannot be swapped. The size of the common queue space plus the total subsystem queue space sizes of all the enabled CICS subsystems cannot exceed 42 K bytes.

The default common queue space size is 4 K bytes.

Define the subsystem type: Specify a 4 for the CICS subsystem.

Display 3.0 General Subsystem Parameters



```
3.0     General Subsystem Parameters

        1.  Location name:                              (8 characters)    _ _ _ _ _ _ _ _
        2.  Subsystem queue space:                          (0-40 K)                  _ _
        3.  Subsystem support swappable:               (0-No   1-Yes)                   _
        4.  Maximum user record length:                     (1-4075)              _ _ _ _
```

Location name: Specify up to 8 characters for the name of the location associated with this configuration. The location name is used in some of the displayed message texts, and must be coded on the SESSION OCL statement. The default is the subsystem configuration member name.

Subsystem queue space: Specify the size, in multiples of 2 K bytes, of the subsystem queue space. The subsystem queue space requirements for each configuration of the CICS subsystem are:

$$S = 3N(R + 21)$$

where:

S = number of bytes required for subsystem queue space
N = number of defined session addresses for each line enabled
R = maximum record length

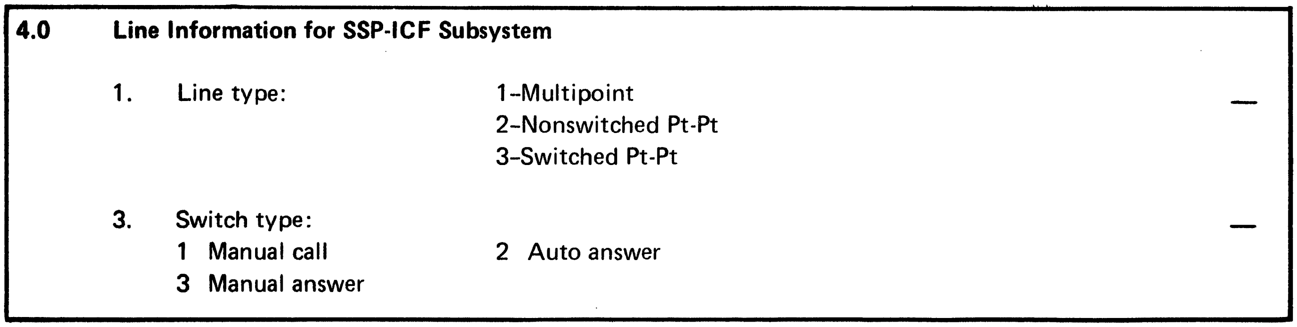
The subsystem queue space cannot be swapped. The size of the common queue space plus the total subsystem queue space sizes of all the enabled CICS subsystems cannot exceed 42 K bytes.

The default subsystem queue space is 4 K bytes.

Subsystem support swappable: Specify whether you want the subsystem to be swappable. Consider the total system performance, the size of the subsystem, and the amount of user main storage when determining whether you want the subsystem to be swappable. The CICS subsystem requires 6 K bytes of main storage.

Maximum user record length: Specify the maximum record length (1 through 4075 bytes) to be sent or received by a System/34 application program using this subsystem configuration. The values specified in the CICS/VS DFHTCT macro for TYPE=LINE, INAREAL = value, and TYPE=TERMINAL, TIOAL = value should be consistent with the maximum user record length. These values are the sizes of the longest records to be sent and received by the CICS subsystem respectively. The maximum user record length also directly affects the space required for buffers in the subsystem or common queue space.

Display 4.0 Line Information for the SSP-ICF Subsystem



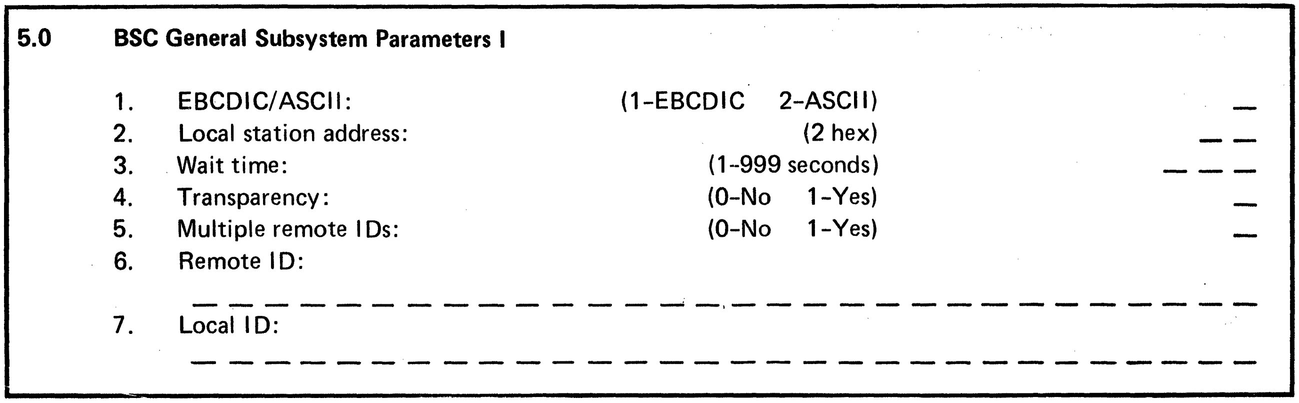
```
4.0     Line Information for SSP-ICF Subsystem

        1.   Line type:            1-Multipoint                    _
                                   2-Nonswitched Pt-Pt
                                   3-Switched Pt-Pt

        3.   Switch type:                                          _
             1  Manual call        2  Auto answer
             3  Manual answer
```

Line type: Specify the line type that is used in your communications environment. The default is 1 (multipoint).

Switch type: This prompt is displayed only if switched line was specified. Specify the switched type you want for the line. There is no default.

If you are going to use the autocall feature, you should specify 1 (manual call). If you are going to use the switched X.21 feature, specify 2 (auto answer). See *Automatic Calling on Switched Lines* in this section for additional information.

Display 5.0 BSC General Subsystem Parameters I



```
5.0     BSC General Subsystem Parameters I

        1.  EBCDIC/ASCII:                       (1-EBCDIC  2-ASCII)        _
        2.  Local station address:                         (2 hex)       _ _
        3.  Wait time:                               (1-999 seconds)   _ _ _
        4.  Transparency:                            (0-No    1-Yes)       _
        5.  Multiple remote IDs:                     (0-No    1-Yes)       _
        6.  Remote ID:
            _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
        7.  Local ID:
            _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
```

EBCDIC/ASCII: Specify which code you require. The code you select must be compatible with the CICS/VS system. The default is 1 (EBCDIC).

Local station address: Specify in hexadecimal the System/34 multipoint line address for this configuration. This value is taken from the entry in the DFTRMLST macro that defines the terminals to be supported. You can specify C2 (B) through D9 (R) for EBCDIC, or 41 (A) through SA (Z) for ASCII. This prompt is displayed only for multipoint lines. Be sure to enter the hexadecimal equivalent in the code used.

Wait time: Specify the number of seconds to maintain line synchronization while not sending or receiving data before a permanent error is indicated. The default is 999, which indicates an infinite wait time.

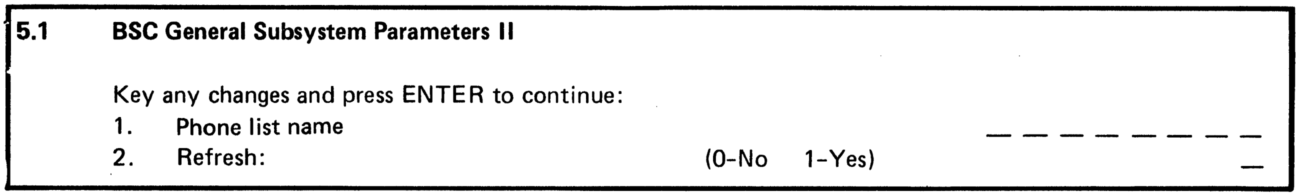
Transparency: Specify whether the EBCDIC data will be transmitted in transparent mode. If you specify 1 (yes) with ASCII data, an error message is displayed. The default is 0 (no). This value must correspond to the TRANSP parameter on the DFHTCT macro in CICS/VS generation.

Remote ID: Specify from 0 to 30 hexadecimal characters to be used for identification of a remote device on a switched line. The number of hexadecimal characters must be a multiple of 2. This field is left-justified, with all unused positions filled with zeros. The hexadecimal characters cannot include any BSC control characters. Therefore, the following hexadecimal characters cannot be used: 01, 02, 03, 10, 1F, 26, 2D, 32, 37, and 3D.

Local ID: Specify from 0 to 30 hexadecimal characters to be used for local switched line identification. The number of hexadecimal characters must be a multiple of 2. The field is left-justified with all unused positions filled with zeros. The hexadecimal characters cannot include any BSC control characters. Therefore, the following hexadecimal characters cannot be used: 01, 02, 03, 10, 1F, 26, 2D, 32, 37, and 3D.

Display 5.1 BSC General Subsystem Parameters II

The following prompt is displayed only if you specified a line type of *switched pt-pt* on display 4.0.



```
5.1    BSC General Subsystem Parameters II

       Key any changes and press ENTER to continue:
       1.  Phone list name                                      _ _ _ _ _ _ _ _
       2.  Refresh:                    (0-No   1-Yes)                         _
```

Phone list name: Specify the name of the load member that contains the list of phone numbers for the autocall feature or the list of numbers for the public data network to be called by the System/34. The load member must be in either #LIBRARY or in the same library as the configuration member.

Use the *refresh* parameter to define how you want the list of numbers processed.

Refresh: Specify 1 (yes) if you want the list reinitialized (calling to begin with the first number on the list) after a successful call. If you enter a list name and do not enter a value for refresh, refresh-yes is assumed.

Specify 0 (no) if you do not want the list reinitialized after a successful call.

Note: See *Automatic Calling on Switched Lines* in this chapter for more information how to use the phone list and refresh parameters.

Display 7.0 Subsystem Inactive Destination Messages

7.0 Subsystem Inactive Destination Messages

1. Subsystem procedure name: (8 characters) _ _ _ _ _ _ _ _
2. Subsystem procedure library name: (8 characters) _ _ _ _ _ _ _ _

Subsystem procedure name: Specify the name of a user-written procedure to be started when data or messages are received for an inactive session (uninvited data). When uninvited data is received, the procedure is run to receive data. The program within this procedure cannot send data on this session. The program can save the data in a file, process the data as it is received, or discard the data. This procedure must exist when the subsystem is enabled.

Subsystem library name: Specify the name of the library containing the inactive destination messages procedure.

Display 10.0 BSC Multipoint Session Addresses

10.0 BSC Multipoint Session Addresses

Define session addresses:
0-Address not defined 1-Address in pool
2-Address reserved
Incoming - Specify 0 or 2 (Blank)____
Outgoing - Specify 0, 1, or 2
A____ B____ C____ D____ E____
F____ G____ H____ I ____ J ____
K____ L____ M____ N____ O____

Define session address: Specify the type of session addresses for each session on a multipoint line. Each address can be defined as 0 (not used), 1 (address in the pool), or 2 (reserved). Reserved addresses can only be acquired if the specific session address is coded on the SESSION OCL statement (SESNADDR parameter). Up to 15 outgoing session addresses and one incoming address can be defined. The multipoint session address is used to equate the session to a CICS/VS TERMID. Each address defined must have a corresponding entry in the DFTRMLST macro in the CICS/VS generation of the network. The default for an outgoing session is 1 (in the pool); the default for an incoming session is 2 (reserved).

CICS/VS CONFIGURATION CONSIDERATIONS

The following are items to consider when configuring CICS/VS for the System/34. For more information about configuring CICS/VS, see the *CICS/VS System Programmer's Reference Manual.*

To include one or more System/34s in an existing or proposed CICS/VS network, the programmer responsible for generating CICS/VS must generate a terminal control program to support one or more System/3 terminals. The terminal control table that defines the network must contain descriptions of the terminals to be supported. The System/34s should be described as System/3 terminals in the DFHTCT macro. For a CICS/VS multipoint network, the DFTRMLST macro is used to define the polling/selection sequences to be used by BTAM. The System/34 CICS subsystem requires an expanded polling list entry as well as an expanded address list definition.

Unlike other CPU-type terminals which use 2-character polling/addressing (control unit address only), the CICS subsystem uses four characters per logical device (control unit address and device address). The control unit address corresponds to the *local station address* specified during the CNFIGICF procedure. The device address must be within the limits of A through O or blank, and refers to the multipoint session addresses defined during the CNFIGICF procedure.

In a multipoint network, the System/34 with the CICS subsystem can be defined as a control unit with multiple devices, or as up to 16 separate CPU-type devices. General polling of System/34s is not supported. The following example shows a listing of the CICS/VS DFTRMLST and DFHTCT macros for a multipoint BSC line.

```
ICFLMP1  DFHTCT  TYPE=SDSCI,DEVICE=SYS/3,BSCODE=EBCDIC,
               MACRF=(R,W),DSCNAME=ICFLMP1
*
*   1                              2    3
*
*
*
POLLICF  DFTRMLST AUTOWLST,(C2C2C1C12D,C2C2C2C22D,C2C2C3C32D,          X
               C2C2C4C42D,C2C2C5C52D,C2C2C6C62D,C2C2C7C72D,            X
               C2C2C8C82D,C2C2C9C92D,C2C2D1D12D,C2C2D2D22D,            X
               C2C2D3D32D,C2C2D4D42D,C2C2D5D52D,C2C2D6D62D,            X
               C2C240402D,3737373737)
*
*
*
*
*                     4
ARICFA   DFTRMLST OPENLST,(E2E2C1C12D)
ARICFB   DFTRMLST OPENLST,(E2E2C2C22D)
ARICFC   DFTRMLST OPENLST,(E2E2C3C32D)
ARICFD   DFTRMLST OPENLST,(E2E2C4C42D)
ARICFE   DFTRMLST OPENLST,(E2E2C5C52D)
ARICFF   DFTRMLST OPENLST,(E2E2C6C62D)
ARICFG   DFTRMLST OPENLST,(E2E2C7C72D)
ARICFH   DFTRMLST OPENLST,(E2E2C8C82D)
ARICFI   DFTRMLST OPENLST,(E2E2C9C92D)
ARICFJ   DFTRMLST OPENLST,(E2E2D1D12D)
ARICFK   DFTRMLST OPENLST,(E2E2D2D22D)
ARICFL   DFTRMLST OPENLST,(E2E2D3D32D)
ARICFM   DFTRMLST OPENLST,(E2E2D4D42D)
ARICFN   DFTRMLST OPENLST,(E2E2D5D52D)
ARICFO   DFTRMLST OPENLST,(E2E2D6D62D)
ARICFX   DFTRMLST OPENLST,(E2E240402D)
*
*
*
*
         DFHTCT  TYPE=LINE,ACCMETH=BTAM,TRMTYPE=SYS/3,                 X
               LINSTAT='OUT OF SERVICE',                               X
     5         INAREAL=4096,BSCODE=EBCDIC,BTAMRLN=1,                   X
               DSCNAME=ICFLMP1,FEATURE=AUTOPOLL,                       X
               LISTADR=(POLLICF,WRAP)
*
*
*
*
         DFHTCT  TYPE=TERMINAL,TRMIDNT=ICFA,TRMADDR=ARICFA,            X
               TRMSTAT=TRANSCEIVE,POLLPOS=1,TRMTYPE=SYS/3,             X
     6         TIOAL=768,CLASS=(CONV,BISYNC)
*
*
*        7                8
*
         DFHTCT  TYPE=TERMINAL,TRMIDNT=ICFB,TRMADDR=ARICFB,            X
               TRMSTAT=TRANSCEIVE,POLLPOS=2,TRMTYPE=SYS/3,             X
               TIOAL=768,CLASS=(CONV,BISYNC)

*
*
*
*
         DFHTCT  TYPE=TERMINAL,TRMIDNT=ICFO,TRMADDR=ARICFO,            X
               TRMSTAT=TRANSCEIVE,POLLPOS=15,TRMTYPE=SYS/3,            X
               TIOAL=768,CLASS=(CONV,BISYNC)
*
*
*
*
         DFHTCT  TYPE=TERMINAL,TRMIDNT=ICFX,TRMADDR=ARICFX,            X
               TRMSTAT=TRANSCEIVE,POLLPOS=16,TRMTYPE=SYS/3,            X
               TIOAL=768,CLASS=(CONV,BISYNC),                          X
               LASTTRM=LINE
```

1 This list is made up of 4-character polling addresses, specified in hexadecimal. The first four hexadecimal characters correspond to the *local station address* specified during CNFIGICF. The next four hexadecimal characters correspond to the *multipoint session address* defined during CNFIGICF and specified on the SESSION OCL statement. These values must be in the range of 'C1C1' through 'D6D6' for interactive sessions. The session reserved for incoming procedure start requests must have the last four hexadecimal characters of '4040'.

2 This is the local station address.

3 This is the multipoint session address.

4 The following lists are made up of 4-character selection addresses, specified in hexadecimal. Each list is used by CICS/VS to send data to a particular session.

5 This length must be greater than or equal to the longest record that the CICS subsystem will transmit.

6 This parameter describes the session as one that can both transmit and receive data and can start tasks on the System/370.

7 This length must be greater than or equal to the longest record to be sent by CICS/VS on this session.

8 The CONV specification in this parameter is optional. It could be replaced by BATCH, which is more common for System/3 type terminals.

STARTING AND ENDING THE CICS SUBSYSTEM

The ENABLE procedure is the means of starting the CICS subsystem on the System/34. The ENABLE procedure associates the subsystem with a particular BSC line. After being enabled, the subsystem monitors the line and waits for application program requests. The ENABLE procedure is described in Chapter 2.

The subsystem can be enabled before CICS/VS is started because no line control sequences are required until the application program requests an operation of the subsystem. If the subsystem is enabled after CICS/VS is started, the CICS/VS operator might have to place the terminal in service.

For a multipoint line, the subsystem responds to polling for each address defined.

The DISABLE procedure stops the subsystem. When a disable is performed, the association between the subsystem and the BSC line is broken, and no further activity occurs on the line. If the subsystem was active on a multipoint line using the multiline communications adapter, the adapter will continue to respond to polling after the disable has been completed.

STARTING CICS SUBSYSTEM APPLICATIONS

System/34 CICS applications are usually started by a display station operator entering a procedure name. The OCL statement that is unique to interactive communications procedures is the SESSION OCL statement. The SESSON statement establishes the parameters required to interface with a particular subsystem for a particular session. The format of the SESSION statement for the CICS subsystem is:

```
// SESSION LOCATION-name,SYMID-session-id

                        ┌ AA ┐                                ┌ YES ┐                  ┌ YES ┐
[,SESNADDR-x]  [,SWTYP- { MA }]  [,PHONE-name]  [,REFRESH-    {     }]  [,RESTORE-    {     }]
                        └ MC ┘                                └ NO  ┘                  └ NO  ┘
```

LOCATION: Specifies the location name associated with this session. The location name is defined during subsystem configuration, and refers to the name of the location with which communications is to take place.

SYMID: Specifies the symbolic ID of the session with which this SESSION statement is associated. The symbolic ID must be two characters, with the first character numeric (0 through 9) and the second character alphabetic (A through Z, #, $, or @). This is the same ID that the application program uses when referring to this session.

SESNADDR: Specifies the address to be used by this session on a multipoint line. This address must be an alphabetic character from A through 0, and must have been defined during subsystem configuration. If this parameter is not specified, the CICS subsystem assigns an address from the address pool. The SESNADDR parameter is used to assign a particular multipoint address to an acquired session.

SWTYP: Specifies the method of making the connection on a switched line. MC indicates that the System/34 operator will initiate the call manually, or that the System/34 will initiate the call automatically (see *Autocall Considerations* in this section for additional information). MA indicates that the System/34 operator will answer the call manually. AA indicates that the System/34 will automatically answer the call. If this parameter is not specified, the connection is established as configured.

PHONE: Specifies the name of the load member that contains the list of numbers for the autocall feature or the list of numbers for the public data network be called by the System/34. The load member must be in either the current user libary or in #LIBRARY. If you do not specify a list name, the name specified during subsystem configuration or the name specified when the subsystem is enabled is used.

Use the *REFRESH* parameter and the *RESTORE* parameter to define how you want the list of numbers processed.

REFRESH: Specifies if you want the list reinitialized (calling to begin with the first number on the list) after a successful call. If you enter a list name and do not enter a value for REFRESH, REFRESH-YES is assumed.

Specify NO if you do not want the list reinitialized after a successful call.

RESTORE: Specifies if you want the list used in a previous job step reinitialized before executing the current job step.

Specify NO if you do not want the list used in a previous job step reinitialized.

If you specify a list and do not specify RESTORE, RESTORE-NO is assumed.

Note: See *Automatic Calling on Switched Lines* in this chapter for more information about how to use the PHONE, REFRESH, and RESTORE parameters.

Incoming Procedure Start Requests

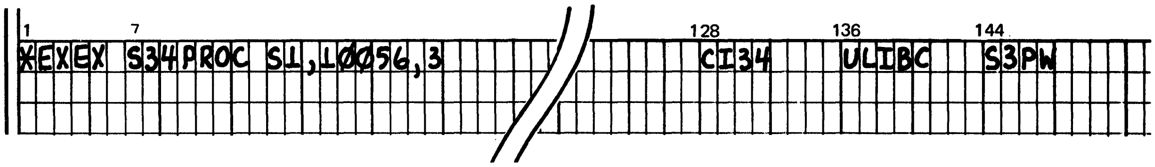
System/34 procedures can also be started by an incoming procedure start request. An application program started in this manner can receive up to 118 bytes of data with the request. The format of an incoming procedure start request accepted by the CICS subsystem is:

```
*EXEX procedure [optional data or procedure parameters] [user id]
          [library name] [password]
```

The *procedure* name must begin in position 7 and must be separated from the data and parameters by one or more blanks. The *data or parameters* are considered to be everything from the first nonblank character following the procedure name through position 127, and are available as positional procedure parameters or as data to the application program.

The *user id* begins in position 128, the *library name* begins in position 136, and the *password* begins in column 144.

The following is a correctly coded example of an incoming procedure request:



1
7
128
136
144

```
*EXEX S34PROC S1,1ØØ56,3      CI34    ULIBC   S3PW
```

More information on incoming procedure start requests and how to write procedures that are to be started by these requests is in Chapter 2.

The CICS/VS application sends the *EXEX request as intrapartition transient data or as an interval control put to a terminal using the CICS/VS command level START command. See the *CICS/VS Application Programmer's Reference Manual (Command Level)* for more information about the START command. CICS/VS sends a DFH2503 message preceding intrapartition transient data, but the CICS subsystem discards the message.

A session started by an incoming *EXEX exists until the application program accepts the data, at which time the subsystem terminates the session. If further operations are to be directed to the CICS subsystem by the System/34 application program, a SESSION OCL statement must be included in the System/34 procedure, and the System/34 program must request that a new session be acquired. The session used to pass data or parameters from the incoming procedure request to the procedure cannot be used for further communication with the subsystem or CICS/VS.

OPERATION CONSIDERATIONS

The following sections describe the operations supported by the CICS subsystem. A complete chart of all interactive communications operations and the subsystems that support them is in each language chapter. This chart also shows the keyword or format name used to code each operation. More information about how an operation is coded is in the appropriate programming language chapter.

Whether an operation completes successfully or unsuccessfully, a return code is given to the application program. All of the return codes that are valid for this subsystem are described at the end of this chapter. A summary chart in Appendix A lists all the return codes and the subsystems for which they are valid.

Acquire Operation

The acquire operation is used to start a session. A session is the line of communication between the application program and the subsystem. The acquire operation must specify the ID given in the SYMID parameter on a SESSION OCL statement. Other parameters on the SESSION statement are used by the subsystem while performing the acquire operation to determine which multipoint address to assign to the session, and what characteristics apply to the BSC line connection.

A return code of 82xx indicates that the acquire operation failed; the minor part of the return code indicates the reason for the failure.

Evoke Operations

An evoke operation is used to initiate a task on the CICS/VS system. The evoke operation causes the CICS/VS transaction ID (procedure name in the evoke parameter list) to be transmitted, along with any user-supplied data, to the CICS/VS system. The CICS subsystem uses only the task ID in the evoke parameter list; all other parameters are ignored.

The following are examples of how the evoke parameter list might be coded in each of the programming languages to start transaction RWP2 and pass it 64 characters of data:

Assembler: EVKLST $EVOK V-ALL,PNAME-'RWP2'

BASIC:
```
030 WRITE #1,USING 40,FORMAT "$$EVOK": "RWP2",DATA$ IOERR ICFERR
040 FORM C 8,X 24,C 64
```

COBOL:

```
01   EVOKE-PARM-LIST.

03   PROCEDURE-NAME   PIC X(4)  VALUE 'RWP2'.
03   FILLER           PIC X(48).
03   DATA-LENGTH      PIC 9999  VALUE 64.
03   USER-DATA        PIC X(64).
```

RPG II:

```
01 OICFILE  E        Ø4
02 O                                   K6 '$$EVOK'
03 O                                    4 'RWP2'
04 O                                   56 '64'
05 O                         DATA     12Ø
```

The evoke operation can be combined with an input operation ($$EVOK) or it can be specified as an output only operation ($$EVOKNI or $$EVOKET). In either case, a single record is transmitted followed by an EOT (BSC end of transmission control character) to indicate the end of this transmission. The record consists of the task ID followed by one blank followed by the data (if present).

A return code of 04xx indicates that the evoke operation could not be handled because input data had been received for this session before the evoke operation was started. The application program should issue an input operation to retrieve the input data, and then reissue the evoke operation. A return code of 80xx or greater indicates that the evoke operation failed; the minor part of the return code indicates the reason for the failure. For combined input and output operations, the minor part of the return code indicates which portion of the operation failed.

Put Operations

The put operation is used to pass data from the System/34 application program to the CICS/VS application program. The put operation can be combined with an input operation ($$SEND) or can be used to send one of a group of records to the CICS/VS system ($$SENDNI). If the put operation is combined with an input operation, one record, followed by an EOT, is sent. Then a record is received by the subsystem. In the case of a BASIC, COBOL, or RPG II program, the input data is made available to the application program on the next accept (READ) operation. Assembler users can choose to wait for the input data by using a put then get operation.

If the put operation is not combined with an input operation, the record is transmitted as part of a batch of records. The put operation that passes the last of a batch of records to the subsystem is the put end of file/chain operation ($$SENDE). The put end of transaction operation ($$SENDET) can also be used. These operations cause the subsystem to send the data associated with the put, followed by an EOT, before returning control to the application program.

If the put operation is not combined with an input operation and is not an end of file operation, the subsystem returns control to the application program immediately after completion of any previous operation for the session. Batch transmission is double buffered, that is, the data from the first put operation is sent as the data from the second is being readied for transmission.

Note: The CICS subsystem is not designed to efficiently support batch communications. See *Performance Considerations* later in this chapter for more information.

Return codes presented for put operations are identical to those for evoke operations. Return codes for batch transmission, however, might indicate the failure of the previous put operation. For combined input and output operations, the minor part of the return code indicates which portion of the operation failed.

Input Operations

The input operations for the CICS subsystem are invite, get, and accept. The invite operation can be issued only as a combined operation with a put or evoke operation ($$SEND, $$EVOK) in BASIC, COBOL, or RPG II. Assembler language users can issue an invite operation explicitly. Either a get or an invite operation signals the subsystem to obtain data on the session for the application program. A get operation causes the application program to wait for the data to be available. When a program issues an invite operation, it receives the data with the next accept operation. The accept operation allows input from any previously invited session or display station.

The end of file indication (EOT) is never received by the CICS subsystem with the input data. Therefore, the return code that indicates end of file is 0300 (no data received and EOT received).

Get Attributes Operation

The get attributes operation for assembler users can be issued at any time to determine the status of a session.

Set Timer Operation

The set timer operation ($$TIMER) results in a timer expired return code (0310) after a specified time interval in hours, minutes, and seconds has elapsed.

Release Operation

The release operation breaks the logical connection between the application program and the subsystem. The subsystem then frees all resources assigned to the session being released. The session address is made available to any application program that issues a subsequent acquire operation.

The release operation can only be issued if no translation is active. If EOT has not been sent following output or received following input, the release operation is not executed. In this case, the return code indicates the status of the session (input or output).

End of Session Operation

The end of session operation ($$EOS) always results in a normal completion return code. The session is always terminated by the end of session operation. If the session is still communicating when the end of session operation is issued, the transmission is aborted by the CICS subsystem, and abnormal termination of the CICS/VS task could result.

PROGRAMMING CONSIDERATIONS

Performance Considerations

The CICS subsystem is not designed to efficiently support batch communications. The transmission of large volumes of data without intervening EOTs prevents efficient use of the communications line in a multitasking environment. Use of the put end of file operation ($$SENDE) or put end of transaction ($$SENDET) allows the line to be freed between records. When the line is free, other applications with CICS sessions can use the line to send or receive data.

Security Considerations

CICS/VS provides security support keyed to the terminal operator rather than to the terminal itself, much like System/34 security. The CICS/VS security support is in the form of two CICS/VS transactions (CSSN and CSSF).

The rules for the use of the CSSN and CSSF transactions are as follows:

- The application program data associated with the CSSN evoke operation must contain the nonblank password (PS-password) and the name (NAME-name) separated by a comma. The password is 1 to 4 characters, and the name is 1 to 20 characters.

- The evoke for the CSSN transaction always results in a response from CICS/VS. If the CICS/VS message indicates that the sign-on was successful, the subsystem discards the message and returns a successful return code. No input operation need be done by the application program. If the message indicates that the sign-on failed, the return code from the evoke indicates that the operation failed and a message is waiting. The application program must then issue an input operation to get the CICS/VS messages.

- The sign-off (CSSF) transaction is used to sever the logical connection between the application program and CICS/VS. The CSSF transaction removes the previously specified password and name from the CICS/VS sign-on table.

- Upon successful completion of the evoke for CSSF transaction, the subsystem makes the CICS message text available to the application program. The application program must then issue an input operation to get the CICS/VS message.

- The CICS subsystem monitors the use of CSSN and CSSF transactions by System/34 application programs. If the application program has issued a successful CSSN evoke, any request to release the session without first issuing a CSSF evoke causes the subsystem to issue an automatic CSSF transaction. When CICS/VS responds that the sign-off is complete, the subsystem proceeds with the release, and in this case the sign-off response message from the CICS/VS system is discarded. The CSSF is sent automatically only if the release operation is used; an end of session operation without a previous CSSF transaction results in abnormal termination.

- After a successful CSSF evoke, the application program can again issue a CSSN evoke to the same session or to another session. A different password and name can be used each time.

CICS/VS Messages

CICS/VS can issue any of several messages of the following format:

DFHccnn text

ccnn represents a CICS/VS message number as described in *CICS/VS Messages and Codes Manual.*

When such a message is received by the CICS subsystem, a message waiting return code is given to the application program for the next operation. If the next operation is an input operation, the message is returned in the application program input area. If the next operation is a put operation, the operation is rejected and a return code indicating a message is waiting is given to the application program. In this case, the program must issue an input operation to obtain the message text and clear the return code. Until this is done, further output operations will be rejected.

CICS/VS messages can exceed the application program data length. If this occurs, the message is truncated on the right, and a return code indicating the truncation is passed to the application program.

Using Switched Lines

On switched lines, you can do the following:

- Call a remote location manually
- Answer a call from a remote location manually
- Call a remote location automatically (if you have the autocall or switched X.21 feature)
- Answer a call from a remote location automatically (if you have the auto answer or switched X.21 feature)

Note: If you have the switched X.21 feature, the System/34 calls and answers automatically. You cannot call or answer manually with the X.21 feature.

To establish communications on a switched line, your application program must send a special record following the acquire operation and before the evoke operation. This record consists of the terminal ID as specified in the TRMIDNT parameter of the DFHTCT TYPE=TERMINAL macro during CICS/VS generation. The record can be any length up to the subsystem maximum record length. CICS does not send a response to this record. The next record should be sent by the evoke operation.

After a connection has been established, the CICS subsystem will not disconnect the line unless a line error or abnormal termination occurs. Your application program can terminate the CICS/VS connection by issuing an evoke then invite operation ($$EVOK) using the CSSF transaction with the keyword GOODNIGHT as data. The line is disconnected. If your program issues another operation for this session, the program will receive an 8193 or 8194 return code (disconnect indication received). Your application program can issue either a release operation and end, or establish the same or a different connection and issue another input or output operation.

Manual Calling and Manual Answering on Switched Lines

If you want the operator to call a remote location manually, specify 1 (manual call) on display 4.0 of the CNFIGICF procedure and do not specify a list name for the phone list name parameter on display 5.1 or for the PHONE parameter on the SESSION statement. The message OPERATOR DIAL REQUIRED is displayed on the system console when the operator is to make the call.

If you want the operator to answer a call from a remote location manually, specify 3 (manual answer) on display 4.0. The message OPERATOR ANSWER REQUIRED is displayed on the system console when the operator is to answer the call.

Note: The message asking the operator to call or answer is displayed when your program sends the special record described previously.

Automatic Calling on Switched Lines

The System/34 can call remote locations automatically on switched lines using either the autocall or switched X.21 feature. To call a remote location(s) automatically, you must do the following:

- Specify the correct configuration parameters.
- Create a list of phone numbers for the autocall unit or a list of numbers for the public data network.
- Enter the name of the list when you configure the subsystem, when you enable the subsystem, or when you write the SESSION OCL statement.
- Enable the subsystem on an autocall or switched X.21 line.
- Send the special CICS record following the acquire operation.

Configuring the Subsystem for the Autocall Feature: When you use the autocall feature, the System/34 calls the remote locations automatically regardless of the switch type parameter you specified during subsystem configuration or enable. However, you should specify 1 (manual call), which allows the operator to enable the subsystem on a manual call line if an autocall line is not available. A message is then displayed to ask the operator to dial the remote location manually.

Configuring the Subsystem for the X.21 Feature: When you use the switched X.21 feature, specify 2 (auto answer) for the switch type parameter. The System/34 calls the location(s) automatically and answers calls automatically when you specify auto answer. You cannot call a location manually or answer a call manually with the X.21 feature.

Creating a List for Autocall or the Public Data Network: The list for autocall or the list for the public data network contains the number or numbers you want the System/34 to call. You create the list and store it in a load member using the DEFINEPN or DEFINX21 procedure. These procedures are described in the *System Support Reference Manual.*

If the list name is specified during subsystem configuration or when the subsystem is enabled, the load member containing the list must be in either #LIBRARY or in the same library as the configuration member. If the list name is specified on the SESSION statement, the load member must be in either #LIBRARY or the current user library.

Following is an example of how the System/34 handles a list. Although the example shows a list for the autocall feature, the System/34 handles a list for the public data network (X.21 feature) in the same manner except where noted.

Phone List Example: The System/34 calls the numbers in the order listed. When a number is called, the call might be unsuccessful. A retry count specified during the DEFINEPN procedure is associated with each number in the list. When a call is unsuccessful, the retry count is decremented by 1, and the next number on the list is called as shown in the following example:

Note: If you are using the switched X.21 feature, the retry count is decremented depending upon the reason for the unsuccessful call. A call progress signal (CPS) gives the reason for the unsuccessful call. The call progress signal is displayed on the system console with the unsuccessful call message. If the call progress signal begins with a 4, 5, or 7, the number is attempted only once, the retry count is set to 0, and the number is marked as unsuccessfully called.

Phone Number	**Retry Count**	
8672906	5 -1 = 4	(The retry count is decremented and the next number is called.)
6286500	1	
6280363	3	

This number will not be called again until the other numbers on the list have been called or until the phone list is reinitialized.

If the retry count reaches 0, a message is displayed on the system console indicating that the call was unsuccessful, and the number is marked as unsuccessfully called.

Phone Number	**Retry Count**	
8672906	5 -1 = 4	
6286500	1 -1 = 0	(Unsuccessfully called)
6280363	3	

When a number is marked as unsuccessfully called, it is not called again until the phone list is reintialized.

If a number is successfully called, an operator message is displayed on the system console, the line connection is established, and the number is marked as successfully called.

Phone Number	**Retry Count**	
8672906	5 -1 = 4	
6286500	1 -1 = 0	(Unsuccessfully called)
6280363	3	(Successfully called)

The number is not called again until the list is reinitialized.

In the previous example, the first number in the list has not been marked as called. Therefore, if an application program uses the phone list without reinitializing it, the first number is the only number that can be called. If the call is successful, the number is marked as successfully called.

Phone Number	**Retry Count**
8672906	4 (Successfully called)
6286500	0 (Unsuccessfully called)
6280363	3 (Successfully called)

The phone list must now be reinitialized to be used again. If the list is not reinitialized and an application program attempts to use the list, a message PHONE LIST EXHAUSTED is displayed on the system console, and a return code of xx86 is returned to the application program.

In the previous example, if the call to the first number in the list is unsuccessful and the retry count reaches 0, the number is marked as unsuccessfully called, a message NO NUMBERS REACHED is displayed on the system console, and a return code of xx85 is returned to the application program. The phone list must be reinitialized to be used again.

Phone Number	**Retry Count**
8672906	0 (Unsuccessfully called)
6286500	0 (Unsuccessfully called)
6280363	3 (Successfully called)

When the list is reinitialized, all retry counts are set to the counts specified during the DEFINEPN procedure, and calling begins with the first number in the list when the list is used again.

The specification of the REFRESH and RESTORE parameters indicates how the list is reinitialized. The REFRESH parameter can be specified during configuration, during ENABLE, or on the SESSION statement. The RESTORE parameter can be specified on the SESSION statement only. Following is a description of the REFRESH and RESTORE parameters.

REFRESH Parameter

If you specify REFRESH-YES or do not specify REFRESH (the default is YES), the list is reinitialized after the first successful call or after all numbers in the list have been marked as unsuccessfully called.

If you specify REFRESH-NO, the list is not reinitialized after a successful call. The list is reinitialized as follows:

- After the PHONE LIST EXHAUSTED message is displayed
- After the NO NUMBERS REACHED message is displayed
- As specified by the RESTORE parameter

RESTORE Parameter

If you specify RESTORE-YES on the SESSION statement, the list specified is reinitialized prior to executing the current step in the procedure. The current user library is searched first for the list. If the list is not found, the system library is searched.

If you specify RESTORE-NO, the list is not reinitialized prior to executing the current step in the procedure. The default is NO.

IF LISTDONE Conditional Expression

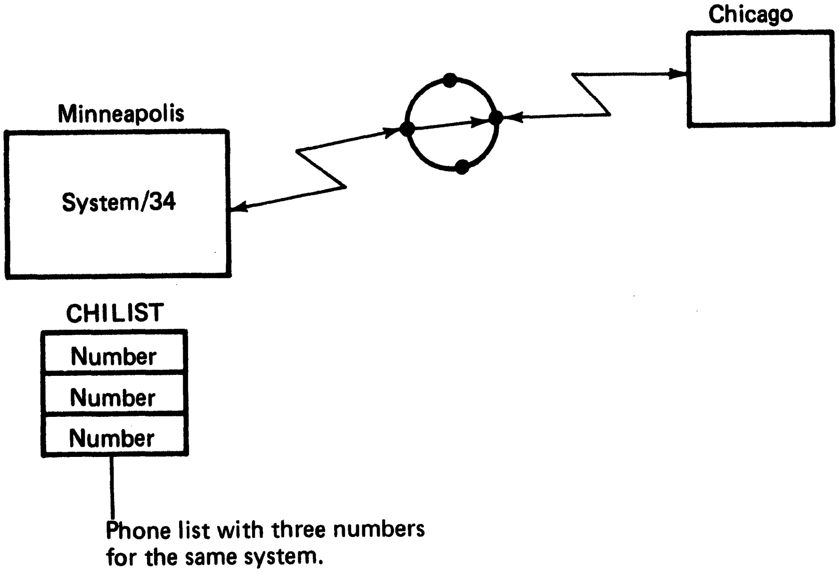
You can use the IF LISTDONE conditional expression to determine if all of the numbers in a list used by a previous job step have been called. If all numbers in the list are marked as successfully called or unsuccessfully called (retry counts are 0), the expression is true. The expression is false if REFRESH-YES was specified on the SESSION statement and any number in the list was successfully called. The expression can only be used to test lists that have been specified on a SESSION statement. If the name of the list tested was not specified on the SESSION statement, the expression is false.

Note: In some countries a delay is required before a call can be placed when you use the autocall feature. If this delay is required in your country, you specify the delay when you create the phone list using the DEFINEPN procedure. See the *System Support Reference Manual* for information about the DEFINEPN procedure and the delay parameter.

The following examples show how you can use the *REFRESH* parameter, the *RESTORE* parameter, and the *IF LISTDONE* statement.

Example of Calling One Location

In this first example, the System/34 calls one location many times. A phone list name was entered on display 5.1 of the CNFIGICF procedure. The list in this example contains three numbers for the same remote system in Chicago. When the remote system is called, it is not important which number is called successfully, but we want the System/34 to begin calling with the first number in the list each time the program uses the list. To do this, 1 (yes) was specified for the refresh parameter on display 5.1.



The list and setup in this example can be used for either batch or interactive communications with the remote system.

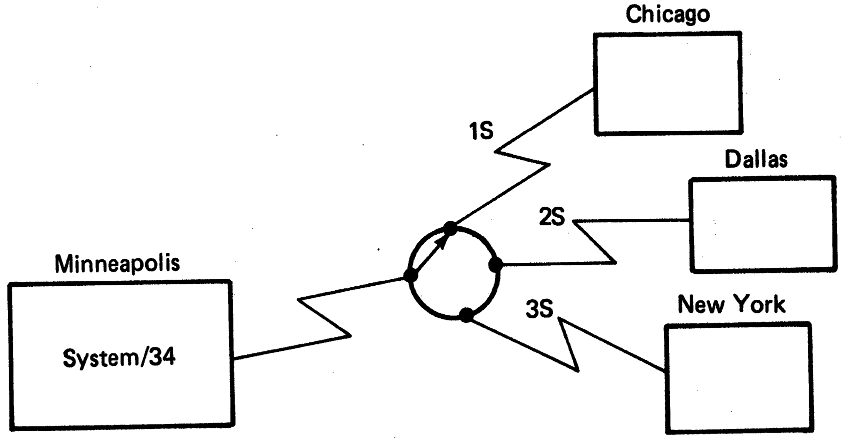
Example of Calling Multiple Locations (Refresh-Yes)

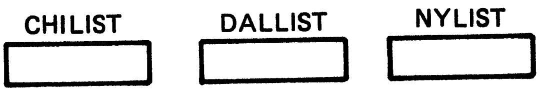
In this example, an MRT program is used for interactive communications with three remote systems.

One phone list was created for each remote system with one number in each list. The phone list names and refresh parameters are specified on the SESSION OCL statements as follows:

```
// LOAD MRTPROG
// SESSION LOCATION-MULTLOC,SYMID-1S,PHONE-CHILIST
// SESSION LOCATION-MULTLOC,SYMID-2S,PHONE-DALLIST
// SESSION LOCATION-MULTLOC,SYMID-3S,PHONE-NYLIST
// RUN
```

For example, the program calls the system in Chicago (session 1S) using the list CHILIST. When the System/34 program processes the data from Chicago, it determines that additional data is needed. The program then calls Dallas (session 2S) using the phone list DALLIST to find the additional data. The data was not at the Dallas location, so the program calls New York (session 3S) using phone list NYLIST and gets the data it needs. After processing the data, the program calls Chicago again and sends the results to the Chicago system.



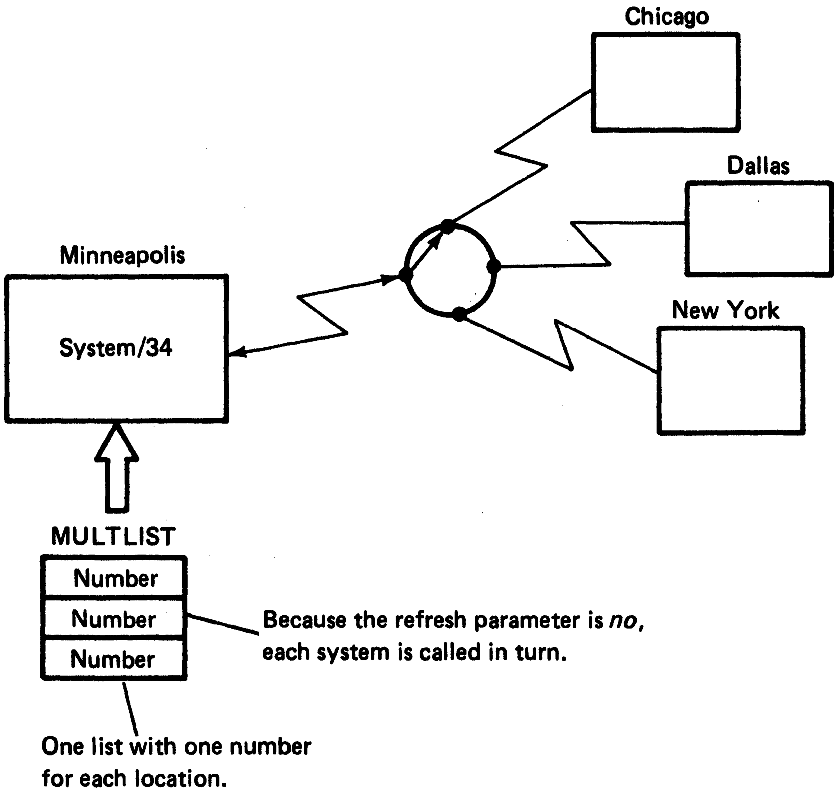


One list for each location.

Example of Calling Multiple Locations (Refresh-No)

In the two previous examples, the refresh parameter was set to *yes*. In this example, the parameter is set to *no*. One phone list is used, which was specified on display 5.1 of the CNFIGICF procedure. The refresh parameter was also set to 0 (no) on this display.

The program in this example loops through the list calling each system in turn to send one or more files of data to each system. The program checks for return code 8285 (NO NUMBERS REACHED) or 8286 (PHONE LIST EXHAUSTED) to determine when all numbers on the list have been called or tried.



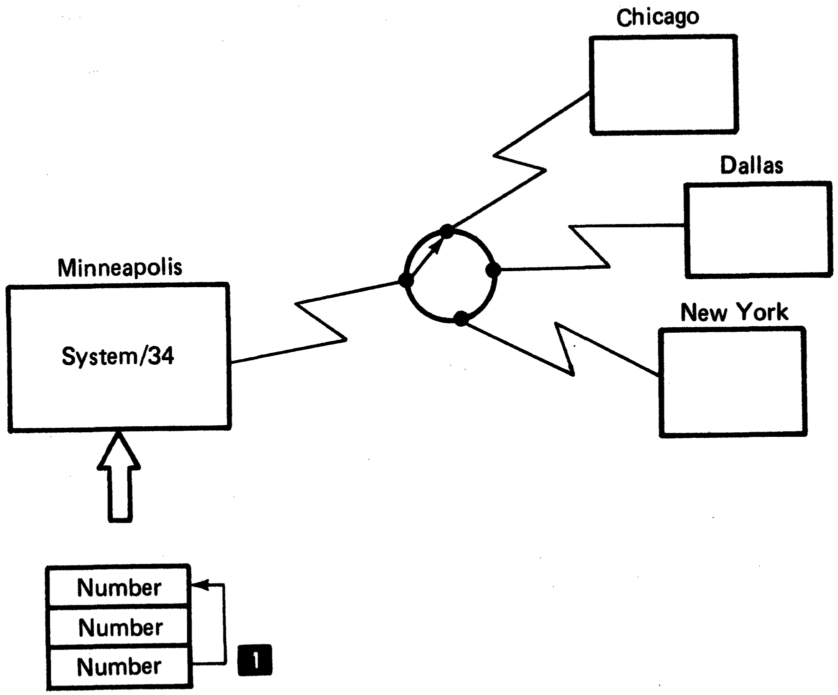
Because the refresh parameter is *no*, each system is called in turn.

One list with one number for each location.

Example of Calling Multiple Locations (RESTORE-YES)

In this example, the phone list is reinitialized by entering RESTORE-YES on the SESSION statement. Two programs (A and B) in one procedure use the same list to call the same remote system. If program A ends before all numbers have been called, the list is not reinitialized. To ensure that the list is reinitialized before program B uses the list, RESTORE-YES is entered on the SESSION statement. This ensures that the first number on the list is called when program B uses the list. For example:

```
// LOAD PROGA
// SESSION LOCATION-MULTLOC,SYMID-1S,
// PHONE-MULTLIST,REFRESH-NO
// RUN
// LOAD PROGB
// SESSION LOCATION-MULTLOC,SYMID-1S,
// PHONE-MULTLIST,REFRESH-NO,RESTORE-YES
// RUN                                1
```



The list is reinitialized before the next program uses it.

Example of Using the IF LISTDONE OCL Statement

You can also use the IF LISTDONE OCL statement to test the status of the list if your program does not test for a list done return code. This example shows how you can use the IF LISTDONE statement to check the list status.

Each time the program is loaded, it calls the next system, transmits or receives one or more files of data, and ends. If the list is done, the procedure also ends; otherwise, the program is loaded again and the next system is called.

```
// TAG TOP
// LOAD PROGA     1                              2             3
// SESSION LOCATION-MULTIPLE,SYMID-1S,PHONE-MULTLIST,REFRESH-NO
// RUN
// IFF LISTDONE-MULTLIST GOTO TOP    4
```

1 Each location is called, one at a time.

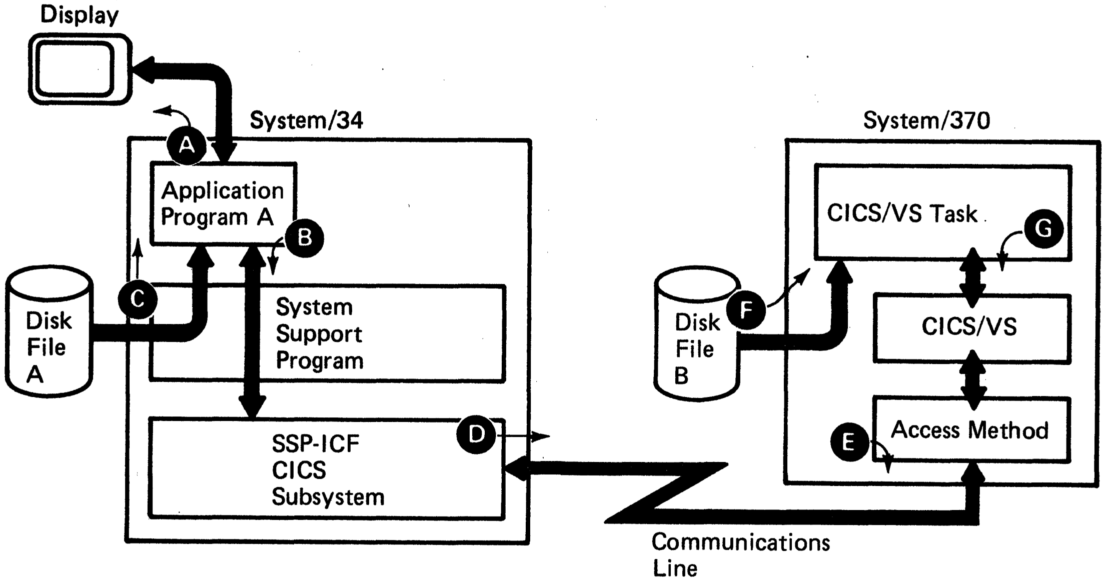
2 The list contains one number for each location.

3 The parameter REFRESH-NO allows each number to be called.

4 If all numbers have not been called, go to TOP.

How to Write Programs that Use the CICS Subsystem

The following example inquiry application is used in the CICS programming example:



1. Application program A (in the System/34) displays a prompt asking an operator to enter an item number requesting the stock status for the item Ⓐ.

2. When the operator enters the item number, program A reads the number and searches file A (the local file) for the item Ⓒ.

3. If the item is found in the local file, program A displays the stock status on the screen Ⓐ.

4. If the item is *not* in the local file, program A uses the CICS subsystem to send the item number to CICS Ⓑ and Ⓓ.

5. Program B (in the remote system) uses the item number to search the remote file for the item Ⓕ.

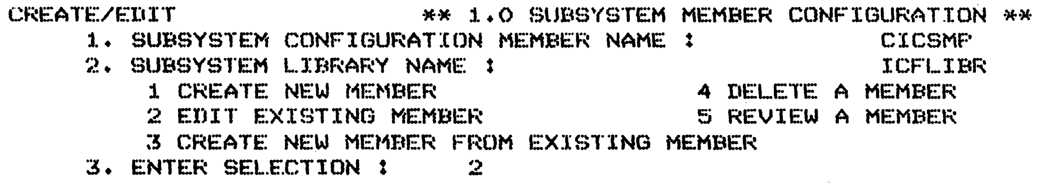
6. If the item is in the remote file, CICS sends the stock status to program A Ⓖ and Ⓔ. If the item is not in the remote file, CICS sends the characters ***.

7. If program A receives the stock status, it displays it. If it receives the characters ***, it displays the message ITEM NOT FOUND Ⓐ.

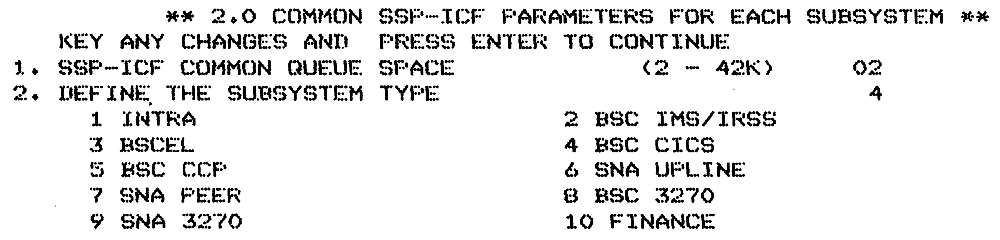
Except for a few minor changes, program A in this example is the program described in Chapter 7 for the Intra subsystem. The changes to the program required for CICS are shown in this chapter following the configuration and OCL examples. If you have not read the description of program A in Chapter 7, see *How to Write Programs that Use the Intra Subsystem* in Chapter 7. The application, configuration, and OCL examples in Chapter 7 are for the Intra subsystem only; you do not need to read those. Following are the configuration parameters and OCL statements for this subsystem.

Configuration Parameters

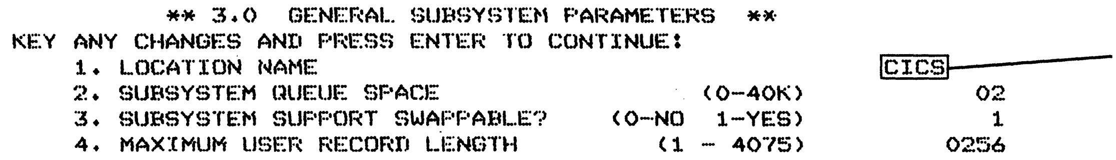
The following configuration parameters are used for this example:



```
CREATE/EDIT                    ** 1.0 SUBSYSTEM MEMBER CONFIGURATION **
      1. SUBSYSTEM CONFIGURATION MEMBER NAME :                CICSMP
      2. SUBSYSTEM LIBRARY NAME :                             ICFLIBR
         1 CREATE NEW MEMBER                       4 DELETE A MEMBER
         2 EDIT EXISTING MEMBER                    5 REVIEW A MEMBER
         3 CREATE NEW MEMBER FROM EXISTING MEMBER
      3. ENTER SELECTION :      2
```

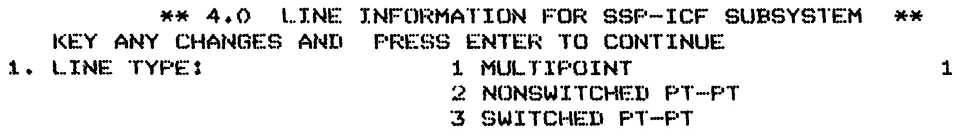


```
          ** 2.0 COMMON SSP-ICF PARAMETERS FOR EACH SUBSYSTEM **
   KEY ANY CHANGES AND  PRESS ENTER TO CONTINUE
1. SSP-ICF COMMON QUEUE SPACE                 (2 - 42K)      02
2. DEFINE THE SUBSYSTEM TYPE                                  4
      1 INTRA                            2 BSC IMS/IRSS
      3 BSCEL                            4 BSC CICS
      5 BSC CCP                          6 SNA UPLINE
      7 SNA PEER                         8 BSC 3270
      9 SNA 3270                         10 FINANCE
```



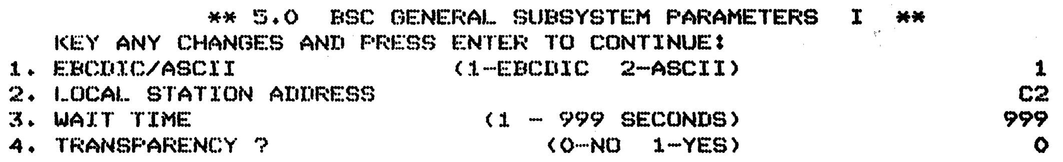
```
          ** 3.0  GENERAL SUBSYSTEM PARAMETERS  **
KEY ANY CHANGES AND PRESS ENTER TO CONTINUE:
     1. LOCATION NAME                                     CICS
     2. SUBSYSTEM QUEUE SPACE                  (0-40K)       02
     3. SUBSYSTEM SUPPORT SWAPPABLE?      (0-NO  1-YES)       1
     4. MAXIMUM USER RECORD LENGTH          (1 - 4075)     0256
```

Also specified on the SESSION statement.

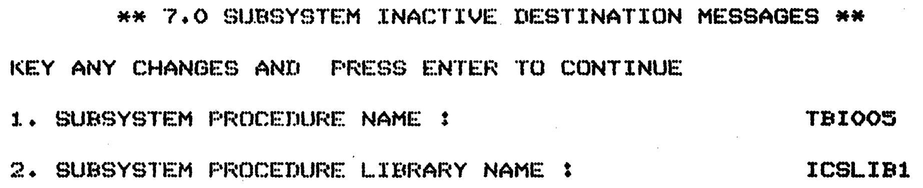


```
          ** 4.0  LINE INFORMATION FOR SSP-ICF SUBSYSTEM  **
   KEY ANY CHANGES AND  PRESS ENTER TO CONTINUE
1. LINE TYPE:                 1 MULTIPOINT                 1
                              2 NONSWITCHED PT-PT
                              3 SWITCHED PT-PT
```

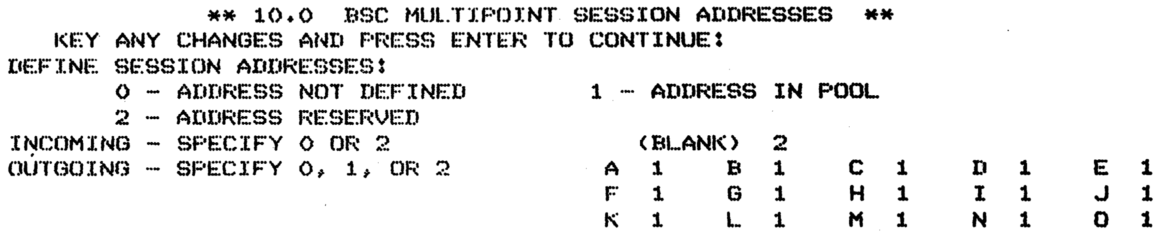
Configuration Parameters (continued)



```
              ** 5.0  BSC GENERAL SUBSYSTEM PARAMETERS  I  **
   KEY ANY CHANGES AND PRESS ENTER TO CONTINUE:
1. EBCDIC/ASCII              (1-EBCDIC  2-ASCII)                1
2. LOCAL STATION ADDRESS                                       C2
3. WAIT TIME                   (1 - 999 SECONDS)              999
4. TRANSPARENCY ?                  (0-NO  1-YES)                0
```



```
        ** 7.0 SUBSYSTEM INACTIVE DESTINATION MESSAGES **

KEY ANY CHANGES AND  PRESS ENTER TO CONTINUE

1. SUBSYSTEM PROCEDURE NAME :                      TBI005

2. SUBSYSTEM PROCEDURE LIBRARY NAME :              ICSLIB1
```



```
              ** 10.0  BSC MULTIPOINT SESSION ADDRESSES  **
   KEY ANY CHANGES AND PRESS ENTER TO CONTINUE:
DEFINE SESSION ADDRESSES:
       0 - ADDRESS NOT DEFINED        1 - ADDRESS IN POOL
       2 - ADDRESS RESERVED
INCOMING - SPECIFY 0 OR 2                (BLANK)  2
OUTGOING - SPECIFY 0, 1, OR 2          A  1    B  1    C  1    D  1    E  1
                                       F  1    G  1    H  1    I  1    J  1
                                       K  1    L  1    M  1    N  1    O  1
```

OCL Statements

The following procedure and OCL statement are used for the BASIC example:

```
BASICR CICSBAS,ITEMBAS,30,,CICSSESS
// SESSION LOCATION-CICS,SYMID-1S
```

The BASICR procedure includes the SESSION statement.

The location name (CICS) is also specified on display 3.0 of the CNFIGICF procedure.

The SYMID (session ID) is explained in the description of the program.

The following OCL statements are used for the COBOL example:

```
// LOAD COBCIC
// FILE NAME-FILEA
// SESSION LOCATION-CICS,SYMID-1S
// RUN
```

The location name (CICS) is also specified on display 3.0 of the CNFIGICF procedure.

The SYMID (session ID) is explained in the description of the program.

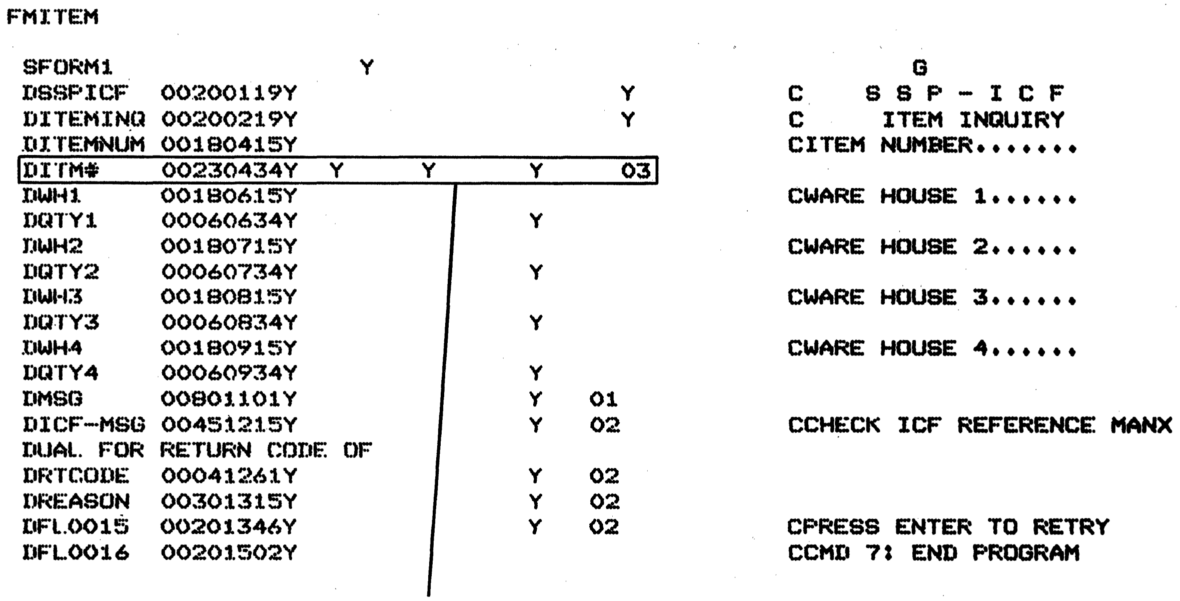
The following OCL statements are used for the RPG II example:

```
// LOAD RPGCIC
// FILE NAME-FILEA
// SESSION LOCATION-CICS,SYMID-1S
// RUN
```

The location name (CICS) is also specified on display 3.0 of the CNFIGICF procedure.

The SYMID (session ID) is explained in the description of the program.

Changes for the Screen Format



```
FMITEM

 SFORM1                 Y                                   G
 DSSPICF  00200119Y                      Y         C    S S P - I C F
 DITEMINQ 00200219Y                      Y         C     ITEM INQUIRY
 DITEMNUM 00180415Y                                CITEM NUMBER.......
 DITM#    00230434Y  Y     Y       Y    03
 DWH1     00180615Y                                CWARE HOUSE 1......
 DQTY1    00060634Y                Y
 DWH2     00180715Y                                CWARE HOUSE 2......
 DQTY2    00060734Y                Y
 DWH3     00180815Y                                CWARE HOUSE 3......
 DQTY3    00060834Y                Y
 DWH4     00180915Y                                CWARE HOUSE 4......
 DQTY4    00060934Y                Y
 DMSG     00801101Y                Y  01
 DICF-MSG 00451215Y                Y  02           CCHECK ICF REFERENCE MANX
 DUAL. FOR RETURN CODE OF
 DRTCODE  00041261Y                Y  02
 DREASON  00301315Y                Y  02
 DFL0015  00201346Y                Y  02           CPRESS ENTER TO RETRY
 DFL0016  00201502Y                                CCMD 7: END PROGRAM
```

The item number parameters have been changed. For the example in Chapter 7, the item number sent to program B was right-adjusted. However, the item number sent to CICS must be left-adjusted.

Changes for the Programming Example

The following examples show the changes required for program A, described in Chapter 7, to permit communications with CICS.

Changes for BASIC

```
00060 DIM ITEM$*23,MES$*80,REASON$*30
00070 OPEN #1: "WS,NAME=FMITEM,RECL=161,LIBR=ICFLIBR"
00080 OPEN #2: "NAME=FILEA,SHR,RECL=50,KEYL=23,KEYP=1,RANDOM",KEYED,INPUT
00090 INDIC$(1:2)="11"
```

See *Changes for the Screen Format* earlier in this example.

The program ICII is evoked at the host system. The password, user ID, and library name are not required in this example.

```
00410 !*---------------------------------------------------------------*
00420 !*                       EVOKE PROCEDURE 'ICII'                  *
00430 !*---------------------------------------------------------------*
00440 EVOK:OPCODE$="EVOK"
00450   WRITE #3,USING 460,FORMAT "$$EVOK": "ICII",ITEM$ IOERR ICFERR
00460   FORM C 8,X 24,C 23
00470 !*---------------------------------------------------------------*
00480 !*        GET INPUT FROM 'ICII'. IF ITEM NUMBER IS FOUND,        *
00490 !*              DISPLAY THE INVENTORY INFORMATION                *
00500 !*---------------------------------------------------------------*
00510 OPCODE$="GET"
00520 READ #3,USING 530: DATA$ IOERR ICFERR
00530 FORM C 3
00540 IF DATA$="***" THEN NOITEM
00550   REREAD #3,USING 560: ITEM$,QTY1$,QTY2$,QTY3$,QTY4$ IOERR ICFERR
00560   FORM X 32,C 23,X 145,4*C 6
00570   INDIC$(1:2)="11"
00580   GOTO DISPLY
```

Input is from ICII at the host system.

Note: This program does not send an end of transaction ($$SENDET). The host system ends the transaction. The program does receive a return code when the end of transaction is received.

Changes for COBOL

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.

    SELECT TRANSACTION-FILE
       ASSIGN TO WORKSTATION-FMITEM-01,
       ORGANIZATION IS TRANSACTION,
       FILE STATUS IS WS-FS, ICF-FS,
       CONTROL-AREA IS WS-CONTROL-AREA.

    SELECT FILEA-FILE ASSIGN TO DISK-FILEA,
       ORGANIZATION IS INDEXED, ACCESS IS RANDOM,
       RECORD KEY IS FILEA-NUMBER.

    SELECT PRINT-FILE ASSIGN TO PRINTER-PRINTER.
```

See *Changes for the Screen Format* earlier in this example.

```
01  EVOKE-RECORD.
    03  PROCEDURE-NAME              PIC X(8) VALUE 'ICII'.
    03  PASSWORD                    PIC X(8).
    03  USER-ID                     PIC X(8).
    03  LIBRARY-NAME                PIC X(8).
    03  FILLER                      PIC X(20).
    03  DATA-LENGTH                 PIC XXXX VALUE '0023'.
    03  ICF-ITEM-NUMBER-OUT         PIC X(23).
```

The program ICII is evoked at the host system.

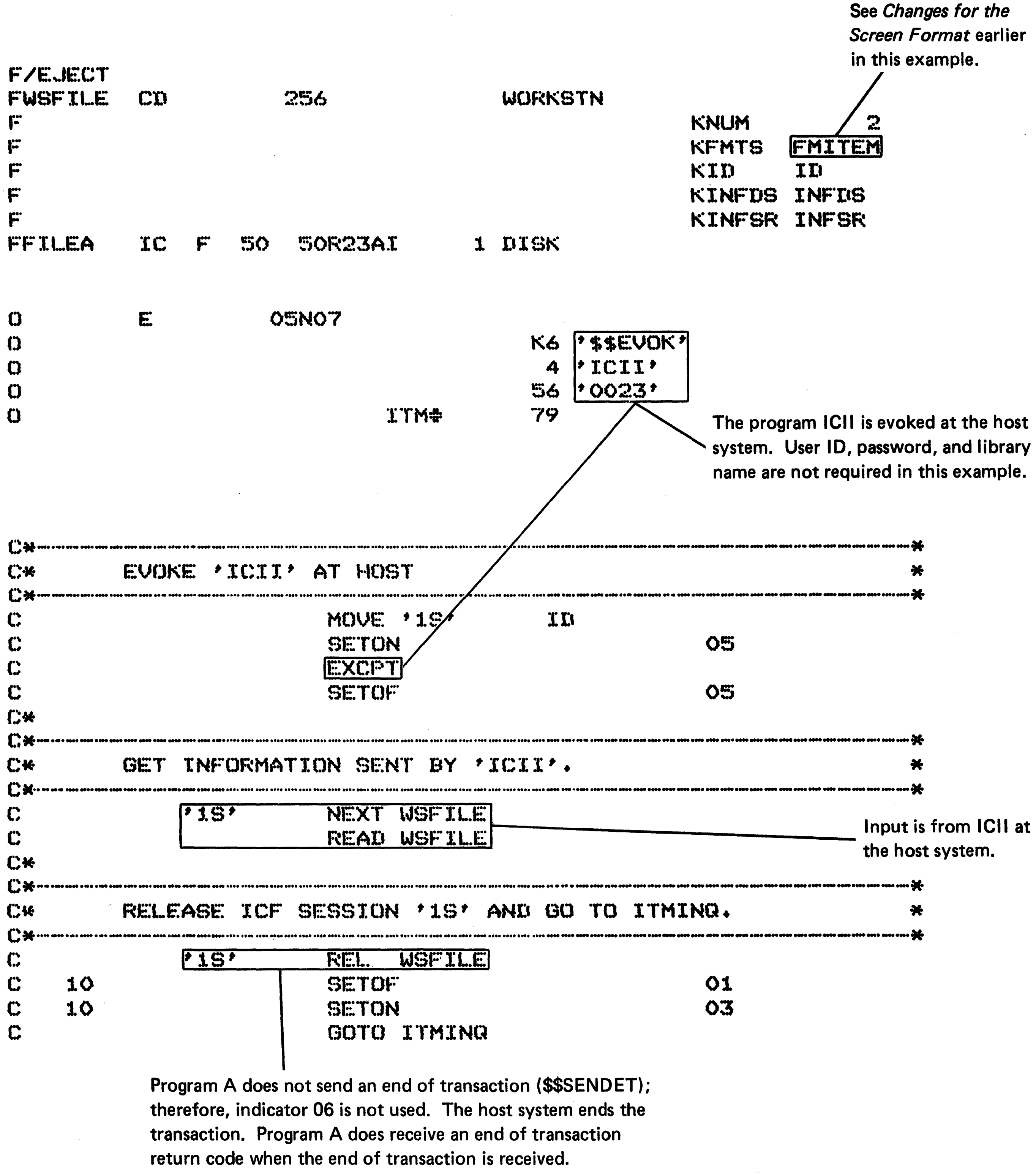
These parameters are not required in this example.

```
*--------------------------------------------------------------*
*    EVOKE 'ICII' AT HOST                                      *
*--------------------------------------------------------------*
     MOVE ITEM-NUMBER TO ICF-ITEM-NUMBER-OUT.
     WRITE TRANSACTION-RECORD FROM EVOKE-RECORD
        FORMAT IS '$$EVOK',  TERMINAL IS ICF-SESSION.
     MOVE 'EVOK' TO OPCODE.
     PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
     IF IO2 = B'O'
        PERFORM SEND-EOS,
        GO TO ITEM-INQUIRY.
*--------------------------------------------------------------*
*    GET INPUT FROM HOST
*--------------------------------------------------------------*
     MOVE SPACES TO ICF-RECORD-IN.
     READ TRANSACTION-FILE RECORD INTO ICF-RECORD-IN,
       TERMINAL IS ICF-SESSION.
     MOVE 'GET' TO OPCODE.
     PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
     IF IO2 = B'O'
        PERFORM SEND-EOS,
        GO TO ITEM-INQUIRY.
*--------------------------------------------------------------*
*    RELEASE SESSION                                           *
*--------------------------------------------------------------*
     DROP ICF-SESSION FROM TRANSACTION-FILE.
     MOVE 'DROP' TO OPCODE.
     PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
     IF IO2 = B'O'
       PERFORM SEND-EOS.
```

Input is from the host system.

Program A does not send $$SENDET (end of transaction). The host system ends the transaction. The program does receive a return code when the end of transaction is received.

Changes for RPG II



See *Changes for the Screen Format* earlier in this example.

```
F/EJECT
FWSFILE  CD        256              WORKSTN
F                                                 KNUM         2
F                                                 KFMTS   FMITEM
F                                                 KID     ID
F                                                 KINFDS  INFDS
F                                                 KINFSR  INFSR
FFILEA   IC  F  50  50R23AI      1 DISK

O        E        05N07
O                                       K6 '$$EVOK'
O                                        4 'ICII'
O                                       56 '0023'
O                            ITM#       79

C*-----------------------------------------------------------*
C*      EVOKE 'ICII' AT HOST                                 *
C*-----------------------------------------------------------*
C                          MOVE '1S'      ID
C                          SETON                     05
C                          EXCPT
C                          SETOF                     05
C*
C*-----------------------------------------------------------*
C*      GET INFORMATION SENT BY 'ICII'.                      *
C*-----------------------------------------------------------*
C             '1S'         NEXT WSFILE
C                          READ WSFILE
C*
C*-----------------------------------------------------------*
C*      RELEASE ICF SESSION '1S' AND GO TO ITMINQ.           *
C*-----------------------------------------------------------*
C             '1S'         REL  WSFILE
C   10                     SETOF                     01
C   10                     SETON                     03
C                          GOTO ITMINQ
```

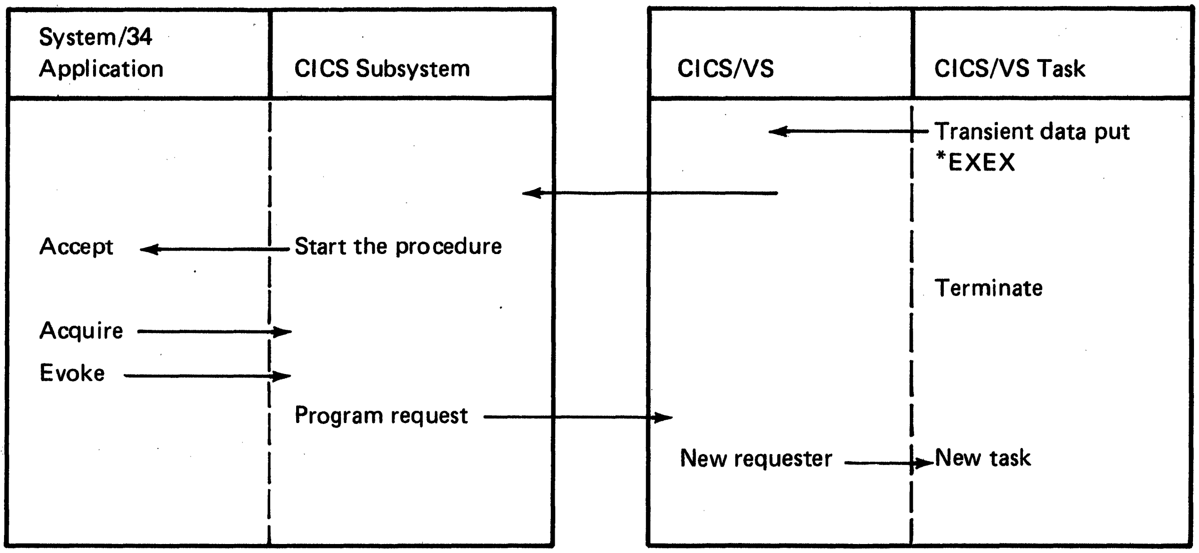
The program ICII is evoked at the host system. User ID, password, and library name are not required in this example.

Input is from ICII at the host system.

Program A does not send an end of transaction ($$SENDET); therefore, indicator 06 is not used. The host system ends the transaction. Program A does receive an end of transaction return code when the end of transaction is received.

Remote Procedure Start Request Example

The following chart shows a sample application program flow for a CICS/VS application program that starts a System/34 procedure.



CICS Subsystem Return Codes

This part of Chapter 10 describes all the return codes that are valid for the CICS 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 four-digit value; the first two digits contain the major code, and the last two 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).

Note: In the return code descriptions, *your program* refers to the local System/34 application program that initiates the operation and receives the return code from the subsystem. The *remote program* refers to the application program in the remote (or host) system with which the System/34 application program is communicating through SSP-ICF.

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 CICS subsystem. Although all the operations shown are valid for CICS, their validity also depends on the logical sequence of communications events occurring between the System/34 and the remote system.

Input Operations to Your Program	Output Operations from Your Program	Combined Operations in Your Program
Accept input		
	Acquire[1]	
	End of session	
	Evoke Evoke end of transaction	Evoke then get[2] Evoke then invite
Get Get attributes[3]		
Invite		
	Put Put end of file Put end of transaction	Put then get[2] Put then invite
	Release	
	Set timer[4]	

[1] Normally, 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.

[2] Valid only in assembler language.

[3] Valid only in assembler and COBOL languages.

[4] For BASIC and RPG II programs, the set 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.

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, or it received a message from the remote system.

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:** The *output* operation just performed by your program was completed successfully; your program can continue to send data.

Normal Action: For the actions that can be taken (in this session) after 0000 is returned for an output operation, refer to the following chart:

If This Output Operation Was:	Then, in This 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.

0001 **Normal Indication:** Your program has received some data on a successful input operation. It can continue to receive input until SSP-ICF returns a code of xx00 (an end of transmission indication, which allows your program to send data), or xx08 (an end of transaction indication).

Normal Action: Issue another input operation. If your program can detect something equivalent to an end of file condition, indicating that the last of the data was just received, it can issue an output operation.

0010 **Normal Indication:** A request to change direction was received from the remote program on a successful *output* operation for your program; the remote program wants to send data as soon as possible. You should allow the remote program to send its data.

Normal Action: Issue an input operation as soon as possible.

0021 **Normal Indication:** A message was received from the remote system on a successful input operation. (The message is now in your program's input buffer.) Your program can continue to receive input.

Normal Action: Handle the message in the input buffer (possibly display it), and issue another input operation. If your program can detect something equivalent to an end of file condition, indicating that the last of the data was just received, it can issue an output operation.

0031 **Normal Indication:** A truncated message was received from the remote system on a successful input operation. (The message was truncated because it was too long for your program's input buffer.) Your program can continue to receive input.

Normal Action: Handle the truncated message (possibly display it) in your program's input buffer, and issue another input operation. If your program can detect something equivalent to an end of file condition, indicating that the last of the data was just received, it can issue an output operation.

Major Code 01 – Successful operation with a new requester.

The new requester is a program on a remote system that initiated a session with your program by sending to the local system a procedure start request. 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 procedure start request, initiated by the remote program, was sent by the remote system in the form of an *EXEX procedure start statement. The request may have included, for your program, data from the remote *program* or a system message from the remote *system*.

Normal Description: The following return code indicates that the input operation issued by your program and responded to by a new requester completed successfully. Your program received some data, no data, or a system message from the requester. If any data was received, it was included in the incoming procedure start request statement. If a system message was received, the procedure that was started was the default destination procedure; the message (now in the subsystem input buffer) will be passed to your program when the next input operation is issued.

If your program is an assembler program, the length of the data or message 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, either data or a message 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**

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 can continue to receive input until SSP-ICF returns a code of xx00 (an end of transmission indication) or xx08 (an end of transaction indication).

Normal Action: Handle any data passed with the request, perform any necessary record keeping[1] for the new requester, and issue another input operation. If your program can detect something equivalent to an end of file condition, indicating that the last of the data was just received, it can issue an output operation.

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, or it received a message from the remote system. 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 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**

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 can continue to receive input until SSP-ICF returns a code of xx00 (an end of transmission indication) or xx08 (an end of transaction indication).

Normal Action: Issue another input operation. If your program can detect something equivalent to an end of file condition, indicating that the last of the data was just received, it can issue an output operation.

[1] For some situations, no record keeping for the session is necessary. In other situations, you should record the session ID of the new requester. You may also want to keep a table containing the IDs of all active requesters, or to maintain a history log of all requests.

0221 **Normal Indication:** A message was received from the remote system on a successful input operation. (The message is now in your program's input buffer.) Also, a stop system request or a disable subsystem request is pending; no new sessions using the subsystem can be initiated. Your program can continue to receive input.

Normal Action: Handle the message in your program's input buffer (possibly display it), and issue another input operation. If your program can detect something equivalent to an end of file condition, indicating that the last of the data was just received, it can issue an output operation.

0231 **Normal Indication:** A truncated message was received from the remote system on a successful input operation. (The message was truncated because it was too long for your program's input buffer.) Also, a stop system request or a disable subsystem request is pending; no new sessions using the subsystem can be initiated. Your program can continue to receive input.

Normal Action: Handle the truncated message (possibly display it) in your program's input buffer, and issue another input operation. If your program can detect something equivalent to an end of file condition, indicating that the last of the data was just received, it can issue an output operation.

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:** An end of transmission indication with *no* data was received on a successful input operation. The last record in the file has been received. Although communications have ended with the remote program, the session with the remote system is still active.

Normal Action: Issue another input operation, issue an output operation, or terminate the transaction with a put end of transaction operation.

0308 **Normal Indication:** An end of transaction indication was received *without* data on a successful input operation. Communications have ended with the remote program, and the session 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.

Major Code 04-34 – Miscellaneous program errors.

Error Description: The operation just attempted by your program failed, or an output exception occurred.

- An operation may have failed because it was issued at the wrong time or because a data record was too long.
- An output exception may have occurred because your program attempted to send output when it should be receiving the output that has already been sent by the remote program.

Recovery Action: Refer to the individual return code descriptions for the appropriate recovery actions.

Code Indication/Action

0412 **Normal (Exception) Indication:** 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 remote program. Your program's output was not sent and should be sent later, after the remote program's data (still waiting in the subsystem input buffer) has been received.

Note: If your program issues another output operation, an error return code of 831C will be received.

Normal Action: Issue an input operation to receive the data waiting in the subsystem input buffer.

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 a new requester) 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 requesting program have been completed.

3401 **Error Indication:** This input operation was rejected because the record length of the data sent by the remote 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/34 Messages Guide* 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 (in COBOL and assembler only) a set timer operation, or by using the TIMER intrinsic function (in BASIC only); (2) continue local processing; or (3) terminate.
- If the session should be started again after the subsystem is enabled, it must be reacquired by your program or restarted by the remote 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 either in this subsystem or in the interrupt handler.

Recovery Action: This subsystem has been disabled; it must be enabled again before communications can resume. Your program can continue local processing, wait[1] to reissue the acquire operation, or terminate.

If more than one subsystem was active when the error occurred, all subsystems that were active when the error occurred should be disabled. (Note that all other active CICS subsystems are automatically disabled when the error occurs; all other types of active subsystems must be manually disabled.)

If all subsystems (of all types) on the system are *not* disabled to recover from the processor check, the common queue space used by the failing subsystem *cannot* be freed. And if it is not freed, that space is wasted, and an indication of insufficient common queue space being available can occur. The indication can occur as a message when the failing subsystem is reenabled or when a different subsystem is enabled. The indication can also occur as a return code to your program for any subsystem that is starting a *new* session (code 8215) or performing an output operation in any *existing* session (code 8315).

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 remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, it can wait[1] until the subsystem has been reenabled to reissue the acquire operation, or it can terminate.

80BD **Error Indication:** Your program attempted to make a connection on an X.21 line, but the X.21 connect task was not active. The X.21 task (which was activated when the IPL process was used to start the system) has terminated abnormally for some reason. The subsystem has been disabled.

Recovery Action: Your program can continue local processing or terminate. Before your program can communicate over an X.21 line: the IPL process must be performed again to reactivate the X.21 task, the subsystem must be reenabled, and the session must be reacquired.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

Major Code 81 – Permanent (nonrecoverable) session error.

Error Description: A nonrecoverable error has occurred in the session; the session cannot be continued and 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/34 Messages Guide* 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 session must be acquired again before communications can resume.

General Recovery Actions: The following general actions can be taken for each 81xx return code. Other specific actions are given in each return code description.

- Several return codes indicate that an error condition must be corrected by changing a value in the subsystem configuration record or in the SESSION statement for your program.
  - To change a parameter value in the subsystem configuration being used by your program, disable the subsystem before making the change in the subsystem's configuration record, and enable the subsystem again to make the change effective.
  - To change a parameter value in the SESSION statement associated with your program, you must terminate your program only.

  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.

- If the session should be started again, it must be reacquired by your program or restarted by the remote program before communications can resume.

- An end of session (EOS or $$EOS) operation should be issued for the session that has terminated. Your program can also continue local processing, or it can terminate.

Note: If the session is started again, it starts from the beginning, not at the point where the session error occurred.

Code Indication/Action

8193 **Error Indication:** A disconnect indication (for switched lines only) was received on an *output* operation. A disconnect time-out in the remote system was exceeded, the line was unexpectedly disconnected, or your program may have sent some invalid data. The session has been terminated.

This return code also occurs as a *normal* indication when your program has issued an evoke then invite operation using the CSSF transaction ID to request that CICS disconnect the line.

Recovery Action: Verify that your program did not cause a time-out and that it did not send data that was invalid. Also, verify that it did not try to send data after the transaction had ended. If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

8194 **Error Indication:** A disconnect indication (for switched lines only) was received on an *input* operation. A disconnect timeout in the remote system was exceeded, or the line was unexpectedly disconnected. The session has been terminated.

Recovery Action: Verify that your program did not cause a time-out. Also, verify that it did not try to receive data after it had received an end of transaction indication. If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

81BC **Error Indication:** An attempt by this subsystem to automatically call a remote location using the autocall or X.21 feature was not successful because the wrong type of list was used to make the call. Either a list of public data network numbers was used to make the call on an autocall line, or a list of phone numbers was used to make the call on an X.21 line. The session has been terminated.

Recovery Action: Change the name specified in the *phone list name* parameter in the subsystem configuration, or in the PHONE parameter of the SESSION statement for this program. Then reissue the acquire operation to restart the session.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

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/34 Messages Guide* 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, first disable the subsystem, make the change in the subsystem's configuration record, and 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, (and depending on what the return code description says):
   a. Simply reissue the acquire operation. It could be successful if the error occurred because there was not enough common queue space available to support a new session or because an isolated line error occurred.
   b. If the acquire operation is again unsuccessful, retry it only a limited number of times. (The limit for retries should be specified in your program.)
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 operation is valid only in an *active* session and cannot, therefore, be issued after an 82xx return code is received. (This restriction does not apply to COBOL and assembler programs, or to the TIMER intrinsic function in BASIC, which also can be used to wait for a specified time interval.)

Code Indication/Action

8213 **Error Indication:** On an unsuccessful acquire operation, a queue space error condition was detected. The session could not be started because no *subsystem* queue space was available at the time.

Recovery Action: Your program can wait[1], then reissue the acquire operation. If an unacceptable number of queue space errors occurs, you can disable the subsystem and change the subsystem configuration by specifying a larger subsystem queue space size in the subsystem queue space parameter. After the subsystem is enabled, reissue the acquire operation to start the session.

8215 **Error Indication:** On an unsuccessful acquire operation, a queue space error condition was detected. The session cannot be started because the size of the *common* queue space, specified during subsystem configuration, is too small.

Recovery Action: Your program can wait[1], then reissue the acquire operation. If an unacceptable number of queue space errors occurs, you can disable all the subsystems that are active in the system, and change the subsystem configuration by specifying a larger common queue space size in the SSP-ICF common queue space parameter. After the subsystem is enabled, reissue the acquire operation to start the session.

821E **Error Indication:** The acquire operation attempted by your program (BASIC programs only) was unsuccessful because there was no SESSION statement specified between the LOAD and RUN statements for your program. The method used to issue the acquire operation is not supported by the CICS subsystem. The session was not started.

Recovery Action: Issue a SESSION statement that specifies, in the SYMID parameter, the identifier of the session to be acquired. The same identifier must be specified in the ID parameter of the OPEN statement.

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, respecify 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.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

8281 **Error Indication:** On an unsuccessful acquire operation, an SSP-ICF error condition was detected. The error caused a processor check either in this subsystem or in the interrupt handler.

Recovery Action: This subsystem has been disabled; it must be enabled again before communications can resume. Your program can continue local processing, wait[1] to reissue the acquire operation, or terminate.

If more than one subsystem was active when the error occurred, all subsystems that were active when the error occurred should be disabled. (Note that all other active CICS subsystems are automatically disabled when the error occurs; all other types of active subsystems must be manually disabled.)

If all subsystems (of all types) on the system are *not* disabled to recover from the processor check, the common queue space used by the failing subsystem *cannot* be freed. And if it is not freed, that space is wasted, and an indication of insufficient common queue space being available can occur. The indication can occur as a message when the failing subsystem is reenabled or when a different subsystem is enabled. The indication can also occur as a return code to your program for any subsystem that is starting a *new* session (code 8215) or performing an output operation in any *existing* session (code 8315).

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 remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, wait[1] to reissue the acquire operation, or terminate.

82A7 **Error Indication:** The acquire operation was unsuccessful because the specified communications line was already in use. The session was not started.

Recovery Action: Your program can wait[1] for the line to become available, then reissue the acquire operation. Otherwise, it can continue local processing or terminate.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

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 100 sessions can be active in the System/34 at one time. The session was not started.

Recovery Action: Your program can wait[1] 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 to be enabled is identified by the location parameter in the SESSION statement. That location name must also be specified in the subsystem configuration record (shown on display 3.0 of the subsystem configuration planning charts). The session was not started.

Recovery Action: Verify that the subsystem name was specified correctly on the LOCATION parameter of the SESSION statement. If the correct name was specified, contact the System/34 system operator and request that the specified subsystem be enabled by executing the ENABLE procedure command at the system console. Then reissue the acquire operation. Otherwise, your program can continue local processing, wait[1] 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[1] until the subsystem has been enabled, and 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[1] until the subsystem is enabled again, and then reissue the acquire operation. Otherwise, your program can continue local processing, or it can terminate.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

82B1 **Error Indication:** On an unsuccessful acquire operation, the session specified by the session address on the SESSION statement was identified as already being in use. (The session address is specified in the SESNADDR parameter of the SESSION statement. The address matches one of the addresses in the list of addresses that were specified during subsystem configuration.) The session was not started.

Recovery Action: (1) Your program can wait[1] for the specified session to become available and reissue the acquire operation. (2) You can remove the SESNADDR parameter value from the SESSION statement and reissue the acquire operation; the subsystem will then assign a session address from those available in the subsystem configuration. (3) Or, your program can continue local processing or terminate.

82B2 **Error Indication:** The acquire operation was not successful because all of the sessions in the address pool are already in use. (The session address was not specified in the SESNADDR parameter of the SESSION statement; any available session in the pool was to be used.) The session was not started.

Recovery Action: Wait[1] for one of the sessions in the pool to become available, then reissue the acquire operation. Otherwise, continue local processing or terminate.

82B4 **Error Indication:** The acquire operation was not successful because all of the resources needed for the session could not be allocated from the assign/free area of the system. All available resources are already being used in the system. The session was not started.

Recovery Action: Wait[1] for the needed resources to become available, then reissue the acquire operation. Otherwise, continue local processing or terminate.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

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/34 Messages Guide* 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, first disable the subsystem, make the change in the subsystem's configuration record, and 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.

   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.

   Note: If the session is started again, it starts from the beginning, not at the point where the session error occurred.

3. If no change is needed in your program or in the subsystem, (and depending on what the return code description says):

   a. Notify the remote location that a change is required on that end to correct the error received.

   b. Retry the operation, if possible. It could be successful if the error occurred because there was not enough common queue space available at the time, because an isolated line error occurred, or because the remote system was not active temporarily.

   c. If another operation is not successful, retry it only a limited number of times. (The limit for retries should be specified in your program.)

   d. Issue a set timer operation in your program so it can wait for a specified time interval before reissuing the operation.

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 an 81xx (session was terminated) or 82xx (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 an 81xx or 82xx return code is handled properly. If you want your program to recover from an improperly handled error condition, issue another acquire operation.

8315 **Error Indication:** On an evoke operation, a queue space error condition was detected. The evoke operation could not be performed because no *common* queue space was available at the time.

Recovery Action: Your program can issue a set timer operation and wait for a period of time, and then reissue the evoke operation. If an unacceptable number of queue space errors occur, you can disable all the subsystems and change the subsystem configuration by specifying a larger common queue space size in the SSP-ICF common queue space parameter. After the subsystem is enabled, reissue the acquire operation to restart the session.

831C **Error Indication:** The output operation issued before this output operation received a return code of 0412 (indicating that the remote program sent 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 data.

Recovery Action: Issue an input operation to receive the 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. 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 before attempting to communicate with the remote 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: If you want your program to recover dynamically, reissue the output operation with a smaller output length. Otherwise, you can either change the record length in your program and recompile it, or you can change the value specified for the maximum user record length parameter in the subsystem configuration. 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.

8329 **Error Indication:** An invalid evoke operation was detected in this session. Your program was evoked by an incoming procedure start request and cannot, therefore, issue any evoke operations in this session.

Recovery Action: If you want your program to dynamically 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.

832B **Error Indication:** On the first output operation, a record length of 0 was detected by the subsystem. The output operation was not performed.

Recovery Action: If a coding error in your program caused the error, correct your program. If the data record is in error, correct it. Then issue another output operation.

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.

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.

8336 **Error Indication:** On the first *output* operation requiring that a switched line connection be established for the session, an invalid remote identifier for the remote system was received from the remote system; the session is still active. The received remote identifier must match the remote identifier specified for this subsystem configuration.

Recovery Action: Verify that the remote identifier specified for this subsystem configuration was specified correctly by the remote ID parameter in the CNFIGICF procedure. If the remote identifier was specified correctly, call the remote location to correct the error at the remote end of the switched line.

8383 **Error Indication:** An MLCA (multiline communications adapter) controller check occurred on an *output* operation; data may have been lost. The session, however, is still active.

Recovery Action: Issue another output operation to send the same data again. If the same error indication continues to occur, issue an end of session operation to end the session. Then your program can continue local processing, or it can terminate.

8384 **Error Indication:** An MLCA (multiline communications adapter) controller check occurred on an *input* operation; data may have been lost. The session, however, is still active.

Recovery Action: Issue another input operation to receive the same data again. If the same error indication continues to occur, issue an end of session operation to end the session. Then if your program started the session, reissue the acquire operation and try again. If your program was evoked, it can wait to be evoked again, continue local processing, or terminate.

8385 **Error Indication:** An attempt by this subsystem to automatically call one or more remote locations using the autocall or X.21 feature was not successful. All available numbers in the list of phone numbers or in the list of public data network numbers were called, but no connection was established. The session, however, is still active.

Recovery Action: Your program can reissue the output operation to automatically call all the numbers in the list again. The list is reinitialized when this error occurs, so that the first number in the list is called first when the output operation is reissued. If this return code continues to be returned to your program while it attempts to complete a call, it should issue a release or end of session operation, and then continue local processing or terminate.

8386 **Error Indication:** An attempt by this subsystem to automatically call one or more remote locations using the autocall or X.21 feature was not successful. All numbers in the list of phone numbers or in the list of public data network numbers were already marked as called. The message *PHONE LIST EXHAUSTED* (SYS-8607) has been displayed on the system console. The session, however, is still active.

Recovery Action: Your program can reissue the output operation to automatically call all the numbers in the list again. The list is reinitialized when this error occurs, so that the first number in the list is called first when the output operation is reissued. If this return code continues to be returned to your program while it attempts to complete a call, it should issue a release or end of session operation, and then continue local processing or terminate.

8391 **Error Indication:** A permanent line error occurred on an unsuccessful *output* operation, but the system operator did not respond to the resulting error message. (You can find out what type of line error occurred by asking the system operator.) This return code will continue to occur if the operator does not respond. The session is still active.

Recovery Action: The system operator should respond to the message to resolve the error indication. Your program can issue a set timer operation to wait while continuing to perform local processing, then it can reissue the output operation to try sending the data again; or it can terminate. (The output operation should be retried only a limited number of times.)

8392 **Error Indication:** A permanent line error occurred on an unsuccessful *input* operation, but the system operator did not respond to the resulting error message. (You can find out what type of line error occurred by asking the system operator.) This return code will continue to occur if the operator does not respond. The session is still active.

Recovery Action: The system operator should respond to the message to resolve the error indication. Your program can issue a set timer operation to wait while continuing to perform local processing, then it can reissue the input operation to try receiving the data again; or it can terminate. (The input operation should be retried only a limited number of times.)

8397 **Error Indication:** An abort transmission sequence was received from the remote system on an *output* operation; the remote system is terminating the line transmission abnormally because it could not or did not want to continue communicating with your program. The session is still active.

Recovery Action: If your program started the session, issue another evoke operation to start a different transaction. If your program was evoked, it can continue local processing, or it can terminate.

8398 **Error Indication:** An abort transmission sequence was received from the remote system on an *input* operation; the remote system is terminating the line transmission abnormally because it could not or did not want to continue communicating with your program. The session is still active.

Recovery Action: Issue an end of session operation to end the session. Then if your program started the session, it can issue another acquire operation to start a different session, it can continue local processing, or it can terminate. If your program was evoked, it can wait to be evoked again (MRT programs only), it can continue local processing, or it can terminate.

8399 **Error Indication:** On an *output* operation, an indication was received that a delay count was exceeded; a condition at your location (external to your program) delayed sending the output from your program. The session is still active.

Recovery Action: Issue an end of session operation to end the session. Then if your program started the session, it can reissue the acquire operation to restart the session and try sending the data again, it can continue local processing, or it can terminate. If your program was evoked, it can wait to be evoked again (MRT programs only), it can continue local processing, or it can terminate.

839A **Error Indication:** On an *input* operation, an indication was received that a delay count was exceeded; a condition at your location (external to your program) delayed receiving the output for your program. The session is still active.

Recovery Action: Issue an end of session operation to end the session. Then if your program started the session, it can reissue the acquire operation to restart the session and try receiving the data again, it can continue local processing, or it can terminate. If your program was evoked, it can wait to be evoked again (MRT programs only), it can continue local processing, or it can terminate.

839B **Error Indication:** On an *output* operation, in a put-versus-put situation, the subsystem detected a block size error in the data it received from the remote system before it detected that both your program and the remote system were attempting to send data at the same time. The remote system sent data, but the length of the data block exceeded the length of the subsystem's line buffer. The session is still active.

Recovery Action: Two error conditions have occurred. You should check your program logic before continuing, and check that the maximum user record length is correct in the subsystem configuration record. If the parameter is correct, notify the remote system programmer and verify that the record length is correct. Then, if your program started the session, reissue the output operation; if your program was evoked, issue an input operation.

839C **Error Indication:** On an *input* operation, the length of the data block sent by the remote system exceeded the length of the subsystem input buffer. The session is still active.

Recovery Action: Check that the maximum user record length is correct in the subsystem configuration record. If the parameter is not correct, issue an end of session operation and terminate your program, then specify the correct record length in the subsystem configuration. If the parameter is correct, notify the remote system programmer and verify that the record length is correct.

Chapter 11. The BSC IMS/IRSS Subsystem

The BSC IMS/IRSS subsystem provides distributed data processing support to users of the System/34 SSP in conjunction with a System/370 using IMS/VS with IRSS. (For the remainder of this chapter, the IMS/IRSS subsystem is simply referred to in text as the IMS subsystem, and the remote System/370 using IMS/VS with IRSS is referred to as the IMS/VS system.) The IMS subsystem provides an interactive interface between System/34 application programs and programs on an IMS/VS system with IRSS. The IMS subsystem can support multiple application programs concurrently communicating with IMS/VS.

The IMS subsystem allows System/34 application programs to initiate tasks on the IMS/VS system. System/34 procedures can be initiated from tasks on the IMS/VS system. System/34 security options as well as IMS/VS security options are supported.

The IMS subsystem communicates with IMS/VS over a multipoint line only. The IMS subsystem allows up to 16 concurrent sessions between the System/34 and IMS/VS over a single multipoint line. Fifteen of these sessions can be initiated by the System/34, and one is reserved for incoming procedure start requests. Each session is viewed by IMS/VS as a physical terminal and, as such, must have an associated PTERM (physical terminal) address. These addresses must be defined at the IMS/VS host and at the System/34.

The interface between IMS/VS and the System/34 IMS subsystem consists of the transmission of data (DA) blocks and synchronization (SY) blocks. Data blocks are used to send and receive data. Synchronization blocks are used between IMS/VS and the IMS subsystem to inform each other of the status of the session, completion of output, restart points, and shutdown initiation. The formatting of these blocks is controlled by the IMS subsystem and is, therefore, transparent to the application program.

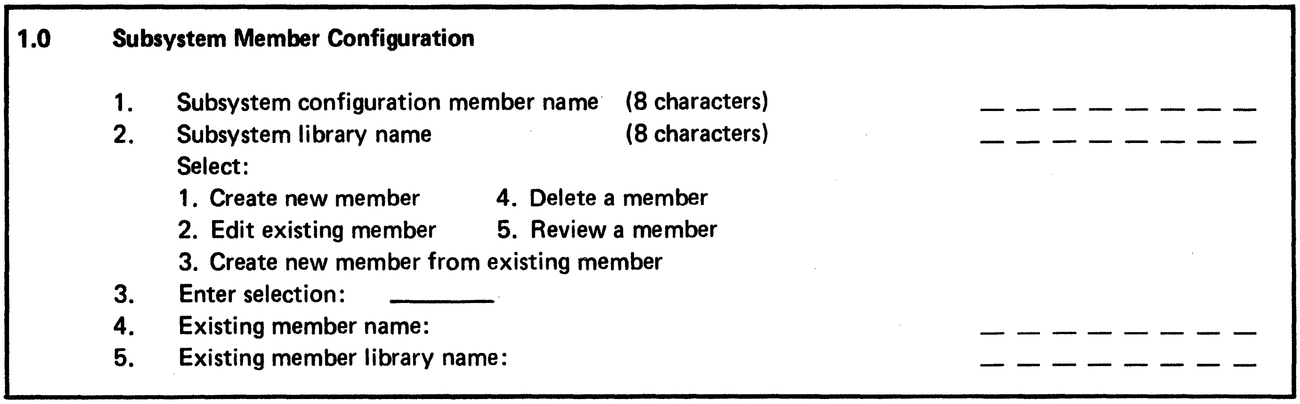
Communication with IMS/VS is in the form of messages. A message is a logical unit of data, for example a response to an inquiry. If the message is longer than the record length being transmitted or is transmitted by multiple output operations, the message is divided into segments. A data block from IMS/VS can contain more than one segment. The multiple segments can contain a single message (chained together and passed to the application program) or multiple messages (deblocked and passed one at a time to the application).

SETTING UP THE IMS/IRSS SUBSYSTEM

The SSP procedures CNFIGSSP and INSTALL are used to include the interactive communications feature and IMS subsystem on the System/34. The general interactive communications and line control support is included when it is requested on the appropriate CNFIGSSP prompt. The IMS subsystem support is copied to the system library when the appropriate responses to the INSTALL procedure prompts are taken. The CNFIGSSP and INSTALL procedures, with their displays and related responses, are described in the *Installation and Modification Reference Manual.*

After the IMS subsystem has been installed, the CNFIGICF procedure is used to tailor the subsystem support to an existing or proposed network. The operation of the CNFIGICF procedure is also contained in the *Installation and Modification Reference Manual.* Before running the CNFIGICF procedure, however, you should fill out a planning chart for each subsystem that you want to define. Copies of the planning chart for each subsystem are available in Appendix F of this manual and in the *Installation and Modification Reference Manual.* The following sections explain how to fill out the planning chart for the IMS subsystem.

Display 1.0 Subsystem Member Configuration



```
1.0     Subsystem Member Configuration

        1.   Subsystem configuration member name  (8 characters)        _ _ _ _ _ _ _ _
        2.   Subsystem library name               (8 characters)        _ _ _ _ _ _ _ _
             Select:
             1. Create new member        4. Delete a member
             2. Edit existing member     5. Review a member
             3. Create new member from existing member
        3.   Enter selection:  ______
        4.   Existing member name:                                      _ _ _ _ _ _ _ _
        5.   Existing member library name:                              _ _ _ _ _ _ _ _
```

Subsystem configuration member name: Specify a name for this configuration of the subsystem. This name is used to store the member in a library, and is referenced in the ENABLE and DISABLE procedures.

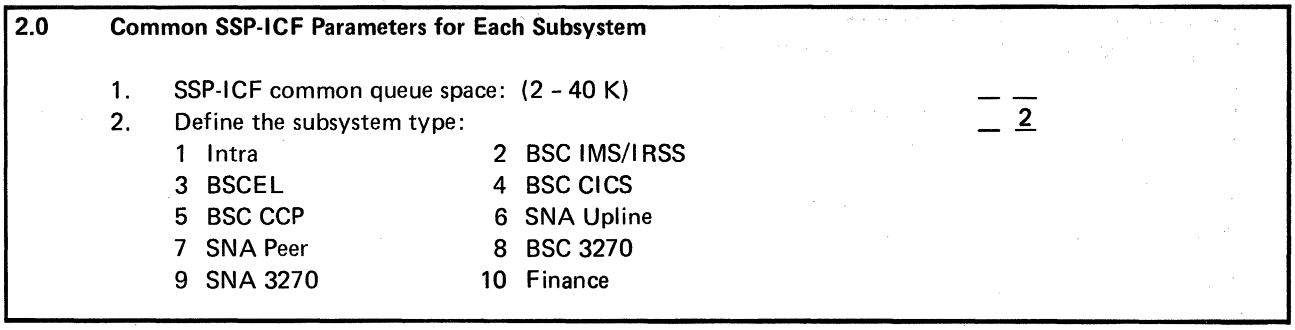
Library name: Specify the name of the library in which the configuration is stored or to be stored. The default is #LIBRARY, however, you should probably store the configuration in a user library.

Enter selection: Specify one of the five options: (1) create a new member, (2) edit an existing member, (3) create a new member from an existing member, (4) delete a member, or (5) review a member without changing it.

Existing member name: This prompt appears if option 3 was selected. Specify the name of the existing subsystem configuration member that is to be used to create the new member. The existing member remains unchanged.

Existing member library name: This prompt appears if option 3 was selected. Specify the name of the library where the existing member resides.

Display 2.0 Common SSP-ICF Parameters for Each Subsystem



```
2.0     Common SSP-ICF Parameters for Each Subsystem

        1.  SSP-ICF common queue space:  (2 - 40 K)                        _ _
        2.  Define the subsystem type:                                     _ 2
            1  Intra                   2  BSC IMS/IRSS
            3  BSCEL                   4  BSC CICS
            5  BSC CCP                 6  SNA Upline
            7  SNA Peer                8  BSC 3270
            9  SNA 3270               10  Finance
```

SSP-ICF common queue space: Specify the size, in multiples of 2 K bytes, of the common queue space. The common queue space requirements for each configuration of the IMS subsystem enabled are:

C = 50P + 250

where:
C = number of bytes required for common queue space
P = number of PTERMs defined

The common queue space value specified for the first subsystem that is enabled becomes the size of the common queue space. Be sure that the value specified for common queue space size takes into account the requirements of any other subsystem that may be active concurrently.

The size of the common queue space plus the total subsystem queue space of all the enabled IMS subsystems cannot exceed 42 K bytes.

The default common queue space size is 4 K bytes.

Define the subsystem type: Specify a 2 for the IMS subsystem.

Display 3.0 General Subsystem Parameters

```
3.0     General Subsystem Parameters

        1.  Location name:                      (8 characters)      _ _ _ _ _ _ _ _
        2.  Subsystem queue space:                   (2-40 K)                   _ _
        3.  Subsystem support swappable:        (0-No   1-Yes)                    _
```

Location name: Specify up to 8 characters for the name of the location associated with this configuration. The location name is used in some of the displayed message texts, and must be coded on the SESSION OCL statement. The default is the subsystem configuration member name. The location name refers to the name of the location with which communications is to take place.

Subsystem queue space: Specify the size, in multiples of 2 K bytes, of the subsystem queue space. The subsystem queue space requirements for each configuration of the IMS subsystem enabled are:

$$S = 27.5P + 1.1\ N + 1595$$

where:

S = number of bytes required for the subsystem queue space
P = number of PTERMs defined
N = maximum total record lengths of all sessions that will be active concurrently

The size of the common queue space plus the total subsystem queue space of all the enabled IMS subsystems cannot exceed 42 K bytes.

The default subsystem queue space size is 4 K bytes.

Subsystem support swappable: Specify whether you want the subsystem to be swappable. Consider the total system performance, the size of the subsystem, and the amount of user main storage when determining whether you want the subsystem to be swappable. The IMS subsystem requires 14 K bytes of main storage.

Display 5.0 BSC General Subsystem Parameters I

5.0 BSC General Subsystem Parameters I

2. Local station address: (2 hex) _ _

Local station address: Specify in hexadecimal the System/34 multipoint station address identified for this configuration. You can specify B (C2) through R (D9). Be sure to enter the hexadecimal equivalent in the code used. This address must correspond to the ADDR parameter of the STATION macro in IMS/VS generation.

Display 7.0 Subsystem Inactive Destination Messages

7.0 Subsystem Inactive Destination Messages

1. Subsystem procedure name: (8 characters) _ _ _ _ _ _ _ _
2. Subsystem procedure library name: (8 characters) _ _ _ _ _ _ _ _

Subsystem procedure name: Specify the name of a user-written procedure to be started when data or messages are received for an inactive session (PTERM). This can happen when the System/34 application program abnormally teminates before all data has been sent by the host system. When uninvited data is received, the procedure is run to receive the data. The program within this procedure cannot send data on the session. The program can save the data in a file, process the data as it is received, or discard the data.

Subsystem library name: Specify the name of the library containing the inactive destination messages procedure.

Display 12.0 BSC IMS/IRSS Subsystem PTERMs

12.0 BSC IMS/IRSS Subsystem PTERMs

1. Subsystem remote program start PTERM: _ _ _ _
2. Subsystem local PTERMs:

_ _ _ _   _ _ _ _   _ _ _ _   _ _ _ _   _ _ _ _
_ _ _ _   _ _ _ _   _ _ _ _   _ _ _ _   _ _ _ _
_ _ _ _   _ _ _ _   _ _ _ _   _ _ _ _   _ _ _ _

Subsystem remote program start PTERM: Specify the physical terminal (PTERM) address associated with the incoming procedure requests. The address must be four hexadecimal characters but cannot be 0000. If not specified, no PTERM is defined for incoming procedure requests, and incoming procedure requests are not allowed. This address must correspond to the ADDR parameter of the TERMINAL macro in IMS/VS generation.

Subsystem local PTERMs: Specify the physical terminal (PTERM) addresses associated with acquired sessions. The addresses are then placed in a pool. A specific PTERM from this pool can be requested on the SESSION OCL statement, or the subsystem can assign one from the pool. A maximum of 15 PTERM addresses can be defined. Each address must be four hexadecimal characters but cannot be 0000. Each address must correspond to an ADDR parameter on a TERMINAL macro in IMS/VS generation.

SETTING UP THE IMS/VS SYSTEM

The System/34 must be defined during IMS/VS generation via macros. Following is a list of macros and the required keywords to support the System/34 IMS subsystem. Only the macros and keywords that pertain directly to the System/34 IMS subsystem are shown. For more information see the *IMS/VS Version 1 Installation Guide*.

- LINEGRP UNITYPE=S/3

- LINE BUFSIZE=(x,y)
  where *x* is the input buffer size and *y* is the output buffer size. The input buffer size should be a minimum of 268 bytes, and the output buffer size should be a maximum of 268 bytes.

- STATION TYPE=(NOASK,POSTPONE)
  The TRNSLM keyword can also be specified if desired.

- TERMINAL
  Each PTERM address specified during configuration of the System/34 must also be defined to IMS/VS via the TERMINAL macro.

- MSGDEL Keyword of the TERMINAL Macro
  The SYSINFO option should be specified for the program start PTERM. The NONIOPCB option is recommended for the other PTERMs. The NONIOPCB option prevents the following types of messages from being sent to System/34 application programs:
  - Message switches
  - Messages inserted by an application program to an alternate PCB
  - /BROADCAST messages
  - DFS059 TERMINAL status messages

  If the application requires sending any of these types of messages, the SYSINFO option can be specified. SYSINFO causes IMS/VS to discard terminal status messages for this terminal.

The following is a sample of the IMS/VS macros for defining sessions with the System/34 IMS subsystem:

```
LINEGRP DDNAME=DDSYS34A,UNITYPE=S/3
 LINE ADDR=032,BUFSIZE=(268,268)
  STATION ADDR=C2,TYPE=(NOASK,POSTPONE),NAME=SYS34A
   TERMINAL ADDR=F0F1,MSGDEL=SYSINFO
    NAME=SYS34A1
   TERMINAL ADDR=F0F2,MSGDEL=NONIOPCB
    NAME=SYS34A2
   TERMINAL ADDR=F0F3,MSGDEL=NONE
    NAME=SYS34A3
   TERMINAL ADDR=F0F4,MSGDEL=SYSINFO
    NAME=SYS34A4
```

The local station address specified on display 5.0 of the CNFIGICF procedure.

The system PTERMs specified on display 12.0 of the CNFIGICF procedure.

STARTING AND ENDING THE IMS SUBSYSTEM

To start the IMS subsystem, the ENABLE procedure must be run. The format of the ENABLE procedure command is in Chapter 2. During enable, the IMS subsystem allocates the required storage for all control blocks and buffers. A receive operation is started, and the subsystem waits for IMS/VS to send a start SY block.

To terminate the IMS subsystem, the DISABLE procedure must be run. The format of the DISABLE procedure command is in Chapter 2. The following actions occur on a held disable:

- The IMS subsystem requests that, as soon as the message in progress is complete, IMS/VS not send additional output for the procedure start PTERM.
- When all active sessions have terminated, the subsystem requests IMS/VS to shut down the PTERM.
- When all the messages that have been received by the System/34 are acknowledged to IMS/VS, IMS/VS shuts down the PTERM, and communications is terminated.

After communications is terminated, regardless of the type of disable, all buffers and control blocks are freed before the subsystem terminates. If the subsystem was active on a multipoint line using the multiline communications adapter, the adapter will continue to respond negatively if polled or selected after the disable has been completed.

STARTING IMS SUBSYSTEM APPLICATIONS

System/34 IMS subsystem applications can be started by a display station operator entering a procedure command or by a request from the remote system. Procedures that are started by a System/34 operator must have a SESSION statement for each session to be started. Requests from the remote system to start a procedure must be in a special format. The following sections describe the SESSION statement and the incoming procedure start requests.

SESSION OCL Statement

The format of the SESSION OCL statement for the IMS subsystem is:

```
// SESSION LOCATION-name ,SYMID-session-id

    [,BATCH- {YES}]
             {NO }

    [,MAXMSG-nnnn]  [,PTERM-xxxx]
```

LOCATION: Specifies the location name associated with this session. The location name is defined during subsystem configuration, and refers to the name of the location with which communication is to take place.

SYMID: Specifies the symbolic ID of the session with which this SESSION statement is associated. The symbolic ID must be two characters, with the first character numeric (0 through 9) and the second character alphabetic (A through Z, #, $, or @). This is the same ID that the application program uses when referring to this session. This ID is the equivalent of the symbolic display station ID as specified on the WORKSTN OCL statement. This parameter has no default.

BATCH: Specifies whether this session will be used for batch activity. YES indicates that batch activity will occur, and results in the line being reserved for use exclusively by this session when the session is acquired. NO indicates that batch activity will not occur, and is the default. For more information on batch activity, see the *Put Operations* and *Input Operations* descriptions later in this chapter.

MAXMSG: Specifies the maximum total length of a single message that will be transmitted by evoke and put operations. This length can be any decimal number from 1 to 4096. This parameter must be specified if BATCH-NO is specified or defaulted.

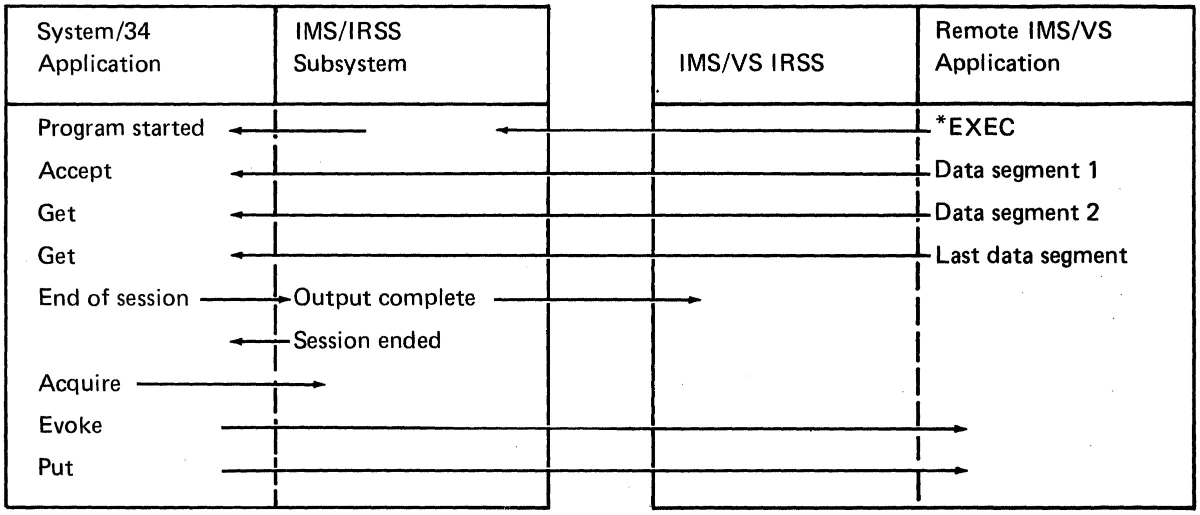
PTERM: Specifies the physical terminal ID to be used for this session. The ID must have been configured during subsystem configuration. The value can be any hexadecimal number from 0001 through FFFF. If you do not specify this parameter, the subsystem selects an address from the list defined during subsystem configuration.

Incoming Procedure Requests

Sessions can be initiated by receiving an *EXEC or *EXEX procedure start request from IMS/VS on the PTERM address specified for this during subsystem configuration. The format of these requests is in Chapter 2. The procedure start request must be contained in one nonspanned segment that consists of no more than 147 bytes. If the procedure cannot be started, an SY block is sent to IMS/VS along with the PTERM address and a system message.

Because only one message can be sent with a procedure request, the session is available for another procedure request as soon as the subsystem sends an output complete SY block. If a communicating program (*EXEC) is started on the System/34 and is given all the message segments, the System/34 program must receive end of file for the message before the subsystem sends an output complete SY block. The output complete SY block is usually sent when the subsystem receives notification that the System/34 application program is terminating normally. If the application program wishes to indicate that it has received end of file for the message but still has a large amount of processing to do, it can issue an end of session operation. When the subsystem sees this operation, the session is logically disconnected from the subsystem and the output complete SY block is sent to IMS/VS.

No output operations can be issued on the session started by the *EXEC request. A System/34 application program that has been started by an *EXEC must acquire a new session and evoke a transaction if data is to be sent to the remote application.



If a noncommunicating program (*EXEX) is started on the System/34, the output complete SY block is sent as soon as the subsystem is assured that the program is scheduled. The session is then available for another procedure start request.

OPERATION CONSIDERATIONS

The following sections describe the operations supported by the IMS subsystem. A complete chart of all interactive communications operations and the subsystems that support them is in each language chapter. The chart also shows the keyword or format name used to code the operation. More information about how an operation is coded is also described in the appropriate programming language chapter.

Whether an operation completes successfully or unsuccessfully, a return code is given to the application program. All of the return codes that are valid for this subsystem are described at the end of this chapter. A summary chart in Appendix A lists all the return codes and the subsystems for which they are valid.

Acquire Operation

The acquire operation is issued to start a new session. If no specific PTERM was requested with the acquire, the subsystem finds an available PTERM and assigns it to this session. If a specific PTERM was requested on the SESSION OCL statement and that PTERM is available, the subsystem assigns it to this session. If a PTERM cannot be assigned or if new sessions are not allowed, a return code that reflects the situation is issued.

Evoke Operations

The evoke operation ($$EVOK, $$EVOKNI, $$EVOKET) is given to the subsystem with a parameter list. This list contains pertinent information for the subsystem; the only parameters allowed by IMS/VS are transaction ID (procedure name) and password. The coding of the evoke parameter list is described in the appropriate programming language chapter.

If an evoke operation without an invite, get, or end of transaction ($$EVOKNI) is followed by a put operation without an invite, get, or end of transaction ($$SENDNI), all put operations that follow the evoke have their segments chained together with the evoke data to form a single message. Otherwise, the evoke data is assumed to be a complete message.

If BATCH-YES was specified on the SESSION OCL statement, the evoke is transmitted as soon as it is issued. If an evoke with data is issued for a BATCH-YES session, the transaction ID, password, and data are sent in one segment. The total length of this segment cannot exceed 256 bytes. IMS/VS combines the segments to form a single message.

If BATCH-NO was specified, the transaction ID, password, and evoke data can be longer than 256 bytes. If the request is longer than 256 bytes, the request is broken into segments before transmission to IMS/VS. IMS/VS combines the segments to form a single message.

Put Operations

When the IMS subsystem receives a put request ($$SENDNI), the application program records are chained together. The chain is considered complete when the application program issues a put end of transaction ($$SENDET), a put then invite ($$SEND), a put then get (assembler only), or an invite operation. The complete message chain is then transmitted by the IMS subsystem to IMS/VS.

If BATCH-YES was specified on the SESSION OCL statement for this session, the records are not accumulated in storage. They are transmitted as each put request is received. Each put request for BATCH-YES cannot exceed 256 bytes.

All nonbatch output requests that are greater than 256 bytes are broken into 256-byte segments before transmission to IMS/VS.

If the maximum output message length specified on the SESSION statement is exceeded before end of message is indicated (by an end of transaction operation or by an input operation), all previous segments of the message are discarded, and the application program receives an error return code (8317).

The put end of transaction operation ($$SENDET) causes the subsystem to turn on the end of message flag. The end of message flag is turned on in this segment if data was given with this request. If no data came with this operation, the end of message flag is set in the preceding segment. If the put end of transaction operation is the first put issued and has no data with it, an invalid operation return code (832B) is given to the application program.

Input Operations

The input operations for the IMS subsystem are invite, get, and accept. The invite operation can be issued only as a combined operation with a put or evoke operation ($$SEND, $$EVOK) in BASIC, COBOL, and RPG II. Assembler language users can issue an invite operation explicitly. Either a get or invite operation signals the subsystem to obtain data on the session for the application program. A get operation causes the application program to wait for data to be available. When a program issues an invite operation, it receives the data with the next accept operation. The accept operation allows input from any previously invited session.

One segment, as received from IMS/VS, is passed to the application program for each input request. If the IMS/VS application program sends records longer than 256 bytes, IMS/VS breaks up the record into 256-byte pieces and sends each piece as part of a spanned segment. The IMS subsystem puts these pieces together to form a complete segment before passing the record to the System/34 application program. If an application program terminates when it has not completely received a message, a system message is displayed on the system console, and an error block (SY10) is sent to IMS/VS.

For each successful input operation (return codes of 00xx through 03xx), the minor part of the return code should be checked for end of transaction (xxx8). The following minor return codes indicate that there is no more data available for this transaction:

Code	**Meaning**
xx08	End of transaction
0118	End of transaction with new requester
xx28	End of transaction with system message in buffer
xx38	End of transaction with truncated system message in buffer

Note: Another input operation after receiving one of these minor return codes could result in an endless wait condition unless additional messages are expected for this session.

Release Operation

The release operation is a request to terminate a session. The release operation is rejected if an input or output message is in progress.

When a System/34 application program is receiving data, the program should not release the session until an end of the transaction code is received. If the program releases the session before the end of transaction, the data is left on the IMS/VS queue. If data is left on an IMS/VS queue, the next program that acquires the associated PTERM receives the data.

End of Session Operation

The end of session operation ($$EOS) always results in a normal completion return code. The session is always terminated by the end of session operation. If the session is still communicating when the end of session operation is issued, the transmission is abnormally terminated by the IMS subsystem, and abnormal termination of the IMS/VS application program could result.

Get Attributes Operation

The get attributes operation (assembler only) can be issued at any time to determine the status of a session.

Set Timer Operation

The set timer operation ($$TIMER) results in a timer expired return code (0310) after a specific time interval in hours, minutes, and seconds has expired.

PROGRAMMING CONSIDERATIONS

IMS/VS System Messages

If an IMS/VS message is received as data for a PTERM that is not assigned to an active session, the message is treated as uninvited data.

When an IMS/VS message is received as data for an active session, the message is given to the application program on the next input operation. If the next operation is not an input operation, the operation is rejected with a return code (0412) indicating that an input operation is required.

Message Switches

Because the IMS subsystem cannot tell the difference between data generated via an IMS/VS message switch and any other data, the message switch is given to the application program as normal data. If the PTERM is not associated with an active session, the message switch is considered uninvited data.

IMS/VS Commands

Any commands sent to IMS/VS via the IMS subsystem must be properly formatted by the application program before being given to the subsystem. The subsystem sends the command as normal data without checking the format. IMS/VS requires that a command have a slash (/) as its first nonblank character and that no other segment can have a slash as its first nonblank character. The /CAN (cancel) command, however, can be sent within a message. Caution should be used when sending IMS/VS commands. Some commands can result in messages that exceed the program buffer sizes.

Data Blocks

A data block contains one or more segments belonging to one or more messages. A segment is fully transmitted by IMS/VS in one transmission, unless its size exceeds the user-specified buffer size, in which case it is changed into a spanned segment, which is transmitted in two or more transmissions.

A block identifier is included with each block for restart purposes. When an input message is enqueued, IMS/VS logs the block identifier with the message. IMS/VS transmits the last logged block identifier back to the System/34 after a restart of IMS/VS. The IMS subsystem can also request this information to be transmitted, thus allowing resynchronization after a previous start.

Data blocks can contain one or more message segments, which can be associated with the same or different PTERMs. IMS/VS sends a complete message for one PTERM before it begins a message for another PTERM.

Data blocks sent to IMS/VS contain one message segment. The System/34 sends a complete message for one PTERM before it begins a message for another PTERM. Therefore, an application program that transmits large messages (batch type) should be scheduled at a time when session activity is at a minimum, because when the IMS subsystem starts transmitting a message, no other messages are started until an end of message is sent for the message in progress. While nonbatch sessions are active, the subsystem does not start a session defined as a BATCH-YES on the SESSION OCL statement. If an attempt is made to acquire a batch session while nonbatch sessions are running, a return code (82AC) is given to the application program indicating that the acquire failed. If a nonbatch session attempts an acquire while a batch session is running, a return code (82AD) is given to the application program indicating that the acquire failed.

Error Blocks Received by System/34

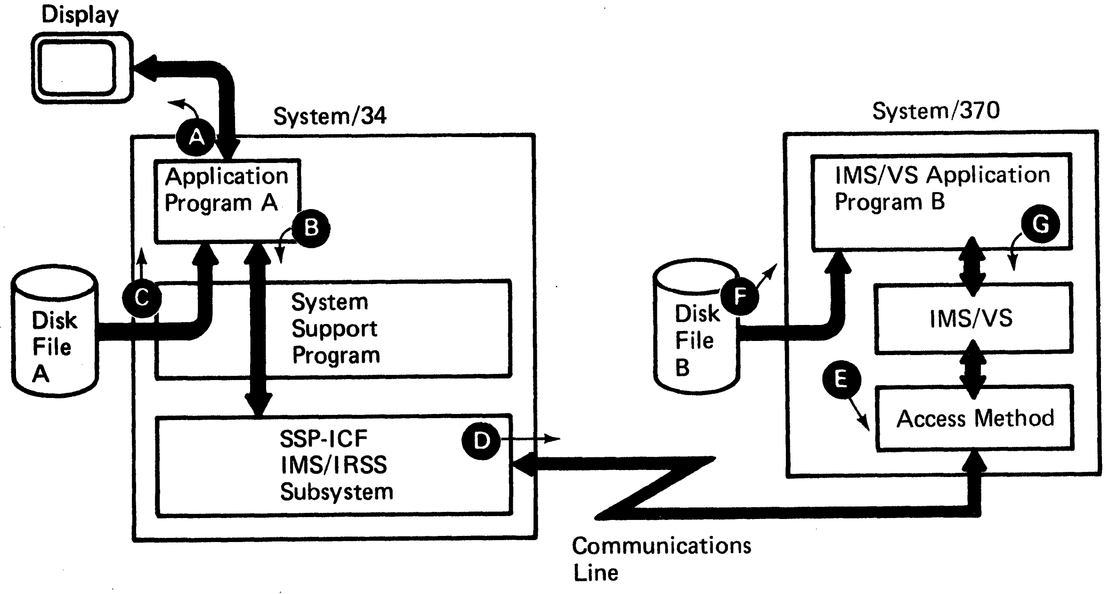
When the IMS subsystem receives an asynchronous (not directed to a specific PTERM) error block (SY1002), the message is displayed on the system console. The message also indicates that this is an asynchronous error. Because this is an asynchronous error, the message is displayed with no halt. The System/34 console operator probably can contact the IMS/VS master terminal operator for more information.

When a synchronous (directed to a specific PTERM) error message is received in an SY block and an input operation is pending for this session, the message is put in the application program data buffer, and a return code (0028) indicating that a system message is in the buffer is given to the application program. If a put operation is pending for this session, a return code (0411) is given to the application program indicating that the output operation has not been performed because a system message is available. At this point, the only accepted operation is an input request for the system message.

Any output message in progress to IMS/VS for this session is canceled.

How to Write Programs that Use the IMS/IRSS Subsystem

The following inquiry application is used in the programming examples:



1. Application program A (in the System/34) displays a prompt asking an operator to enter an item number requesting the stock status for the item Ⓐ.

2. When the operator enters the item number, program A reads the number and searches file A (the local file) for the item Ⓒ.

3. If the item is found in the local file, program A displays the stock status on the screen Ⓐ.

4. If the item is *not* in the local file, program A uses the IMS/IRSS subsystem to send the item number to the host system Ⓑ and Ⓓ.

5. Program B (in the remote system) uses the item number to search the remote file for the item Ⓕ.

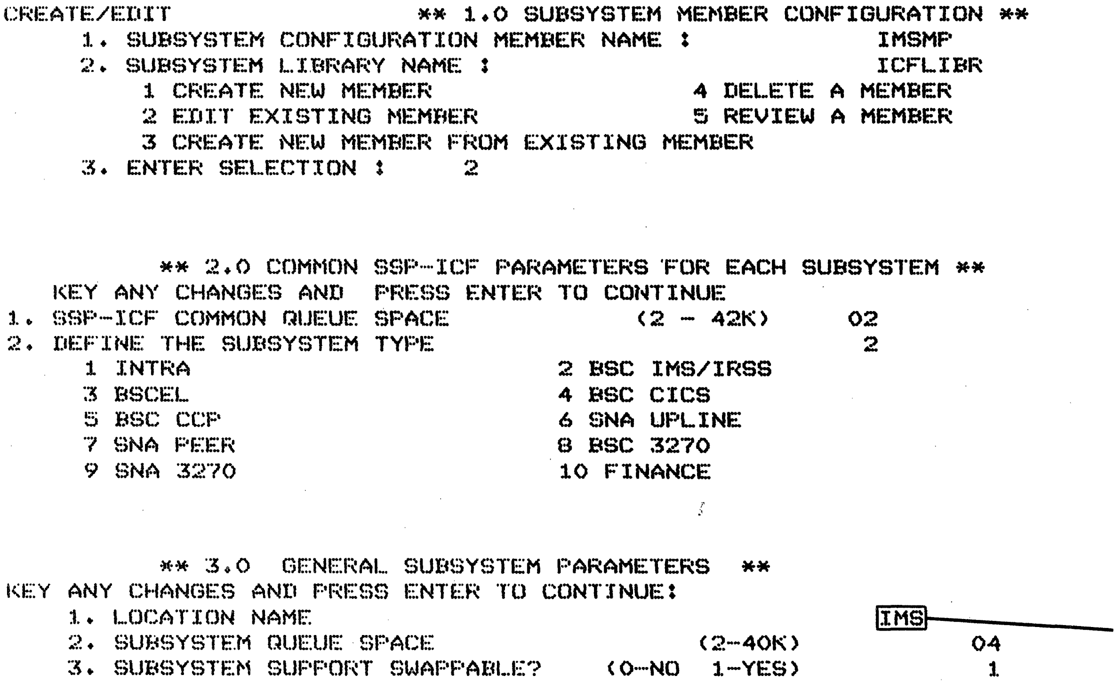
6. If the item is in the remote file, program B sends the stock status to program A Ⓖ and Ⓔ. If the item is not in the remote file, program B sends the characters ***.

7. If program A receives the stock status, it displays it. If it receives the characters ***, it displays the message ITEM NOT FOUND Ⓐ.

Except for a few minor changes, program A in this example is the program described in Chapter 7 for the Intra subsystem. The changes to the program required for IMS are shown in this chapter following the configuration and OCL examples. If you have not read the description of program A in Chapter 7, see *How to Write Programs that Use the Intra Subsystem* in Chapter 7. The configuration and OCL examples in Chapter 7 are for the Intra subsystem only; you do not need to read those. Following are the configuration parameters and OCL statements for this subsystem.

Configuration Parameters

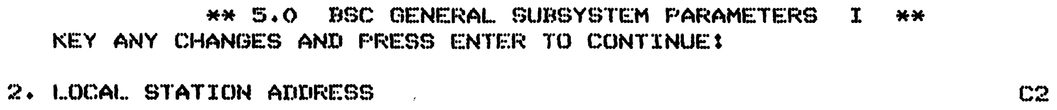
The following configuration parameters are used for this example. For a description of the configuration parameters, see *Setting up the IMS/IRSS Subsystem* at the beginning of this chapter.



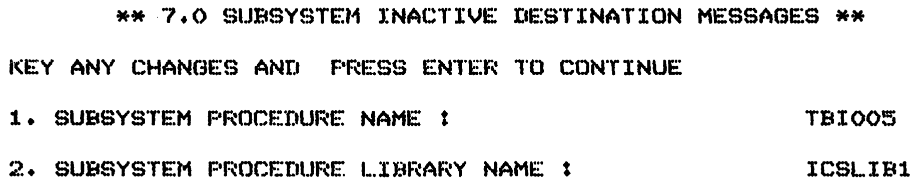
```
CREATE/EDIT                    ** 1.0 SUBSYSTEM MEMBER CONFIGURATION **
      1. SUBSYSTEM CONFIGURATION MEMBER NAME :              IMSMP
      2. SUBSYSTEM LIBRARY NAME :                           ICFLIBR
         1 CREATE NEW MEMBER                      4 DELETE A MEMBER
         2 EDIT EXISTING MEMBER                   5 REVIEW A MEMBER
         3 CREATE NEW MEMBER FROM EXISTING MEMBER
      3. ENTER SELECTION :      2


          ** 2.0 COMMON SSP-ICF PARAMETERS FOR EACH SUBSYSTEM **
   KEY ANY CHANGES AND  PRESS ENTER TO CONTINUE
1. SSP-ICF COMMON QUEUE SPACE                    (2 - 42K)      02
2. DEFINE THE SUBSYSTEM TYPE                                     2
      1 INTRA                           2 BSC IMS/IRSS
      3 BSCEL                           4 BSC CICS
      5 BSC CCP                         6 SNA UPLINE
      7 SNA PEER                        8 BSC 3270
      9 SNA 3270                        10 FINANCE


          ** 3.0  GENERAL SUBSYSTEM PARAMETERS  **
KEY ANY CHANGES AND PRESS ENTER TO CONTINUE:
     1. LOCATION NAME                                          IMS
     2. SUBSYSTEM QUEUE SPACE                    (2-40K)          04
     3. SUBSYSTEM SUPPORT SWAPPABLE?    (0-NO  1-YES)             1
```

The location name is also specified on the SESSION statement.

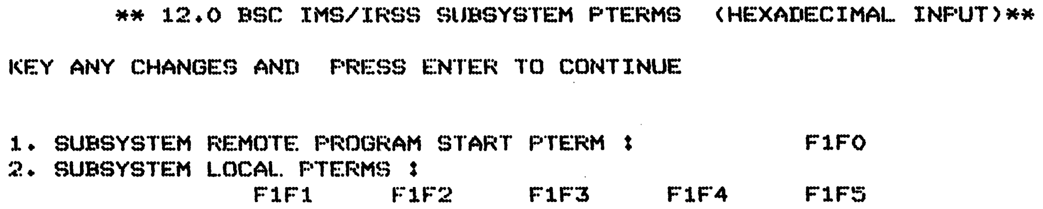
Configuration Parameters (continued)



```
                ** 5.0  BSC GENERAL SUBSYSTEM PARAMETERS  I  **
   KEY ANY CHANGES AND PRESS ENTER TO CONTINUE:

2. LOCAL STATION ADDRESS                                             C2
```



```
        ** 7.0 SUBSYSTEM INACTIVE DESTINATION MESSAGES **

KEY ANY CHANGES AND  PRESS ENTER TO CONTINUE

1. SUBSYSTEM PROCEDURE NAME :                        TBI005

2. SUBSYSTEM PROCEDURE LIBRARY NAME :                ICSLIB1
```



```
        ** 12.0 BSC IMS/IRSS SUBSYSTEM PTERMS  (HEXADECIMAL INPUT)**

KEY ANY CHANGES AND  PRESS ENTER TO CONTINUE


1. SUBSYSTEM REMOTE PROGRAM START PTERM :            F1F0
2. SUBSYSTEM LOCAL PTERMS :
             F1F1     F1F2     F1F3     F1F4     F1F5
```

OCL Statements

The following OCL statements are used for the BASIC example:

```
BASICR IMSBAS,ITEMBAS,30,,IMSSESS

// SESSION LOCATION-IMS,SYMID-1S,MAXMSG-512
```

The BASICR procedure includes the SESSION statement.

The location name (IMS) is also specified on display 3.0 of the CNFIGICF procedure.

The following OCL statements are used for the COBOL example:

```
// LOAD COBIMS
// FILE NAME-FILEA
// SESSION LOCATION-IMS,SYMID-1S,MAXMSG-266,BATCH-YES
// RUN
```

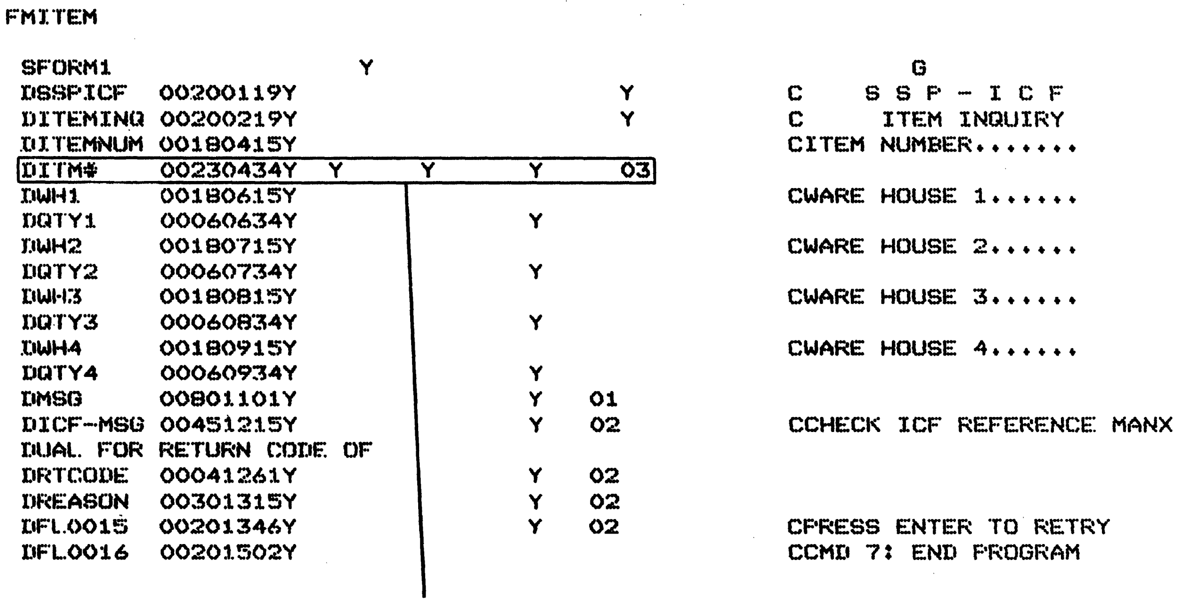
The location name (IMS) is also specified on display 3.0 of the CNFIGICF procedure.

The following OCL statements are used for the RPG II example:

```
// LOAD RPGIMS
// FILE NAME-FILEA
// SESSION LOCATION-IMS,SYMID-1S,MAXMSG-512
// RUN
```

The location name (IMS) is also specified on display 3.0 of the CNFIGICF procedure.

Changes for the Screen Format



```
FMITEM

 SFORM1                Y                                   G
 DSSPICF  00200119Y                     Y          C   S S P - I C F
 DITEMINQ 00200219Y                     Y          C     ITEM INQUIRY
 DITEMNUM 00180415Y                                CITEM NUMBER.......
 DITM#    00230434Y  Y     Y      Y     03
 DWH1     00180615Y                                CWARE HOUSE 1......
 DQTY1    00060634Y               Y
 DWH2     00180715Y                                CWARE HOUSE 2......
 DQTY2    00060734Y               Y
 DWH3     00180815Y                                CWARE HOUSE 3......
 DQTY3    00060834Y               Y
 DWH4     00180915Y                                CWARE HOUSE 4......
 DQTY4    00060934Y               Y
 DMSG     00801101Y               Y   01
 DICF-MSG 00451215Y               Y   02           CCHECK ICF REFERENCE MANX
 DUAL. FOR RETURN CODE OF
 DRTCODE  00041261Y               Y   02
 DREASON  00301315Y               Y   02
 DFL0015  00201346Y               Y   02           CPRESS ENTER TO RETRY
 DFL0016  00201502Y                                CCMD 7: END PROGRAM
```

The item number parameters have been changed. For the example in Chapter 7, the item number sent to program B was right-adjusted. However, the item number sent to CICS must be left-adjusted.

Changes for the Programming Example

The following examples show the changes required for Program A, described in Chapter 7, to permit communications with IMS.

Changes for BASIC

See *Changes for the Screen Format* earlier in this example.

Evoke the program MSCGT005 at the host system. The password, user ID, and library name were not required for this example.

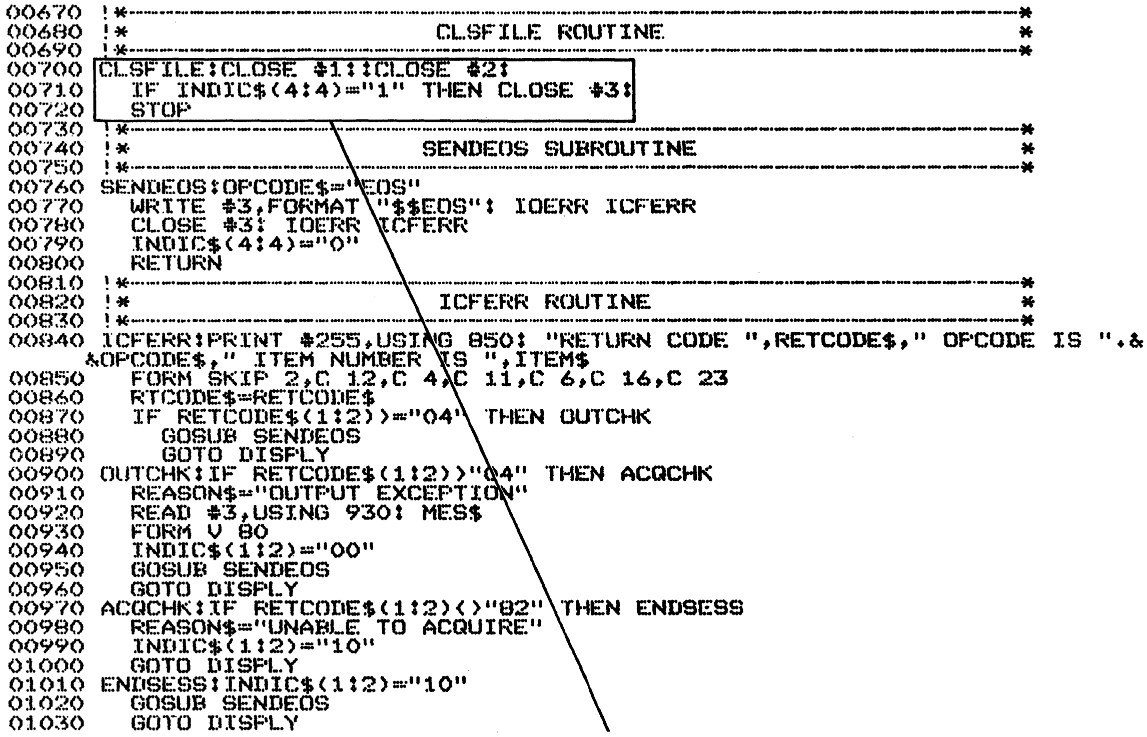
```
00010 !*****************************************************************
00020 !*                                                               *
00030 !*             IMSBAS - ITEM INQUIRY WRITTEN IN BASIC            *
00040 !*                                                               *
00050 !*****************************************************************
00060 DIM ITEM$*23,ITEMNO$*23,MES$*80,REASON$*30
00070 OPEN #1: "WS,NAME=FMITEM,RECL=161,LIBR=ICFLIBR"
00080 OPEN #2: "NAME=FILEA,SHR,RECL=50,KEYL=23,KEYP=1,RANDOM",KEYED,INPUT
00090 INDIC$(1:2)="11"
00100 !*--------------------------------------------------------------*
00110 !*           DISPLAY SCREEN REQUESTING ITEM NUMBER.             *
00120 !*                   IF CMD 7, CLOSE ALL FILES                  *
00130 !*--------------------------------------------------------------*
00140 DISPLY:WRITE #1,USING 150,FORMAT "FORM1",INDIC INDIC$: ITEM$,QTY1$,&
     &QTY2$,QTY3$,QTY4$,MES$,RTCODE$,REASON$
00150   FORM C 23,4*C 6,C 80,C 4,C 30
00160   MES$="":RTCODE$="":REASON$=""
00170   READ #1,USING 180: ITEM$ CONV 220
00180   FORM C 23
00190   INDIC$(1:3)="110"
00200   IF CMDKEY=7 THEN CLSFILE
00210   IF ITEM$<>RPT$(" ",23) THEN GOTO READFILE
00220     MES$="INVALID ITEM NUMBER ENTERED"
00230     INDIC$(1:1)="0":INDIC$(3:3)="1"
00240     GOTO DISPLY
00250 !*--------------------------------------------------------------*
00260 !*   READ LOCAL FILE 'FILEA' FOR REQUESTED ITEM NUMBER.IF        *
00270 !*   ITEM IS FOUND LOCALLY, DISPLAY ITEM INFORMATION.IF         *
00280 !*   ITEM IS NOT FOUND LOCALLY, SEND ITEM NUMBER TO             *
00290 !*   'MSCGT005' USING ICF.                                      *
00300 !*--------------------------------------------------------------*
00310 READFILE:READ #2,USING 320,KEY=ITEM$: QTY1$,QTY2$,QTY3$,QTY4$ NOKEY ICF
00320   FORM X 26,4*C 6
00330   GOTO DISPLY
00340 !*--------------------------------------------------------------*
00350 !*                   ACQUIRE ICF-SESSION (1S)                   *
00360 !*--------------------------------------------------------------*
00370 ICF:IF INDIC$(4:4)="1" THEN EVOK
00380   OPCODE$="ACQ"
00390   OPEN #3: "WS,ID=1S,RECL=512" IOERR ICFERR
00400   INDIC$(4:4)="1"
00410 !*--------------------------------------------------------------*
00420 !*                  EVOKE PROCEDURE 'MSCGT005'                  *
00430 !*--------------------------------------------------------------*
00440 EVOK:OPCODE$="EVOK"
00450   WRITE #3,USING 460,FORMAT "$$EVOK": "MSCGT005",ITEM$ IOERR ICFERR
00460   FORM C 8,X 24,C 23
00470 !*--------------------------------------------------------------*
00480 !*                   GET INPUT FROM 'MSCGT005'                  *
00490 !*--------------------------------------------------------------*
00500 OPCODE$="GET"
00510 READ #3,USING 520: DATA$,ITEMNO$,QTY1$,QTY2$,QTY3$,QTY4$ IOERR ICFERR
00520 FORM C 3,X 29,C 23,X 145,4*C 6
00530 READ #3: IOERR ICFERR
00540 READ #3: IOERR ICFERR,EOF 600
00550 !*--------------------------------------------------------------*
00560 !*   IF ITEM NUMBER IS NOT FOUND, DISPLAY MESSAGE 'ITEM          *
00570 !*   NOT FOUND' TO THE SCREEN. IF THE ITEM IS FOUND,            *
00580 !*   DISPLAY THE INVENTORY INFORMATION.                         *
00590 !*--------------------------------------------------------------*
00600 INDIC$(1:2)="11"
00610 IF DATA$="***" THEN NOITEM
00620   ITEM$=ITEMNO$
00630   GOTO DISPLY
00640 NOITEM:MES$="ITEM NUMBER "&ITEM$&" NOT FOUND"
00650   INDIC$(1:1)="0":INDIC$(3:3)="1"
00660   GOTO DISPLY
```

This read operation checks for end of file condition.

The second record does not contain usable data.

IMS sends two records; therefore, two read operations are required to get the data. The first record contains the stock status or indicates that the item was not found.

Changes for BASIC (continued)



```
00670 !*---------------------------------------------------------------------*
00680 !*                        CLSFILE ROUTINE                              *
00690 !*---------------------------------------------------------------------*
00700 CLSFILE:CLOSE #1::CLOSE #2:
00710    IF INDIC$(4:4)="1" THEN CLOSE #3:
00720    STOP
00730 !*---------------------------------------------------------------------*
00740 !*                       SENDEOS SUBROUTINE                            *
00750 !*---------------------------------------------------------------------*
00760 SENDEOS:OPCODE$="EOS"
00770    WRITE #3,FORMAT "$$EOS": IOERR ICFERR
00780    CLOSE #3: IOERR ICFERR
00790    INDIC$(4:4)="0"
00800    RETURN
00810 !*---------------------------------------------------------------------*
00820 !*                         ICFERR ROUTINE                              *
00830 !*---------------------------------------------------------------------*
00840 ICFERR:PRINT #255,USING 850: "RETURN CODE ",RETCODE$," OPCODE IS ",&
     &OPCODE$," ITEM NUMBER IS ",ITEM$
00850    FORM SKIP 2,C 12,C 4,C 11,C 6,C 16,C 23
00860    RTCODE$=RETCODE$
00870    IF RETCODE$(1:2)>="04" THEN OUTCHK
00880       GOSUB SENDEOS
00890       GOTO DISPLY
00900 OUTCHK:IF RETCODE$(1:2)>"04" THEN ACQCHK
00910    REASON$="OUTPUT EXCEPTION"
00920    READ #3,USING 930: MES$
00930    FORM V 80
00940    INDIC$(1:2)="00"
00950    GOSUB SENDEOS
00960    GOTO DISPLY
00970 ACQCHK:IF RETCODE$(1:2)<>"82" THEN ENDSESS
00980    REASON$="UNABLE TO ACQUIRE"
00990    INDIC$(1:2)="10"
01000    GOTO DISPLY
01010 ENDSESS:INDIC$(1:2)="10"
01020    GOSUB SENDEOS
01030    GOTO DISPLY
```

This program does not send an end of transaction. The host system ends the transaction.

Changes for COBOL

```
SELECT TRANSACTION-FILE
    ASSIGN TO WORKSTATION-FMITEM-01,
    ORGANIZATION IS TRANSACTION,
    FILE STATUS IS WS-FS, ICF-FS,
    CONTROL-AREA IS WS-CONTROL-AREA.
```

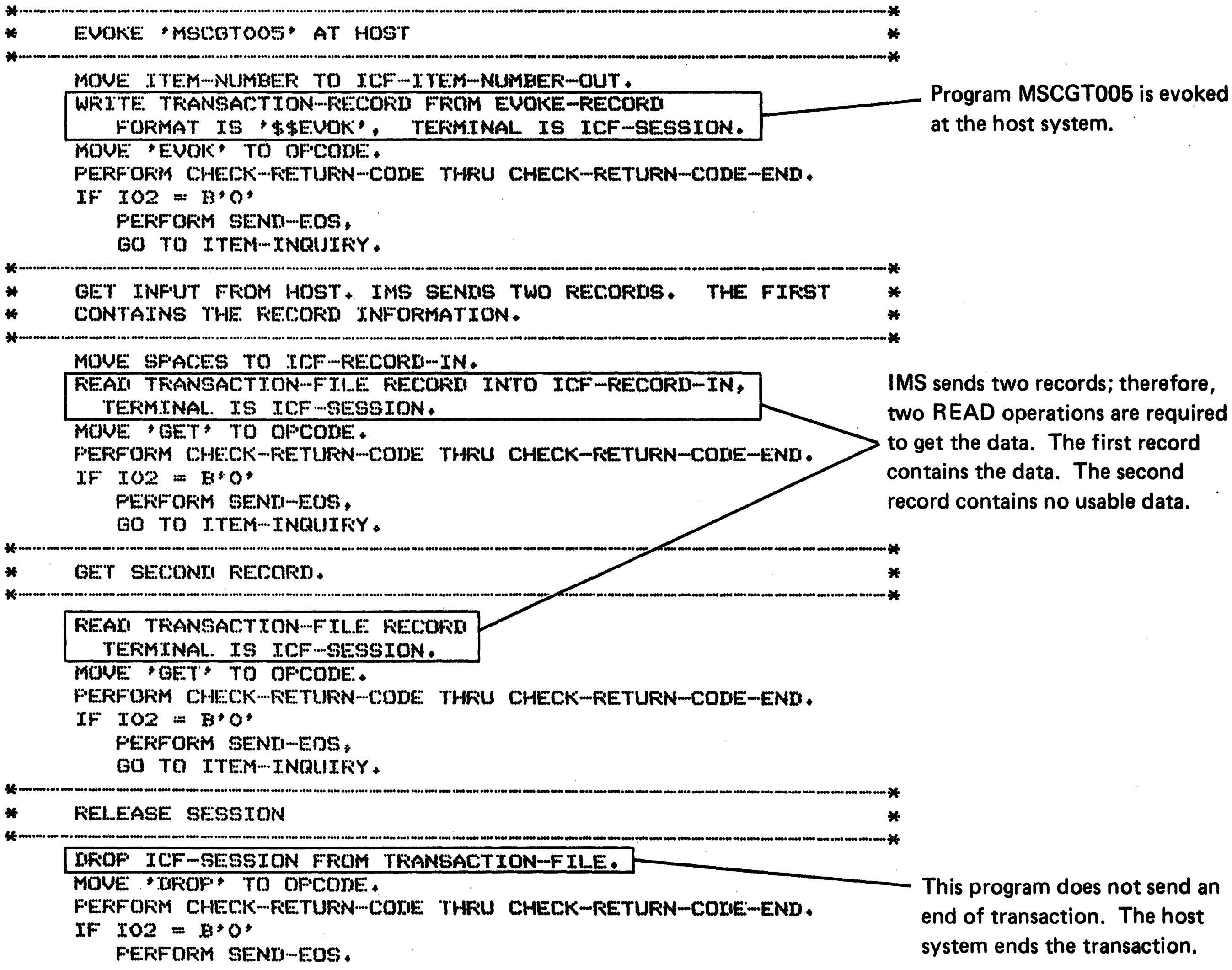
See *Changes for the Screen Format* earlier in this example.

```
DATA DIVISION.
FILE SECTION.
FD  TRANSACTION-FILE, LABEL RECORDS ARE OMITTED.
01  TRANSACTION-RECORD               PIC X(512).


01  EVOKE-RECORD.
    03  PROCEDURE-NAME               PIC X(8) VALUE 'MSCOT005'.
    03  PASSWORD                     PIC X(8).
    03  USER-ID                      PIC X(8).
    03  LIBRARY-NAME                 PIC X(8).
    03  FILLER                       PIC X(20).
    03  DATA-LENGTH                  PIC XXXX VALUE '0023'.
    03  ICF-ITEM-NUMBER-OUT          PIC X(23).

01  ICF-RECORD-IN.
    03  ICF-RECORD-CHECK.
      05  FIRST-3-CHARACTERS         PIC X(3).
      05  REST-OF-DATA               PIC X(509).

    03  ICF-RECORD-OK REDEFINES ICF-RECORD-CHECK.
      05  FILLER                     PIC X(32).
      05  ICF-ITEM-NUMBER-IN         PIC X(23).
      05  FILLER                     PIC X(145).
      05  ICF-QTY-1                  PIC 9(6).
      05  ICF-QTY-2                  PIC 9(6).
      05  ICF-QTY-3                  PIC 9(6).
      05  ICF-QTY-4                  PIC 9(6).
      05  FILLER                     PIC X(288).
```

The record length was increased because the record from IMS was longer than 256 bytes.

These parameters were not required for this example.

Changes for COBOL (continued)



```
*-----------------------------------------------------------------*
*     EVOKE 'MSCGT005' AT HOST                                    *
*-----------------------------------------------------------------*
      MOVE ITEM-NUMBER TO ICF-ITEM-NUMBER-OUT.
      WRITE TRANSACTION-RECORD FROM EVOKE-RECORD
         FORMAT IS '$$EVOK',  TERMINAL IS ICF-SESSION.
      MOVE 'EVOK' TO OPCODE.
      PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
      IF IO2 = B'O'
         PERFORM SEND-EOS,
         GO TO ITEM-INQUIRY.
*-----------------------------------------------------------------*
*     GET INPUT FROM HOST. IMS SENDS TWO RECORDS.  THE FIRST      *
*     CONTAINS THE RECORD INFORMATION.                            *
*-----------------------------------------------------------------*
      MOVE SPACES TO ICF-RECORD-IN.
      READ TRANSACTION-FILE RECORD INTO ICF-RECORD-IN,
        TERMINAL IS ICF-SESSION.
      MOVE 'GET' TO OPCODE.
      PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
      IF IO2 = B'O'
         PERFORM SEND-EOS,
         GO TO ITEM-INQUIRY.
*-----------------------------------------------------------------*
*     GET SECOND RECORD.                                          *
*-----------------------------------------------------------------*
      READ TRANSACTION-FILE RECORD
        TERMINAL IS ICF-SESSION.
      MOVE 'GET' TO OPCODE.
      PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
      IF IO2 = B'O'
         PERFORM SEND-EOS,
         GO TO ITEM-INQUIRY.
*-----------------------------------------------------------------*
*     RELEASE SESSION                                             *
*-----------------------------------------------------------------*
      DROP ICF-SESSION FROM TRANSACTION-FILE.
      MOVE 'DROP' TO OPCODE.
      PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
      IF IO2 = B'O'
         PERFORM SEND-EOS.
```

Program MSCGT005 is evoked at the host system.

IMS sends two records; therefore, two READ operations are required to get the data. The first record contains the data. The second record contains no usable data.

This program does not send an end of transaction. The host system ends the transaction.

Changes for RPG II

See *Changes for the Screen Format* earlier in this example.

```
FWSFILE  CD         512                WORKSTN
F                                                          KNUM       2
F                                                          KFMTS  FMITEM
F                                                          KID    ID
F                                                          KINFDS INFDS
F                                                          KINFSR INFSR
FFILEA   IC  F  50  50R23AI        1 DISK

O        E          05N07
O                                         K6 '$$EVOK'
O                                          8 'MSCGT005'
O                                         56 '0023'
O                              ITM#       79
```

The program MSCGT005 is evoked at the host system. The password, user ID, and library name are not required for this example.

Changes for RPG II (continued)

```
C*----------------------------------------------------------------------*
C*      EVOKE 'MSCGT005' AT HOST                                        *
C*----------------------------------------------------------------------*
C                   MOVE '1S'     ID
C                   SETON                    05
C                   EXCPT
C                   SETOF                    05
C*
C*----------------------------------------------------------------------*
C*      GET INFORMATION SENT BY 'MSCGT005'. IMS SENDS TWO               *
C*      RECORDS.  THE FIRST CONTAINS THE ITEM INFORMATION.              *
C*----------------------------------------------------------------------*
C          '1S'     NEXT WSFILE
C                   READ WSFILE
C                   MOVELITM#     SAVITM 23
C                   Z-ADDWSQTY1   QTY1
C                   Z-ADDWSQTY2   QTY2
C                   Z-ADDWSQTY3   QTY3
C                   Z-ADDWSQTY4   QTY4
C*
C*----------------------------------------------------------------------*
C*      GET SECOND RECORD                                               *
C*----------------------------------------------------------------------*
C          '1S'     NEXT WSFILE
C                   READ WSFILE
C                   MOVELSAVITM   ITM#
C*
C*----------------------------------------------------------------------*
C*      RELEASE ICF SESSION '1S' AND GO TO ITMINQ.                      *
C*----------------------------------------------------------------------*
C          '1S'     REL  WSFILE
C   10              SETOF                    01
C   10              SETON                    03
C                   GOTO ITMINQ
```

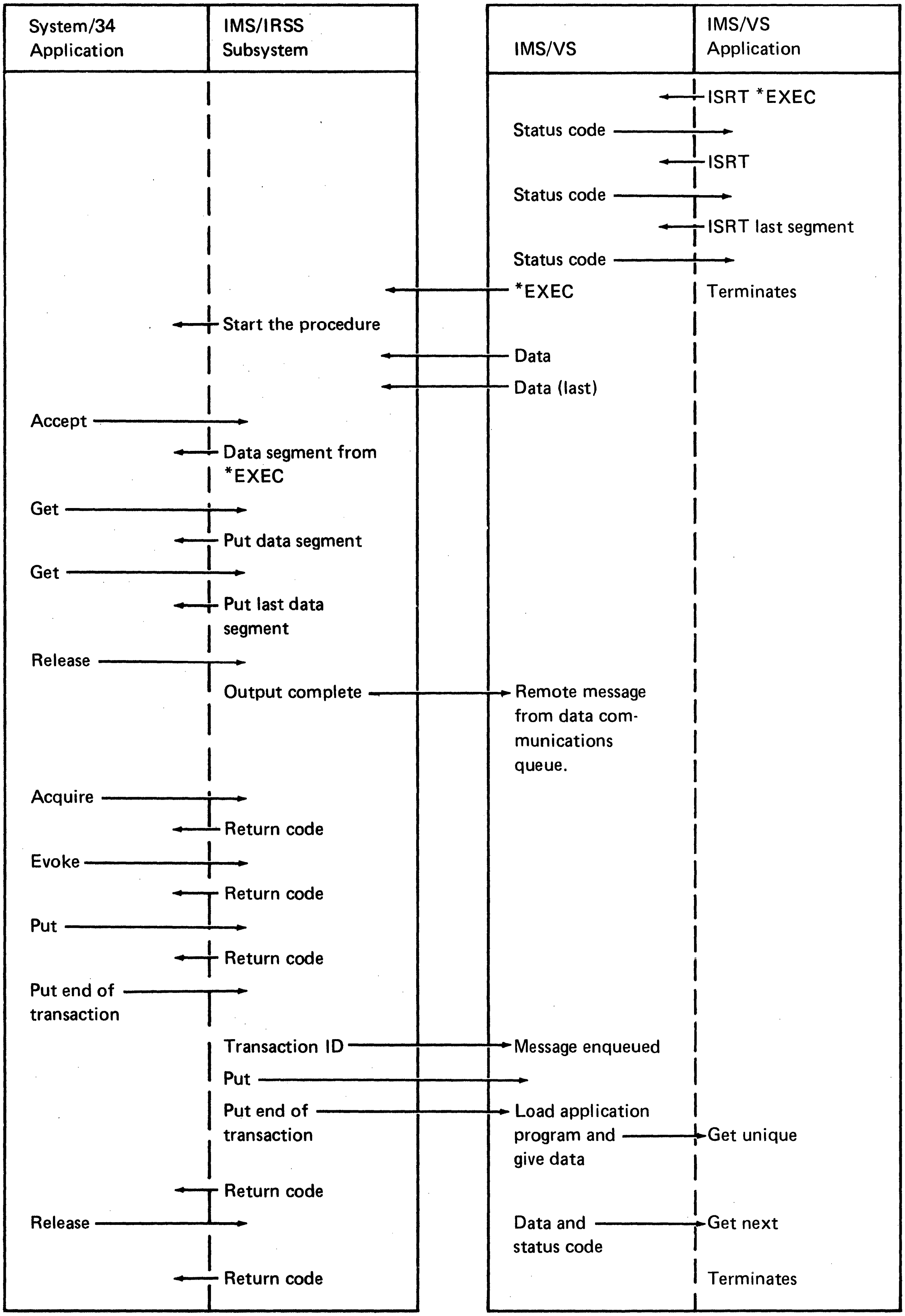
Input is from MSCGT005 at the host system.

IMS sends two records; therefore, two READ operations are required to get the data. The first record contains the data. The second record contains no usable data.

This program does not send an end of transaction ($$SENDET); therefore, indicator 06 is not used. The host system ends the transaction.

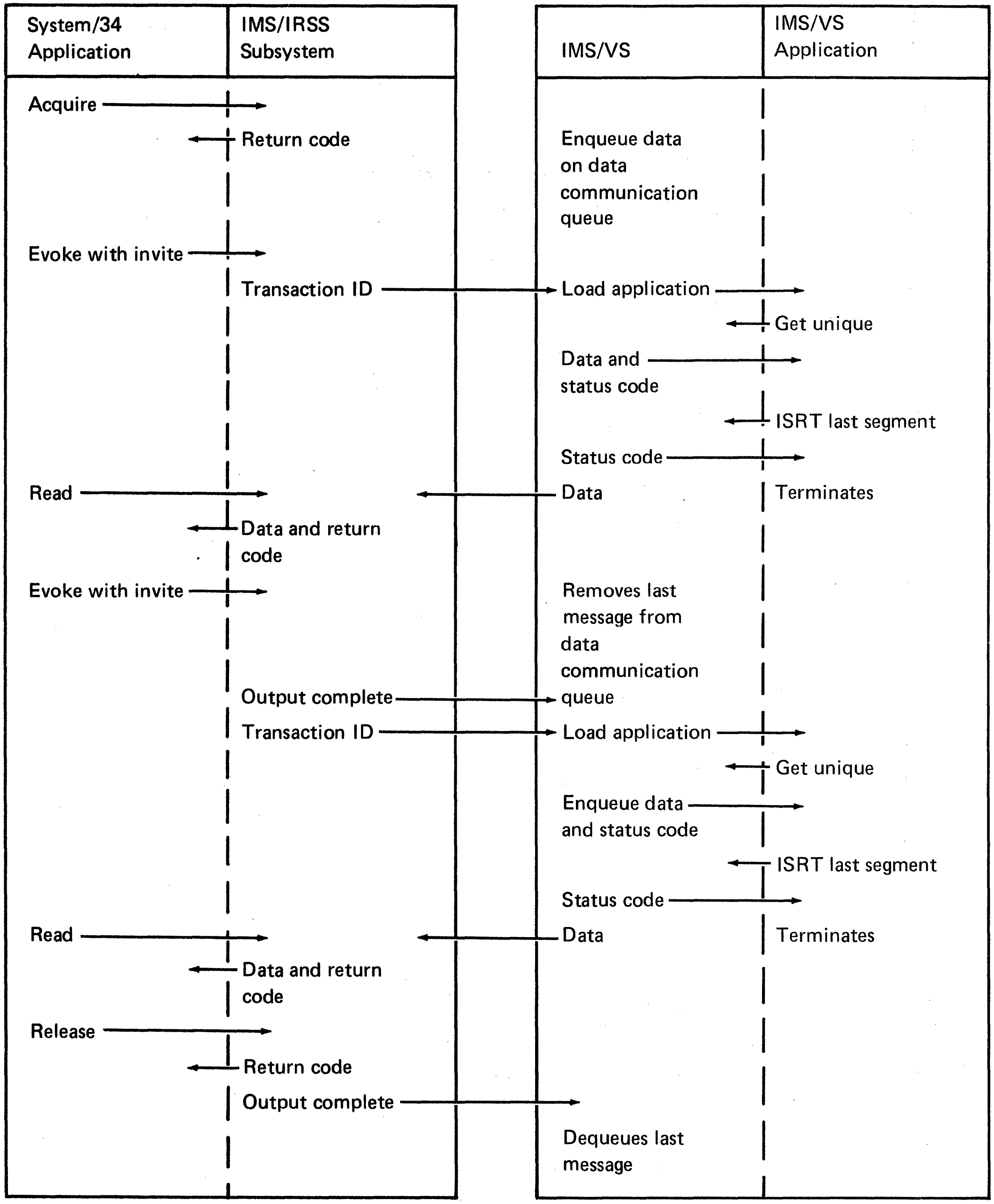
Remote Procedure Start Request Example

The following example shows an IMS/VS program starting a System/34 procedure, and the System/34 program attaching to the IMS/VS application program that originally performed the procedure request.


System/34 Application
IMS/IRSS Subsystem
IMS/VS
IMS/VS Application
ISRT *EXEC
Status code
ISRT
Status code
ISRT last segment
Status code
*EXEC
Terminates
Start the procedure
Data
Data (last)
Accept
Data segment from *EXEC
Get
Put data segment
Get
Put last data segment
Release
Output complete
Remote message from data communications queue.
Acquire
Return code
Evoke
Return code
Put
Return code
Put end of transaction
Transaction ID
Message enqueued
Put
Put end of transaction
Load application program and give data
Get unique
Return code
Release
Data and status code
Get next
Return code
Terminates

Inquiry Application Example

The following example shows an inquiry program on the System/34 starting an inquiry program on the IMS/VS system.



IMS/IRSS Subsystem Return Codes

This part of Chapter 11 describes all the return codes that are valid for the IMS 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 four-digit value; the first two digits contain the major code, and the last two 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).

Note: In the return code descriptions, *your program* refers to the local System/34 application program that initiates the operation and receives the return code from the subsystem. The *remote program* refers to the application program in the remote (or host) system with which the System/34 application program is communicating through SSP-ICF.

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 IMS subsystem. Although all the operations shown are valid for IMS, their validity also depends on the logical sequence of communications events occurring between the System/34 and the remote system.

Input Operations to Your Program	Output Operations from Your Program	Combined Operations in Your Program
Accept input		
	Acquire[1]	
	End of session	
	Evoke Evoke end of transaction	Evoke then get[2] Evoke then invite
Get Get attributes[3]		
Invite		
	Put Put end of transaction	Put then get[2] Put then invite
	Release	
	Set timer[4]	

[1] Normally, 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.

[2] Valid only in assembler language.

[3] Valid only in assembler and COBOL languages.

[4] For BASIC and RPG II programs, the set 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.

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, or it received a message from the remote system.

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:** The *output* operation just performed by your program was completed successfully; your program can continue to send data.

Normal Action: For the actions that can be taken (in this session) after 0000 is returned for an output operation, refer to the following chart:

If This Output Operation Was:	Then, in This 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.

0001 **Normal Indication:** Your program has received some data on a successful input operation. It can continue to receive input until SSP-ICF returns a code of xx00 (an end of transmission indication, which allows your program to send data), or xx08 (an end of transaction indication).

Normal Action: Issue another input operation. If your program can detect something equivalent to an end of file condition, indicating that the last of the data was just received, it can issue an output operation.

0008 **Normal Indication:** An end of transaction indication was received with the last of the data on a successful input operation. Communications have ended with the remote *program*, but the session with the remote *system* is still active.

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.

0028 **Normal Indication:** An end of transaction indication was received with a remote 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. Communications have ended with the remote *program*, but the session with the remote *system* is still active.

Normal Action: Handle the message in your program's 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.

0038 **Normal Indication:** An end of transaction indication was received with a *truncated* remote 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. Communications have ended with the remote *program*, but the session with the remote *system* is still active.

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, either issue an end of session operation or terminate.

Major Code 01 – Successful operation with a new requester.

The new requester is a program on a remote system that initiated a session with your program by sending a procedure start request to the local system. 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 procedure start request, initiated by the remote program, was sent as data by the remote system in the form of an *EXEC or *EXEX procedure start statement. The request may have included, for your program, data from the remote program.

Normal Description: Each of the following return codes indicates that the *input* operation issued by your program and responded to by a new requester completed successfully. (Your program cannot issue any output operations in a session started by a procedure start request.) Your program may have received some data from the requester; the data (if any) was included in the incoming procedure start request statement.

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: Continue with the next input operation until an end of transaction minor return code is received.

Code **Indication/Action**

0100 **Normal Indication:** On a successful input operation from a new requester, a procedure start request was received, and some data may have been received with the request. The remote program may now want to continue to send data, or it may want to receive data; your program should issue the appropriate operation.

Normal Action: Handle any data that may have been passed with the request, perform any necessary record keeping[1] for the new requester, and issue an input operation.

[1] For some situations, no record keeping for the session is necessary. In other situations, you should record the session ID of the new requester. You may also want to keep a table containing the IDs of all active requesters, or to maintain a history log of all requests.

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 can continue to receive input until SSP-ICF returns a code of xx00 (an end of transmission indication) or xx08 (an end of transaction indication).

Normal Action: Handle any data passed with the request, perform any necessary record keeping[1] for the new requester, and issue another input operation. If your program can detect something equivalent to an end of file condition, indicating that the last of the data was just received, it can issue an acquire operation to start a different session.

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 remote program. Its communications have ended with your program; however, the session is still active between the local and remote systems.)

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[1] for the new requester of the transaction that has ended. Then, either issue an end of session operation or terminate your program.

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 remote program.) The session has been ended.

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[1] for the new requester of the transaction that has ended. Then, because your program was evoked, issue an end of session operation or terminate.

[1] For some situations, no record keeping for the session is necessary. In other situations, you should record the session ID of the new requester. You may also want to keep a table containing the IDs of all active requesters, or to maintain a history log of all requests.

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, or it received a message from the remote system. 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 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**

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 can continue to receive input until SSP-ICF returns a code of xx00 (an end of transmission indication) or xx08 (an end of transaction indication).

Normal Action: Issue another input operation. If your program can detect something equivalent to an end of file condition, indicating that the last of the data was just received, it can issue an output operation.

0208 **Normal Indication:** An end of transaction indication was received with the last of the data on a successful input operation. Although communications have ended with the remote program, the session with the remote system is still active. 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 (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.

0228 **Normal Indication:** An end of transaction indication was received with a remote 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. Although communications have ended with the remote program, the session with the remote system is still active. 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 your program's input buffer (display it, for example). If 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 your program was evoked, either issue an end of session operation or terminate.

0238 **Normal Indication:** An end of transaction indication was received with a *truncated* remote 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. Although communications have ended with the remote program, the session with the remote system is still active. 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 (to start another program), it can issue a release operation (to perform local processing), or it can terminate. If your program was evoked, either issue an end of session operation or terminate.

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:** Either an end of transmission indication or an end of message indication was received with *no* data on a successful input operation. The last record in the file has been received. Although communications have ended with the remote program, the session with the remote system is still active.

Normal Action: Handle the message that is in your program's input buffer (possibly display it). Also, 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 your program was evoked, it must either issue an end of session operation or terminate.

0308 **Normal Indication:** An end of transaction indication was received *without* data on a successful input operation. Although communications have ended with the remote program, the session with the remote system is still active.

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.

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 remote 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 will be received.

General Recovery Actions: Issue an input operation to receive data or a message from the remote system.

Code Indication/Action

0411 **Normal Indication:** The remote 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 remote 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. Your program must receive the data before it can perform an output operation.

Normal Action: Issue an input operation to receive the data.

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 a new requester) 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 requesting program have been completed.

3401 **Error Indication:** This input operation was rejected because the record length of the data sent by the remote 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/34 Messages Guide* 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 (in COBOL and assembler only) a set timer operation, or by using the TIMER intrinsic function (in BASIC only); (2) continue local processing; or (3) terminate.
- If the session should be started again after the subsystem is enabled, it must be reacquired by your program or restarted by the remote 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 either in this subsystem or in the interrupt handler.

Recovery Action: This subsystem has been disabled; it must be enabled again before communications can resume. Your program can continue local processing, wait[1] to reissue the acquire operation, or terminate.

If more than one subsystem was active when the error occurred, all subsystems that were active when the error occurred should be disabled. (Note that all other active IMS subsystems are automatically disabled when the error occurs; all other types of active subsystems must be manually disabled.)

If all subsystems (of all types) on the system are *not* disabled to recover from the processor check, the common queue space used by the failing subsystem *cannot* be freed. And if it is not freed, that space is wasted, and an indication of insufficient common queue space being available can occur. The indication can occur as a message when the failing subsystem is reenabled or when a different subsystem is enabled. The indication can also occur as an 8315 return code for any subsystem that is performing an output operation in a session.

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 remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, it can wait[1] until the subsystem has been reenabled and reissue the acquire operation, or it can terminate.

8083 **Error Indication:** An MLCA (multiline communications adapter) controller check occurred on an *output* operation. The subsystem has been disabled.

Recovery Action: Communications with the remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, wait[1] until the subsystem is enabled again, or terminate.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

8084 **Error Indication:** An MLCA (multiline communications adapter) controller check occurred on an *input* operation. The subsystem has been disabled.

Recovery Action: Communications with the remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, wait[1] until the subsystem is enabled again, or terminate.

Major Code 81 – Permanent (nonrecoverable) session error.

Error Description: A nonrecoverable error has occurred in the session; the session cannot be continued and 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/34 Messages Guide* 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 session must be acquired again before communications can resume.

General Recovery Actions: The following general actions can be taken for each 81xx return code. Other specific actions are given in each return code description.

- Several return codes indicate that an error condition must be corrected by changing a value in the subsystem configuration record or in the SESSION statement for your program.
  - To change a parameter value in the subsystem configuration being used by your program, disable the subsystem before making the change in the subsystem's configuration record, and enable the subsystem again to make the change effective.
  - To change a parameter value in the SESSION statement associated with your program, you must terminate your program only.

  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.

- If the session should be started again, it must be reacquired by your program or restarted by the remote program before communications can resume.

- An end of session (EOS or $$EOS) operation should be issued for the session that has terminated. Your program can also continue local processing, or it can terminate.

Note: If the session is started again, it starts from the beginning, not at the point where the session error occurred.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

Code **Indication/Action**

8183 **Error Indication:** An MLCA (multiline communications adapter) controller check occurred on an *output* operation; data may have been lost. The session has been terminated.

Recovery Action: If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

8184 **Error Indication:** An MLCA (multiline communications adapter) controller check occurred on an *input* operation; data may have been lost. The session has been terminated.

Recovery Action: If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

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/34 Messages Guide* 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, first disable the subsystem, make the change in the subsystem's configuration record, and 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, (and depending on what the return code description says):
   a. Notify the remote location that a change is required on that end to correct the error received.
   b. Simply reissue the acquire operation. It could be successful if the error occurred because there was not enough common queue space available to support a new session, because an isolated line error occurred, or because the remote system was not active temporarily.
   c. If the acquire operation is again unsuccessful, retry it only a limited number of times. (The limit for retries should be specified in your program.)

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 operation is valid only in an *active* session and cannot, therefore, be issued after an 82xx return code is received. (This restriction does not apply to COBOL and assembler programs, or to the TIMER intrinsic function in BASIC, which also can be used to wait for a specified time interval.)

Code Indication/Action

820D **Error Indication:** On an unsuccessful acquire operation, a normal shutdown indication was received from the IMS host system. All IMS sessions are being terminated; your session was not started.

Recovery Action: Your program can wait[1] for the IMS host system to be started again, then reissue the acquire operation to start the session. Or, it can continue local processing or terminate.

8213 **Error Indication:** On an unsuccessful acquire operation, a queue space error condition was detected. The session could not be started because no *subsystem* queue space was available at the time.

Recovery Action: Your program can wait[1] for a period of time, then reissue the acquire operation. If an unacceptable number of queue space errors occur, you can disable the subsystem and change the subsystem configuration by specifying a larger subsystem queue space size in the subsystem queue space parameter. After the subsystem is enabled, reissue the acquire operation to start the session.

821E **Error Indication:** The acquire operation attempted by your program (BASIC programs only) was unsuccessful because there was no SESSION statement specified between the LOAD and RUN statements for your program. The method used to issue the acquire operation is not supported by the IMS subsystem. The session was not started.

Recovery Action: Issue a SESSION statement that specifies, in the SYMID parameter, the identifier of the session to be acquired. The same identifier must be specified in the ID parameter of the OPEN statement.

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, respecify 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.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

8281 **Error Indication:** On an unsuccessful acquire operation, an SSP-ICF error condition was detected. The error caused a processor check either in this subsystem or in the interrupt handler.

Recovery Action: This subsystem has been disabled; it must be enabled again before communications can resume. Your program can continue local processing, wait[1] to reissue the acquire operation, or terminate.

If more than one subsystem was active when the error occurred, all subsystems that were active when the error occurred should be disabled. (Note that all other active IMS subsystems are automatically disabled when the error occurs; all other types of active subsystems must be manually disabled.)

If all subsystems (of all types) on the system are *not* disabled to recover from the processor check, the common queue space used by the failing subsystem *cannot* be freed. And if it is not freed, that space is wasted, and an indication of insufficient common queue space being available can occur. The indication can occur as a message when the failing subsystem is reenabled or when a different subsystem is enabled. The indication can also occur as an 8315 return code for any subsystem that is performing an output operation in a session.

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 remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, wait[1] 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 100 sessions can be active in the System/34 at one time. The session was not started.

Recovery Action: Your program can wait[1] for another session to end and then reissue the acquire operation. Otherwise, your program can continue local processing or terminate.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

82A9 **Error Indication:** The acquire operation just perfo.med was not successful because the specified IMS host system has not been started. The session was not started.

Recovery Action: Call the remote location and request that the host system be made active. Then reissue the acquire operation. Otherwise, your program can continue local processing, wait[1] to reissue the acquire operation, 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 to be enabled is identified by the location parameter in the SESSION statement. That location name must also be specified in the subsystem configuration record (shown on display 3.0 of the subsystem configuration planning charts). The session was not started.

Recovery Action: Verify that the subsystem name was specified correctly on the LOCATION parameter of the SESSION statement. If the correct name was specified, contact the System/34 system operator and request that the specified subsystem be enabled by executing the ENABLE procedure command at the system console. Then reissue the acquire operation. Otherwise, your program can continue local processing, wait[1] 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[1] until the subsystem has been enabled, and then reissue the acquire operation to start the session.

82AC **Error Indication:** On an acquire operation for a session to be used for a batch job, the operation was not successful because one or more interactive sessions are currently using the line. (The line cannot have any interactive sessions on it when a batch session is active.) The batch session was not started.

Recovery Action: Your program can wait[1] for all of the interactive sessions to end, and then reissue the acquire operation to start the batch session. Otherwise, it can continue local processing or terminate.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

82AD **Error Indication:** An acquire operation for an interactive session was not successful because a batch session is currently active on the line. (The line cannot have any interactive sessions on it when a batch session is active.) The interactive session was not started.

Recovery Action: Your program can wait[1] for the line to become available, and then reissue the acquire operation to start the interactive session. Otherwise, it can continue local processing or terminate.

82AE **Error Indication:** On an unsuccessful acquire operation, an invalid PTERM (physical terminal) address was detected. The PTERM address specified in the PTERM parameter of the SESSION statement does not match any of the PTERM addresses specified in the subsystem configuration. The session was not started.

Recovery Action: Either change the address value specified in the SESSION statement, or disable the subsystem and check all of the PTERM addresses in the subsystem configuration with the remote location. Correct any errors, enable the subsystem again, and reissue the acquire operation.

82AF **Error Indication:** On an unsuccessful acquire operation, an incorrect PTERM (physical terminal) address was detected. The PTERM address specified in the PTERM parameter of the SESSION statement was for the *remote* PTERM address that is reserved for incoming procedure start requests from the IMS host system only. The session was not started.

Recovery Action: Change the PTERM parameter value given on the SESSION statement to identify one of the subsystem *local* PTERM addresses that are specified in the subsystem configuration, or do not specify any address for the PTERM parameter (the subsystem will assign one of the available addresses for the session). Reissue the acquire operation.

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[1] until the subsystem is enabled again, and then reissue the acquire operation. Otherwise, your program can continue local processing, or it can terminate.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

82B1 **Error Indication:** On an unsuccessful acquire operation, the session specified by the session address on the SESSION statement was identified as already being in use. (The session address is specified in the PTERM parameter of the SESSION statement. The address matches one of the addresses in the list of addresses that were specified during subsystem configuration.) The session was not started.

Recovery Action: (1) Your program can wait[1] for the specified session to become available and reissue the acquire operation. (2) You can remove the PTERM parameter value from the SESSION statement and reissue the acquire operation; the subsystem will then assign a session address from those available in the subsystem configuration. (3) Or, your program can continue local processing or terminate.

82B3 **Error Indication:** The acquire operation was not successful because all of the sessions specified in the subsystem configuration are already in use. The session was not started.

Recovery Action: Wait[1] for one of the sessions in the subsystem to become available, then reissue the acquire operation. Otherwise, continue local processing or terminate.

82B4 **Error Indication:** The acquire operation was not successful because all of the resources needed for the session could not be allocated from the assign/free area of the system. All available resources are already being used in the system. The session was not started.

Recovery Action: Wait[1] for the needed resources to become available, then reissue the acquire operation. Otherwise, continue local processing or terminate.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

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/34 Messages Guide* 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, first disable the subsystem, make the change in the subsystem's configuration record, and 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.

   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.

   Note: If the session is started again, it starts from the beginning, not at the point where the session error occurred.
3. If no change is needed in your program or in the subsystem, (and depending on what the return code description says):
   a. Notify the remote location that a change is required on that end to correct the error received.
   b. Retry the operation, if possible. It could be successful if the error occurred because there was not enough common queue space available at the time, because an isolated line error occurred, or because the remote system was not active temporarily.
   c. If another operation is not successful, retry it only a limited number of times. (The limit for retries should be specified in your program.)
   d. Issue a set timer operation in your program so it can wait for a specified time interval before reissuing the operation.

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 an 81xx (session was terminated) or 82xx (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 an 81xx or 82xx return code is handled properly. If you want your program to recover from an improperly handled error condition, issue another acquire operation.

830D **Error Indication:** On an *output* operation, a normal shutdown indication was received from the IMS host system. All IMS sessions are being terminated.

Recovery Action: Your program can wait (using the start timer operation) for the IMS host system to be started again, then reissue the acquire operation to restart the session. Or, it can continue local processing or terminate.

830E **Error Indication:** On an *input* operation, a normal shutdown indication was received from the IMS host system. All IMS sessions are being terminated.

Recovery Action: Your program can wait (using the start timer operation) for the IMS host system to be started again, then reissue the acquire operation to restart the session. Or, it can continue local processing or terminate.

8313 **Error Indication:** On an *output* operation, a queue space error condition was detected. The output operation could not be performed because no *subsystem* queue space was available at the time.

Recovery Action: Your program can issue a set timer operation and wait for a period of time, then reissue the output operation. If an unacceptable number of queue space errors occur, you can disable the subsystem and change the subsystem configuration by specifying a larger subsystem queue space size in the subsystem queue space parameter. After the subsystem is enabled, reissue the acquire operation to restart the session.

8315 **Error Indication:** On an evoke operation, a queue space error condition was detected. The evoke operation could not be performed because no *common* queue space was available at the time.

Recovery Action: Your program can issue a set timer operation and wait for a period of time, and then reissue the evoke operation. If an unacceptable number of queue space errors occur, you can disable all the subsystems and change the subsystem configuration by specifying a larger common queue space size in the SSP-ICF common queue space parameter. After the subsystem is enabled, reissue the acquire operation to restart the session.

8317 **Error Indication:** On an IMS subsystem *output* operation, an indication was received that the maximum output record length has been exceeded. A message from your program, which was being accumulated in the subsystem output buffer, exceeds the maximum user record length specified for the subsystem configuration. The message was not sent.

Recovery Action: Either the record length specified in your program must be changed, or the IMS subsystem must be reconfigured before your program can send the message to the IMS host system. The maximum user record length parameter must be respecified with a value that is large enough for the longest record to be sent or received. After the subsystem has been reconfigured and enabled, the session can be started again. If your program started the session, reissue the acquire operation. If your program was evoked, it can wait to be evoked again, it can continue local processing, or it can terminate.

831A **Error Indication:** An evoke operation failed to complete successfully, or the remote program terminated abnormally. A message from the remote system describing why it failed is waiting in the subsystem input buffer. The evoke operation could have been 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 and continue local processing, or it can terminate.

831C **Error Indication:** The output operation issued before this output operation received a return code of 0411 or 0412 (indicating that the remote 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.

831D **Error Indication:** The previous output operation was invalid. No output operations can be issued in this session because it was started by an incoming procedure start request, and only input operations are allowed.

Recovery Action: Issue an input operation to receive the message or the data; and if output must be sent by your program, issue an acquire operation to acquire a different session. After your program has terminated, correct your program so that only input operations are issued when it has been evoked by a new requester.

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, however, 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 remote 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: If you want your program to recover dynamically, reissue the output operation with a smaller output length. Otherwise, you can either change the record length in your program and recompile it, or you can change the value specified for the maximum user record length parameter in the subsystem configuration. 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.

8320 **Error Indication:** On a *batch output* operation to the IMS host system, your program attempted to send a data record that was longer than 256 bytes. The record was not sent.

Recovery Action: Correct your program so that it sends records that are no longer than 256 bytes.

8322 **Error Indication:** A put with no invite operation was erroneously followed by a release operation. The release operation is not 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 other 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 remote 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 remote program. The session is still active.

Recovery Action: If you want your program to dynamically 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 incoming procedure start request and cannot, therefore, issue any evoke operations in this session.

Recovery Action: If you want your program to dynamically 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.

832B **Error Indication:** On the first output operation, a record length of 0 was detected by the subsystem. The output operation was not performed.

Recovery Action: If a coding error in your program caused the error, correct your program. If the data record is in error, correct it. Then issue another output operation.

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.

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.

8334 **Error Indication:** On an evoke operation issued by your program, no procedure name was included with the operation.

Recovery Action: Correct the evoke operation statement by supplying the correct procedure name, and reissue the operation.

8391 **Error Indication:** A permanent line error occurred on an *output* operation, and the system operator has taken a recovery option in response to the line error message. (You can find out what type of line error occurred by asking the system operator.) The session, however, is still active.

Recovery Action: Your program can immediately issue another output operation to try sending the data again; or it can issue a set timer operation to wait for a period of time before reissuing the output operation.

8392 **Error Indication:** A permanent line error occurred on an *input* operation, and the system operator has taken a recovery option in response to the line error message. (You can find out what type of line error occurred by asking the system operator.) The session, however, is still active.

Recovery Action: Your program can immediately issue another input operation to try receiving data again; or it can issue a set timer operation to wait for a period of time before reissuing the input operation.

Chapter 12. The BSC 3270 Support Subsystem

The BSC 3270 subsystem provides communications support between System/34 and the following host systems using a BSC 3270 network:

- CICS/VS
- IMS/VS
- Model 15 CCP

In addition, the subsystem supports the 3270 Device Emulation Program Product and its communications with TSO and the above host systems. The 3270 Device Emulation Program Product is described in the *3270 Device Emulation User's Guide.*

The subsystem permits a 3270 BSC line to be used concurrently for:

- Existing host system program communications with remote 3270 terminals
- System/34 BASIC, COBOL, RPG II and assembler programs communicating with host system programs
- System/34 display stations and printers operating as terminals communicating with host system programs (This function requires the 3270 Device Emulation Program Product.)

Additionally, if the network is controlled by VTAM, concurrent access to multiple host system programs (CICS/VS and IMS/VS, for example) is possible.

For System/34 program to host system program communications, the subsystem makes the System/34 application program appear to the host system as a remote 3277 display station. Thus, data sent by the host system that would have been displayed on a 3277 display screen is instead used to satisfy a System/34 application program input operation. Conversely, data from a System/34 application program output operation is presented to the remote system as if entered from a 3277 display station.

The subsystem isolates the System/34 application program from 3270 device control information, or allows the System/34 application to send and receive data streams complete with all 3270 device control information.

The 3270 subsystem supports up to 32 concurrent sessions on one multipoint line.

SETTING UP THE 3270 SUBSYSTEM

The SSP procedures CNFIGSSP and INSTALL are used to include the interactive communications and 3270 subsystem support on the System/34. The general interactive communications support is included by requesting it on the appropriate CNFIGSSP prompt. The 3270 subsystem support is then copied to the system library by taking the appropriate responses to the INSTALL procedure prompts. The CNFIGSSP and INSTALL procedures, with their displays and related responses, are described in the *Installation and Modification Reference Manual.*

After the 3270 subsystem has been installed, the CNFIGICF procedure is used to tailor the subsystem support to an existing or proposed network. The operation of the CNFIGICF procedure is also explained in the *Installation and Modification Reference Manual.* Before running the CNFIGICF procedure, however, you should fill out a planning chart for each subsystem that you want to define. Copies of the planning chart for each subsystem are available in Appendix F of this manual and in the *Installation and Modification Reference Manual.* The following sections explain how to fill out the planning chart for the 3270 subsystem.

Display 1.0 Subsystem Member Configuration

1.0 Subsystem Member Configuration

1. Subsystem configuration member name (8 characters) _ _ _ _ _ _ _ _

2. Subsystem library name (8 characters) _ _ _ _ _ _ _ _

Select:

1. Create new member 4. Delete a member

2. Edit existing member 5. Review a member

3. Create new member from existing member

3. Enter selection: ____

4. Existing member name: _ _ _ _ _ _ _ _

5. Existing member library name: _ _ _ _ _ _ _ _

Subsystem configuration member name: Specify a name for this configuration of the subsystem. The name is used to store the member in a library, and is referenced in the ENABLE and DISABLE procedures.

Library name: Specify the name of the library in which the configuration is stored or is to be stored. The default is #LIBRARY, however, you should probably store the configuration in a user library.

Enter selection: Specify one of the five options: (1) create a new member, (2) edit an existing member, (3) create a new member from an existing member, (4) delete a member, or (5) review a member without changing it.

Existing member name: This prompt appears if option 3 was selected. Specify the name of the existing subsystem configuration member that is to be used to create the new member. The existing member remains unchanged.

Existing member library name: This prompt appears if option 3 was selected. Specify the library name where the existing member resides.

Display 2.0 Common SSP-ICF Parameters for Each Subsystem

```
2.0     Common SSP-ICF Parameters for Each Subsystem

        1.   SSP-ICF common queue space:  (2 - 42 K)                 _ _
        2.   Define the subsystem type:                              _ 8
             1  Intra                2  BSC IMS/IRSS
             3  BSCEL                4  BSC CICS
             5  BSC CCP              6  SNA Upline
             7  SNA Peer             8  BSC 3270
             9  SNA 3270            10  Finance
```

SSP-ICF common queue space: Specify the size, in multiples of 2 K bytes, of the common queue space. The common queue space requirements for the BSC 3270 subsystem are:

$$C = 104A + 1264$$

where:

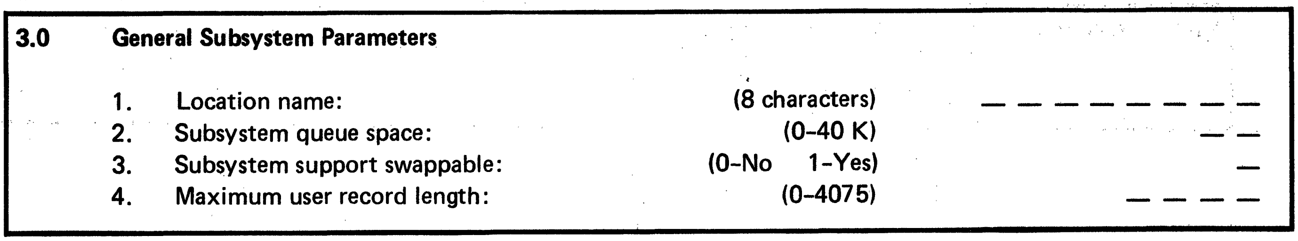
C = number of bytes required for common queue space
A = number of acquired sessions that will be active concurrently (*not* including device emulation sessions)

The common queue space value specified for the first subsystem that is enabled becomes the size of the common queue space. Be sure that the value specified for common queue space size takes into account the requirements of any other subsystem that might be active concurrently.

The size of the common queue space plus the total subsystem queue space of all the enabled 3270 subsystems cannot exceed 42 K bytes. The default common queue space size is 4 K bytes.

Define the subsystem type: Specify an 8 for the BSC 3270 subsystem.

Display 3.0 General Subsystem Parameters



3.0 **General Subsystem Parameters**

1. Location name: (8 characters) _ _ _ _ _ _ _ _
2. Subsystem queue space: (0-40 K) _ _
3. Subsystem support swappable: (0-No 1-Yes) _
4. Maximum user record length: (0-4075) _ _ _ _

Location name: Specify up to 8 characters for the name of the location associated with this configuration. The location name is used in some of the displayed message texts, and must be coded on the SESSION OCL statement. The default is the subsystem configuration member name. The location name refers to the name of the location with which communications is to take place.

Subsystem queue space: Specify the size, in multiples of 2 K bytes, of the subsystem queue space. The subsystem queue space requirements for each configuration of the 3270 subsystem enabled are:

$$S = 2L + 2AR + C + 8$$

where:

S = number of bytes required for the subsystem queue space
L = line buffer length
A = maximum number of sessions that will be active concurrently (*not* including device emulation sessions)
R = maximim record length
C = 0 if maximum record length is 0, otherwise 256

If no other subsystems will be active concurrently, these requirements should be added to the common queue space requirements, and the subsystem queue space should be 0.

The size of the common queue space plus the total subsystem queue space of all the enabled 3270 subsystems cannot exceed 42 K bytes.

The default subsystem queue space size is 4 K bytes.

Subsystem support swappable: Specify whether you want the subsystem to be swappable. Consider the total system performance, the size of the subsystem, and the amount of user main storage when determining whether you want the subsystem to be swappable. The 3270 subsystem requires 8 K bytes of main storage. The subsystem should be specified as swappable if the subsystem will be used for the 3270 Device Emulation Program Product only.

Maximum user record length: Specify the maximum record length (0 through 4075 bytes) to be sent or received by any System/34 application program using this subsystem configuration. This length excludes BSC control characters and messages sent by the host system (CCP messages, for example). If the subsystem will be used only for 3270 device emulation, the maximum record length should be set to 0. The default is 1024 bytes.

Display 4.0 Line Information for SSP-ICF Subsystem

4.0 Line Information for SSP-ICF Subsystem

2. Local station address: (2 hex) _ _

Local station address: Specify the System/34 multipoint line address for this configuration. The local station address for the System/34 must correspond to the address configured for the System/34 during host system configuration. (See the chart at the end of this section for the relationship of the local station address to the host configurations.) The following are valid local station addresses: 40, C2, C3, C4, C5, C6, C7, C8, C9, 4A, 4B, 4C, 4D, 4E, 4F, D1, D2, D3, D4, D5, D6, D7, D8, D9, 5A, FB, 5C, 5D, 5E, and 5F.

Display 14.0 3270 Subsystem General Parameters

14.0 3270 Subsystem General Parameters

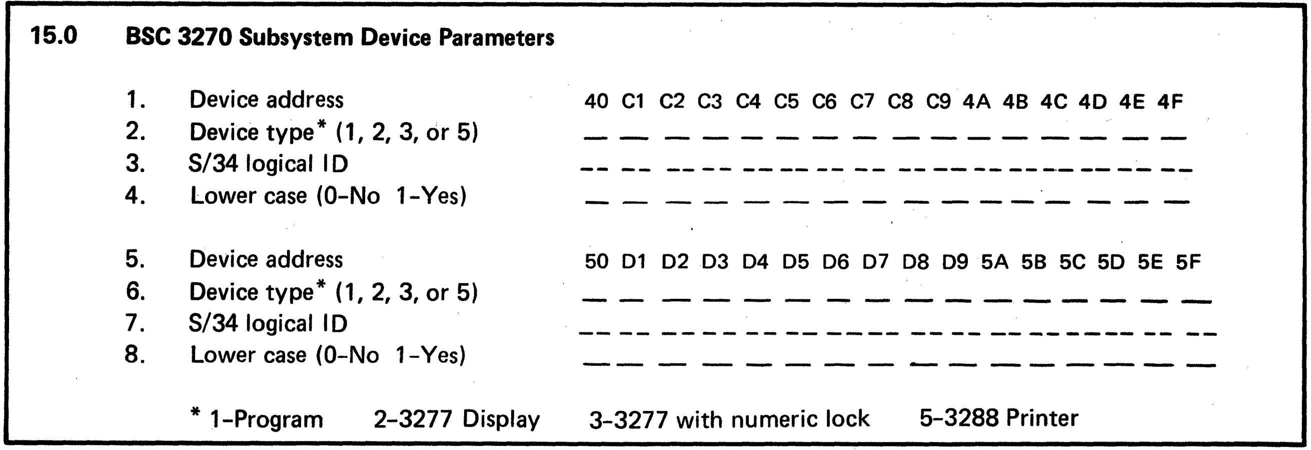
1. Line buffer size: (256-4096) _ _ _ _
2. Delay count: (0-255) _ _ _

Line buffer size: Specify, in decimal, the number of bytes in the line buffer. The line buffer size should be greater than or equal to the largest transmission that will be received from the host system, up to a maximum of 4096 bytes. The size includes any BSC control characters and 3270 device control characters, as well as any messages sent by the host system (CCP messages, for example). The default size of 2040 bytes should be changed only if queue space limitations necessitate a decrease, or if line buffer overflows predictably occur (necessitating an increase).

Delay count: Specify the number of times the BSC interrupt handler should negatively respond before terminating an operation. A delay count greater than zero should be specified only if return codes that indicate the delay count was exceeded while transmitting or receiving (8399 or 839A) are received intermittently because of high System/34 processor overload. The delay count should not be increased if the return codes are received in the following situations:

- A display station using BSC 3270 device emulation experiences an error or is powered off.
- System/34 programs regularly experience one or both return codes (8399 or 839A). Increasing the delay count in this situation would severely degrade the performance of the entire 3270 network.

Display 15.0 BSC 3270 Subsystem Device Parameters



```
15.0   BSC 3270 Subsystem Device Parameters

       1.  Device address               40 C1 C2 C3 C4 C5 C6 C7 C8 C9 4A 4B 4C 4D 4E 4F
       2.  Device type* (1, 2, 3, or 5) __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
       3.  S/34 logical ID              __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
       4.  Lower case (0-No  1-Yes)     __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __

       5.  Device address               50 D1 D2 D3 D4 D5 D6 D7 D8 D9 5A 5B 5C 5D 5E 5F
       6.  Device type* (1, 2, 3, or 5) __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
       7.  S/34 logical ID              __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
       8.  Lower case (0-No  1-Yes)     __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __

       * 1-Program   2-3277 Display   3-3277 with numeric lock   5-3288 Printer
```

Device address: Each address represents one of the allowable device address for a 3270 session. The device address corresponds to values specified during host system generation. (The chart that follows shows the relationship between the device addresses and the host system configuration.) These addresses can be specified for a particular session by using the DEVADDR parameter on the SESSION OCL statement.

Device type: Specify the type of device that will be attached to the device address. Possible values and their meanings are:

blank Undefined. The address will not be used for this configuration.

1 Program. The device address will be used to address an application program on the System/34. Only addresses with device type 1 can be acquired.

2 3277 Display. The device address will be used to address a 5251 Display Station. The 3270 Device Emulation Program Product is used to make the 5251 appear to be a 3277 Display Station.

3 3277 Display with Numeric Lock. The device address will be used to address a 5251 Display Station. The 3270 Device Emulation Program Product is used to make the 5251 appear to be a 3277 Display Station that has the Numeric Lock feature.

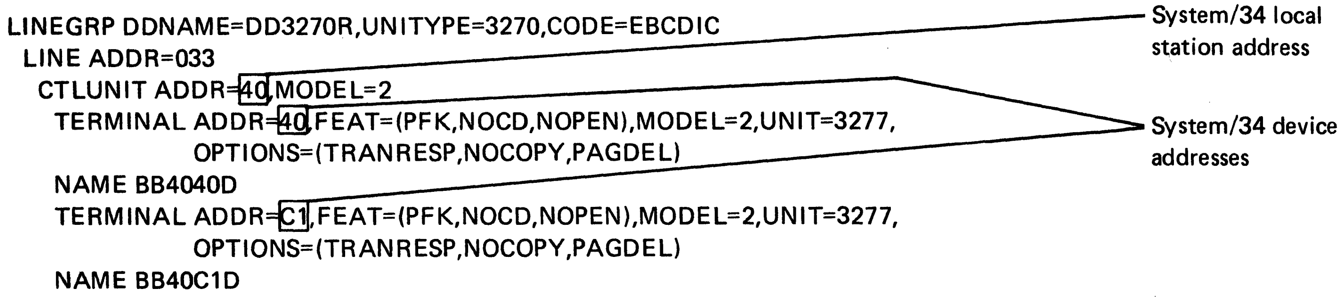
5 3288 Printer. The device address will be used to address a printer. The 3270 Device Emulation Program Product is used to make the printer on the System/34 appear to be a 3288 line printer.

S/34 logical ID: Do not specify this parameter for device type 1 (program). Specify the logical ID of the work station to be used for the device address. If the logical ID is specified, it must match the logical ID of a local display station or a local printer. The device address is reserved for use exclusively by that System/34 work station. A System/34 logical ID can appear only once in a configuration. If no logical ID is specified with a device address, any appropriate device type can use the device address on a first-come, first-served basis.

Lowercase: Do not specify this parameter for device types 1 (program) or 5 (printer). Specify whether lowercase is allowed on the display station. Specify 0 (no) or 1 (yes). If lowercase is not specified, all lowercase characters are displayed and sent to the host system as uppercase characters.

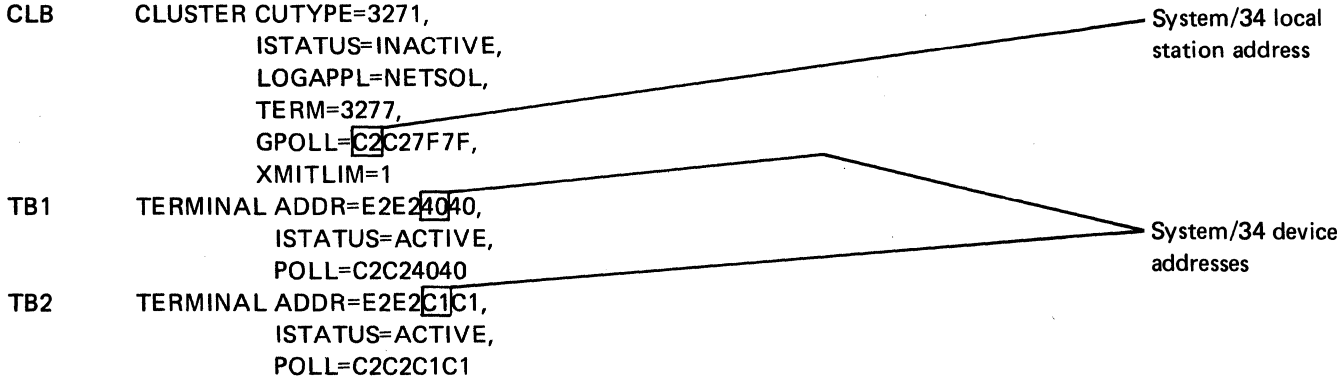
The following figures show the relationship between System/34 local station addresses and device addresses and those specified during host system configuration.

IMS/VS Stage 1 Generation Input for BTAM Terminals



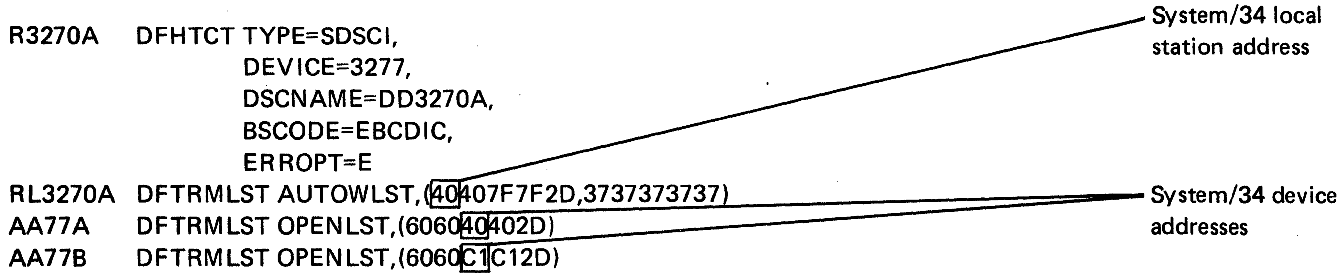
```
LINEGRP DDNAME=DD3270R,UNITYPE=3270,CODE=EBCDIC
  LINE ADDR=033
    CTLUNIT ADDR=40,MODEL=2
      TERMINAL ADDR=40,FEAT=(PFK,NOCD,NOPEN),MODEL=2,UNIT=3277,
                 OPTIONS=(TRANRESP,NOCOPY,PAGDEL)
      NAME BB4040D
      TERMINAL ADDR=C1,FEAT=(PFK,NOCD,NOPEN),MODEL=2,UNIT=3277,
                 OPTIONS=(TRANRESP,NOCOPY,PAGDEL)
      NAME BB40C1D
```

NCP Stage 1 Generation Input (for IMS/VS or CICS/VS VTAM Terminals)



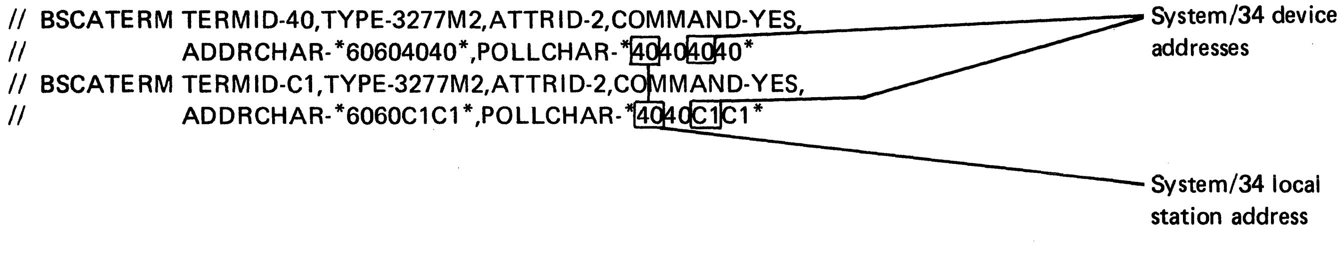
```
CLB      CLUSTER CUTYPE=3271,
                 ISTATUS=INACTIVE,
                 LOGAPPL=NETSOL,
                 TERM=3277,
                 GPOLL=C2C27F7F,
                 XMITLIM=1
TB1      TERMINAL ADDR=E2E24040,
                  ISTATUS=ACTIVE,
                  POLL=C2C24040
TB2      TERMINAL ADDR=E2E2C1C1,
                  ISTATUS=ACTIVE,
                  POLL=C2C2C1C1
```

CICS/VS Terminal Control Table Input for BTAM Terminals



```
R3270A   DFHTCT TYPE=SDSCI,
                DEVICE=3277,
                DSCNAME=DD3270A,
                BSCODE=EBCDIC,
                ERROPT=E
RL3270A  DFTRMLST AUTOWLST,(40407F7F2D,3737373737)
AA77A    DFTRMLST OPENLST,(60604040402D)
AA77B    DFTRMLST OPENLST,(6060C1C12D)
```

CCP Assignment Set Input



```
// BSCATERM TERMID-40,TYPE-3277M2,ATTRID-2,COMMAND-YES,
//          ADDRCHAR-*60604040*,POLLCHAR-*40404040*
// BSCATERM TERMID-C1,TYPE-3277M2,ATTRID-2,COMMAND-YES,
//          ADDRCHAR-*6060C1C1*,POLLCHAR-*4040C1C1*
```

HOST SYSTEM CONFIGURATION

Some or all of the following items may be required before communications with the System/34 can take place:

- CICS/VS, IMS/VS, VTAM, NCP, or CCP generation. This generation is necessary only if the existing generation parameters do not accommodate the remote 3270 support provided by the System/34.

- Generation of CICS/VS tables, IMS/VS control blocks, or a CCP assignment set. This generation is necessary only if the existing parameters do not define a remote 3271 cluster suitable for use by the System/34.

- Write application programs that communicate with specific System/34 application programs. Existing programs written for remote 3270 communications are most suitably run, without modification, by System/34 display stations using the 3270 Device Emulation Program Product.

Specific generation considerations for each host system are in the appropriate section on that host system later in this chapter.

STARTING AND ENDING THE 3270 SUBSYSTEM

The sequence of operations necessary to establish communications between the System/34 and the host system depends on:

- The host system generation parameters

- The current state of the host system communications program (CCP, for example)

- Any special requirements of the host system application program

The first step, in any case, is for the System/34 operator to run the ENABLE procedure. After being enabled, the 3270 subsystem begins to monitor the communications line. The subsystem responds to BSC control sequences as a 3271 Model 2 cluster control unit would respond if all its devices were powered off.

After the ENABLE procedure has been run, System/34 application programs can issue an acquire operation to the 3270 subsystem. The host system is made aware of a successful acquire operation in the same manner as a 3271 cluster control unit notifies the host system that power has been turned on at a 3277 display station.

The DISABLE procedure stops the subsystem. When the disable is performed, the 3270 subsystem ceases to monitor the communications line. When the subsystem terminates, the host system sees the System/34 as a 3271 cluster control unit that has been powered off.

SESSION OCL STATEMENT

The System/34 OCL statement that is unique to interactive communications is the SESSION statement. One SESSION statement must be given for each session used by the application program. The SESSION statement establishes the parameters required by the subsystem to interface with the program for a particular session.

The format of the SESSION statement for the 3270 subsystem is:

```
// SESSION LOCATION-name,SYMID-session-id,

[DATAID-cc,FLDLTH-length] [,DEVADDR-xx]

[               (CICS  ) ]
[ ,HOSTNAME-   { IMS    }]
[               (OTHER ) ]
```

LOCATION: Specifies the location name associated with this session. This name is the same as the location name specified during subsystem configuration.

SYMID: Specifies the symbolic ID of the session with which this SESSION statement is associated. The symbolic ID must be two characters, with the first character numeric (0 through 9) and the second character alphabetic (A through Z, #, $, or @). This is the same ID that the application program uses when referring to this session.

DATAID: Specifies the two characters that identify the way data is to be handled. This parameter and the FLDLTH parameter are mutually inclusive. When specified, they indicate that the System/34 application program should be isolated from 3270 device control information. This is accomplished as follows:

- An input operation performed on a session with the DATAID parameter specified receives only application data. All 3270 device control information is removed. See *Input Operations* later in this chapter for more details.

- 3270 device control information is added to the data provided by a put operation whenever the first two characters of the data are the same as the DATAID parameter. The FLDLTH parameter determines exactly what 3270 device control information is added.

FLDLTH: Specifies the length of fields and format of output data. If the first two characters of an output record match the DATAID parameter, the subsystem adds all necessary 3270 device control information to make the record a valid 3270 format.

If the value of the FLDLTH parameter is greater than 0, the subsystem formats the data as if it were entered from a 3277 display station with the following format:

- Row 1, column 1 is an input field delimiter (an attribute indicating that the field is unprotected).
- The next consecutive screen positions, starting with row 1, column 2, become an input field. FLDLTH defines the length of this and the following input fields.
- The next consecutive screen position is another input field delimiter (an attribute indicating that the field is unprotected) followed by the next input field.

The last step is repeated until all the data from the System/34 has been formatted.

If the value of the FLDLTH parameter is 0, the subsystem formats the data as if it were entered from a 3277 display station with no field delimiters (attributes) on the screen.

Regardless of the value of the FLDLTH parameter, the cursor position is returned as row 1, column 1, and the Enter key pressed code (enter AID byte) is returned to the host system.

DEVADDR: Specifies the 2-character hexadecimal device address associated with this session. The host system uses this address when sending data to this session and when identifying data sent by this session. If this parameter is not specified, the first available device address that was configured for programs will be used.

HOSTNAME: Specifies the host type for this session. CICS indicates that the host system is CICS/VS; IMS indicates that the host system is IMS/VS; OTHER indicates that the host system is CCP, and is the default.

OPERATION CONSIDERATIONS

System/34 application programs communicate with host system applications by issuing operation requests to the 3270 subsystem. The following sections describe each of the operations supported by the 3270 subsystem. A complete chart of all interactive communications operations and the subsystems that support them is in the appropriate language chapter. The chart also shows the keyword or format name used to code the operation in that language. More information about how each operation is coded is in the appropriate programming language chapter.

Whether an operation completes successfully or unsuccessfully, a return code is given to the application program. All of the return codes that are valid for the 3270 subsystem are described at the end of this chapter. A summary chart in Appendix A lists all the return codes and the subsystems for which they are valid.

Acquire Operation

A session with the 3270 subsystem is established by issuing an acquire operation. The acquire operation includes an ID that corresponds to the SYMID parameter of a SESSION OCL statement. The SESSION statement is used by the BSC 3270 subsystem when performing the acquire operation to determine the environment of the session being acquired.

The host system identifies the session by the physical device address assigned when the session is acquired. The device address assigned to the session is determined by the configuration being used and the SESSION OCL statement. If the DEVADDR parameter is specified on the SESSION statement and the specified address corresponds to an address defined as a program, that address is assigned to the session. If the DEVADDR parameter is not specified, the first available program address is allocated to the session. If the DEVADDR is the wrong type of address or is currently in use, the acquire operation fails with a return code indicating that the address is unavailable (82B1).

If the DEVADDR parameter is not specified and there are no available program device addresses, the acquire operation fails with a return code indicating that no sessions are available (82B3).

The successful completion of an acquire operation is directly analogous to powering on a 3277 display station. After the acquire operation completes, the System/34 application program must issue any input and output operations necessary to begin communications with the host system (sending a VTAM logon and receiving the logon messages, for example). See the appropriate host system considerations later in this chapter for more information on host requirements.

Evoke Operations

The function performed by the evoke operation depends on the HOSTNAME parameter of the SESSION OCL statement. This section describes general evoke processing. More specific information is in the host system considerations later in this chapter.

In the case of the BSC 3270 subsystem, an evoke operation is treated as a special type of output operation. Three forms of the evoke operation are supported: evoke ($$EVOKNI), evoke then invite ($$EVOK), and evoke then get (assembler only). Evoke operations are valid only for sessions defined as follows:

- The DATAID parameter must be specified on the SESSION statement.
- The FLDLTH parameter must be specified on the SESSION statement.
- The HOSTNAME parameter on the SESSION statement must be either IMS or CICS.

If an evoke operation is issued for a session that is not defined in this manner, the evoke is rejected as an unsupported operation (return code of 831E).

For any valid evoke operation, the subsystem always adds 3270 device control information before sending the data to the host system. The DATAID and FLDLTH parameters must be specified, but have no effect on the data sent by the evoke operation.

The evoke operation is normally used by a System/34 application program when starting a transaction or issuing a system command to the host system. The System/34 application program places the procedure name in the evoke parameter list. Any data associated with the evoke operation is placed in the application program record area.

The 3270 subsystem processes the evoke operation as follows:

- The procedure name from the evoke parameter list becomes the first eight characters of data sent to the host system.
- The user data length specifies the length of data to be appended to the procedure name and sent to the host system.
- The user data is appended to the procedure name and becomes the ninth through the last characters sent to the host system.

All other evoke parameters are ignored by the 3270 subsystem.

Data included with the evoke operation must conform to the restrictions for output data as described under *Put Operations*.

Put Operations

Three forms of put operations are supported by the 3270 subsystem: put end of file ($$SENDE), put then invite ($$SEND), and put then get (assembler only).

Processing of the put operation depends on the DATAID and FLDLTH parameters of the SESSION OCL statement and the first two characters of the output data. If the first two characters of the data supplied by the System/34 application program match the DATAID parameter, the subsystem adds all necessary 3270 device control information to make the record a valid 3270 format. For more information on the format generated, see the SESSION OCL statement description earlier in this chapter.

If the first two characters of the data supplied by the System/34 application program do not match the DATAID parameter or the DATAID parameter is not specified, the subsystem does not add any 3270 device control information. The subsystem does frame the data with the start of text and end of text control characters and the control unit and device addresses to ensure valid 3270 BSC protocol. The System/34 program must ensure that a valid read modified format is sent to the host system. The System/34 programmer must thoroughly understand the *3270 Information Display System Components Description Manual* to ensure valid formatting.

A System/34 application program can only send valid character (graphic) data to the host system. This precludes sending binary or packed numbers, object programs, and data with imbedded device control information. In general, any attempt to send data with a value of less than hex 40 causes unpredictable results.

Input Operations

To receive data from a host application, the System/34 program must issue an accept or get operation to an invited session or a get to an uninvited session. This could include the get or invite portion of a put or evoke operation. An input request remains outstanding until the host system sends data to the physical device address allocated to this session.

If the DATAID parameter was specified, all 3270 device control information is removed from the data received from the host system before the data is passed to the System/34 application program.[1]

[1] There are three 3270 orders that require special processing not performed by the 3270 subsystem: the *repeat to address* order, the *program tab* order, and the *erase all unprotected to address* order. These orders are removed from the data received, but a minor return code (12) indicating the occurrence of one or more of these orders is returned with the data. These orders are not normally generated by the host system, but are used as selectable options to decrease communications line overhead. For more information on these orders, see the *3270 Information Display System Components Description Manual.*

If the first two characters of the received data (excluding 3270 device control information) are different from the value of the DATAID parameter, a minor return code (20) indicates that the received data is a message.[2] Received data consisting solely of 3270 device control information (for example, an erase all unprotected command) does not satisfy an input operation request.

If no DATAID parameter was specified, all data received from the host system (including 3270 device control information, but excluding BSC control characters) is passed to the System/34 application program. Neither the 12 nor the 20 minor return code is set.

In all instances, a valid supported command sequence (write, erase write, erase all unprotected, or read modified) must be received by the 3270 subsystem or the subsystem sends a command reject to the host system, and issues a return code (8338) indicating that an invalid or unsupported command was received.

The following is an example of how the 3270 subsystem handles data received from the host system.

If the data stream received was:

> STX-ESC-Write control-SBA-@@-Start field-Attribute-Insert
> cursor-ABCD-ETX hex 02 27 F1 40 11 40 40 1D 40 13 C1C2C3C4 03

and the DATAID parameter was AB, the System/34 application program receives the ABCD with a minor return code of 00.

If the DATAID parameter was BB, the System/34 application program receives the ABCD with a minor return code of 20.

If the DATAID parameter was not specified, the System/34 application program receives:

> ESC-Write-Write control-SBA-@@-Start field-Attribute-Insert
> cursor-ABCD hex 27 F1 40 11 40 40 1D 40 13 C1C2C3C4

with a minor return code of 00.

For a more complete description of what these data streams mean, see the *3270 Information Display System Components Description Manual.*

[2] If an unsupported order (minor return code 12) and a message condition (minor return code 20) exist simultaneously, the message condition is indicated in the return code.

Release Operation

Before the System/34 application program terminates, a release operation should be issued for each session. The operation is performed unless the host system has sent data not yet received by the application program or unless the session is invited. If a release operation is issued to the 3270 subsystem while data from the host system is waiting to be passed to the System/34 application program, the release is not performed. A return code indicating that data is waiting (0412) is passed to the program, which must then issue an input operation to retrieve the data. After all data has been retrieved, the release operation can be retried and will complete normally.

A successful release operation terminates the session normally. When the system/34 application program terminates, all sessions not previously released are terminated abnormally. Termination of a session appears to the host system as the 3277 display station being powered off. The 3270 subsystem does not purge host system output queues, terminate host transactions, or log off the host applications; these and other logical cleanup activities are the responsibility of the System/34 application program. Failure to complete these activities could result in errors for the next session using the same device address.

End of Session Operation

The end of session operation is treated as an unconditional release, and is always performed. The return code is always 0000.

Get Attributes Operation

The get attributes operation can be issued at any time to determine the status of a session.

Set Timer Operation

The set timer operation results in a timer expired return code (0310) after a specified time interval in hours, minutes, and seconds has elapsed.

HOST SYSTEM STATUS INFORMATION

The 3270 subsystem generates status information that can be transmitted to the host system. The following are the status codes and their meanings that can be sent by the 3270 subsystem:

- Operation check (OC): No input buffer is currently available for data sent by the host system; or, the escape command did not immediately follow the start of text in the data stream sent by the host system; or, the data sent by the host system exceeded the maximum user record length.

- Command reject (CR): A copy command was received from the host system; or, an invalid command was received by the host system.

- Intervention required (IR): No session has acquired the specified 3270 device address.

- Device busy (DB): No input buffer is currently available for data from the host system.

- Device end (DE): A session has now been acquired for the specified 3270 device address; or, an input buffer is now available to receive data from the host system.

- Control check (CC): The System/34 processor is overloaded.

- Unit specify (US): A command was sent to a busy device.

CONSIDERATIONS FOR HOST SYSTEMS USING VTAM

In general, the System/34 should be generated to VTAM as a remote 3271 cluster with as many terminals as necessary. The following parameters for stage 1 input to NCP generation affect the startup procedures at both the System/34 and the host system:

- Group CRETRY=255 eliminates the need for the 'vary active, vary inactive' sequence that is otherwise required to restart polling to a cluster that did not respond to polling (because it was powered off, for example). Network performance can be degraded, however, because the host system must wait for a time-out of all nonresponding clusters on each polling cycle.

- CLUSTER ISTATUS=ACTIVE eliminates the need for the vary active command that is otherwise required to start polling the cluster after NCP is initially loaded into the 370x.

- CLUSTER LOGAPPL=name or TERMINAL LOGAPPL=name eliminates the need for the logon request that is otherwise required to attach a terminal to the named program (CICS/VS for example). This parameter does not prevent terminals from logging off of the named program.

- TERMINAL ISTATUS=ACTIVE eliminates the need for the vary active command that is otherwise required to begin accepting input from a terminal after NCP is initially loaded into the 370x.

Depending on the above parameters, the VTAM operator might have to perform a vary active command or a 'vary inactive, vary active' command sequence for the System/34 after the subsystem is enabled. Any display station signed on to the 3270 Device Emulation Program Product and any System/34 application programs that have performed an acquire operation can receive host system messages as a result of these commands.

Also depending on the above parameters, System/34 application programs might have to issue a logon request to VTAM. If the logon request must be issued, check your host system instructions for the format of the request.

CONSIDERATIONS FOR IMS/VS HOSTS

System/34 Programming Considerations

All 3270 devices are automatically defined to operate with the message formatting service provided by IMS/VS. Consequently, the System/34 program must assure that the format of data sent to IMS/VS corresponds to the active device input format for that device. There are three ways that the System/34 programmer can accomplish this:

- Use the evoke operation
- Use a put operation with the first two characters of data matching the DATAID parameter
- Use a put operation with the first two characters of data not matching the DATAID parameter

The evoke operation emulates the 3277 Clear key before transmitting the data provided with the evoke. As a result, IMS/VS switches from formatted mode to unformatted mode, in which basic edit is used instead of message format services to process incoming messages. Basic edit removes the attention identifier and cursor address, and changes all start buffer address sequences to blanks.

The evoke operation should be used to send IMS/VS commands (for example, /FORMAT). These commands could include:

- Session initialization and termination
- Establishing formats
- Exception condition handling
- Responses to IMS/VS messages

The put operation is recommended for sending transaction requests and data records to IMS/VS because of the relatively high overhead involved with the evoke operation.

A put operation can be issued whose first two characters of data match the value of the DATAID parameter of the SESSION OCL statement. This operation will be successful only if both of the following are true:

- The active device input format has its first input field at row 1, column 2; that is, the DFLD macro has the POS=(1,2) parameter.
- The active device input format has its second and succeeding input fields following the first at fixed intervals of one greater than the value of the FLDLTH parameter on the SESSION statement.

A put operation can be issued whose first two characters of data differ from the value of the DATAID parameter of the SESSION statement. The operation will be successful only if the format of the data corresponds, without further modification by the subsystem, to the active device input format. The format must consist of the following fields, in order:

1. The attention identifier byte

2. The cursor positions (2 bytes)

3. A set buffer address character (hex 11), an input field position (2 bytes), and the data

Item 3 must be included for each field. The input field position must match the POS parameter on the DFLD macro for the input field being returned. Not all fields defined in the device input format must be returned, but all fields returned must be in ascending order by position. The data following the input field position is only that portion to be returned to the associated input field. If no data fields are to be returned, omit item 3 entirely. A complete description of the values that can be used in these fields is contained in the *3270 Information Display System Component Description Manual.*

Consecutive input operations (without an intervening output operation) will cause an indefinite wait. After IMS/VS has successfully sent output to a 3270 terminal, the screen is protected from further output until the terminal responds. The IMS/VS NEXTMSGP function can be used, however, for dequeuing consecutive messages. After the System/34 program has successfully completed an input operation, the program should issue a put operation consisting solely of the 3270 code for the PF key defined as NEXTMSGP. (For example, for PF11 the code is the # character, hex 7B.) Then the System/34 application program should issue another input operation to receive the next message. If the message queue is empty, the System/34 program receives an IMS/VS message (DFS290 for example).

Sessions with IMS/VS

In a typical System/34 application, a session with IMS/VS would consist of the following operations:

1. An acquire operation

2. An evoke operation

3. A put operation

4. An input operation

5. A release operation

The following describes an example of this type of session.

The System/34 application program performs an acquire operation for the session identified by the SYMID parameter of the following SESSION OCL statement:

```
// SESSION LOCATION-IMS1,SYMID-1S,FLDLTH-1919,DATAID-XY,HOSTNAME-IMS
```

The DATAID characters, XY, are the first two characters of the transaction name that this System/34 program will execute.

Next, the System/34 program performs an evoke operation with the data:

```
/FORMAT S34MOD
```

The S34MOD references an IMS/VS message format services message output description as follows:

```
S34MOD     MSG       TYPE=OUTPUT,
                     SOR=S34FMT,
                     NXT=S34MID
           LPAGE     SOR=S34PAG
           SEG
           MFLD      S34FLD,
                     LTH=1919
           MSGEND
```

The format (S34FMT) referenced by the SOR parameter is defined as follows:

```
S34FMT     FMT
           DEV       TYPE=(3270,2),
                     FEAT=(PFK,NOCD,NOPEN),
                     SYSMSG=S34FLD,
                     DSCA=X'0080'
           DIV       TYPE=INOUT
S34PAG     DPAGE     FILL=NULL
S34FLD     DFLD      LTH=1919,POS=(1,2)
           FMTEND
```

The message input descriptor (S34MID) referenced by the NXT parameter in the S34MOD definition is defined as follows:

```
S34MID     MSG       TYPE=INPUT,
                     SOR=S34FMT,
                     NXT=S34MOD
           LPAGE     SOR=S34PAG
           SEG
           MFLD      S34FLD,
                     LTH=1919,
                     FILL=NULL
           MSGEND
```

The System/34 application program then performs a put operation with the first two characters of data (that is, the first two characters of the transaction name) matching the value of the DATAID parameter specified on the SESSION OCL statement (XY in this example).

Depending on the characteristics of the IMS/VS application program started by step 3, the System/34 application can issue an input operation, perform another put or evoke operation, or end the session. When the System/34 application program is ready to end the session, it issues a release operation and the session is terminated.

IMS/VS Generation Considerations

The TERMINAL macro is input to the stage one generation of IMS/VS. The following parameters should be specified for System/34 programs:

```
FEAT=(PFK,NOCD,NOPEN),
MODEL=2,
UNIT=3277,
MSGDEL=NONIOPCB
```

The MSGDEL parameter prevents IMS/VS from sending messages designed for a display station to the System/34 program.

Startup for IMS/VS

Following the enable of the System/34 BSC 3270 subsystem the host system IMS/VS operator may have to perform a /START or /RESTART command for the PTERMs associated with the System/34. As a result of these commands, an IMS/VS message is queued to each PTERM not generated with the MSGDEL=NONIOPCB parameter. This startup consideration applies only to non-VTAM networks.

IMS/VS Message Formatting Services Considerations

For System/34 programs that do all the formatting of data sent to IMS/VS (that is, programs that do not use the DATAID parameter or do not use characters that match the DATAID parameter), there are no restrictions on message formatting services formats.

For System/34 programs that let the 3270 subsystem do the formatting of data sent to IMS/VS (that is, do put operations that match the value of the DATAID parameter), the device input format must:

- Have its first input field at row 1, column 2 (the DFLD macro with the POS=1,2 parameter)
- Have all subsequent input fields, if any, at fixed intervals of one greater than the value of the FLDLTH parameter of the SESSION OCL statement

For example, if the SESSION statement specified FLDLTH-79, then a device input format with three input fields would be:

```
S34FMT2   FMT
          DEV      TYPE=(3270,2)
                   FEAT=(PFK,NOCD,NOPEN),
                   SYSMSG=S34FLD1
          DIV      TYPE=INOUT
S34FLD1   DFLD     POS=(1,2)
S34FLD2   DFLD     POS=(2,2)
S34FLD3   DFLD     POS=(3,2)
          FMTEND
```

Use of the SYSMSG parameter is recommended, because the active format is preserved whenever IMS/VS sends a message.

The DSCA parameter on the DEV macro should indicate that the erase all unprotected and program tab functions should not be used.

IMS/VS Programming Considerations

A response alternate PCB (ALTRESP=YES parameter) should not be used for System/34 programs. Should an IMS/VS application program operating in conversational or response mode satisfy the I/O PCB requirement by using an alternate PCB, the System/34 program would be unable to continue processing.

Coding Example

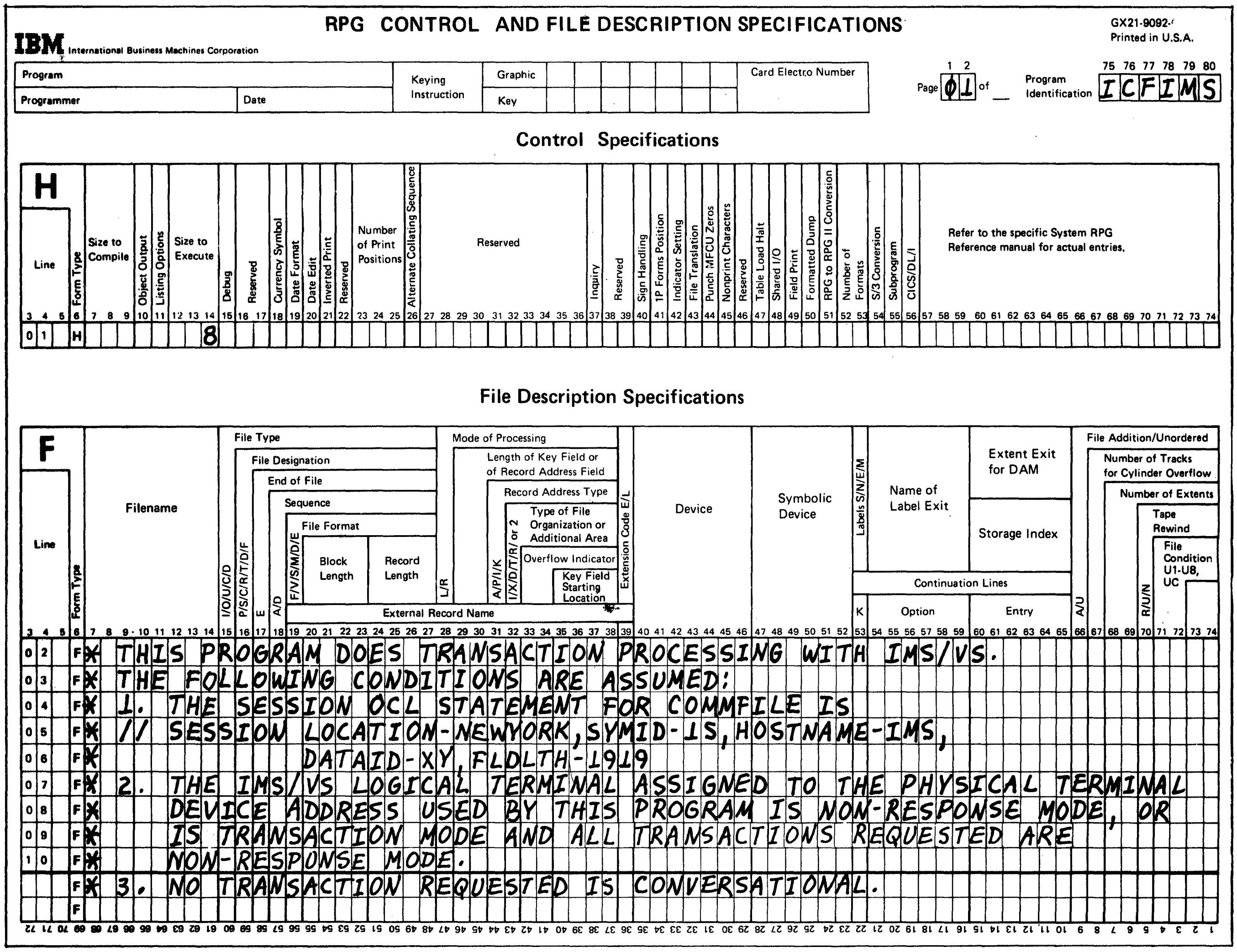
The following sample programs use the 3270 subsystem to communicate with IMS/VS.

When the System/34 program is started, it transmits a data file to the host system, one record at a time. The programs operate in a transaction-processing environment; that is, the host transaction is started, performs a single input operation, and terminates.

The first function of the System/34 program is to acquire a session (1S) with the 3270 subsystem. Next the program must issue an evoke with no invite operation ($$EVOKNI) to send the IMS/VS/FORMAT command. Then, a data record is read from the System/34 file. A put end of file operation ($$SENDE) is issued to send the data record. The first eight characters of data are the host system application name; the name is followed by the data record. The process of reading and sending a record continues until the entire file is sent. the session is then released, and the System/34 program terminates.

```
1   IDENTIFICATION DIVISION.
2   PROGRAM-ID.  ICFIMS.
3   SECURITY.  THIS PROGRAM DOES TRANSACTION PROCESSING WITH IMS/VS.
        THE FOLLOWING CONDITIONS ARE ASSUMED:
        1.  THE SESSION OCL STATEMENT FOR COMM-FILE IS
            // SESSION LOCATION-NEWYORK,SYMID-1S,HOSTNAME-IMS,
               DATAID-XY,FLDLTH-1919
        2.  THE IMS/VS LOGICAL TERMINAL ASSIGNED TO THIS PHYSICAL
            TERMINAL (DEVICE) ADDRESS USED BY THIS PROGRAM IS
            NON-RESPONSE MODE, OR IS TRANSACTION MODE AND ALL
            TRANSACTIONS REQUESTED ARE NON-RESPONSE MODE.
        3.  NO TRANSACTION REQUESTED IS CONVERSATIONAL.
        4.  NO TRANSACTION REQUESTED QUEUES A RESPONSE BACK TO
            ITS REQUESTOR.
        5.  NO LOGON OR LOGOFF TO IMS/VS IS REQUIRED.
        6.  EACH RECORD READ FROM LOCAL-TRANS-FILE BEGINS WITH
            AN IMS/VS TRANSACTION NAME.  EACH TRANSACTION NAME
            BEGINS WITH THE CHARACTERS "XY".
4   ENVIRONMENT DIVISION.
5   CONFIGURATION SECTION.
6   SOURCE-COMPUTER.  IBM-S34.
7   OBJECT-COMPUTER.  IBM-S34.
8   SPECIAL-NAMES.
9       SYSTEM-CONSOLE IS CONSOLE.
10  INPUT-OUTPUT SECTION.
11  FILE-CONTROL.
12      SELECT COMM-FILE ASSIGN TO WORKSTATION-FORMAT-01,
            ORGANIZATION IS TRANSACTION,
            FILE STATUS IS WS-RETURN-CODE, ICF-RETURN-CODE,
            CONTROL-AREA IS IMS-CONTROL-AREA.
13      SELECT LOCAL-TRANS-FILE ASSIGN TO DISK-IMSTRANS,
            ORGANIZATION IS SEQUENTIAL,
            ACCESS MODE IS SEQUENTIAL.
14  DATA DIVISION.
15  FILE SECTION.
16  FD  COMM-FILE LABEL RECORDS ARE OMITTED.
17  01  COMM-RECORD                    PIC X(128).
18  FD  LOCAL-TRANS-FILE LABEL RECORDS ARE STANDARD.
19  01  IMS-TRANSACTION                PIC X(80).
20  WORKING-STORAGE SECTION.
```

```
   ****************************************************************
   *    STATUS BYTES FOR IMS/VS SESSION I/O                       *
   ****************************************************************
21  01  IMS-RETURN-CODE.
        03  WS-RETURN-CODE           PIC X(2).
23      03  ICF-RETURN-CODE          PIC X(4).
24          88  NORMAL-COMPLETION    VALUE '0000'.
25          88  MESSAGE-RECEIVED     VALUE '0020'.
26  01  IMS-CONTROL-AREA.
27      03  WS-AID-BYTE              PIC 99.
28      03  WS-ID                    PIC XX.
29      03  FILLER                   PIC X(8).
   ****************************************************************
   *    FORMAT REQUEST TO BE SENT TO IMS/VS                       *
   ****************************************************************
30  01  FORMAT-REQUEST.
31      03  FORMAT-COMMAND           PIC X(8) VALUE '/FORMAT'.
32      03  FILLER                   PIC X(8).
33      03  FILLER                   PIC X(8).
34      03  FILLER                   PIC X(8).
35      03  FILLER                   PIC X(20) VALUE SPACES.
36      03  DATA-LENGTH              PIC 9(4) VALUE 9.
37      03  FORMAT-NAME              PIC X(9) VALUE 'S34MOD   '.
   ****************************************************************
   *    TRANSACTION REQUEST TO SEND TO IMS/VS                     *
   ****************************************************************
38  01  TRANSACTION-REQUEST.
39      03  TRANSACTION-LENGTH       PIC S9(4) VALUE 80.
40      03  TRANSACTION-TEXT         PIC X(80).
   ****************************************************************
   *    ERROR MESSAGE FOR ABNORMAL COMPLETION                     *
   ****************************************************************
41  01  ERROR-MESSAGE.
42      03  FILLER                   PIC X(25)
            VALUE 'ABNORMAL COMPLETION CODE '.
43      03  ABEND-CODE               PIC X(4).
44      03  FILLER                   PIC X(18)
            VALUE ' AT RECORD NUMBER '.
45      03  RECORD-NUMBER            PIC 9(4) VALUE 0.
46      03  FILLER                   PIC X(26) VALUE SPACES.
   ****************************************************************
   *    PROCEDURE DIVISION BEGINS HERE                            *
   ****************************************************************
47  PROCEDURE DIVISION.
48  INITIATE-FILES.
49      OPEN I-O COMM-FILE.
50      OPEN INPUT LOCAL-TRANS-FILE.
51      READ COMM-FILE AT END GO TO IGNORE-END.
53  IGNORE-END.
54      ACQUIRE '1S' FOR COMM-FILE.
55      IF NOT NORMAL-COMPLETION GO TO DISPLAY-ERROR.
57      WRITE COMM-RECORD FROM FORMAT-REQUEST FORMAT IS '$$EVOKNI'.
58  SEND-TRANS-FILE.
59      READ LOCAL-TRANS-FILE INTO TRANSACTION-TEXT,
60          AT END GO TO END-OF-JOB.
61      ADD 1 TO RECORD-NUMBER.
62      WRITE COMM-RECORD FROM TRANSACTION-REQUEST,
            FORMAT IS '$$SENDE'.
63      IF NORMAL-COMPLETION GO TO SEND-TRANS-FILE,
65      ELSE GO TO DISPLAY-ERROR.
67  END-OF-JOB.
68      DROP '1S' FROM COMM-FILE.
69      CLOSE COMM-FILE, LOCAL-TRANS-FILE.
70      STOP RUN.
71  DISPLAY-ERROR.
72      MOVE ICF-RETURN-CODE TO ABEND-CODE.
73      DISPLAY ERROR-MESSAGE UPON CONSOLE.
74      GO TO END-OF-JOB.
```



IBM International Business Machines Corporation

RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

GX21-9092-
Printed in U.S.A.

Program | Keying Instruction | Graphic | Key | Card Electro Number
Programmer | Date

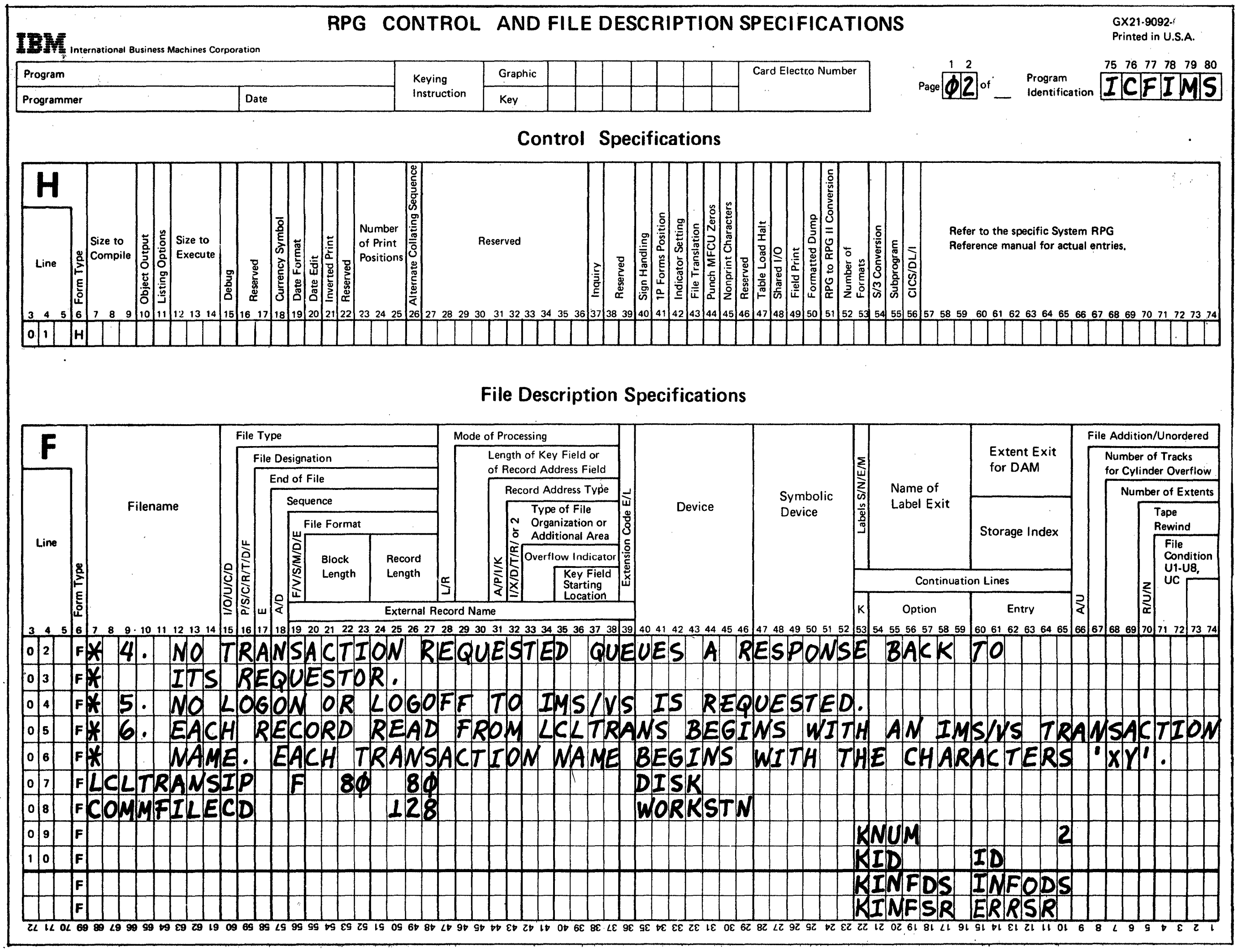
Page 01 of __ Program Identification ICFIMS

Control Specifications

```
01H       8
```

File Description Specifications

```
02F* THIS PROGRAM DOES TRANSACTION PROCESSING WITH IMS/VS.
03F* THE FOLLOWING CONDITIONS ARE ASSUMED:
04F* 1. THE SESSION OCL STATEMENT FOR COMMFILE IS
05F* // SESSION LOCATION-NEWYORK,SYMID-1S,HOSTNAME-IMS,
06F*            DATAID-XY,FLDLTH-1919
07F* 2. THE IMS/VS LOGICAL TERMINAL ASSIGNED TO THE PHYSICAL TERMINAL
08F*      DEVICE ADDRESS USED BY THIS PROGRAM IS NON-RESPONSE MODE, OR
09F*      IS TRANSACTION MODE AND ALL TRANSACTIONS REQUESTED ARE
10F*      NON-RESPONSE MODE.
  F* 3. NO TRANSACTION REQUESTED IS CONVERSATIONAL.
  F
```



IBM International Business Machines Corporation

RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

GX21-9092-
Printed in U.S.A.

Program | Keying Instruction | Graphic | Key | Card Electro Number
Programmer | Date

Page Ø2 of

Program Identification ICFIMS

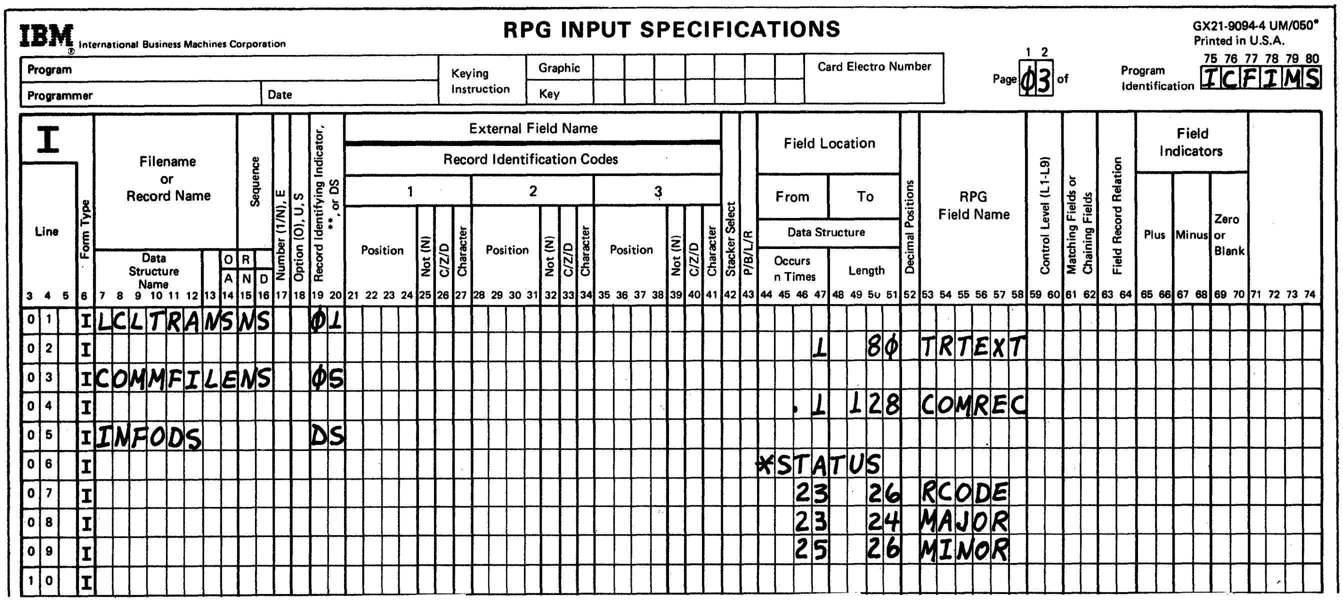
Control Specifications

```
01 H
```

Refer to the specific System RPG Reference manual for actual entries.

File Description Specifications

```
02 F* 4. NO TRANSACTION REQUESTED QUEUES A RESPONSE BACK TO
03 F*    ITS REQUESTOR.
04 F* 5. NO LOGON OR LOGOFF TO IMS/VS IS REQUESTED.
05 F* 6. EACH RECORD READ FROM LCLTRANS BEGINS WITH AN IMS/VS TRANSACTION
06 F*    NAME. EACH TRANSACTION NAME BEGINS WITH THE CHARACTERS 'XY'.
07 FLCLTRANSIP  F   8Ø  8Ø           DISK
08 FCOMMFILECD        128            WORKSTN
09 F                                              KNUM        2
10 F                                              KID    ID
   F                                              KINFDS INFODS
   F                                              KINFSR ERRSR
```



IBM International Business Machines Corporation

RPG INPUT SPECIFICATIONS

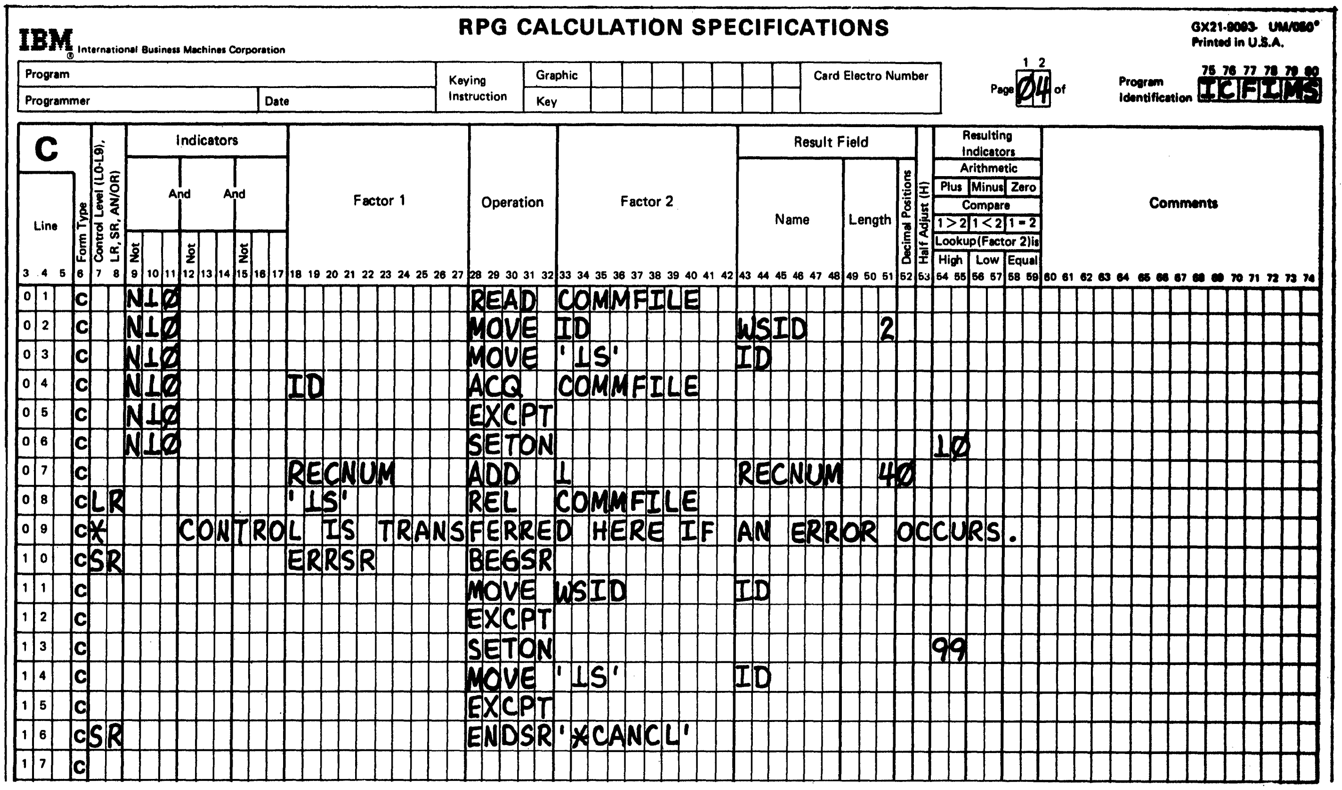
GX21-9094-4 UM/050*
Printed in U.S.A.

Program | Keying Instruction | Graphic | Key | Card Electro Number
Programmer | Date

Page Ø3 of

Program Identification ICFIMS

```
01 ILCLTRANSNS  Ø1
02 I                                        1  8Ø TRTEXT
03 ICOMMFILENS  ØS
04 I                                        1 128 COMREC
05 IINFODS      DS
06 I                                     *STATUS
07 I                                       23  26 RCODE
08 I                                       23  24 MAJOR
09 I                                       25  26 MINOR
10 I
```



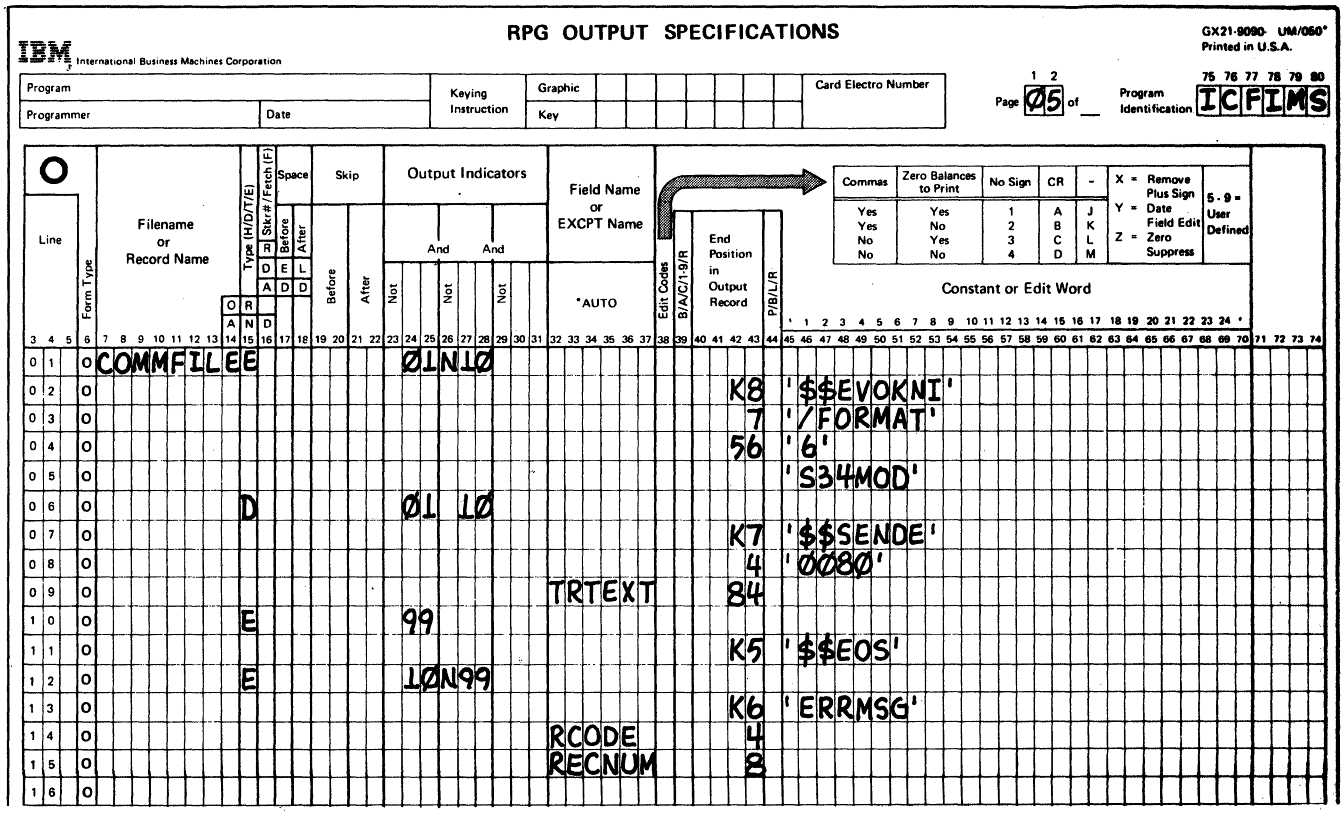
IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

GX21-9093 UM/050* Printed in U.S.A.

Page Ø4 of — Program Identification ICFIMS

```
01 C  N1Ø                READ COMMFILE
02 C  N1Ø                MOVE ID        WSID    2
03 C  N1Ø                MOVE '1S'      ID
04 C  N1Ø      ID        ACQ  COMMFILE
05 C  N1Ø                EXCPT
06 C  N1Ø                SETON                     1Ø
07 C           RECNUM    ADD  1         RECNUM  4Ø
08 CLR         '1S'      REL  COMMFILE
09 C*    CONTROL IS TRANSFERRED HERE IF AN ERROR OCCURS.
10 CSR         ERRSR     BEGSR
11 C                     MOVE WSID      ID
12 C                     EXCPT
13 C                     SETON                     99
14 C                     MOVE '1S'      ID
15 C                     EXCPT
16 CSR                   ENDSR'*CANCL'
17 C
```



IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21-9090 UM/050* Printed in U.S.A.

Page Ø5 of — Program Identification ICFIMS

```
01 OCOMMFILEE        Ø1N1Ø
02 O                                   K8 '$$EVOKNI'
03 O                                    7 '/FORMAT'
04 O                                   56 '6'
05 O                                      'S34MOD'
06 O        D        Ø1 1Ø
07 O                                   K7 '$$SENDE'
08 O                                    4 'ØØ8Ø'
09 O                         TRTEXT    84
10 O        E        99
11 O                                   K5 '$$EOS'
12 O        E        1ØN99
13 O                                   K6 'ERRMSG'
14 O                         RCODE      4
15 O                         RECNUM     8
16 O
```

CONSIDERATIONS FOR HOST SYSTEMS USING CCP

CCP Assignment Set Considerations

The following parameter requirements are imposed for System/34 program to CCP program communications using the 3270 subsystem:

- TYPE-3277M2 is mandatory on the BSCATERM statement.

- ADDRCHAR-6161xxxx,POLLCHAR-C1C1xxxx is not supported on the BSCATERM statement.

- ADDRCHAR-F0F0xxxx,POLLCHAR-5050xxxx is not supported on the BSCATERM statement.

- SWITCHED-NO is mandatory on the TERMATTR statement.

- BLKL should be the same as the System/34 line buffer size.

The following parameters eliminate the need for the CCP operator to retry command-capable terminals in ERP:

- ERTIME-n on the SYSTEM statement

- AUTOERP-YES on the BSCATERM statement

The following are examples of SYSTEM, BSCATERM, and TERMATTR statements:

```
//SYSTEM MINUPA-20K,MINTBUF-600,PASSWORD-CMS,ERTIME-6
//TERMATTR ATTRID-1,DATAFORM-MESSAGE,BLKL-2040,SWITCHED-NO
//BSCATERM TERMID-C1,TYPE-3277M2,ATTRID-1,COMMAND-YES,AUTOERP-YES,
//ADDRCHAR-*6060CICI*,POLLCHAR-*4040C1C1*
```

Startup for CCP

Following the enable of the System/34 BSC 3270 subsystem, an acquire operation must be performed by the System/34 application program for each session (device) address that is defined to CCP as command-capable. The acquire must be done before the CCP operator places the terminal on line. Following the acquire, the CCP operator may have to vary a command-capable terminal on line or retry a command-capable terminal in ERP (unless AUTOERP-YES was specified). This must be done before CCP will accept any input from that device address.

System/34 programs acquiring session (device) addresses corresponding to CCP terminals that are not command-capable should be started before the terminal is referenced by a CCP application program.

Signing On to CCP

System/34 application programs defined as command mode terminals to CCP must send the /ON command and then send the program request(s). The format of the /ON command is:

'ƀƀ/ONƀpassword

A subsequent input operation should receive a CCP message (A01 for example).

The format of a program request is:

'ƀƀprogram name

If the first two characters of the program name match the DATAID parameter, the 'ƀƀ can be omitted.

New CCP Applications

New CCP applications written to communicate with System/34 application programs should not use DFF to format the data streams. The System/34 application should specify the DATAID parameter on the SESSION OCL statement. Using maps for input and output causes increased overhead at both the host system and the System/34, and is not required.

Each output operation performed by the CCP application program should have the following format:

	ESC -	Write -	Write control -	Data ID -	Data
For example, hex	27	F1	40	C2C2	C1C2C3C4

The System/34 application program should check the return code associated with each input operation to differentiate application program data from messages sent by CCP. Input received from the CCP application program will consist of the data ID followed by data, whereas messages from CCP will consist solely of the message text.

Each output operation performed by the System/34 application program should consist of data ID followed by data. The corresponding input operation for the CCP application program would receive the message in the following format:

	cu -	dv -	'ƀƀ -	Data ID -	Data
For example, hex	40	C1	7D4040	C2C2	C1C2C3C4

where 'ƀƀ is the AID byte followed by the cursor address.

CCP Applications That Use DFF

Each output operation performed by the CCP application program should have the following format:

Format name-Data ID-Data

where:

format name is the name of the DFF format.

The input and output operations performed by the System/34 application program should be handled as in the previous section on new CCP programs.

Input operations performed by the CCP application program should expect to receive messages from the System/34 application program in the following format:

'-Data ID-Data

where:

' is the AID code for the Enter key.

The DFF format used to receive messages from the System/34 application program must be defined as follows:

- All input-capable fields must precede the first output only field.
- All input-capable fields must be the same length.
- The length of the input fields must be the same as the value of the FLDLTH parameter on the System/34 SESSION OCL statement.
- The first input-capable field must begin at row 1, column 2.
- All input-capable fields must be contiguous, separated by only an attribute byte.

The DFF format used to send messages to the System/34 application program should not define fields with the modified data tag (in the field attribute byte) set on. The copy operation will be rejected by the System/34 3270 subsystem. The erase operation performs no function at the System/34.

Programming Considerations for CCP Application Programs

The -14 return code (RPG 1M) may occur intermittently while sending messages to a System/34 application program or to System/34 display stations signed on to the 3270 Device Emulation Program Product. This message indicates that a previous message is queued for the System/34 application program. The CCP application program should resend the message after an optional wait.

Coding Example

When the System/34 program is started, it transmits a data file to the host system, one record at a time. The programs operate in a transaction processing environment; that is, the host transaction is started, performs a single input operation, and terminates.

The first function of the System/34 program is to acquire a session (IS) with the 3270 subsystem. Next, a put then invite operation ($$SEND) is issued to send the /ON command to CCP. An input operation is performed to obtain the response message. Then, a data record is read from the System/34 file. A put then invite operation ($$SEND) is issued to send the data record. The first six characters of data are the program name; the name is followed by the data record. An input operation is then required to obtain the CCP application program termination message (assuming the CCP application program defined ENDMSG-YES). The process of reading and sending a record continues until the entire file is sent. The session is then released and the System/34 program terminates.

```
 1  IDENTIFICATION DIVISION.
 2  PROGRAM-ID.  ICFCCP.
 3  SECURITY.  THIS PROGRAM DOES TRANSACTION PROCESSING WITH CCP.
        THE FOLLOWING CONDITIONS ARE ASSUMED:
        1.  THE SESSION OCL STATEMENT FOR COMM-FILE IS
            // SESSION LOCATION-NEWYORK,SYMID-1S,HOSTNAME-OTHER,
               DATAID-XY,FLDLTH-80
        2.  THE CCP LOGICAL TERMINAL ASSIGNED TO THIS PHYSICAL
            TERMINAL (DEVICE) ADDRESS USED BY THIS PROGRAM IS
            A COMMAND-CAPABLE TERMINAL.
        3. THE ACQUIRE OPERATION IS EXECUTED BEFORE CCP IS STARTED
            OR BEFORE THE CCP TERMINAL IS ACTIVATED.
        4.  NO TRANSACTION REQUESTED SENDS A RESPONSE BACK TO
            ITS REQUESTOR.
        5.  EACH RECORD READ FROM LOCAL-TRANS-FILE BEGINS WITH
            A CCP TRANSACTION NAME.  EACH TRANSACTION NAME
            BEGINS WITH THE CHARACTERS "XY".
 4  ENVIRONMENT DIVISION.
 5  CONFIGURATION SECTION.
 6  SOURCE-COMPUTER.  IBM-S34.
 7  OBJECT-COMPUTER.  IBM-S34.
 8  SPECIAL-NAMES.
 9      SYSTEM-CONSOLE IS CONSOLE.
10  INPUT-OUTPUT SECTION.
11  FILE-CONTROL.
12      SELECT COMM-FILE ASSIGN TO WORKSTATION-FORMAT-01,
            ORGANIZATION IS TRANSACTION,
            FILE STATUS IS WS-RETURN-CODE, ICF-RETURN-CODE,
            CONTROL-AREA IS CCP-CONTROL-AREA.
13      SELECT LOCAL-TRANS-FILE ASSIGN TO DISK-CCPTRANS,
            ORGANIZATION IS SEQUENTIAL,
            ACCESS MODE IS SEQUENTIAL.
14  DATA DIVISION.
15  FILE SECTION.
16  FD  COMM-FILE LABEL RECORDS ARE OMITTED.
17  01  COMM-RECORD                     PIC X(128).
18  FD  LOCAL-TRANS-FILE LABEL RECORDS ARE STANDARD.
19  01  CCP-TRANSACTION                 PIC X(80).
20  WORKING-STORAGE SECTION.
```

```
****************************************************************
*   STATUS BYTES FOR CCP SESSION I/O                           *
****************************************************************
01  CCP-RETURN-CODE.
    03  WS-RETURN-CODE            PIC X(2).
    03  ICF-RETURN-CODE           PIC X(4).
        88  NORMAL-COMPLETION     VALUE '0000'.
        88  MESSAGE-RECEIVED      VALUE '0020'.
01  CCP-CONTROL-AREA.
    03  WS-AID-BYTE               PIC 99.
    03  WS-ID                     PIC XX.
    03  FILLER                    PIC X(8).
****************************************************************
*   SIGNON REQUEST TO BE SENT TO CCP                           *
****************************************************************
01  SIGNON-REQUEST.
    03  SIGNON-LENGTH             PIC S9(4) VALUE 16.
    03  SIGNON-AID                PIC X VALUE QUOTE.
    03  SIGNON-CURSOR             PIC XX VALUE '  '.
    03  SIGNON-TEXT               PIC X(13) VALUE '/ON PASSWORD'.
****************************************************************
*   TRANSACTION REQUEST TO SEND TO CCP                         *
****************************************************************
01  TRANSACTION-REQUEST.
    03  TRANSACTION-LENGTH        PIC S9(4) VALUE 80.
    03  TRANSACTION-TEXT          PIC X(80).
****************************************************************
*   MESSAGE BUFFER TO RECEIVE CCP MESSAGES                     *
****************************************************************
01  CCP-MESSAGE.
    03  CCP-MESSAGE-NUMBER        PIC X(3).
        88  CCP-SIGNON-COMPLETE   VALUE 'A01'.
        88  CCP-TRANS-COMPLETE    VALUE 'S03'.
    03  CCP-MESSAGE-TEXT          PIC X(23).
    03  FILLER                    PIC X(54).
****************************************************************
*   ERROR MESSAGE FOR ABNORMAL ICF COMPLETION                  *
****************************************************************
01  ERROR-MESSAGE1.
    03  FILLER                    PIC X(25)
        VALUE 'ABNORMAL COMPLETION CODE '.
    03  ABEND-CODE                PIC X(4).
    03  FILLER                    PIC X(18)
        VALUE ' AT RECORD NUMBER '.
    03  RECORD-NUMBER             PIC 9(4) VALUE 0.
    03  FILLER                    PIC X(26) VALUE SPACES.
****************************************************************
*   ERROR MESSAGE FOR ABNORMAL CCP COMPLETION                  *
****************************************************************
01  ERROR-MESSAGE2.
    03  FILLER                    PIC X(23)
        VALUE 'UNEXPECTED CCP MESSAGE'.
    03  MESSAGE-CODE              PIC X(3).
    03  FILLER                    PIC X(18)
        VALUE ' AT RECORD NUMBER '.
    03  RECORD-NUMBER             PIC 9(4) VALUE 0.
    03  FILLER                    PIC X(4) VALUE ' -- '.
    03  MESSAGE-TEXT              PIC X(23).
****************************************************************
*   PROCEDURE DIVISION BEGINS HERE                             *
****************************************************************
PROCEDURE DIVISION.
```

```
INITIATE-FILES.
    OPEN I-O COMM-FILE.
    OPEN INPUT LOCAL-TRANS-FILE.
    READ COMM-FILE AT END GO TO IGNORE-END.
IGNORE-END.
    ACQUIRE '1S' FOR COMM-FILE.
    IF NOT NORMAL-COMPLETION GO TO DISPLAY-ERROR.
    WRITE COMM-RECORD FROM SIGNON-REQUEST FORMAT IS '$$SEND'.
    READ COMM-FILE INTO CCP-MESSAGE TERMINAL IS '1S'.
    IF NOT MESSAGE-RECEIVED GO TO DISPLAY-ERROR.
    IF NOT CCP-SIGNON-COMPLETE GO TO DISPLAY-MESSAGE.
SEND-TRANS-FILE.
    READ LOCAL-TRANS-FILE INTO TRANSACTION-TEXT,
        AT END GO TO END-OF-JOB.
    ADD 1 TO RECORD-NUMBER IN ERROR-MESSAGE1.
    ADD 1 TO RECORD-NUMBER IN ERROR-MESSAGE2.
    WRITE COMM-RECORD FROM TRANSACTION-REQUEST,
        FORMAT IS '$$SEND'.
    READ COMM-FILE INTO CCP-MESSAGE TERMINAL IS '1S'.
    IF MESSAGE-RECEIVED
        IF CCP-TRANS-COMPLETE GO TO SEND-TRANS-FILE,
        ELSE GO TO DISPLAY-MESSAGE,
    ELSE GO TO DISPLAY-ERROR.
END-OF-JOB.
    DROP '1S' FROM COMM-FILE.
    CLOSE COMM-FILE, LOCAL-TRANS-FILE.
    STOP RUN.
DISPLAY-ERROR.
    MOVE ICF-RETURN-CODE TO ABEND-CODE.
    DISPLAY ERROR-MESSAGE1 UPON CONSOLE.
    GO TO END-OF-JOB.
DISPLAY-MESSAGE.
    MOVE CCP-MESSAGE-NUMBER TO MESSAGE-CODE.
    MOVE CCP-MESSAGE-TEXT TO MESSAGE-TEXT.
    DISPLAY ERROR-MESSAGE2 UPON CONSOLE.
    GO TO END-OF-JOB.
```

IBM International Business Machines Corporation

RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

GX21-9092-
Printed in U.S.A.

Program		Keying Instruction	Graphic		Card Electro Number
Programmer	Date		Key		

Page (1 2) Ø1 of __ Program Identification (75 76 77 78 79 80) ICFCCP

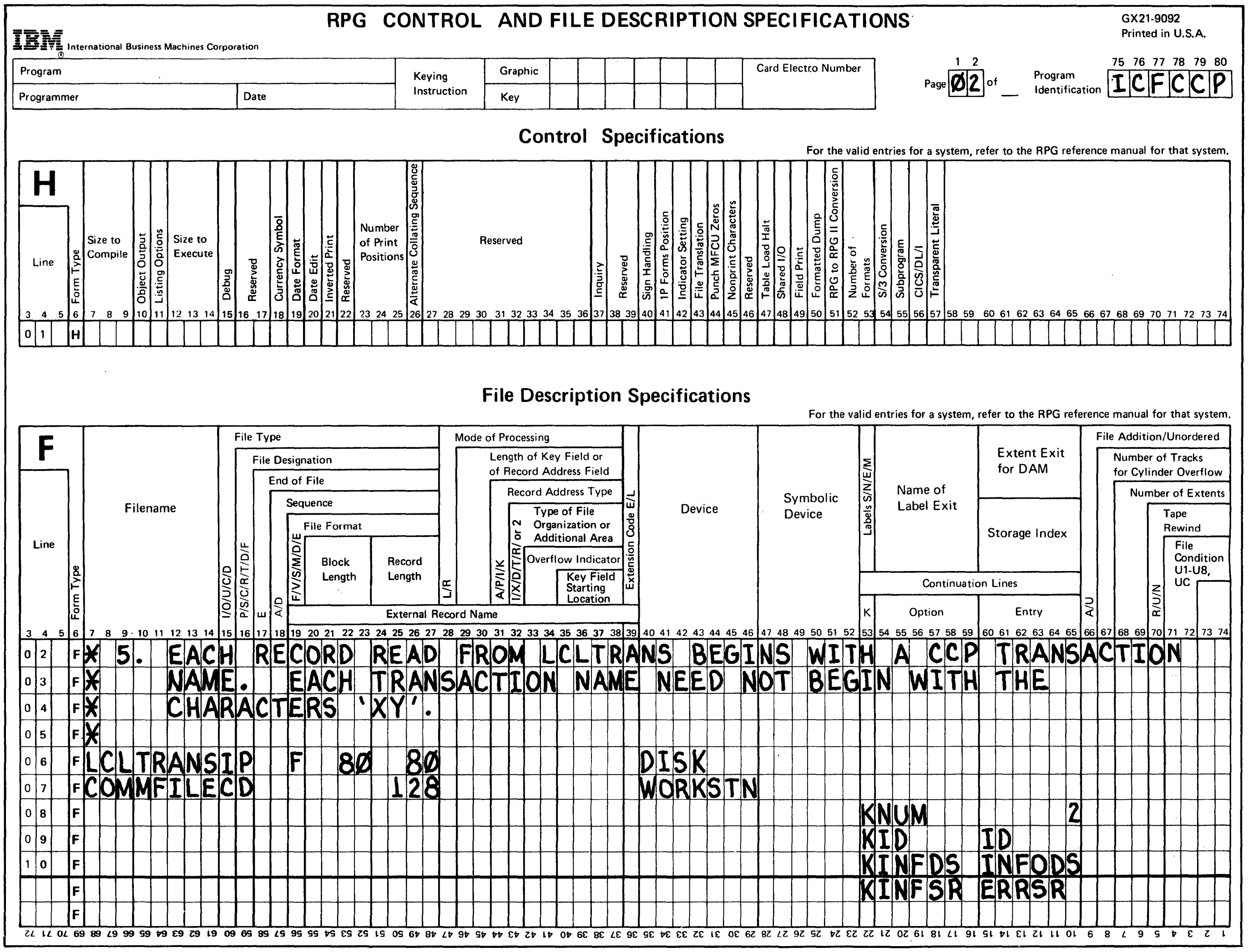
Control Specifications

Line (3-5)	Form Type (6)	Size to Compile (7-9)	Object Output (10)	Listing Options (11)	Size to Execute (12-14)	Debug (15)	Reserved (16-17)	Currency Symbol (18)	Date Format (19)	Date Edit (20)	Inverted Print (21)	Reserved (22)	Number of Print Positions (23-25)	Alternate Collating Sequence (26)	Reserved (27-36)	Inquiry (37)	Reserved (38-39)	Sign Handling (40)	1P Forms Position (41)	Indicator Setting (42)	File Translation (43)	Punch MFCU Zeros (44)	Nonprint Characters (45)	Reserved (46)	Table Load Halt (47)	Shared I/O (48)	Field Print (49)	Formatted Dump (50)	RPG to RPG II Conversion (51)	Number of Formats (52-53)	S/3 Conversion (54)	Subprogram (55)	CICS/DL/I (56)	Refer to the specific System RPG Reference manual for actual entries. (57-74)
01	H				8																													

File Description Specifications

Line (3-5) | Form Type (6) | Filename (7-14) | File Type: I/O/U/C/D (15) | File Designation: P/S/C/R/T/D/F (16) | End of File: E (17) | Sequence: A/D (18) | File Format: F/V/S/M/D/E (19) | Block Length (20-23) | Record Length (24-27) | Mode of Processing: L/R (28) | Length of Key Field or of Record Address Field (29-30) | Record Address Type: A/P/I/K (31) | Type of File Organization or Additional Area: I/X/D/T/R/ or 2 (32) | Overflow Indicator (33-34) | Key Field Starting Location (35-38) | External Record Name (19-28) | Extension Code E/L (39) | Device (40-46) | Symbolic Device (47-52) | Labels S/N/E/M (53) | Name of Label Exit (54-59) | Extent Exit for DAM (60-65) | Storage Index (60-65) | Continuation Lines: K (53), Option (54-59), Entry (60-65) | A/U (66) | File Addition/Unordered (66) | Number of Tracks for Cylinder Overflow (67-68) | Number of Extents (69) | Tape Rewind: R/U/N (70) | File Condition U1-U8, UC (71-72) (73 74)

```
02   F* THIS PROGRAM DOES TRANSACTION PROCESSING WITH CCP.
03   F* THE FOLLOWING CONDITIONS ARE ASSUMED:
04   F* 1. THE SESSION OCL STATEMENT FOR COMMFILE IS
05   F*    // SESSION LOCATION-NEWYORK,SYMID-1S,HOSTNAME-OTHER,
06   F*       DATAID-XY,FLDLTH-8Ø
07   F* 2. THE CCP TERMINAL ASSIGNED TO THE PHYSICAL TERMINAL (DEVICE)
08   F*    ADDRESS USED BY THIS PROGRAM IS COMMAND CAPABLE.
09   F* 3. THE ACQUIRE OPERATION IS ISSUED BEFORE CCP IS STARTED, OR
10   F*    BEFORE THE CCP TERMINAL IS ACTIVATED.
     F* 4. NO TRANSACTION REQUESTED RESPONDS TO THE REQUESTER.
     F
```



IBM International Business Machines Corporation

RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

GX21-9092
Printed in U.S.A.

Page Ø2 of __ Program Identification ICFCCP

Control Specifications

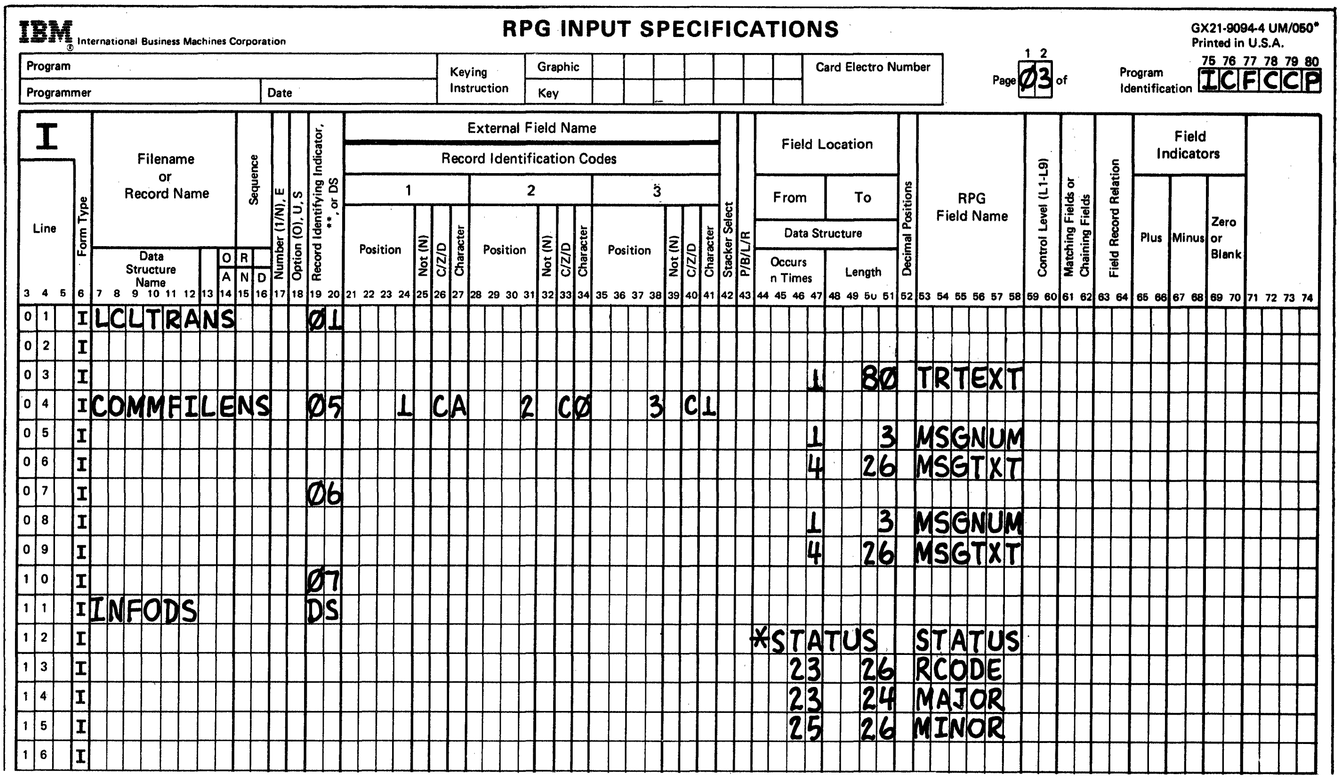
For the valid entries for a system, refer to the RPG reference manual for that system.

```
01 H
```

File Description Specifications

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

```
02 F* 5. EACH RECORD READ FROM LCLTRANS BEGINS WITH A CCP TRANSACTION
03 F*    NAME.  EACH TRANSACTION NAME NEED NOT BEGIN WITH THE
04 F*    CHARACTERS 'XY'.
05 F*
06 FLCLTRANSIP  F   8Ø  8Ø           DISK
07 FCOMMFILECD         128           WORKSTN
08 F                                              KNUM        2
09 F                                              KID    ID
10 F                                              KINFDS INFODS
   F                                              KINFSR ERRSR
   F
```



IBM International Business Machines Corporation

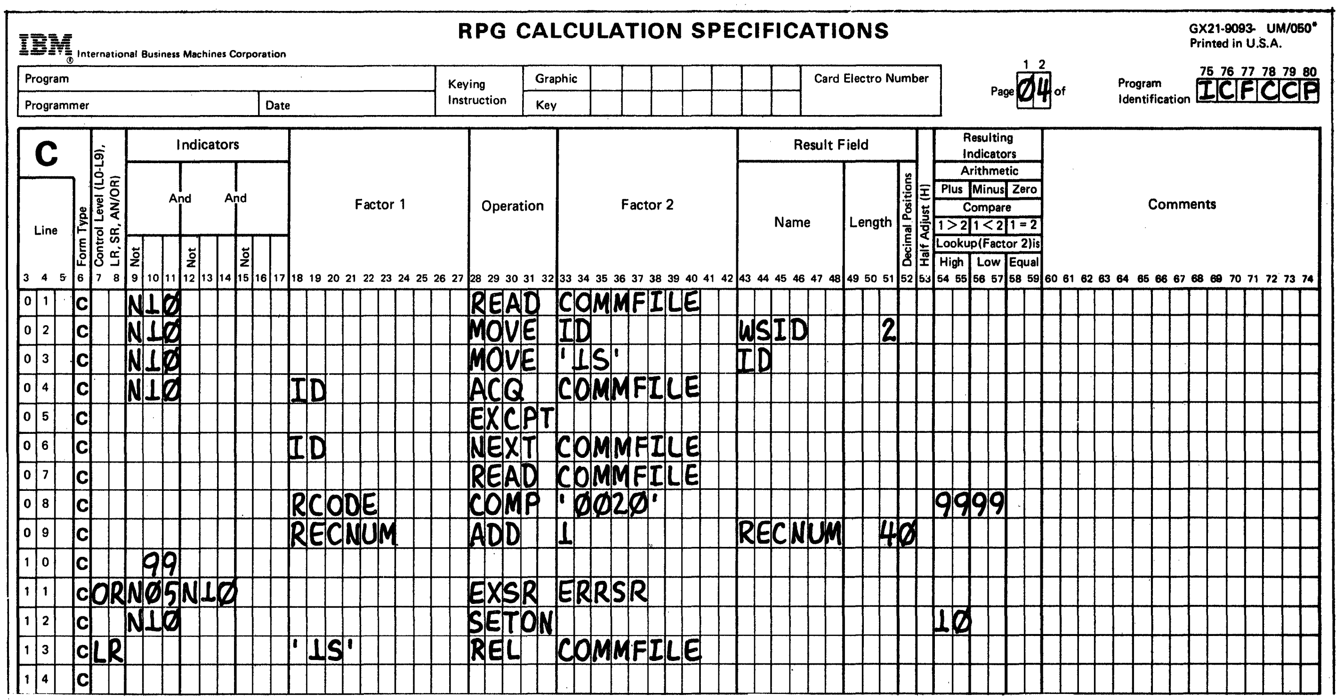
RPG INPUT SPECIFICATIONS

GX21-9094-4 UM/050*
Printed in U.S.A.

Program | Programmer | Date | Keying Instruction: Graphic, Key | Card Electro Number

Page Ø3 of | Program Identification ICFCCP

Line	Form Type	Filename or Record Name	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, **, or DS	1 Position	Not (N)	C/Z/D	Character	2 Position	Not (N)	C/Z/D	Character	3 Position	Not (N)	C/Z/D	Character	Stacker Select	P/B/L/R	Field Location From	To	Decimal Positions	RPG Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Minus	Zero or Blank
01	I	LCLTRANS				Ø1																								
02	I																													
03	I																				1	8Ø		TRTEXT						
04	I	COMMFILE	NS			Ø5	1		C	A	2		C	Ø	3		C	1												
05	I																				1	3		MSGNUM						
06	I																				4	26		MSGTXT						
07	I					Ø6																								
08	I																				1	3		MSGNUM						
09	I																				4	26		MSGTXT						
10	I					Ø7																								
11	I	INFODS				DS																								
12	I																				*STATUS			STATUS						
13	I																				23	26		RCODE						
14	I																				23	24		MAJOR						
15	I																				25	26		MINOR						
16	I																													



IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

GX21-9093- UM/050*
Printed in U.S.A.

Program | Programmer | Date | Keying Instruction: Graphic, Key | Card Electro Number

Page Ø4 of | Program Identification ICFCCP

Line	Form Type	Control Level (LØ-L9), LR, SR, AN/OR	Not	Indicator	Not	Indicator	Not	Indicator	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Plus / 1>2 / High	Minus / 1<2 / Low	Zero / 1=2 / Equal	Comments
01	C		N	1Ø						READ	COMMFILE								
02	C		N	1Ø						MOVE	ID	WSID	2						
03	C		N	1Ø						MOVE	'1S'	ID							
04	C		N	1Ø					ID	ACQ	COMMFILE								
05	C									EXCPT									
06	C								ID	NEXT	COMMFILE								
07	C									READ	COMMFILE								
08	C								RCODE	COMP	'ØØ2Ø'					99	99		
09	C								RECNUM	ADD	1	RECNUM	4	Ø					
10	C			99															
11	C	OR	N	Ø5	N	1Ø				EXSR	ERRSR								
12	C		N	1Ø						SETON						1Ø			
13	C	LR							'1S'	REL	COMMFILE								
14	C																		

IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

GX21-9093- UM/050*
Printed in U.S.A.

Program | Programmer | Date | Keying Instruction | Graphic | Key | Card Electro Number

Page Ø5 of

Program Identification ICFCCP

```
01 C*    CONTROL IS TRANSFERRED HERE IF AN ERROR OCCURS.
02 CSR         ERRSR     BEGSR
03 C  N1Ø      RECNUM    SUB  1         RECNUM
04 C                     SETON                     98
05 C                     MOVE WSID      ID
06 C                     EXCPT
07 C                     SETON                     97
08 C                     MOVE '1S'      ID
09 C                     EXCPT
10 CSR                   ENDSP'*CANCL'
11 C
```

IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21-9090-4 UM/050*
Printed in U.S.A.

Program | Programmer | Date | Keying Instruction | Graphic | Key | Card Electro Number

Page Ø6 of

Program Identification ICFCCP

```
01 OCOMMFILEE        Ø1N1Ø
02 O                                   K6 '$$SEND'
03 O                                    4 '16'
04 O                                   2Ø '''  /ON PASSWORD'
05 O        E        Ø1 1Ø
06 O                                   K6 '$$SEND'
07 O                                    4 '8Ø'
08 O                         TRTEXT    84
09 O        E        97
10 O                                   K5 '$$EOS'
11 O        E        98
12 O                                   K6 'ERRMSG'
13 O                 99                24 'ABNORMAL COMPLETION CODE'
14 O                N99                24 'UNEXPECTED CCP MESSAGE  '
15 O                 99      RCODE     28
16 O                N99      MSGNUM    32
17 O                         RECNUM    32
18 O
```

CONSIDERATIONS FOR HOST SYSTEMS USING CICS/VS

Newly Developed CICS Applications

New CICS/VS applications written to communicate with the BSC 3270 subsystem should not use BMS maps to format the data streams. Using maps for input and output causes increased overhead at both the host system and the System/34, and is not required.

If the CICS command level interface is being used, the command ISSUE COPY should not be used. A command reject will result, taking the unit out of service. In addition, ISSUE PRINT and ISSUE ERASEAUP should not be used. These operations perform no function at the System/34, unless the session is acquired with no DATAID characters. In this case, the System/34 application program is expected to process these commands.

The SESSION OCL statement should include the DATAID parameter. As a result, the System/34 application program, after issuing an input request, receives only the data characters, including the two DATAID characters. The subsystem removes any 3270 command or order sequences.

Conversely, System/34 put operations should be issued with the first two characters matching the DATAID characters for the session. As a result, all required 3270 protocol is inserted by the 3270 subsystem. When the CICS application issues an input request, it receives the data, including the DATAID characters. The aid byte is always reported as the Enter key, and the cursor address is always row 1, column 1.

CICS transactions should be initiated via evoke operation requests by the System/34 application program. This requires the HOSTNAME-CICS parameter on the SESSION OCL. The DATAID parameter and FLDLTH-0 must also be specified on the SESSION statement.

This method provides maximum isolation from 3270 protocol dependencies. However, if a system message or other data stream is received that does not match the DATAID characters, the System/34 application program receives a minor return code of 20 on the input operation. Such messages and data must be processed by the System/34 application program. Broadcast messages and other routed messages can be prevented during CICS generation. See *CICS System Generation Considerations* later in this chapter for details.

Existing CICS Applications That Do Not Use BMS Maps

When possible, DATAID characters should be chosen to use the output formatting function of the BSC 3270 subsystem. Because it is not always possible to define DATAID characters that match the data stream sent to the host, the System/34 application might not be completely isolated from 3270 device protocol. When DATAID characters do not evoke output data stream formatting, a data format such as the following example must be generated by the System/34 application program:

Aid Byte	**Cursor Address**	**Data**
hex 7D (enter key)	hex 4040 (row 1, column 1)	ABCD (data characters)

When DATAID characters are not specified, a data format such as the following example must be expected:

Escape	**Command**	**Data**
hex 27	hex F14B (write-write control)	ABCD (data characters)

CICS applications can be initiated using evoke operation requests. This requires the HOSTNAME-CICS, FLDLTH-0, and DATAID parameters on the SESSION OCL statement. The DATAID characters do not have to match the data to have the host output data stream formatted by the subsystem.

CICS Applications That Use BMS Maps

The BSC 3270 subsystem provides the System/34 application with a limited data stream formatting function. To use this function on output, the DATAID characters must be specified and must match the first 2 data bytes in an output request. If the CICS application program requires a data stream format other than that generated by the subsystem, the System/34 application must construct the format. The formatting provided by the subsystem on System/34 application output requests is described earlier in this chapter. The formatting function provided by the 3270 subsystem on input request is more general and can be used to accept a wider variety of host system application data maps than it can respond to. One way to take advantage of the asymmetry in the formatting function provided is to do transaction processing.

For example, the following application flow can be used:

1. An evoke operation initiates a transaction at the host system. All input data is supplied with the evoke data.

2. The host application performs the requested processing and issues a response to the System/34 program using a previously prepared output map.

3. The host system transaction terminates.

4. The data sent to the System/34 application evokes the formatting function of the subsystem.

5. The System/34 application receives only the data (including field headings, if present).

6. The process can be repeated.

Simple inquiry programs could be developed in this fashion, provided the data fields can be located by the System/34 application program.

An alternative method for interfacing to a CICS application that uses maps is to provide all 3270 data stream formatting within the System/34 application. This method is accomplished by not specifying the DATAID characters on the SESSION OCL statement. The formatting function and evoke operation support cannot be used for the session. While this method of operation places no restrictions on the data stream, it requires a thorough understanding of 3270 protocols and data streams and should be used carefully. (See *3270 Information Display System Components Description Manual* for more information on 3270 protocols.)

If the formatting support provided by the 3270 subsystem is to be used to interface to CICS applications using maps, the map sets used by the CICS application must meet the following criteria:

Input maps:

- All input-capable (ATTRB=UNPROT) fields must precede the first output only field.

- All input fields must be the same length.

- The length of each field must equal the FLDLTH parameter of the session.

- The first field must begin in row 1, column 1.

- All fields must be contiguous, separated only by a single space between fields on a 3277 display.

Output maps:

- No output field can be written with the modified data tag on, because the response to a poll might be unpredictable (ATTRB≠FSET).

- No tab support is provided; therefore, tabs should not be used.

Input/output maps must meet criteria for both input and output maps.

The following is an example of a valid map set definition for use with 3270 subsystem:

```
PIMO1LDFHMSDTYPE=MAP,LANG=COBOL,
            MODE=INOUT,
            STORAGE=AUTO,TERM=ALL
PIMAP1A  DFHMDI COLUMN=1,CTRL=(PRINT,L80),DATA=BLOCK,LINE=1,SIZE=(24,80)
PIMAPF1  DFHMDF  POS=(1,1),ATTRB=(UNPROT,IC),LENGTH=79
PIMAPF2  DFHMDF  POS=(2,1),ATTRB=(UNPROT),LENGTH=79
PIMAPF3  DFHMDF  POS=(3,1),ATTRB=(UNPROT),LENGTH=79
PIMAPF4  DFHMDF  POS=(4,1),ATTRB=(UNPROT),LENGTH=79
PIMAPF5  DFHMDF  POS=(5,1),ATTRB=(UNPROT),LENGTH=79 (field length=79)
PIMAPF6  DFHMDF  POS=(6,1),ATTRB=(UNPROT),LENGTH=79
PIMAPF7  DFHMDF  POS=(7,1),ATTRB=(UNPROT),LENGTH=79
PIMAP1B  DFHMDI  COLUMN=1,CTRL=(PRINT,L40),DATA=BLOCK,LINE=1,SIZE=(24,40)
PIMBF01  DFHMDF  POS=(1,1),ATTRB=(UNPROT),LENGTH=39
PIMBF02  DFHMDF  POS=(1,41),ATTRB=(UNPROT),LENGTH=39
PIMBF03  DFHMDF  POS-(2,1),ATTRB=(UNPROT),LENGTH=39
PIMBF04  DFHMDF  POS=(2,41),ATTRB=(UNPROT),LENGTH=39
PIMBF05  DFHMDF  POS=(3,1),ATTRB=(UNPROT),LENGTH=39
PIMBF06  DFHMDF  POS=(3,41),ATTRB=(UNPROT),LENGTH=39
PIMBF07  DFHMDF  POS=(4,1),ATTRB=(UNPROT),LENGTH=39
PIMBF08  DFHMDF  POS=(4,41),ATTRB=(UNPROT),LENGTH=39 (field length=39)
PIMBF09  DFHMDF  POS=(5,1),ATTRB=(UNPROT),LENGTH=39
PIMBF10  DFHMDF  POS=(5,41),ATTRB=(UNPROT),LENGTH=39
PIMBF11  DFHMDF  POS=(6,1),ATTRB=(UNPROT),LENGTH=39
PIMBF12  DFHMDF  POS=(6,41),ATTRB=(UNPROT),LENGTH=39
PIMBF13  DFHMDF  POS=(7,1),ATTRB=(UNPROT),LENGTH=39
PIMBF14  DFHMDF  POS=(7,41),ATTRB=(UNPROT),LENGTH=39
PIMBF15  DFHMDF  POS=(8,1),ATTRB=(UNPROT),LENGTH=39
PIMBF16  DFHMDF  POS=(8,41),ATTRB=(UNPROT),LENGTH=39
         DFHMSD TYPE=FINAL
         END
```

The above map set example illustrates multiple fields. If a map set is to be defined solely for use with the 3270 subsystem, however, a better alternative is only one field, starting at row 1, column 1, for each map. The length of the field is determined by the length of the data record to be sent or received.

General Program Flow

A typical System/34 application interfacing to a CICS transaction would follow this pattern:

1. Acquire a session with the following SESSION OCL statement:

   a. // SESSION (Non-BMS CICS application using BSC 3270 subsystem formatting function)
      - FLDLTH-0
      - DATAID-user specified
      - HOST-CICS

   b. // SESSION (BMS application using BSC 3270 subsystem input/output formatting function)
      - FLDLTH-user specified
      - DATAID-user specified
      - HOST-CICS

   c. // SESSION (BMS application using map sets that do not meet the BSC 3270 subsystem requirements for subsystem support)
      - FLDLTH not specified
      - DATAID not specified
      - HOST not specified

   In addition, the SYMID and LOCATION parameters are required in all cases, and a device address can optionally be used.

2. Start a transaction:

   If a or b, issue an evoke request with all necessary transaction qualification and data in the output buffer. The transaction name is supplied to the 3270 subsystem via the evoke parameter list.

   If c, format an output data stream corresponding to that generated by an actual 3277 when evoking a transaction.

   For example (assume that CICS expects input from a cleared screen):

   Aid-Cursor address-Transaction ID-space-Data

3. Issue input and output requests:

   If a or b, on output, place DATAID characters in first 2 bytes of output buffer. On input, expect DATAID characters as first 2 bytes in input buffer.

   If c, on output, format the data stream to meet the input map being used by the CICS application.

   For example (assume that the output length is 50 bytes and that the input map is PIMAP1B in the sample map set provided):

   > Aid-Cursor address-Set buffer address sequence-39 bytes of data-set buffer address sequence-11 bytes of data

   If C, on input, expect the data stream generated by the CICS application output map.

   For example, assume that the output map used by the CICS application is PIMAP1B in the sample map set provided and the output length is 50, and the operation does not send data only or map only. The data stream would be:

   > ESC-Command Sequence-Set buffer address sequence-Insert cursor-Start field sequence-39 bytes of data-Set buffer address sequence-Start field sequence-11 bytes of data

4. Release the session:

   When processing is finished, the session should be released, making the device address available for other programs. The session is automatically released when the System/34 program terminates. However, a release operation does not ensure the termination of an active CICS transaction associated with the session being released. The System/34 application must synchronize termination with the CICS application. This function is not performed by the 3270 subsystem. The subsequent use of a session address that remains logically connected to a CICS transaction could produce unpredictable results.

CICS System Generation Considerations

Certain considerations during CICS configuration can make the 3270 subsystem easier to use. These include terminal control table considerations and the following stage 1 consideration:

> The PUNSOL=NO parameter can be specified on the BFUSG parameter to ensure that unsolicited input messages are not purged. These messages include those generated by a 3277 display when the Reset key is pressed followed by the Enter key. When a System/34 program does two consecutive output operations, a sequence occurs that is analogous to pressing Reset and entering more data. File transfer to the host is an example of an application that is likely to issue consecutive output requests. In any case, this parameter might affect the operational requirements of a System/34 application.

CICS Table Considerations

The following program control table option can be used to cause a time-out on a transaction that hangs because a System/34 program terminates without satisfying a host system transaction input request:

```
DFHPCT   TYPE=ENTRY
           .
           .
           .
           .
         RTIMOUT=mmss
```

The following terminal control table can be used to prevent the delivery of broadcast messages to terminal addresses used for System/34 application programs:

```
DFHTCT TYPE=TERMINAL,BMSFEAT=NOROUTEALL
```

In addition, CONNECT=AUTO issues a VTAM SIMLOGON, which causes a connection to be established to VTAM when the terminal is brought into service. This makes it unnecessary for the S/34 application to issue a VTAM logon after acquiring a session.

The FEATURE=COPY parameter should not be included because the copy command is not supported by the 3270 subsystem.

Specifying PGESTAT=AUTOPAGE eliminates the need to issue paging commands to retrieve paged data.

Use TRMSTAT=TRANSACTION, so no routed messages are delivered without being requested. This is recommended for sessions connected to a System/34 application program that is not designed to process messages but does allow input and output processing. TRMSTAT=TRANSCEIVE is the same as TRANSACTION, but messages are sent automatically.

TRMSTAT=TERMINAL receives messages, but cannot send input. The TRMSTAT parameter should be selected to tailor the environment to the applications that will be used.

Using VF=NO and HF=NO prevents program tab requests from being sent. The 3270 subsystem does not support program tab requests.

The values selected for these parameters should be considered carefully. They can be used to tailor the host system environment that a System/34 program must interface with. For more information on CICS system generation, see *CICS/VS System Programmer's Reference Manual.*

Startup for CICS/VS

Following the enabling of the System/34 3270 subsystem, the host system CICS/VS operator may have to:

- Run the CSMT transaction to place the line and/or cluster corresponding to the System/34 in service. This applies only to non-VTAM terminals.

- Run the CSMT transaction to place the terminals associated with the System/34 in service. The CICS/VS terminals that are associated with System/34 device addresses reserved for programs should be placed in service before a System/34 program uses the session (device) address.

Sample Programs

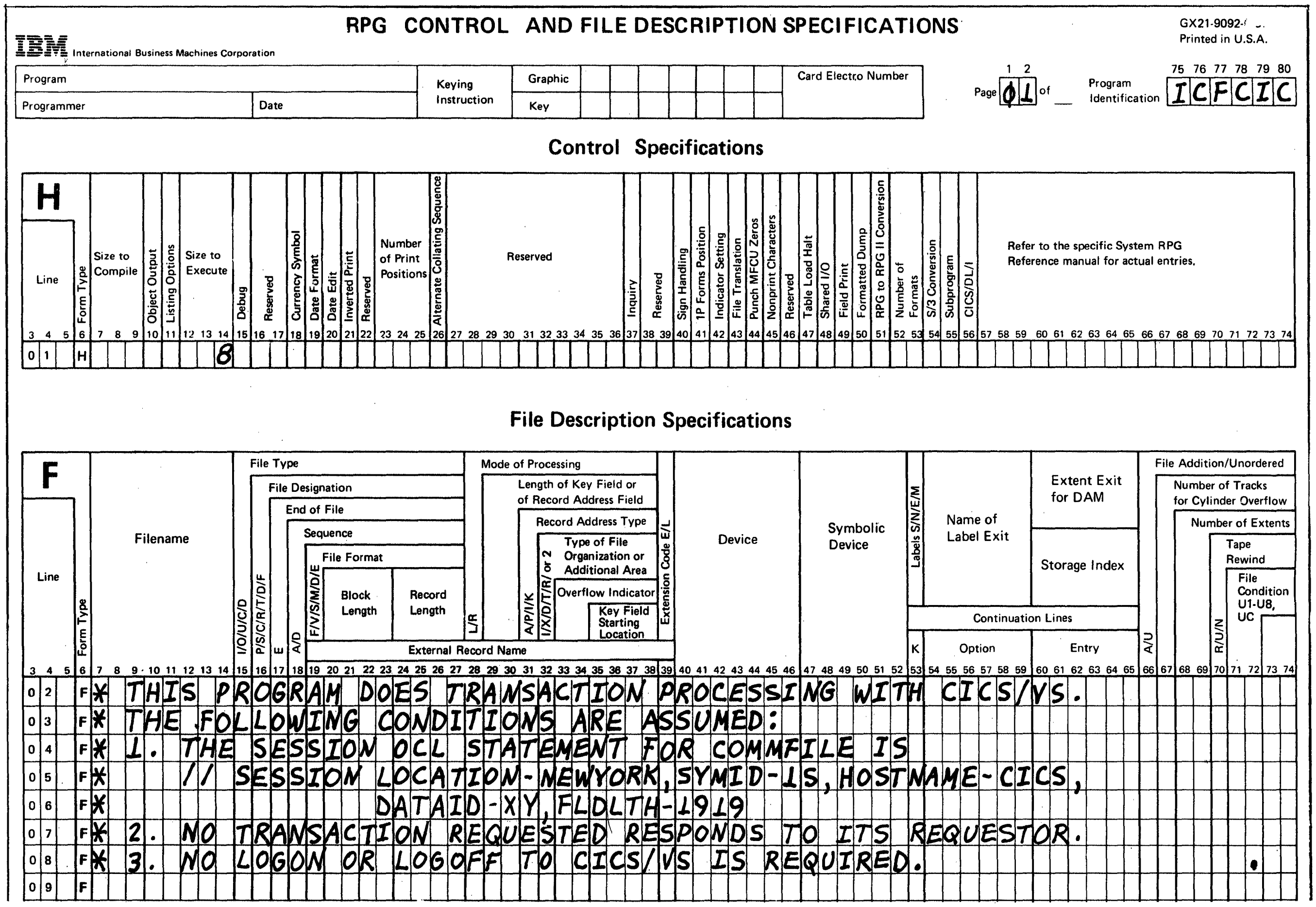
The following sample programs use the 3270 subsystem to communicate with CICS/VS.

When the System/34 program is started, it transmits a data file to the host system, one record at a time. The programs operate in a transaction-processing environment; that is, the host transaction is started, performs a single input operation, and terminates.

The first function of the System/34 program is to acquire a session (1S) with the 3270 subsystem. Then a data record is read form the System/34 file. An evoke with no invite operation ($$EVOKNI) is used to start the CICS/VS program and to send the data record. The process of reading and sending a record continues until the entire file is sent. The session is then released and the System/34 program terminates.

```
 1  IDENTIFICATION DIVISION.
 2  PROGRAM-ID.  ICFCICS.
 3  SECURITY.  THIS PROGRAM DOES TRANSACTION PROCESSING WITH CICS/VS.
        THE FOLLOWING CONDITIONS ARE ASSUMED:
        1.  THE SESSION OCL STATEMENT FOR COMM-FILE IS
            // SESSION LOCATION-NEWYORK,SYMID-1S,HOSTNAME-CICS,
               DATAID-XY,FLDLTH-0
        2.  NO TRANSACTION REQUESTED IS CONVERSATIONAL.
        3.  NO LOGON OR LOGOFF TO CICS/VS IS REQUIRED.
        4.  EACH RECORD READ FROM LOCAL-TRANS-FILE BEGINS WITH
            A CICS/VS TRANSACTION NAME.  EACH TRANSACTION NAME
            NEED NOT BEGIN WITH THE CHARACTERS "XY".
 4  ENVIRONMENT DIVISION.
 5  CONFIGURATION SECTION.
 6  SOURCE-COMPUTER.  IBM-S34.
 7  OBJECT-COMPUTER.  IBM-S34.
 8  SPECIAL-NAMES.
 9      SYSTEM-CONSOLE IS CONSOLE.
10  INPUT-OUTPUT SECTION.
11  FILE-CONTROL.
12      SELECT COMM-FILE ASSIGN TO WORKSTATION-FORMAT-01,
            ORGANIZATION IS TRANSACTION,
            FILE STATUS IS WS-RETURN-CODE, ICF-RETURN-CODE,
            CONTROL-AREA IS CICS-CONTROL-AREA.
13      SELECT LOCAL-TRANS-FILE ASSIGN TO DISK-CICSTRAN,
            ORGANIZATION IS SEQUENTIAL,
            ACCESS MODE IS SEQUENTIAL.
14  DATA DIVISION.
15  FILE SECTION.
16  FD  COMM-FILE LABEL RECORDS ARE OMITTED.
17  01  COMM-RECORD                     PIC X(128).
18  FD  LOCAL-TRANS-FILE LABEL RECORDS ARE STANDARD.
19  01  CICS-TRANSACTION.
20      03  CICS-TRANS-NAME             PIC X(8).
21      03  CICS-TRANS-TEXT             PIC X(72).
22  WORKING-STORAGE SECTION.
   ***************************************************************
   *    STATUS BYTES FOR CICS/VS SESSION I/O                     *
   ***************************************************************
23  01  CICS-RETURN-CODE.
24      03  WS-RETURN-CODE              PIC X(2).
25      03  ICF-RETURN-CODE             PIC X(4).
26          88  NORMAL-COMPLETION       VALUE '0000'.
27          88  MESSAGE-RECEIVED        VALUE '0020'.
28  01  CICS-CONTROL-AREA.
29      03  WS-AID-BYTE                 PIC 99.
30      03  WS-ID                       PIC XX.
31      03  FILLER                      PIC X(8).
   ***************************************************************
   *    TRANSACTION REQUEST TO BE SENT TO CICS/VS                *
   ***************************************************************
32  01  TRANSACTION-REQUEST.
33      03  TRANSACTION-NAME            PIC X(8).
34      03  FILLER                      PIC X(8).
35      03  FILLER                      PIC X(8).
36      03  FILLER                      PIC X(8).
37      03  FILLER                      PIC X(20) VALUE SPACES.
38      03  DATA-LENGTH                 PIC 9(4) VALUE 72.
39      03  TRANSACTION-TEXT            PIC X(72).
   ***************************************************************
   *    ERROR MESSAGE FOR ABNORMAL COMPLETION                    *
   ***************************************************************
40  01  ERROR-MESSAGE.
41      03  FILLER                      PIC X(25)
            VALUE 'ABNORMAL COMPLETION CODE '.
42      03  ABEND-CODE                  PIC X(4).
43      03  FILLER                      PIC X(18)
            VALUE ' AT RECORD NUMBER '.
44      03  RECORD-NUMBER               PIC 9(4) VALUE 0.
45      03  FILLER                      PIC X(26) VALUE SPACES.
```

```
      ******************************************************************
      *    PROCEDURE DIVISION BEGINS HERE                             *
      ******************************************************************
 46   PROCEDURE DIVISION.
 47   INITIATE-FILES.
 48       OPEN I-O COMM-FILE.
 49       OPEN INPUT LOCAL-TRANS-FILE.
 50       READ COMM-FILE AT END GO TO IGNORE-END.
 52   IGNORE-END.
 53       ACQUIRE '1S' FOR COMM-FILE.
 54       IF NOT NORMAL-COMPLETION GO TO DISPLAY-ERROR.
 56   SEND-TRANS-FILE.
 57       READ LOCAL-TRANS-FILE INTO CICS-TRANSACTION,
 58            AT END GO TO END-OF-JOB.
 59       ADD 1 TO RECORD-NUMBER.
 60       MOVE CICS-TRANS-NAME TO TRANSACTION-NAME.
 61       MOVE CICS-TRANS-TEXT TO TRANSACTION-TEXT.
 62       WRITE COMM-RECORD FROM TRANSACTION-REQUEST,
              FORMAT IS '$$EVOKNI'.
 63       IF NORMAL-COMPLETION GO TO SEND-TRANS-FILE,
 65       ELSE GO TO DISPLAY-ERROR.
 67   END-OF-JOB.
 68       DROP '1S' FROM COMM-FILE.
 69       CLOSE COMM-FILE, LOCAL-TRANS-FILE.
 70       STOP RUN.
 71   DISPLAY-ERROR.
 72       MOVE ICF-RETURN-CODE TO ABEND-CODE.
 73       DISPLAY ERROR-MESSAGE UPON CONSOLE.
 74       GO TO END-OF-JOB.
```



IBM International Business Machines Corporation

RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

GX21-9092-
Printed in U.S.A.

Program	Keying Instruction	Graphic		Card Electro Number
Programmer / Date		Key		

Page 01 of __   Program Identification (75 76 77 78 79 80) ICFCIC

Control Specifications

H

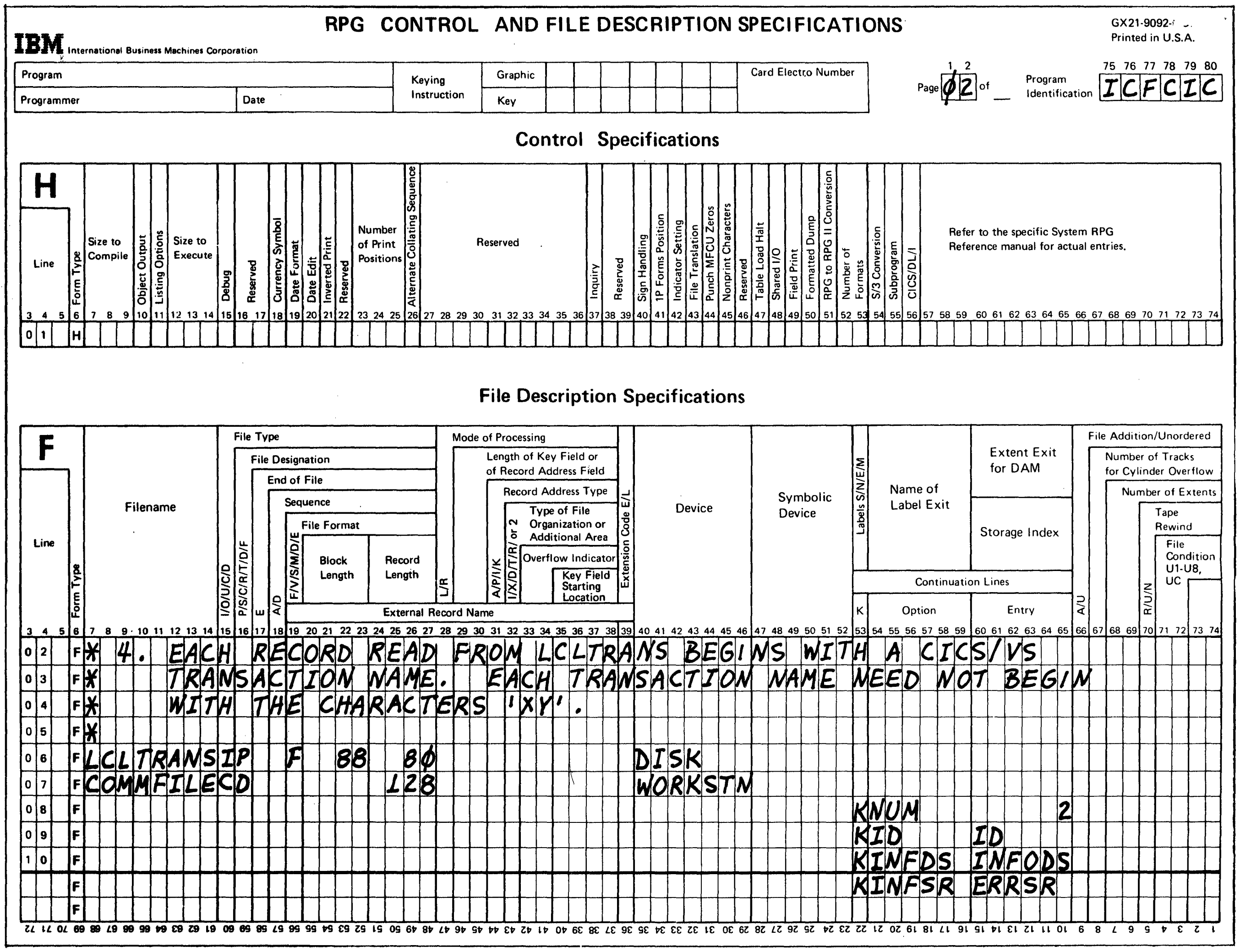
Line (3 4 5)	Form Type (6)	Size to Compile (7 8 9)	Object Output (10)	Listing Options (11)	Size to Execute (12 13 14)	Debug (15)	Reserved, Currency Symbol, Date Format, Date Edit, Inverted Print, Reserved, Number of Print Positions, Alternate Collating Sequence, Reserved, Inquiry, Reserved, Sign Handling, 1P Forms Position, Indicator Setting, File Translation, Punch MFCU Zeros, Nonprint Characters, Reserved, Table Load Halt, Shared I/O, Field Print, Formatted Dump, RPG to RPG II Conversion, Number of Formats, S/3 Conversion, Subprogram, CICS/DL/I (16–56)	Refer to the specific System RPG Reference manual for actual entries. (57–74)
01	H				8			

File Description Specifications

F

Line | Form Type | Filename | File Type | File Designation | End of File | Sequence | File Format | Block Length | Record Length | Mode of Processing | 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 | Device | Symbolic Device | Labels S/N/E/M | Name of Label Exit | Extent Exit for DAM | Storage Index | Continuation Lines: K, Option, Entry | A/U | File Addition/Unordered | Number of Tracks for Cylinder Overflow | Number of Extents | Tape Rewind | File Condition U1-U8, UC | External Record Name

Line	Form Type	
02	F	* THIS PROGRAM DOES TRANSACTION PROCESSING WITH CICS/VS.
03	F	* THE FOLLOWING CONDITIONS ARE ASSUMED:
04	F	* 1. THE SESSION OCL STATEMENT FOR COMMFILE IS
05	F	* // SESSION LOCATION-NEWYORK,SYMID-1S,HOSTNAME-CICS,
06	F	* DATAID-XY,FLDLTH-1919
07	F	* 2. NO TRANSACTION REQUESTED RESPONDS TO ITS REQUESTOR.
08	F	* 3. NO LOGON OR LOGOFF TO CICS/VS IS REQUIRED.
09	F	



IBM International Business Machines Corporation

RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

GX21-9092-
Printed in U.S.A.

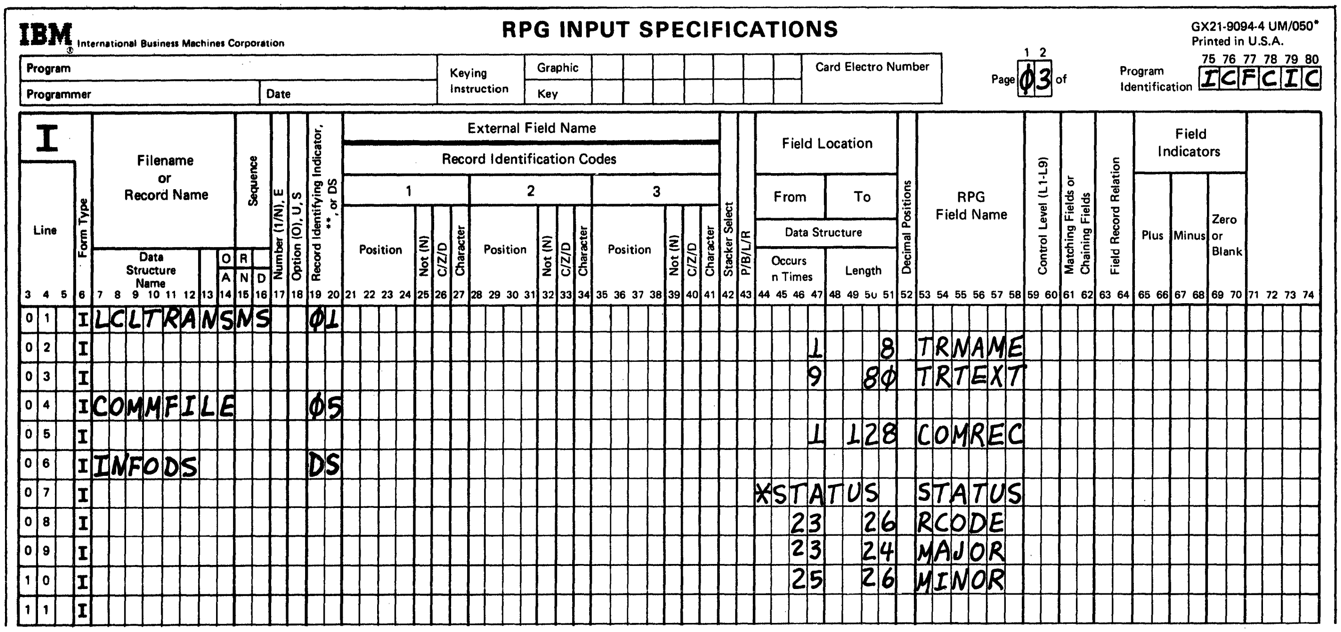
Page Ø2 of   Program Identification ICFCIC

Control Specifications

```
01 H
```

File Description Specifications

```
02 F* 4. EACH RECORD READ FROM LCLTRANS BEGINS WITH A CICS/VS
03 F*       TRANSACTION NAME.  EACH TRANSACTION NAME NEED NOT BEGIN
04 F*       WITH THE CHARACTERS 'XY'.
05 F*
06 FLCLTRANSIP  F  88  8Ø            DISK
07 FCOMMFILECD        128            WORKSTN
08 F                                              KNUM        2
09 F                                              KID    ID
10 F                                              KINFDS INFODS
   F                                              KINFSR ERRSR
   F
```



IBM International Business Machines Corporation

RPG INPUT SPECIFICATIONS

GX21-9094-4 UM/050*
Printed in U.S.A.

Page Ø3 of   Program Identification ICFCIC

```
01 ILCLTRANSNS  Ø1
02 I                                        1   8 TRNAME
03 I                                        9  8Ø TRTEXT
04 ICOMMFILE    Ø5
05 I                                        1 128 COMREC
06 IINFODS      DS
07 I                                     *STATUS  STATUS
08 I                                       23  26 RCODE
09 I                                       23  24 MAJOR
10 I                                       25  26 MINOR
11 I
```

IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

GX21-9093- UM/050* Printed in U.S.A.

Program | Programmer | Date | Keying Instruction | Graphic | Key | Card Electro Number

Page 04 of

Program Identification ICFCIC

```
01 C  N10                READ COMMFILE
02 C  N10                MOVE ID        WSID    2
03 C  N10                MOVE '1S'      ID
04 C  N10      ID        ACQ  COMMFILE
05 C  N10                SETON
06 C           RECNUM    ADD  1         RECNUM  40
07 CLR         '1S'      REL  COMMFILE
08 C*    CONTROL IS TRANSFERRED HERE IF AN ERROR OCCURS
09 CSR         ERRSR     BEGSR
10 C                     MOVE WSID      ID
11 C                     EXCPT
12 C                     SETON
13 C                     MOVE '1S'      ID
14 C                     EXCPT
15 CSR                   ENDSR'*CANCL'
16 C
```

IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21-9090-4 UM/050* Printed in U.S.A.

Program | Programmer | Date | Keying Instruction | Graphic | Key | Card Electro Number

Page 05 of

Program Identification ICFCIC

```
01 OCOMMFILED         01
02 O                                   K8 '$$EVOKNI'
03 O                         TRNAME     8
04 O                                   56 '72'
05 O                         TRTEXT   128
06 O        E        99
07 O                                   K5 '$$EOS'
08 O        E        10N99
09 O                                   K6 'ERRMSG'
10 O                         RCODE      4
11 O                         RECNUM     8
12 O
```

3270 Subsystem Return Codes

This part of Chapter 12 describes all the return codes that are valid for the 3270 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 four-digit value; the first two digits contain the major code, and the last two 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: In the return code descriptions, *your program* refers to the local System/34 application program that initiates the operation and receives the return code from the subsystem. The *remote program* refers to the application program in the remote (or host) system with which the System/34 application program is communicating through SSP-ICF.

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 3270 subsystem. Although all the operations shown are valid for 3270, their validity also depends on the logical sequence of communications events occurring between the System/34 and the remote system.

Input Operations to Your Program	Output Operations from Your Program	Combined Operations in Your Program
Accept input		
	Acquire[1]	
	End of session	
	Evoke	Evoke then get[2] Evoke then invite
Get Get attributes[3]		
Invite		
	Put end of file	Put then get[2] Put then invite
	Release	
	Set timer[4]	

[1] Normally, 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.

[2] Valid only in assembler language.

[3] Valid only in assembler and COBOL languages.

[4] For BASIC and RPG II programs, the set 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.

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, or it received a message from the remote system.

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:** The *output* operation just performed by your program was completed successfully; your program can continue to send data.

Normal Action: For the actions that can be taken (in this session) after 0000 is returned for an output operation, refer to the following chart:

If This Output Operation Was:	Then, in This 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.

0012 **Normal Indication:** On a successful input operation, one or more 3270 orders were received that are not supported by the 3270 subsystem. These orders have been removed from the data now in your program's input buffer. The three types of unsupported orders are: the *repeat to address* order, the *program tab* order, and the *erase all unprotected to address* order. (For more information on these orders, see the *3270 Information Display System Components Description Manual.*)

Normal Action: Handle the data as you would on a normal input operation. Your program can then issue another input operation or it can issue an output operation.

0020 **Normal Indication:** A message from the remote system and an end of transmission indication were received on a successful input operation. The received data is a message because the first 2 characters of the data were different from the 2 characters specified in the DATAID parameter of the SESSION statement. (The message is now in your program's input buffer.) The remote system may now want to receive some data.

Normal Action: Handle the message in the input buffer (possibly display it). Your program now has control of the session; issue an output operation or another input operation.

0030 **Normal Indication:** A truncated message from the remote system and an end of transmission indication were received on a successful input operation. The remote system may now want to receive some data. (The message was truncated because it was too long for your program's input buffer.)

Normal Action: Handle the truncated message (possibly display it) in the input buffer, and issue an output operation.

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, or it received a message from the remote system. 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 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

0212 **Normal Indication**: On a successful input operation, one or more 3270 orders were received that are not supported by the 3270 subsystem. These orders have been removed from the data now in your program's input buffer. The three types of unsupported orders are: the *repeat to address* order, the *program tab* order, and the *erase all unprotected to address* order. (For more information on these orders, see the *3270 Information Display System Components Description Manual.*) 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 data as you would on a normal input operation. Your program can then issue another input operation or it can issue an output operation.

0220 **Normal Indication:** A message from the remote system and an end of transmission indication were received on a successful input operation. (The message is now in your program's input buffer.) The remote system may now want 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: Handle the message in the input buffer (possibly display it), and issue an output operation or another input operation.

0230 **Normal Indication:** A truncated message from the remote system and an end of transmission indication were received on a successful input operation. The remote system may now want to receive some data. (The message was truncated because it was too long for your program's input buffer.) 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 (possibly display it) in the input buffer, and issue an output operation.

Major Code 03 – Successful operation, but no data received.

Normal Description: The 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.

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.

Major Codes 04-34 – Miscellaneous program errors.

Error Description: The operation just attempted by your program failed, or an output exception occurred.

- An operation may have failed because it was issued at the wrong time or because a data record was too long.
- An output exception may have occurred because your program attempted to send output when it should be receiving the output that has already been sent by the remote program.

Recovery Action: Refer to the individual return code descriptions for the appropriate recovery actions.

Code **Indication/Action**

0412 **Normal (Exception) Indication:** 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 remote program. Your program's output was not sent and should be sent later, after the remote program's data (still waiting in the subsystem input buffer) has been received.

Note: If your program issues another output operation, an error return code of 831C will be received.

Normal Action: Issue an input operation to receive the data waiting in the subsystem input buffer.

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 a new requester) 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 requesting program have been completed.

3401 **Error Indication**: This input operation was rejected because the record length of the data sent by the remote 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/34 Messages Guide* 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 (in COBOL and assembler only) a set timer operation, or by using the TIMER intrinsic function (in BASIC only); (2) continue local processing; or (3) terminate.
- If the session should be started again after the subsystem is enabled, it must be reacquired by your program or restarted by the remote 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 either in this subsystem or in the interrupt handler.

Recovery Action: This subsystem has been disabled; it must be enabled again before communications can resume. Your program can continue local processing, wait[1] to reissue the acquire operation, or terminate.

If more than one subsystem was active when the error occurred, all subsystems that were active when the error occurred should be disabled. (Note that all other active 3270 subsystems are automatically disabled when the error occurs; all other types of active subsystems must be manually disabled.)

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

If all subsystems (of all types) on the system are *not* disabled to recover from the processor check, the common queue space used by the failing subsystem *cannot* be freed. And if it is not freed, that space is wasted, and an indication of insufficient common queue space being available can occur. The indication can occur as a message when the failing subsystem is reenabled or when a different subsystem is enabled. The indication can also occur as a return code to your program for any subsystem that is starting a *new* session (code 8215) or performing an output operation in any *existing* session (code 8315).

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 remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, it can wait[1] until the subsystem has been reenabled to reissue the acquire operation, or it can terminate.

8083 **Error Indication:** An MLCA (multiline communications adapter) controller check occurred on an *output* operation. The subsystem has been disabled.

Recovery Action: Communications with the remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, wait[1] until the subsystem is enabled again, or terminate.

8084 **Error Indication:** An MLCA (multiline communications adapter) controller check occurred on an *input* operation. The subsystem has been disabled.

Recovery Action: Communications with the remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, wait[1] until the subsystem is enabled again, or terminate.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

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/34 Messages Guide* 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, first disable the subsystem, make the change in the subsystem's configuration record, and 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, (and depending on what the return code description says):
   a. Notify the remote location that a change is required on that end to correct the error received.
   b. Simply reissue the acquire operation. It could be successful if the error occurred because there was not enough common queue space available to support a new session, or because an isolated line error occurred.
   c. If the acquire operation is again unsuccessful, retry it only a limited number of times. (The limit for retries should be specified in your program.)
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 operation is valid only in an *active* session and cannot, therefore, be issued after an 82xx return code is received. (This restriction does not apply to COBOL and assembler programs, or to the TIMER intrinsic function in BASIC, which also can be used to wait for a specified time interval.)

Code **Indication/Action**

8213 **Error Indication:** On an unsuccessful acquire operation, a queue space error condition was detected. The session could not be started because no *subsystem* queue space was available at the time.

Recovery Action: Your program can wait[1] for a period of time, then reissue the acquire operation. If an unacceptable number of queue space errors occur, you can disable the subsystem and change the subsystem configuration by specifying a larger subsystem queue space size in the subsystem queue space parameter. After the subsystem is enabled, reissue the acquire operation to start the session.

8215 **Error Indication:** On an unsuccessful acquire operation, a queue space error condition was detected. The session cannot be started because the size of the *common* queue space, specified during subsystem configuration, is too small.

Recovery Action: Your program can wait[1] for a period of time, then reissue the acquire operation. If an unacceptable number of queue space errors occur, you can disable all the subsystems that are active in the system, and change the subsystem configuration by specifying a larger common queue space size in the SSP-ICF common queue space parameter. After the subsystem is enabled, reissue the acquire operation to start the session.

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, respecify 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.

[1] **For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.**

8281 **Error Indication:** On an unsuccessful acquire operation, an SSP-ICF error condition was detected. The error caused a processor check either in this subsystem or in the interrupt handler.

Recovery Action: This subsystem has been disabled; it must be enabled again before communications can resume. Your program can continue local processing, wait[1] to reissue the acquire operation, or terminate.

If more than one subsystem was active when the error occurred, all subsystems that were active when the error occurred should be disabled. (Note that all other active 3270 subsystems are automatically disabled when the error occurs; all other types of active subsystems must be manually disabled.)

If all subsystems (of all types) on the system are *not* disabled to recover from the processor check, the common queue space used by the failing subsystem *cannot* be freed. And if it is not freed, that space is wasted, and an indication of insufficient common queue space being available can occur. The indication can occur as a message when the failing subsystem is reenabled or when a different subsystem is enabled. The indication can also occur as a return code to your program for any subsystem that is starting a *new* session (code 8215) or performing an output operation in any *existing* session (code 8315).

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 remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, wait[1] 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 100 sessions can be active in the System/34 at one time. The session was not started.

Recovery Action: Your program can wait[1] for another session to end and then reissue the acquire operation. Otherwise, your program can continue local processing or terminate.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

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 to be enabled is identified by the location parameter in the SESSION statement. That location name must also be specified in the subsystem configuration record (shown on display 3.0 of the subsystem configuration planning charts). The session was not started.

Recovery Action: Verify that the subsystem name was specified correctly on the LOCATION parameter of the SESSION statement. If the correct name was specified, contact the System/34 system operator and request that the specified subsystem be enabled by executing the ENABLE procedure command at the system console. Then reissue the acquire operation. Otherwise, your program can continue local processing, wait[1] 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[1] until the subsystem has been enabled, and then reissue the acquire operation to start the session.

82AE **Error Indication:** On an unsuccessful acquire operation, an invalid device address was detected. The device address specified in the DEVADDR parameter of the SESSION statement does not match any of the device addresses specified in the subsystem configuration. The session was not started.

Recovery Action: Either change the address value specified in the SESSION statement, or disable the subsystem and check all of the device addresses in the subsystem configuration with the remote location. Correct any errors, enable the subsystem again, and reissue the acquire operation.

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[1] until the subsystem is enabled again, and then reissue the acquire operation. Otherwise, your program can continue local processing, or it can terminate.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

82B1 **Error Indication:** On an unsuccessful acquire operation, the session specified by the session address on the SESSION statement was identified as already being in use. (The session address is specified in the DEVADDR parameter of the SESSION statement. The address matches one of the addresses in the list of addresses that were specified during subsystem configuration.) The session was not started.

Recovery Action: (1) Your program can wait[1] for the specified session to become available and reissue the acquire operation. (2) You can remove the DEVADDR parameter value from the SESSION statement and reissue the acquire operation; the subsystem will then assign a session address from those available in the subsystem configuration. (3) Or, your program can continue local processing or terminate.

82B3 **Error Indication:** The acquire operation was not successful because all of the sessions specified in the subsystem configuration are already in use. The session was not started.

Recovery Action: Wait[1] for one of the sessions in the subsystem to become available, then reissue the acquire operation. Otherwise, continue local processing or terminate.

82B4 **Error Indication:** The acquire operation was not successful because all of the resources needed for the session could not be allocated from the assign/free area of the system. All available resources are already being used in the system. The session was not started.

Recovery Action: Wait[1] for the needed resources to become available, then reissue the acquire operation. Otherwise, continue local processing or terminate.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

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/34 Messages Guide* 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, first disable the subsystem, make the change in the subsystem's configuration record, and 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.

   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.

   Note: If the session is started again, it starts from the beginning, not at the point where the session error occurred.
3. If no change is needed in your program or in the subsystem, (and depending on what the return code description says):
   a. Notify the remote location that a change is required on that end to correct the error received.
   b. Retry the operation, if possible. It could be successful if the error occurred because there was not enough common queue space available at the time.
   c. If another operation is not successful, retry it only a limited number of times. (The limit for retries should be specified in your program.)
   d. Issue a set timer operation in your program so it can wait for a specified time interval before reissuing the operation.

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 an 81xx (session was terminated) or 82xx (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 an 81xx or 82xx return code is handled properly. If you want your program to recover from an improperly handled error condition, issue another acquire operation.

8315 **Error Indication:** On an evoke operation, a queue space error condition was detected. The evoke operation could not be performed because no *common* queue space was available at the time.

Recovery Action: Your program can issue a set timer operation and wait for a period of time, and then reissue the evoke operation. If an unacceptable number of queue space errors occur, you can disable all the subsystems and change the subsystem configuration by specifying a larger common queue space size in the SSP-ICF common queue space parameter. After the subsystem is enabled, reissue the acquire operation to restart the session.

831C **Error Indication:** The output operation issued before this output operation received a return code of 0412 (indicating that the remote program sent 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 data.

Recovery Action: Issue an input operation to receive the 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. 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 before attempting to communicate with the remote 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: If you want your program to recover dynamically, reissue the output operation with a smaller output length. Otherwise, you can either change the record length in your program and recompile it, or you can change the value specified for the maximum user record length parameter in the subsystem configuration. 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.

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 a release operation. If a coding error in your program caused the error, correct your program.

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.

8334 **Error Indication:** On an evoke operation issued by your program, no procedure name was included with the operation.

Recovery Action: Correct the evoke operation statement by supplying the correct procedure name, and reissue the operation.

8338 **Error Indication:** An invalid or unsupported 3270 command was received from the remote system. The session is still active.

Recovery Action: Reissue the input operation, or issue an output operation. If the error continues to occur, notify the remote location that a change is required to the remote program to correct the error condition.

8339 **Error Indication:** The data (a 3270 command sent by the remote system) was rejected by the subsystem because it was busy processing the last operation; the command is unavailable to your program.

Recovery Action: Your program can reissue the input operation, and the remote system can resend the command. Or your program can issue an output operation now and receive the command later on.

839C **Error Indication:** On an *input* operation, the length of the data block sent by the remote system exceeded the length of the subsystem's input buffer. The session, however, is still active.

Recovery Action: Check that the maximum user record length is correct in the subsystem configuration record. If the parameter is correct, notify the remote system programmer and verify that the record length is correct. Then, if your program started the session, reissue the input operation.

Chapter 13. The Finance Subsystem

The Finance subsystem allows users of the System/34 SSP to communicate with the 3601/4701 Finance Controller and its attached devices and with the 3694 Document Processor. The Finance subsystem provides an interface between application programs on the System/34 and application programs on the 3601/4701 or 3694, including the IBM 3600 Online Terminal Support for System/34 Field Developed Program (FDP) and the Check Processing Executive/3694 Program Product. The System/34 application programs can be written in assembler, BASIC, COBOL, or RPG II. The encryption/decryption subroutines cannot be used with the BASIC language.

The Finance subsystem allows the 3601/4701 and the 3694 to initiate procedures on the System/34. System/34 security options are supported.

The Finance subsystem provides support for communicating on a point-to-point switched or nonswitched line or on a multipoint line. Remote 5251 display stations, other System/34s using the Peer subsystem, 3601/4701 Finance Controllers, and 3694 Document Processors can coexist on the same multipoint communications line.

The support provided with the Finance subsystem also includes a data encryption/decryption routine that can be accessed by a System/34 application program, and a utility to transmit an operational diskette image from the System/34 to the 3601/4701 Finance Controller. The data encryption/decryption routines are available only in the United States.

The Finance subsystem allows up to sixteen 3601/4701 or 3694 controllers (or any combinaton of 3601s and 3694s) on a multipoint line and up to 30 concurrent logical work station sessions with each 3601/4701 controller. However, the System/34 supports a maximum of only 100 concurrent SSP-ICF sessions.

SETTING UP THE FINANCE SUBSYSTEM

The SSP procedures CNFIGSSP and INSTALL are used to include the interactive communications feature and Finance subsystem support on the System/34. The general interactive communications and line control support is included when it is requested on the appropriate CNFIGSSP prompt. The Finance subsystem support is copied to the system library when the appropriate responses to the INSTALL procedure prompts are taken. The CNFIGSSP and INSTALL procedures, with their displays and related responses, are described in the *Installation and Modification Reference Manual.*

Note: If the Finance subsystem, remote work station support, and/or the Peer subsystem are to run concurrently on the same line, that line must be specified as multipoint during microcode configuration.

After the Finance subsystem has been installed, the CNFIGICF procedure is used to tailor the subsystem support to an existing or proposed network. The operation of the CNFIGICF procedure is also explained in the *Installation and Modification Reference Manual*. Before running the CNFIGICF procedure, however, you should fill out a planning chart for each subsystem configuration that you want to define. Copies of the planning chart for each subsystem are available in Appendix F of this manual and in the *Installation and Modification Reference Manual*. The following sections explain how to fill out the planning chart for the Finance subsystem.

Display 1.0 Subsystem Member Configuration

1.0 Subsystem Member Configuration

1. Subsystem configuration member name (8 characters) _ _ _ _ _ _ _ _
2. Subsystem library name (8 characters) _ _ _ _ _ _ _ _

   Select:

   1. Create new member 4. Delete a member

   2. Edit existing member 5. Review a member

   3. Create new member from existing member
3. Enter selection: ______
4. Existing member name: _ _ _ _ _ _ _ _
5. Existing member library name: _ _ _ _ _ _ _ _

Subsystem configuration member name: Specify a name for this configuration of the subsystem. This name is used to store the member in a library, and is referenced in the ENABLE and DISABLE procedures.

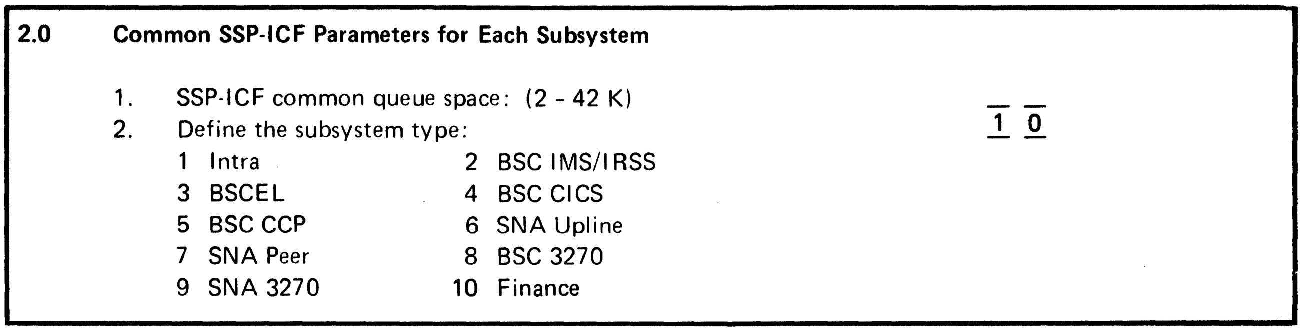
Library name: Specify the name of the library in which the configuration member is stored or to be stored. The default is #LIBRARY.

Enter selection: Specify one of the five options: (1) create a new member, (2) edit an existing member, (3) create a new member from an existing member, (4) delete a member, or (5) review a member without changing it.

Existing member name: This prompt appears if option 3 was selected. Specify the name of the existing subsystem configuration member that is to be used to create the new member. The existing member remains unchanged.

Existing member library name: This prompt appears if option 3 was selected. Specify the library name where the existing member resides.

Display 2.0 Common SSP-ICF Parameters for Each Subsystem



```
2.0      Common SSP-ICF Parameters for Each Subsystem

           1.   SSP-ICF common queue space:  (2 - 42 K)                       _ _
           2.   Define the subsystem type:                                    1 0
                1  Intra               2  BSC IMS/IRSS
                3  BSCEL               4  BSC CICS
                5  BSC CCP             6  SNA Upline
                7  SNA Peer            8  BSC 3270
                9  SNA 3270           10  Finance
```

SSP-ICF common queue space: Specify the size, in multiples of 2 K bytes, of the common queue space. The common queue space requirement is:

C = 122X + 64Y + 120Z

where:

C = number of bytes required for common queue space
X = total number of communications lines on which the Finance subsystem will be communicating

Y = number of remote locations that will be active concurrently

Z = total number of logical work station sessions that will be active concurrently for all configurations

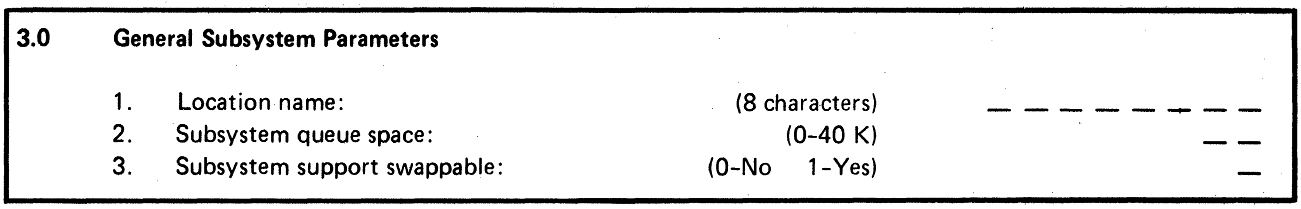
The common queue space value specified for the first subsystem that is enabled becomes the size of the common queue space. Be sure that the value specified for common queue space size takes into account the requirements of any other subsystem that might be active concurrently.

The size of the common queue space plus the total subsystem queue space of all the enabled finance subsystems cannot exceed 42 K bytes.

The default common queue space size is 4 K bytes.

Define the subsystem type: Specify 10 for the Finance subsystem.

Display 3.0 General Subsystem Parameters



```
3.0     General Subsystem Parameters

        1.  Location name:                      (8 characters)     _ _ _ _ _ _ _ _
        2.  Subsystem queue space:                    (0-40 K)                 _ _
        3.  Subsystem support swappable:         (0-No   1-Yes)                  _
```

Location name: Specify up to 8 characters for the name of the location associated with this configuration. The location name is used in some of the displayed message texts. The default is the subsystem configuration member name.

Subsystem queue space: Specify the size, in multiples of 2 K bytes, of the subsystem queue space. No subsystem queue space is required; however, improved performance may be obtained by using subsystem queue space.

Whenever a record is received, it must be moved to an intermediate buffer.

For 3601/4701 sessions, the Finance subsystem attempts to assign an intermediate buffer out of the subsystem queue space. If no subsystem queue space is available, a disk work area is used for the buffer. When the disk work area is used, main storage requirements for the subsystem queue space are reduced, but the amount of time required to move data between the application program and the intermediate buffer is greatly increased.

Factors affecting subsystem queue space requirements for 3601/4701 sessions are:

- The number of concurrent sessions. Each active session competes for intermediate buffers from subsystem queue space. The more sessions competing, the greater the possibility of a failure to obtain an intermediate buffer.

- The size of records. The amount of storage required for an intermediate buffer is equal to the size of the record to be received.

- Frequency of transmission. Sessions that receive records frequently may be more likely to obtain an intermediate buffer from disk than those that receive records infrequently.

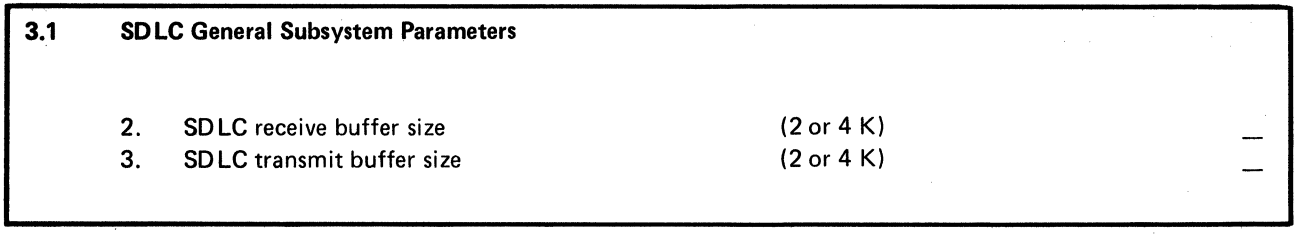
- Response requirements. Interactive programs that receive infrequently may be able to accept the slower response times that occur when intermediate buffers are on disk.

For 3694 sessions, the Finance subsystem assigns a pacing control table and seven pacing buffers in the subsystem queue space. If no subsystem queue space is available, the subsystem assigns the table and buffers in common queue space. Each 3694 session requires approximately 2 K bytes of queue space.

The default subsystem queue space size is 4 K bytes.

Subsystem support swappable: Specify whether you want the subsystem to be swappable. Consider the total system performance, the size of the subsystem, and the amount of user main storage when determining whether you want the subsystem to be swappable. Specifying swappable should make the subsystem operate more efficiently for batch activity, but could degrade performance for the rest of the system. Interactive applications should run with the subsystem nonswappable. If multiple finance configurations are enabled, the swappable status of the first configuration enabled is used. The default is 0 (no). The Finance subsystem requires 16 K bytes of main storage.

Display 3.1 SDLC General Subsystem Parameters

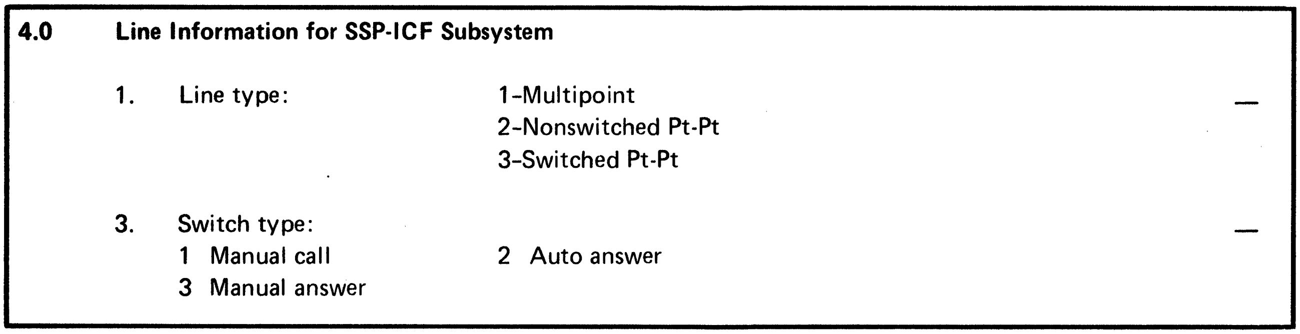


```
3.1    SDLC General Subsystem Parameters


       2.   SDLC receive buffer size                  (2 or 4 K)         _
       3.   SDLC transmit buffer size                 (2 or 4 K)         _
```

SDLC receive buffer size: Specify the amount of storage for SDLC receive buffers. Specify either 2 K bytes (7 buffers) or 4 K bytes (15 buffers). This storage is allocated as nonswappable. The default is 2 K bytes.

SDLC transmit buffer size: Specify the amount of storage for SDLC transmit buffers. Specify either 2 K bytes (7 buffers) or 4 K bytes (15 buffers). This storage is allocated as nonswappable. The default is 2 K bytes.

Display 4.0 Line Information for SSP-ICF Subsystem

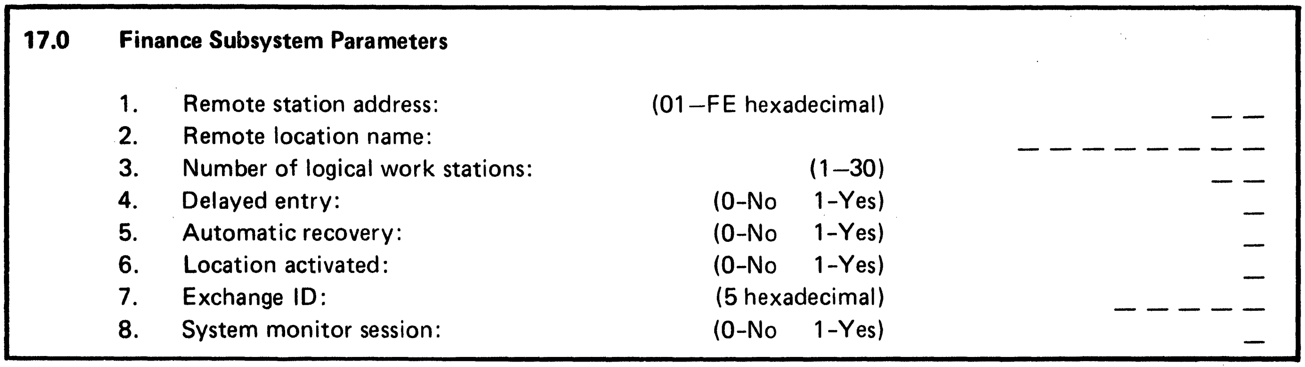


```
4.0    Line Information for SSP-ICF Subsystem

       1.   Line type:             1-Multipoint                          _
                                   2-Nonswitched Pt-Pt
                                   3-Switched Pt-Pt

       3.   Switch type:                                                 _
            1  Manual call         2  Auto answer
            3  Manual answer
```

Line type: Specify the line type that is suitable to your communications environment. There is no default line type.

Switch type: This prompt is displayed only if switched pt-pt line was specified. Specify the switch type you want for the line. The default is 1 (manual call). If you are using the switched X.21 feature, specify 2 (auto answer).

Display 17.0 Finance Subsystem Parameters



```
17.0    Finance Subsystem Parameters

        1.  Remote station address:           (01–FE hexadecimal)              _ _
        2.  Remote location name:                                   _ _ _ _ _ _ _ _
        3.  Number of logical work stations:               (1–30)               _ _
        4.  Delayed entry:                          (0-No   1-Yes)                _
        5.  Automatic recovery:                     (0-No   1-Yes)                _
        6.  Location activated:                     (0-No   1-Yes)                _
        7.  Exchange ID:                        (5 hexadecimal)         _ _ _ _ _
        8.  System monitor session:                 (0-No   1-Yes)                _
```

Remote station address: Specify the station address of the remote location. Specify the address as two hexadecimal digits between 01 and FE. This address must be the same as the control unit address configured in the 3601/4701 or 3694 controller. If remote work stations and/or the Peer subsystem are also active on the same line, this address must not be the same as any station address configured for them.

Remote location name: Specify a name for the remote station. The name must be from 1 to 8 characters and must be unique for each remote location.

Number of logical work stations: Specify the maximum number of logical work stations that can be active with this location. Specify a value from 1 to 30. The default is 1. This value should be the same as the maximum number of work station IDs configured in the 3601/4701 controller or 1 if this location is a 3694.

Delayed entry: Specify whether you want polling to continue if the station does not respond when the subsystem is enabled. If you specify 0 (no) and the station does not respond to polling, the System/34 operator is notified via a message. If you specify 1 (yes) and the station does not respond, polling continues until the station does respond or until the subsystem is disabled. The default is 0 (no).

Automatic recovery: Specify whether automatic recovery should be attempted for this station if a recoverable link error occurs. If you specify 0 (no) and a recoverable link error occurs, the System/34 operator receives an error message and must respond. If you specify 1 (yes) and a recoverable link error occurs, the System/34 operator receives an informational message and recovery is attempted. The default is 0 (no).

Location activated: This prompt is not displayed if a switched pt-pt line is selected. Specify whether the remote location should be activated when the subsystem is enabled. The default is 1 (yes).

Exchange ID: Specify five hexadecimal digits that uniquely identify this station. This ID must be the same as the XID configured in the 3601/4701 or 3694 controller. The default is 00000.

System monitor session: Specify whether you want to activate the system monitor session for this location. The system monitor session is used to transmit an operational diskette to a 3601/4701; no other sessions are allowed. Normally, you should specify 0 (no) for all locations. When the system monitor session is required, you can change this value to 1 (yes) when the subsystem is enabled by using the SHOW parameter. (See *ENABLE Procedure* later in this chapter for more information.)

STARTING AND ENDING THE FINANCE SUBSYSTEM

The ENABLE procedure starts the Finance subsystem and/or activates communication with a remote location. The DISABLE procedure ends the subsystem and/or de-activates communication with a remote location. The following sections describe the ENABLE and DISABLE procedures.

ENABLE Procedure

The ENABLE procedure can be used to activate the Finance subsystem or to activate a particular location on a multipoint line. ENABLE performs the following functions:

- Ensures compatibility between the configuration and the communication hardware.
- Determines whether the line requested is available.
- Loads and attaches the subsystem if it is not already active.
- Loads and attaches the SDLC task if it is not already active.
- Assigns storage for required data areas and buffers.

If the configured line type does not correspond to the actual line type, a message is issued, and the enable is terminated. You can use the SPECIFY procedure to change the line type. The SPECIFY procedure is described in the *SSP Reference Manual*.

Enable also ensures that all remote location names associated with this configuration are unique on the system. If a subsystem is active and one of the location names matches a remote location name in the finance configuration being enabled, a message is issued and the enable for that location is terminated. The operator can also choose to cancel the entire ENABLE procedure.

When the ENABLE procedure is used to activate a particular location on a multipoint line and the subsystem configuration is already active, enable ensures that the subsystem configuration is on the specified line before enabling the location. If the first enable specifies a location name, enable performs the functions required to activate the subsystem.

The format of the ENABLE procedure command for the Finance subsystem is:

$$\text{ENABLE name,}\begin{bmatrix}\underline{\text{\#LIBRARY}}\\ \text{library name}\end{bmatrix}\text{ , }\begin{bmatrix}\text{line number}\end{bmatrix}\text{ , }\begin{bmatrix}\text{SHOW}\\ \underline{\text{NOSHOW}}\end{bmatrix}\text{ , }\begin{bmatrix}\text{location}\end{bmatrix}$$

name: Specifies the name of the subsystem configuration to be enabled.

library name: Specifies the name of the library that contains the specified subsystem configuration. The default is #LIBRARY.

line number: Specifies the communications line for which this subsystem is to be enabled. If this is not the first enable of this subsystem configuration, the line number need not be specified.

SHOW/NOSHOW: Specifies whether subsystem configuration parameters are to be displayed before the subsystem is enabled. If SHOW is specified, they are displayed, and some of the configuration parameters can be overridden. The subsystem configuration is not permanently altered. NOSHOW is the default.

location: Specifies the name of the remote location to be enabled. If this parameter is not specified, all locations configured with location activated (1-yes) are activated.

DISABLE Procedure

The DISABLE procedure can be used to terminate communication with a particular remote location on a multipoint line or to terminate communication with all remote locations in a configuration.

When a disable is requested to de-activate a subsystem configuration or location, the following functions are performed:

- If no sessions are active for the configuration or location being disabled, the disable is performed, and the main storage being used is freed. If the last active location for a configuration is being disabled, the entire configuration is disabled. When the last finance configuration is disabled, the subsystem is terminated. If the Finance subsystem is the last subsystem to use the SDLC task, the SDLC task is also terminated.

- If sessions are active for the location(s) being disabled, a message is issued to the system operator with the following options:
  - Hold the disable. New sessions and transactions are prevented from being started and, when all sessions complete, a normal disable occurs. Each successful operation receives a 02xx return code, indicating that a disable is pending.
  - Retry the disable. Check again for any active sessions.
  - Cancel active sessions and then disable. Active sessions are immediately terminated, and the disable is performed.
  - Ignore the disable request. The DISABLE procedure is canceled and must be run again when a disable is desired.

The format of the DISABLE procedure command for the Finance subsystem is:

```
'DISABLE name,[location]
```

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

location: Specifies the name of the remote location to be disabled. If this parameter is not specified, all locations are disabled.

STARTING FINANCE SUBSYSTEM APPLICATIONS

System/34 Finance subsystem applications can be started by a display station operator entering a procedure command or by a request from a remote 3601/4701 controller. Procedures that are started by a System/34 operator must have a SESSION OCL statement for each session to be started. The following sections describe the SESSION statement and the incoming procedure start requests.

SESSION OCL Statement

The format of the SESSION statement for the finance subsystem is:

```
// SESSION LOCATION-name,SYMID-session id,LWSID-logical work station id
```

LOCATION: Specifies the name of the remote location associated with the session. This name must have been configured as a remote location name for the configuration being used.

SYMID: Specifies the symbolic ID of the session with which this SESSION statement is associated. The symbolic ID must be 2 characters, with the first character numeric (0 through 9) and the second character alphabetic (A through Z, #, $, or @). This is the same ID that the application program uses when referring to this session. This ID is the equivalent of the symbolic display station ID as specified on the WORKSTN OCL statement. This parameter has no default.

LWSID: Specifies the ID of the logical work station for which this session is intended. The ID can be a decimal value from 1 through 31.

Incoming Procedure Requests

For remote applications to initiate procedures on the System/34, the 3601/4701 controller must send an *EXEC procedure start request. The format of the procedure start request is:

```
*EXEC procedure name [optional data]
```

procedure name is the name (1 to 8 characters) of the procedure to be started. The procedure name must start in position 7.

optional data is 1 to 512 bytes of optional data following the procedure name. A blank must be included between the procedure name and optional data.

OPERATION CONSIDERATIONS

The following operations are supported by the Finance subsystem. A complete chart of all interactive communications operations and the subsystems that support them is in each language chapter. The chart also shows the keyword or format name used to code the operation in that language. More information about how an operation is coded is also described in the appropriate programming language chapter.

Whether, an operation completes successfully or unsuccessfully, a return code is given to the application program. All of the return codes that are valid for this subsystem are described at the end of this chapter. A summary chart in Appendix A lists all the return codes and the subsystems for which they are valid.

Acquire Operation

The acquire operation is issued by the application program to allocate a logical work station session. If the requested logical work station session is available and active, the session is assigned to the issuing program.

Put Operations

The put operation is used to pass data from the System/34 application to a 3601/4701 or 3694. The put operation can be issued alone ($$SENDNI) or combined with an input operation ($$SEND).

If the put operation is combined with an input operation, the record is sent to the remote system. Then a record is received by the subsystem. In the case of a BASIC, COBOL, or RPG II program, the input data is made available to the application program on the next accept (READ) operation. Assembler users can choose to wait for the input data by using a put then get operation.

The current put operation is not started until the previous put operation is complete. If the previous put operation failed, the current put is not performed, and the application program is notified via the appropriate return code.

Input Operations

The input operations for the finance subsystem are invite, get, and accept. The invite operation can be issued only as a combined operation with a put operation ($$SEND) in BASIC, COBOL, and RPG II. Assembler language users can issue an invite operation explicitly. Either a get or invite operation signals the subsystem to obtain data on the session for the application program. A get operation causes the application program to wait for the data to be available. When a program issues an invite operation, it receives the data with a subsequent accept operation. The accept operation allows input from any previously invited session.

Release Operation

The release operation is issued by the application program to terminate a session it acquired or by a remotely initiated MRT program to pass the session on to the next job step. If the session is still communicating when the end of session operation is issued, the transmission is abnormally terminated by the Finance subsystem, and abnormal termination of the remote application program could result. See Chapter 2 for more information about the release operation.

End of Session Operation

The end of session operation ($$EOS) always results in a normal completion return code. The session is always terminated by the end of session operation. If the session is still communicating when the end of session operation is issued, the transmission is abnormally terminated by the Finance subsystem, and abnormal termination of the remote application program could result.

Get Attributes Operation

The get attributes operation (assembler only) can be issued at any time to determine the status of a session. If you are using BASIC, you can use the ATTRIBUTE$ intrinsic function to determine the status of a session.

Set Timer Operation

The set timer operation ($$TIMER) results in a timer expired return code (0310) after a specific time interval (in hours, minutes, and seconds) has expired.

TRANSMITTING AN OPERATIONAL DISKETTE IMAGE TO A 3601/4701

Application programs that execute in the 3601/4701 are generated on a System/370 that has been configured to support the 3600 Finance Communication System. The output from the System/370 is a basic exchange diskette file. This file can be transmitted from the System/34 to the 3601/4701 controller. The Finance subsystem transmits this file using the system monitor session. The 3601/4701 uses this file to create an operational diskette that it uses during IPL. Transmitting an operational diskette image is done infrequently, usually only for 3601/4701 installation, changes to 3601/4701 application programs, or 3601/4701 engineering changes.

To transmit an operational diskette image, you must perform the following functions:

1. Perform an IPL on the 3601/4701 with the starter diskette.

2. Transfer the file from the diskette to the disk using the TRANSFER procedure.

3. Enable the Finance subsystem with the configuration that specifies the system monitor session, or use the SHOW parameter of the ENABLE procedure to specify the system monitor session.

4. Enter the following command to run the System/34 LOAD3601 procedure to transmit the file. The LOAD3601 procedure is supplied with the Finance subsystem.

   ```
   LOAD3601 filename,location
   ```

   where:

   filename is the name of the basic exchange file that contains the diskette image to be transmitted to the 3601/4701.

   location is the name of the location of the 3601/4701 to which the diskette image will be transmitted.

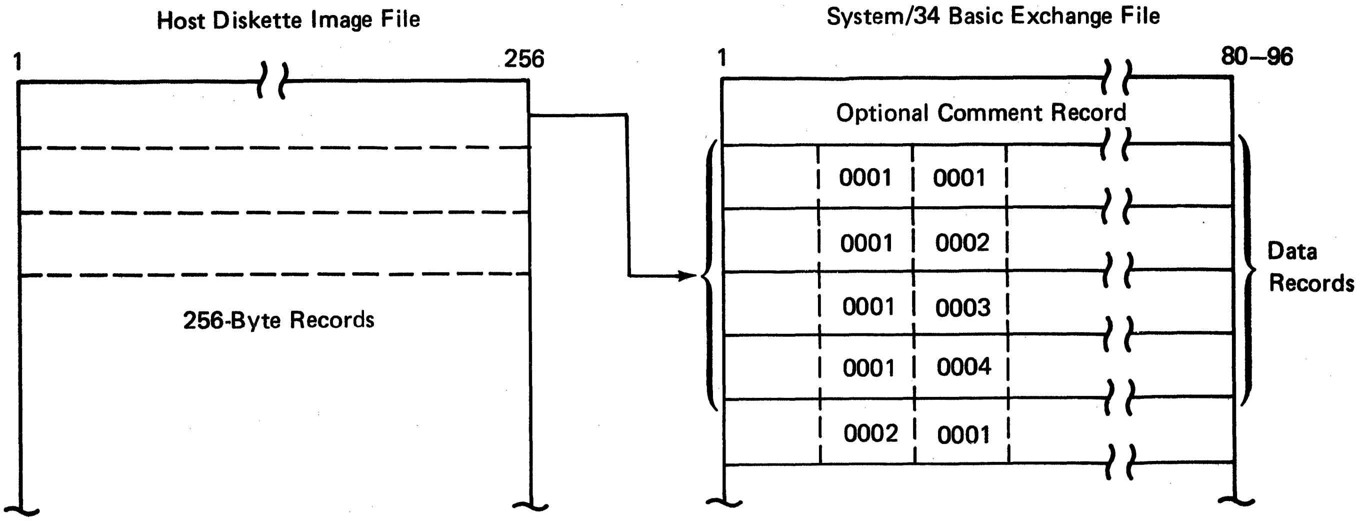
5. Replace the starter diskette on the 3601/4701 with a diskette formatted to receive the file.

6. When the LOAD3601 procedure completes, disable the Finance subsystem.

7. Perform an IPL on the 3601/4701 using the new 3601/4701 operation diskette.

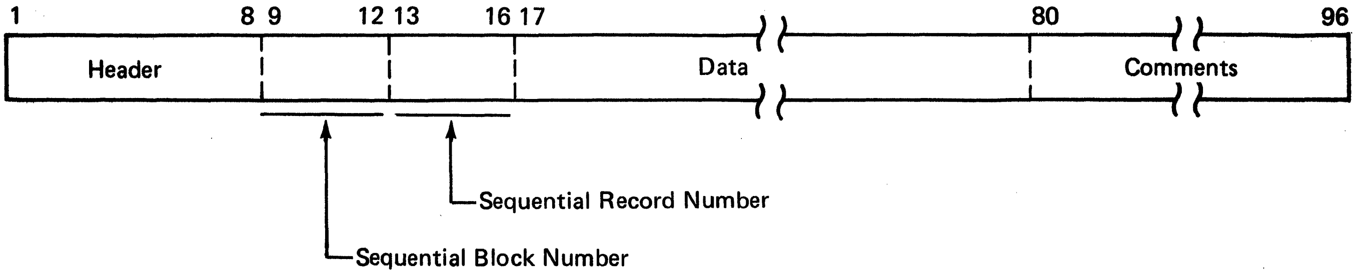
8. Enable the Finance subsystem without the system monitor session and begin normal operation.

If you cannot obtain the basic exchange diskette file, you can create the file as follows. At the host system, create a diskette image file using the Host Diskette Image Create (HDIC) program. This program is part of the 3600 Finance Communication System Host Support for an IBM System/370, 3031, 3032, 3033, or 4300. The diskette image file must be converted into a basic exchange file that can be transmitted by the LOAD3601 procedure. The following illustration shows the format of the diskette image and basic exchange files:



The first record in the file is an optional comment record. You can write any information in this record to identify the file.

The remaining records will contain the data from the diskette image file. As shown in the previous illustration, you must convert each 256-byte record from the diskette image file into four 64-byte records then write the records onto the basic exchange file. The required format for each data record is shown in the following illustration:



Each record can be from 80 to 96 bytes long. You can use positions 1 through 8 for an optional header or comments and 80 through 96 (if used) for optional comments. These positions are ignored by the LOAD3601 procedure.

The LOAD3601 procedure uses the sequential block number and sequential record number to check for correct sequence when it processes the file.

The data is 64 bytes of data from the diskette image file.

ENCRYPTION/DECRYPTION SUBROUTINES

Included with the Finance subsystem support are subroutines that provide encryption and decryption for assembler, COBOL, or RPG II application programs (the encryption/decryption subroutines cannot be used with the BASIC language). The encryption/decryption routines are available in the United States only. The algorithm used is the National Bureau of Standards Data Encryption Standard (DES). The subroutines can be used to encrypt or decrypt sensitive data or to generate personal identification numbers (PINs).

DES is a key-driven cryptographic method; that is, the encryption and decryption of data is controlled by keys supplied to DES. Because there are 2^{64} possible keys, it is extremely unlikely that encrypted data could be decrypted by simply going through the possible keys. Thus, the level of security provided by DES depends on the level of security provided for the keys. Program, and programmer accessibility to keys must be considered when implementing DES.

Because the data is encrypted by the routines in your program, and *not* by the Finance subsystem, the data can be decrypted on the communications line by anyone having the keys. For more information about data security and cryptography, see *IBM Data Security Through Cryptography*, GC22-9062.

The subroutines are SUBR30 for RPG II, SUBR31 for COBOL, and #SBDE for assembler.

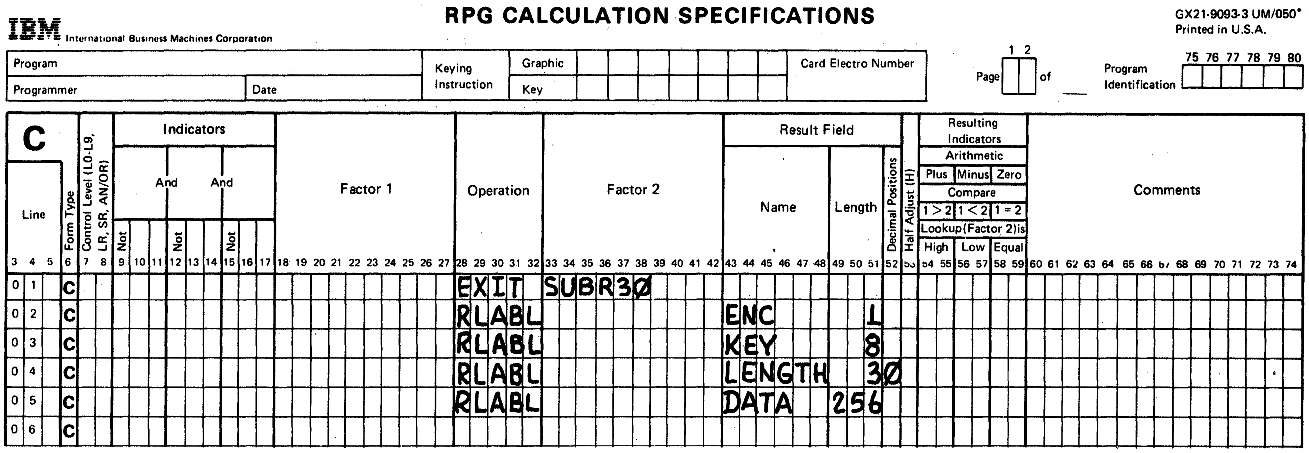
SUBR30 for RPG II

SUBR30 provides two functions: general encryption/decryption and specific encryption. Each function has a specific set of parameters.

General Encryption/Decryption

For the general encryption/decryption function, input data must be defined as alphameric because the resulting encrypted/decrypted data is a random rearrangement of the input data.

To call the general encryption/decryption function, make the following entries on the calculation specifications:



IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

GX21-9093-3 UM/050*
Printed in U.S.A.

Program | Keying Instruction | Graphic | Key | Card Electro Number | Page of | Program Identification 75 76 77 78 79 80

Programmer | Date

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators	Comments
01	C				EXIT	SUBR3Ø						
02	C				RLABL		ENC	1				
03	C				RLABL		KEY	8				
04	C				RLABL		LENGTH	3	Ø			
05	C				RLABL		DATA	256				
06	C											

ENC is a 1-character field that contains an E or a D:

E = Encryption is to be performed.

D = Decryption is to be performed.

To enter the appropriate code in the ENC field, you can use a MOVE operation before calling the subroutine.

KEY is an 8-byte field that contains the key used to encrypt or decrypt the data.

LENGTH is a 3-digit numeric field that specifies the total length of data to be encrypted or decrypted. The value defined for LENGTH must be less than or equal to the length defined for DATA. For example, if the value in the length field is 128, the length defined for data must be 128 or greater. DES encrypts and decrypts data in blocks of 8 bytes. Thus, the value for LENGTH should be an increment of 8. To enter the appropriate value in the LENGTH field, you can use a MOVE operation before calling the subroutine.

DATA is a field (up to 256 bytes long) that contains the data to be encrypted or decrypted. When SUBR30 returns control to your program, DATA contains the encrypted or decrypted data.

There is no explicit return code field for SUBR30. However, if an error occurs, SUBR30 overlays the ENC field with one of the following return codes:

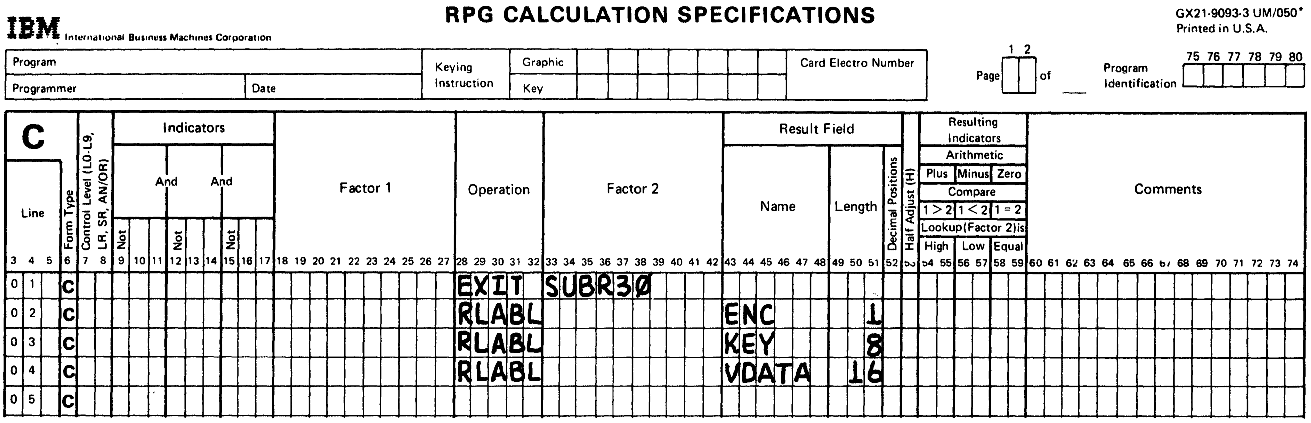
1 = The value of ENC is not E, D, or V (V indicates specific encryption).

3 = The value of LENGTH is not numeric data.

Specific Encryption

The specific encryption function treats character input data as if it were hexadecimal data. Specific encryption converts character input data to its hexadecimal equivalent, encrypts the hexadecimal equivalent, converts the hexadecimal results to character data, and passes the character data back to the calling routine. Input data must be in the range A through F (uppercase only) and 0 through 9 because hexadecimal data can have only these values.

To call the specific encryption function, make the following entries on the calculation specifications:



IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

GX21-9093-3 UM/050*
Printed in U.S.A.

Program | Programmer | Date | Keying Instruction: Graphic, Key | Card Electro Number | Page 1 2 of | Program Identification 75 76 77 78 79 80

Line (3-5)	Form Type (6)	Control Level (L0-L9, LR, SR, AN/OR) (7-8)	Indicators (Not, And, Not, And, Not) (9-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field Name (43-48)	Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Arithmetic Plus Minus Zero; Compare 1>2 1<2 1=2; Lookup(Factor 2)is High Low Equal (54-59)	Comments (60-74)
01	C				EXIT	SUBR3Ø						
02	C				RLABL		ENC	1				
03	C				RLABL		KEY	8				
04	C				RLABL		VDATA	16				
05	C											

ENC is a 1-character field that contains a V. V indicates that specific encryption is to be performed.

KEY is an 8-byte field that contains the key used to encrypt the data.

VDATA is a 16-character field that contains the data to be encrypted. When SUBR30 returns control to your program, VDATA contains the encrypted character data. No matter what length is defined for VDATA, SUBR30 assumes that VDATA is a 16-character, left-adjusted field.

There is no explicit return code field for SUBR30. However, if an error occurs, SUBR30 overlays the ENC field with one of the following return codes:

1 = The value of ENC was not V, E, or D.

2 = The data in VDATA is not valid data (0 through 9, or A through F).

Programming Considerations

- Subroutine parameters
  - DES encrypts or decrypts data in 8-byte blocks by using an 8-byte key. Unpredictable results can occur if the key is not defined as an 8-byte field. Unpredictable results can also occur if the value of LENGTH is not an increment of 8, or if the value of LENGTH is not equal to or less than the length of DATA and the length of DATA is not an increment of 8.
  - Unpredictable results can occur if LENGTH is not defined as a 3-digit numeric field.
  - The ENC field is overlaid only if an error occurs.

- Data definitions
  - For general encryption/decryption, SUBR30 treats KEY and DATA as bit strings, and DES encrypts/decrypts the bits. Therefore, you cannot assume that the resulting data is a particular data type; that is, you should define KEY and DATA as alphameric, but they may contain unprintable or nondisplayable data. ENC must be alphameric data, and LENGTH must be numeric data.
  - For specific encryption, SUBR30 makes no assumption about the format of the data defined for KEY. ENC and VDATA must be alphameric data. After using DES to encrypt VDATA, SUBR30 converts the resulting data to alphameric data before returning control to the calling program.

SUBR31 for COBOL

SUBR31 provides two functions, general encryption/decryption and specific encryption.

General Encryption/Decryption

The general encryption/decryption function encrypts or decrypts data based on a user-supplied key. The format for the COBOL call to SUBR31 for general encryption/decryption is:

```
CALL 'SUBR31' USING ENC,KEY,LENGTH,DATA
```

ENC is a 1-byte alphanumeric field that contains an E or a D. An E indicates that encryption is to be performed. A D indicates that decryption is to be performed.

KEY is an 8-byte field that contains the key to be used to encrypt the data.

LENGTH is a 3-byte field that specifies the total length of the data to be encrypted. The value defined for LENGTH must be less than or equal to the length defined for DATA. For example, if the value in the length field is 128, the length defined for data must be 128 or greater. DES encrypts data in 8-byte blocks. Thus, the value for LENGTH should be in increments of 8. SUBR31 accepts values of LENGTH up to 256. If LENGTH contains a value greater that 256, SUBR31 assumes that the value is 256.

DATA contains the data that is to be encrypted/decrypted. When control is returned by SUBR31 to your program, DATA contains the encrypted/decrypted data. The DATA field should be defined as binary (computational-4).

There is no explicit return code field for SUBR31. If an error occurs, SUBR31 writes a return code over the contents of the ENC field. The following are the return codes issued by SUBR31:

1 = The value of ENC was not E, D, or V.

3 = The value of LENGTH is not numeric.

Specific Encryption

The specific encryption function encrypts character data as if it were hexadecimal data. The input data is considered to be a character representation of hexadecimal data. SUBR31 translates the input data into its hexadecimal equivalent, encrypts the hexadecimal equivalent, converts the results into character data, and passes the character data back to your program.

Input data must be in the character data range A through F (uppercase only) and 0 through 9 because hexadecimal data can have only these values.

The format for the COBOL call to SUBR31 for specific encryption is:

```
CALL 'SUBR31' USING ENC,KEY,VDATA
```

ENC is a 1-byte alphanumeric field that contains a V. A V indicates that specific encryption is to be performed.

KEY is an 8-byte field that contains the key to be used to encrypt the data.

VDATA is a 16-byte alphanumeric field that contains the data to be encyrpted. When control is returned by SUBR31 to your program, VDATA contains the encrypted data.

There is no explicit return code field for SUBR31. If an error occurs, SUBR31 writes a return code over the contents of the ENC field. The following are the return codes issued by SUBR31:

1 = The value of ENC was not E, D, or V.

2 = The data in VDATA contains characters other than A through F or 0 through 9.

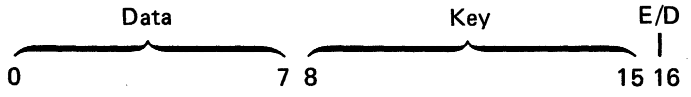
Programming Considerations

- Subroutine parameters
  - DES encrypts or decrypts data in 8-byte blocks by using an 8-byte key. Unpredictable results can occur if the key is not defined as an 8-byte field. Unpredictable results can also occur if the value of LENGTH is not an increment of 8, or if the value of LENGTH is not equal to or less than the length of DATA and the length of DATA is not an increment of 8.
  - Unpredictable results can occur if LENGTH is not defined as a 3-digit numeric field.
  - The ENC field is overlaid only if an error occurs.

- Data definitions
  - For general encryption/decryption, SUBR31 treats KEY and DATA as bit strings, and DES encrypts/decrypts the bits. Therefore, you cannot assume that the resulting data is a particular data type; that is, you should define KEY and DATA as alphanumeric, but they may contain unprintable or nondisplayable data. ENC must be alphanumeric data, and LENGTH must be numeric data.
  - For specific encryption, SUBR31 makes no assumption about the format of the data defined for KEY. ENC and VDATA must be alphanumeric data. After using DES to encyrpt VDATA, SUBR31 converts the resulting data to alphanumeric data before returning control to the calling program.

#SBDE for Assembler

When #SBDE is used, index register 1 must contain the address of the leftmost byte of a 17-byte formatted data stream. This formatted data stream is structured as follows:



Data is the data to be encrypted/decrypted.

Key is an 8-byte key used to encrypt the data.

E/D indicates whether data is to be encrypted or decrypted:

E = Encryption

D = Decryption

3601/4701 AND 3694 PROGRAMMING CONSIDERATIONS

The following information is required when writing an application program for a 3601/4701 or a 3694 that will communicate with the Finance subsystem.

Control Fields and Indicators

The following are the read type definitions and the read flag definiιions that the 3601/4701 or 3694 can receive when communicating with the Finance subsystem.

Read Type Definitions (SMSCRT)

Hex Value	Meaning
00	Ready
01	Positive DR1 response to data
05	Negative DR1 response to data
09	Positive DR1 response on a nondata message
0D	Negative DR1 response on a nondata message
10	Data
23	LU status (3601/4701 only)
41	Shutdown
82	Start data traffic
A0	Bind session
A1	Unbind session

Read Flag Definitions (SMSCRF)

Hex Value	Meaning
01	Definite response protocol (DR1)
05	Exception response protocol (DR1)

The following are the write type definitions and the write flag definitions that the 3601/4701 or 3694 can transmit to the Finance subsystem.

Write Type Definitions (SMSCWT)

Hex Value	Meaning
07	Negative response
10	Data
42	Shutdown complete
A0	Initiate session
A1	Terminate session
C2	Request shutdown (3694 only)

Write Flag Definitions (SMSCWF)

Hex Value	Meaning
01	Definite response protocol (DR1)
05	Exception response protocol (DR1)
0B	Definite response protocol (DR1) and first or last in chain
0D	Exeception response protocol (DR1) and first or last in chain

Message Types

The following are the message types supported by the Finance subsystem that can be transmitted to or received from the 3601/4701 or 3694.

Messages Transmitted to the 3601/4701 or 3694

Bind (session control): The Finance subsystem transmits a bind to initiate a logical work station session with a 3601/4701 or a logical session with a 3694.

Data (normal flow): A System/34 application program transmits data to the 3601/4701 or 3694 application program after the Finance subsystem transmits the Start Data Traffic message and receives a positive response from the 3601/4701 or 3694.

LU Status (normal flow control): For the 3601/4701 only, the Finance subsystem transmits LU status to the 3601/4701 application program to provide the 3601/4701 application program with information about the status of the session. Following is the format of the LUSTAT message:

Bytes Meaning

1 and 20000 = Indicates that user status follows
3 and 40000 = SSP-ICF session does not exist
    0001 = Program terminated normally
    0002 = Program terminated abnormally
    0003 = Resource now available
    0004 = Disable pending
    10xx = Procedure start failed: *xx* can
        indicate one of the following:
            00 = No meaning
            01 = Sign-on had invalid library name
            02 = Disk I/O error in security record
            03 = Job initiation stopped by system operator
            04 = Resources not available to start job now
            05 = Resource security file not found
            06 = Cannot log security information
                to history file
            07 = No user list in resource security
                file for library
            08 = Unauthorized request for a user library
            09 = Invalid procedure name

Ready (independent of session): The Finance subsystem passes the ready message to the application program to indicate that the 3601/4701 or 3694 application program can transmit an initiate message.

Shutdown (expedited flow control): The Finance subsystem transmits the shutdown message to begin terminating a logical work session. The 3601/4701 or 3694 application program should stop sending data and prepare for session termination. When the 3601/4701 or 3694 is ready for session termination, it should send a shutdown complete message to the Finance subsystem.

Start Data Traffic (session control): The Finance subsystem transmits the start data traffic message to the 3601/4701 or 3694 application program to indicate that the session is established and communications can begin.

Unbind (session control): The Finance subsystem transmits the unbind message to the 3601/4701 or 3694 application program to terminate a logical work station session.

Received Messages

Data (normal flow): The 3601/4701 or 3694 application program can transmit data after the session is established and after it receives the start data traffic message.

Initiate (independent of session): The 3601/4701 or 3694 application program must transmit the initiate message to the Finance subsystem. The initiate message causes the Finance subsystem to transmit the bind message to initiate a session. The initiate message must be in the following format:

Field	**Description**
1-10	Initiate header: must be hexadecimal 004040404040404040F3
11	Symbolic name length: must be hexadecimal 08
12-19	Symbolic name: must be 'SFSbbbbb' for the 3601/4701; 'DTNCHXVS' for the 3694
20-21	Must be hexadecimal 0000
22	Length (in binary) of the optional data following this byte
23-n	Optional data

3601/4701 Optional Data: If optional data is transmitted, it contains identification and authorization from the 3601/4701 to control access to System/34 resources. The data must be in the following format and must conform to System/34 syntax:

[userid],[password],[library name]

where:

userid is the identification (1 to 8 characters) of the logical work station.

password is the password (4 characters) required to access System/34 resources.

library is the name (1 to 8 characters) of the library that can contain procedures to be started by the logical work station.

3694 Optional Data: If the 3694 sends optional data, there must be at least 10 bytes of data in the following format:

Field	Description
1	Request code
2-4	User ID; these 3 characters are added to USER to form the user ID.
5-8	Password
9-10	These 2 characters are added to LIB to form the library name. If these characters are 00, the library name is #LIBRARY.

Request Shutdown; 3694 only (expedited flow): The 3694 application program transmits the request shutdown message to request that the session be terminated.

Shutdown Complete; 3601/4701 only (expedited flow control): The 3601/4701 application program transmits the shutdown complete message to indicate that it is ready for session termination.

Terminate; 3601/4701 only (independent of session): The 3601/4701 application program transmits the terminate message to request that the session be terminated. The format of the terminate message must be:

Field	Description
1-2	Terminate header: must be hexadecimal 00F3
3	Symbolic name length: must be hexadecimal 08
4-11	Symbolic name: must be the characters SFSbbbbb

Responses

The Finance subsystem can return one of the following negative responses (-RSP) when it receives an initiate message:

Hexadecimal Response **Reason**

08090000 Mode inconsistency: The LU-to-LU session to be started by this initiate message has not been reset.

08380000 Queueing not supported: Bit 7 of byte 3 of the initiate message is on.

0835xxxx Invalid parameter in the initiate message. *xxxx* is an index (beginning at 0) to the byte in the initiate message that caused the error. This response can be caused by one of the following:

- Byte 3 (other than bit 7) is not 00.
- Bytes 14-21 do not contain SFSbbbbb (3601/4701) or DTNCHXVS (3694).
- Byte 22 is not 00.
- Byte 23 is not 00.
- The userid is longer than 8 bytes.
- The password is not 4 bytes in length.
- The library name is longer than 8 bytes.

08120000 Insufficient resource: The requested resources are temporarily unavailable. The initiate message can be retransmitted.

08040001 Invalid password: The password is not valid for the userid specified.

080Fxxxx End user not authorized: *xxxx* can be one of the following:

0001 = Invalid user ID
0002 = Not authorized for specified library
0003 = User ID authorized for menu input only
0004 = No user list in resource security library

Hexadecimal Response	Reason
084Bxxxx	Requested resources not available: *xxxx* can be one of the following: 0000 = No meaning 0001 = Invalid library 0002 = Password security file not found on disk 0003 = Disk I/O error in security record 0004 = No resources available to start job now 0005 = Resource security file not found 0006 = Cannot log security information to history file
08320000	Invalid count field: User data is less than 10 bytes (3694 only) in length.

The Finance subsystem can transmit the following negative responses to data messages:

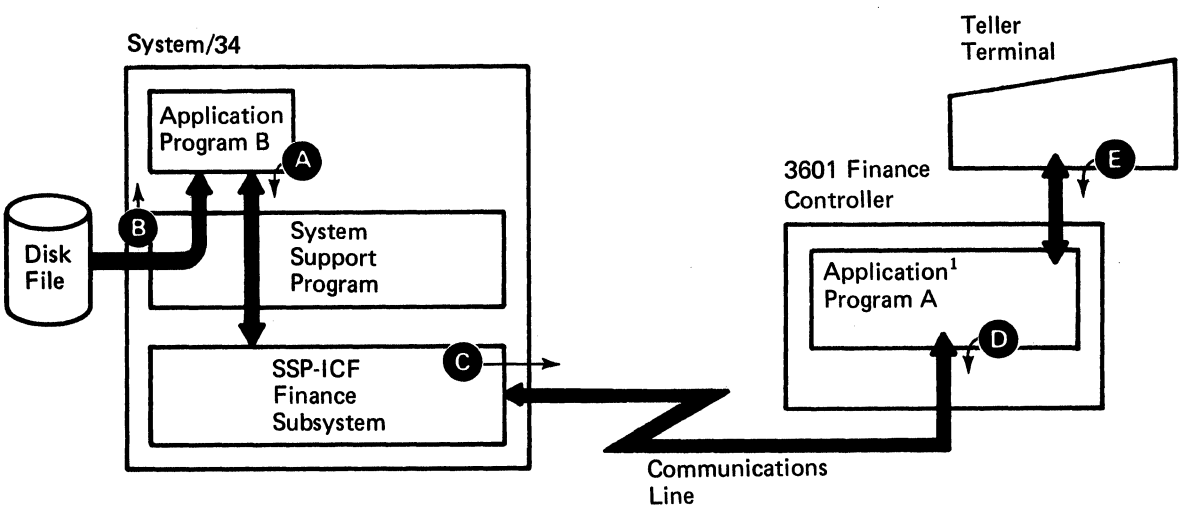
Hexadecimal Response	Meaning
0809xxxx	Mode inconsistency: *xxxx* can indicate one of the following conditions: 0001 = The SSP-ICF session is pending active 0002 = The SSP-ICF session is active 0003 = The SSP-ICF session is reset 0004 = The SSP-ICF session is not available
08120001	Insufficient resource: The buffer is currently in use.
084C0001	Insufficient resource: The buffer is too small.

3694 PROGRAMMING CONSIDERATIONS

Communications between System/34 application programs and the CHX/3694 program is controlled by function management headers (FMH). For information about the function management headers, see the *Check Processing Executive/VS: Program Logic Manual*, LY20-2556. For other information about programming for the 3694, see the *Check Processing Executive/VS: Program Reference and Operations Manual*, SH20-2496 and the *Check Processing Executive/3694: Program Reference and Operations Manual*, SH20-2495.

How to Write Programs that Use the Finance Subsystem

The following inquiry application is used in the programming examples:



1. An operator enters an item number on the teller terminal Ⓔ connected to the 3601. Application program A in the 3601 sends the item number to application program B in the System/34 Ⓓ.

2. Program B reads the number from program A and searches the disk data file for the item Ⓑ.

3. If the item is in the file, program B sends the information about the item to program A Ⓐ and Ⓒ. If the item is not in the file, program B sends the message *** ITEM NOT FOUND to program A.

4. Program A displays either the information about the item requested or the message *** ITEM NOT FOUND.

[1] The application program in the 3601 for this example is the Online Terminal Support for System/34 program.

Program B in this example is similar to program B described in Chapter 7 for the Intra subsystem. The changes to the program required for the Finance subsystem are shown in this chapter following the configuration and OCL examples. If you have not read the example in Chapter 7, see *How to Write Programs that Use the Intra Subsystem* in Chapter 7 for a description of how to write a program that uses the Interactive Communications Feature, then read this example.

Configuration Parameters

The following configuration parameters are used for this example. For a description of the configuration parameters, see *Setting up the Finance Subsystem* at the beginning of this chapter.

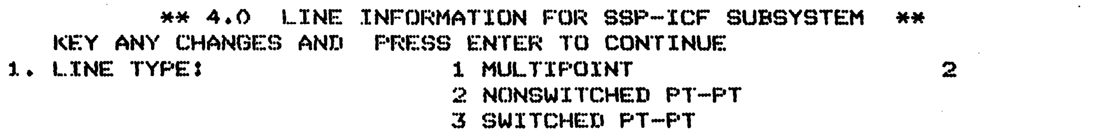
```
CREATE/EDIT                   ** 1.0 SUBSYSTEM MEMBER CONFIGURATION **
      1. SUBSYSTEM CONFIGURATION MEMBER NAME :               SFSFF
      2. SUBSYSTEM LIBRARY NAME :                            ICFLIBR
         1 CREATE NEW MEMBER                    4 DELETE A MEMBER
         2 EDIT EXISTING MEMBER                 5 REVIEW A MEMBER
         3 CREATE NEW MEMBER FROM EXISTING MEMBER
      3. ENTER SELECTION :      2
```

```
            ** 2.0 COMMON SSP-ICF PARAMETERS FOR EACH SUBSYSTEM **
   KEY ANY CHANGES AND  PRESS ENTER TO CONTINUE
1. SSP-ICF COMMON QUEUE SPACE                  (2 - 42K)      02
2. DEFINE THE SUBSYSTEM TYPE                                  10
     1 INTRA                              2 BSC IMS/IRSS
     3 BSCEL                              4 BSC CICS
     5 BSC CCP                            6 SNA UPLINE
     7 SNA PEER                           8 BSC 3270
     9 SNA 3270                           10 FINANCE
```

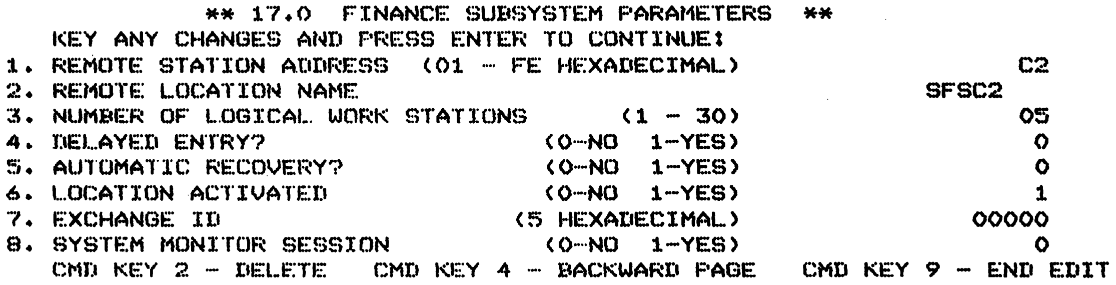
```
           ** 3.0  GENERAL SUBSYSTEM PARAMETERS   **
KEY ANY CHANGES AND PRESS ENTER TO CONTINUE:
     1. LOCATION NAME                                          SFS
     2. SUBSYSTEM QUEUE SPACE                      (0-40K)           00
     3. SUBSYSTEM SUPPORT SWAPPABLE?     (0-NO  1-YES)                1
```

```
               ** 3.1  SDLC GENERAL SUBSYSTEM PARAMETERS   **
   KEY ANY CHANGES AND PRESS ENTER TO CONTINUE:


2. SDLC RECEIVE BUFFER SIZE               (2 OR 4K)                    2
3. SDLC TRANSMIT BUFFER SIZE              (2 OR 4K)                    2
```

Configuration Parameters (continued)



```
          ** 4.0  LINE INFORMATION FOR SSP-ICF SUBSYSTEM  **
   KEY ANY CHANGES AND  PRESS ENTER TO CONTINUE
1. LINE TYPE:                  1 MULTIPOINT                    2
                               2 NONSWITCHED PT-PT
                               3 SWITCHED PT-PT
```



```
             ** 17.0  FINANCE SUBSYSTEM PARAMETERS  **
   KEY ANY CHANGES AND PRESS ENTER TO CONTINUE:
1. REMOTE STATION ADDRESS   (01 - FE HEXADECIMAL)                 C2
2. REMOTE LOCATION NAME                                        SFSC2
3. NUMBER OF LOGICAL WORK STATIONS       (1 - 30)                 05
4. DELAYED ENTRY?                   (0-NO  1-YES)                  0
5. AUTOMATIC RECOVERY?              (0-NO  1-YES)                  0
6. LOCATION ACTIVATED               (0-NO  1-YES)                  1
7. EXCHANGE ID                     (5 HEXADECIMAL)             00000
8. SYSTEM MONITOR SESSION           (0-NO  1-YES)                  0
   CMD KEY 2 - DELETE   CMD KEY 4 - BACKWARD PAGE   CMD KEY 9 - END EDIT
```

OCL Statements

The following OCL statements are used for the BASIC example:

```
BASICR SFSBAS,ITEMBAS,30
```

The following OCL statements are used for the COBOL example:

```
// LOAD COBSFS
// FILE NAME-FILEB
// RUN
```

The following OCL statements are used for the RPG II example:

```
// LOAD RPGSFS
// FILE NAME-FILEB
// RUN
```

Note: A SESSION statement is not required. The 3601/4701 sends a remote procedure start request, which evokes program B and begins a session.

Programming Examples for Program B

The following examples show the BASIC, COBOL, and RPG II programs for the finance inquiry application.

BASIC Programming Example

```
00010 !***********************************************************
00020 !*   SFSBAS RECIEVES AN ITEM NUMBER FROM THE REQUESTOR     *
00030 !*   THROUGH ICF. THE FILE 'FILEB' IS SEARCHED. IF THE     *
00040 !*   ITEM IS FOUND, THE ITEM INFORMATION IS RETURNED TO    *
00050 !*   THE REQUESTOR. IF THE ITEM IS NOT FOUND,              *
00060 !*   '*** ITEM NOT FOUND' IS RETURNED TO THE REQUESTOR.    *
00070 !***********************************************************
00080 DIM ITEM$*23
00090 OPEN #1: "NAME=FILEB,SHR,RECL=50,KEYL=23,KEYP=1,RANDOM",KEYED,INPUT
00100 OPEN #2: "WS,RECL=256" IOERR ICFERR
00110 !*---------------------------------------------------------*
00120 !*     SET UP FLAGS                                        *
00130 !*     CSFLAG$ - CLEAR SCREEN FLAG                         *
00140 !*     NLINE$ - NEW LINE FLAG                              *
00150 !*     END1$, END11$, END2$, END22$ - END OF DATA FLAGS    *
00160 !*---------------------------------------------------------*
00170 CSFLAG$=HEX$("OC")
00180 NLINE$=HEX$("15")
00190 END1$=HEX$("FF")
00200 END11$=HEX$("FF")
00210 END2$=HEX$("FF")
00220 END22$=HEX$("FF")
00230 !*---------------------------------------------------------*
00240 !*                  ACCEPT INPUT SENT                      *
00250 !*---------------------------------------------------------*
00260 OPCODE$="ACCEPT"
00270 WAITIO IOERR ICFERR
00280 READ #2,USING 290: ITEM$ IOERR ICFERR
00290 FORM V 23
00300 ITEM$=RPAD$(ITEM$,23)
00310 !*---------------------------------------------------------*
00320 !*      READ FILEB FILE, USING ITEM NUMBER SENT AS KEY     *
00330 !*---------------------------------------------------------*
00340 READFILE:READ #1,USING 350,KEY=ITEM$: QTY1,QTY2,QTY3,QTY4 NOKEY &
      &ERRORKEY
00350   FORM X 26,4*N 6
00360 !*---------------------------------------------------------*
00370 !*             SEND INFORMATION TO REQUESTOR               *
00380 !*---------------------------------------------------------*
00390 SENDDATA:OPCODE$="SENDNI"
00400   WRITE #2,USING 410,FORMAT "$$SENDNI": "      ON N N  N  ",CSFLAG$.&
     &"ITEM# - ",ITEM$,NLINE$,"QTY 1 - ",QTY1,NLINE$,"QTY 2 - ",QTY2,NLINE$,&
     &"QTY 3 - ",QTY3,NLINE$,"QTY 4 - ",QTY4,END1$,END11$,END2$,END22$
00410   FORM C 16,C 1,C 8,C 23,C 1,C 8,N 6,C 1,C 8,N 6,C 1,C 8,N 6,C 1.&
      &C 8,N 6,4*C 1
00420 !*---------------------------------------------------------*
00430 !*                  CLOSE FILES AND END                    *
00440 !*---------------------------------------------------------*
00450 CLSFILES:CLOSE #1::CLOSE #2:
00460   STOP
00470 !*---------------------------------------------------------*
00480 !*                    ERRORKEY ROUTINE                     *
00490 !*---------------------------------------------------------*
00500 ERRORKEY:OPCODE$="SEND"
00510   WRITE #2,USING 520,FORMAT "$$SENDNI": "      ON N N  N  ",CSFLAG$.&
     &"*** ITEM NOT FOUND",END1$,END11$,END2$,END22$
00520   FORM C 16,C 1,C 18,4*C 1
00530   GOTO CLSFILES
00540 !*---------------------------------------------------------*
00550 !*                     ICFERR ROUTINE                      *
00560 !*---------------------------------------------------------*
00570 ICFERR:IF RETCODE$="0100" THEN CONTINUE
00580   PRINT #255,USING 590: "RETURN CODE ",RETCODE$," OPCODE IS ",&
      &OPCODE$," ITEM NUMBER IS ",ITEM$
00590   FORM SKIP 2,C 12,C 4,C 11,C 6,C 16,C 23
00600   STOP
```

Flags are set up to control the teller terminal display.

Read the item number from the 3601.

The item number is padded with blanks if less than 23 characters.

Read the data file using the item number from the 3601 as a key.

If the item is in the file, send the data and flags to the 3601. If the item is not in the file, send the error message.

If the return code is 0100, the item number has been received successfully. The program continues and reads the item number. If the return code is not 0100, the return code and other information are printed and the program ends.

```
*****************************************************************
*                                                               *
*    COBSFS - ITEM INQUIRY WRITTEN IN COBOL                     *
*                                                               *
*****************************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. COBSFS.

 ENVIRONMENT DIVISION.

 CONFIGURATION SECTION.
 SOURCE-COMPUTER.  IBM-S34.
 OBJECT-COMPUTER.  IBM-S34.

 INPUT-OUTPUT SECTION.
 FILE-CONTROL.

     SELECT TRANSACTION-FILE
        ASSIGN TO WORKSTATION,
        ORGANIZATION IS TRANSACTION,
        FILE STATUS IS WS-FS, ICF-FS,
        CONTROL-AREA IS WS-CONTROL-AREA.

     SELECT FILEB-FILE ASSIGN TO DISK-FILEB,
        ORGANIZATION IS INDEXED, ACCESS IS RANDOM,
        RECORD KEY IS FILEB-NUMBER.

     SELECT PRINT-FILE ASSIGN TO PRINTER-PRINTER.

/
 DATA DIVISION.
 FILE SECTION.
 FD  TRANSACTION-FILE, LABEL RECORDS ARE OMITTED.
 01  TRANSACTION-RECORD          PIC X(256).

 FD  FILEB-FILE, LABEL RECORDS ARE STANDARD.
 01  FILEB-RECORD.
     03  FILEB-NUMBER            PIC X(23).
     03  FILLER                  PIC XXX.
     03  FILEB-QTYS.
       05  FILEB-QTY-1           PIC 9(6).
       05  FILEB-QTY-2           PIC 9(6).
       05  FILEB-QTY-3           PIC 9(6).
       05  FILEB-QTY-4           PIC 9(6).

 FD  PRINT-FILE, LABEL RECORDS ARE OMITTED.
 01  PRINT-RECORD                PIC X(132).
```

```
WORKING-STORAGE SECTION.

01  ICF-ITEM-NUMBER-IN          PIC X(23) VALUE SPACES.

01  CONTROL-CHARACTERS.
    03  CLEAR-SCREEN-CC         PIC 9999 COMP-4 VALUE 0012.
    03  CLEAR-SCREEN-X    REDEFINES  CLEAR-SCREEN-CC.
        05  FILLER              PIC X.
        05  CLEAR-SCREEN        PIC X.
    03  NEW-LINE-CC             PIC 9999 COMP-4 VALUE 0021.
    03  NEW-LINE-X        REDEFINES  NEW-LINE-CC.
        05  FILLER              PIC X.
        05  NEW-LINE            PIC X.
    03  END-OF-DATA-CC          PIC 99999 COMP-4 VALUE 65535.
    03  END-OF-DATA-X  REDEFINES  END-OF-DATA-CC.
        05  FILLER              PIC XX.
        05  END-OF-DATA         PIC XX.

01  WS-DUMMY-AREAS.
    03  WS-CONTROL-AREA.
      05  AID-BYTE              PIC 99.
      05  ICF-SESSION           PIC XX.
      05  FILLER                PIC X(8).
    03  RETURN-CODES.
      05  WS-FS                 PIC XX.
      05  ICF-FS.
         07  MAJOR-RETURN-CODE  PIC XX.
         07  MINOR-RETURN-CODE  PIC XX.

01  ICF-RECORD-OUT.
    03  DATA-LENGTH             PIC X(4) VALUE '0112'.
    03  RECORD-HEADER           PIC X(16)
                           VALUE '     ON N N  N  '.
    03  CLEAR-SCREEN-FLAG-1     PIC X.
    03  FILLER                  PIC X(8) VALUE 'ITEM# - '.
    03  ICF-ITEM-NUMBER-OUT     PIC X(23).
    03  ICF-QTYS.
      05  NEW-LINE-1            PIC X.
      05  FILLER                PIC X(8) VALUE 'QTY 1 - '.
      05  ICF-QTY-1             PIC 9(6).
      05  NEW-LINE-2            PIC X.
      05  FILLER                PIC X(8) VALUE 'QTY 2 - '.
      05  ICF-QTY-2             PIC 9(6).
      05  NEW-LINE-3            PIC X.
      05  FILLER                PIC X(8) VALUE 'QTY 3 - '.
      05  ICF-QTY-3             PIC 9(6).
      05  NEW-LINE-4            PIC X.
      05  FILLER                PIC X(8) VALUE 'QTY 4 - '.
      05  ICF-QTY-4             PIC 9(6).
    03  END-OF-DATA-FLAG-1      PIC XX.
    03  END-OF-DATA-FLAG-2      PIC XX.

01  ERROR-RECORD-OUT.
    03  DATA-LENGTH             PIC XXXX VALUE '0039'.
    03  RECORD-HEADER           PIC X(16)
                           VALUE '     ON N N  N  '.
    03  CLEAR-SCREEN-FLAG-2     PIC X.
    03  FILLER                  PIC X(18)
                           VALUE '*** ITEM NOT FOUND'.
    03  END-OF-DATA-FLAG-11     PIC XX.
    03  END-OF-DATA-FLAG-22     PIC XX.

01  PRINT-CODES.
    03  FILLER                  PIC X(14) VALUE 'RETURN CODE '.
    03  PRINT-RETURN-CODE       PIC XXXX.
    03  FILLER                  PIC X(11)  VALUE ' OPCODE IS '.
    03  OPCODE                  PIC X(6).
```

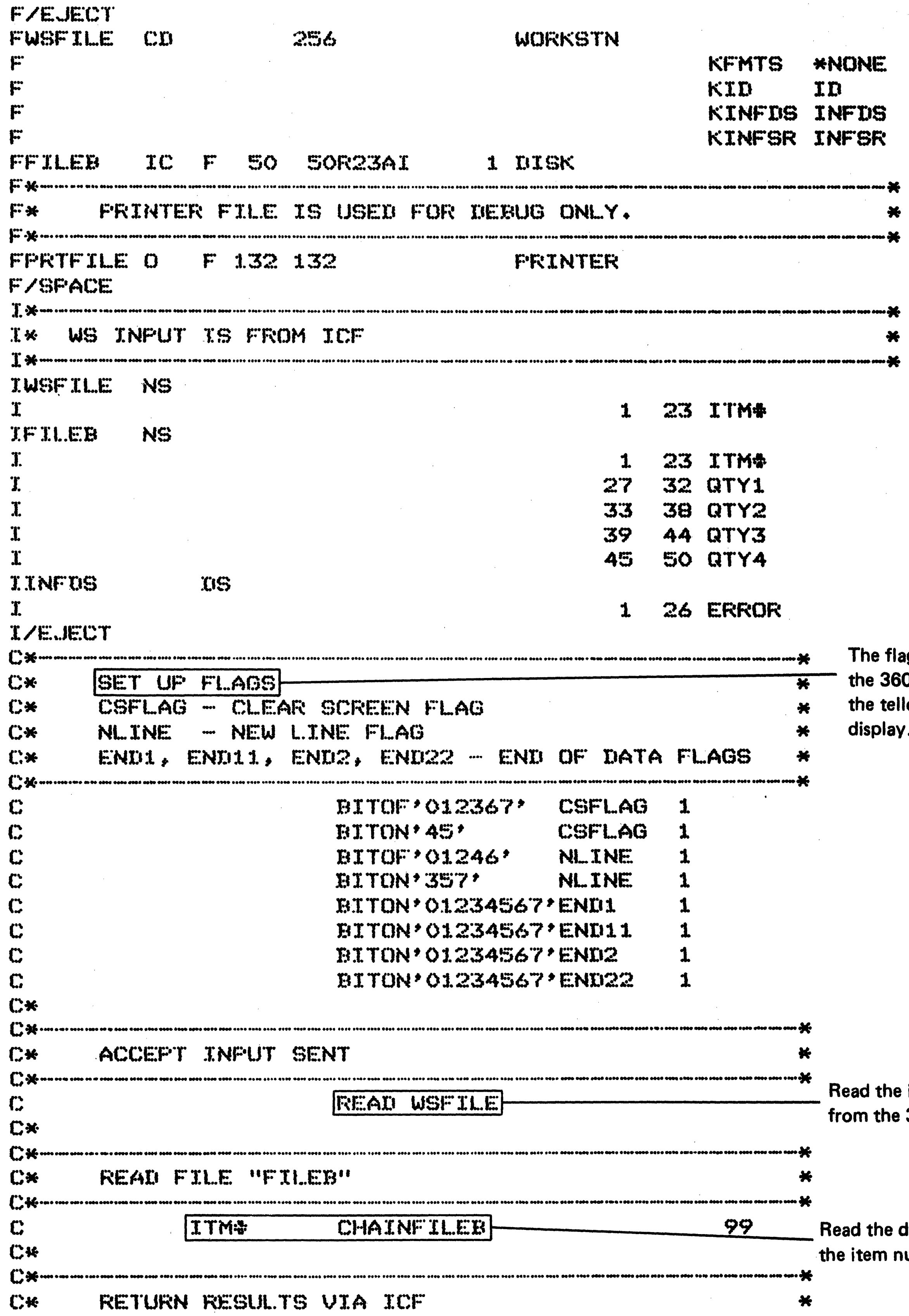
The control characters are added to control the teller terminal display.

COBOL Programming Example (continued)

```
 PROCEDURE DIVISION.
 OPEN-FILES.
     OPEN I-O TRANSACTION-FILE.
     OPEN OUTPUT PRINT-FILE.
     OPEN INPUT FILEB-FILE.
*--------------------------------------------------------------*
*    SET UP FLAGS.                                             *
*--------------------------------------------------------------*
     MOVE CLEAR-SCREEN TO CLEAR-SCREEN-FLAG-1,
                          CLEAR-SCREEN-FLAG-2.
     MOVE NEW-LINE TO NEW-LINE-1, NEW-LINE-2,
                      NEW-LINE-3, NEW-LINE-4.
     MOVE END-OF-DATA  TO  END-OF-DATA-FLAG-1,
                           END-OF-DATA-FLAG-2,
                           END-OF-DATA-FLAG-11,
                           END-OF-DATA-FLAG-22.
*--------------------------------------------------------------*
*    ACCEPT ITEM NUMBER SENT BY REQUESTOR.                     *
*--------------------------------------------------------------*
     READ TRANSACTION-FILE RECORD INTO ICF-ITEM-NUMBER-IN.
     MOVE 'ACCEPT' TO OPCODE.
     PERFORM WRITE-CODES THRU WRITE-CODES-END.
*--------------------------------------------------------------*
*    READ FILEB FILE, USING ITEM NUMBER SENT AS KEY.           *
*--------------------------------------------------------------*
 READ-FILEB-FILE.
     MOVE SPACES TO FILEB-RECORD.
     MOVE ICF-ITEM-NUMBER-IN TO FILEB-NUMBER.
     READ FILEB-FILE,
       INVALID KEY
         MOVE ERROR-RECORD-OUT TO TRANSACTION-RECORD,
         GO TO SEND-DATA.
     MOVE FILEB-NUMBER TO ICF-ITEM-NUMBER-OUT.
     MOVE FILEB-QTY-1 TO ICF-QTY-1.
     MOVE FILEB-QTY-2 TO ICF-QTY-2.
     MOVE FILEB-QTY-3 TO ICF-QTY-3.
     MOVE FILEB-QTY-4 TO ICF-QTY-4.
     MOVE ICF-RECORD-OUT TO TRANSACTION-RECORD.
*--------------------------------------------------------------*
*    SEND INFORMATION TO REQUESTOR.                            *
*--------------------------------------------------------------*
 SEND-DATA.
     WRITE TRANSACTION-RECORD FORMAT IS '$$SENDNI',
        TERMINAL IS ICF-SESSION.
     MOVE 'SENDNI' TO OPCODE.
     PERFORM WRITE-CODES THRU WRITE-CODES-END.
*--------------------------------------------------------------*
*    RELEASE SESSION, CLOSE FILES AND END.                     *
*--------------------------------------------------------------*
 CLOSE-FILES.
     DROP ICF-SESSION FROM TRANSACTION-FILE.
     CLOSE TRANSACTION-FILE, FILEB-FILE, PRINT-FILE.
     STOP RUN.


 WRITE-CODES.
     MOVE ICF-FS TO PRINT-RETURN-CODE.
     WRITE PRINT-RECORD FROM PRINT-CODES
        AFTER ADVANCING 2 LINES.
 WRITE-CODES-END.
```

Read the data from the 3601.

Read the data file using the item number from the 3601. If the item is in the file, send the data to the 3601. If the data is not in the file, send the message ITEM NOT FOUND.



```
F/EJECT
FWSFILE  CD        256            WORKSTN
F                                              KFMTS  *NONE
F                                              KID    ID
F                                              KINFDS INFDS
F                                              KINFSR INFSR
FFILEB   IC  F  50  50R23AI     1 DISK
F*---------------------------------------------------------*
F*    PRINTER FILE IS USED FOR DEBUG ONLY.                 *
F*---------------------------------------------------------*
FPRTFILE O   F 132 132            PRINTER
F/SPACE
I*---------------------------------------------------------*
I*  WS INPUT IS FROM ICF                                   *
I*---------------------------------------------------------*
IWSFILE  NS
I                                        1  23 ITM#
IFILEB   NS
I                                        1  23 ITM#
I                                       27  32 QTY1
I                                       33  38 QTY2
I                                       39  44 QTY3
I                                       45  50 QTY4
IINFDS       DS
I                                        1  26 ERROR
I/EJECT
C*---------------------------------------------------*
C*    SET UP FLAGS                                   *
C*    CSFLAG - CLEAR SCREEN FLAG                     *
C*    NLINE  - NEW LINE FLAG                         *
C*    END1, END11, END2, END22 - END OF DATA FLAGS   *
C*---------------------------------------------------*
C                     BITOF'012367'  CSFLAG  1
C                     BITON'45'      CSFLAG  1
C                     BITOF'01246'   NLINE   1
C                     BITON'357'     NLINE   1
C                     BITON'01234567'END1    1
C                     BITON'01234567'END11   1
C                     BITON'01234567'END2    1
C                     BITON'01234567'END22   1
C*
C*---------------------------------------------------*
C*    ACCEPT INPUT SENT                              *
C*---------------------------------------------------*
C                     READ WSFILE
C*
C*---------------------------------------------------*
C*    READ FILE "FILEB"                              *
C*---------------------------------------------------*
C           ITM#      CHAINFILEB                99
C*
C*---------------------------------------------------*
C*    RETURN RESULTS VIA ICF                         *
```

The flags are sent to the 3601 to control the teller terminal display.

Read the item number from the 3601.

Read the data file using the item number.

```
C*---------------------------------------------------*
C                     SETON                     03
C                     EXCPT
C*
C*---------------------------------------------------*
C*    RELEASE SESSION                               *
C*---------------------------------------------------*
C           ID        REL  WSFILE
C                     SETON                     LR
C*
C*****************************************************
C*                                                   *
C*                  I N F S R                        *
C*                                                   *
C*****************************************************
CSR         INFSR     BEGSR
C                     DEBUGPRTFILE   ERROR
C                     ENDSR'*CANCL'
C/EJECT
O*---------------------------------------------------*
O*    OUTPUT $$SENDNI. IF ITEM IS NOT FOUND,        *
O*    INDICATOR 99 IS ON, AND OUTPUT                *
O*    ***ITEM NUMBER NOT FOUND                      *
O*---------------------------------------------------*
OWSFILE  E        03N99
O                                   K8 '$$SENDNI'
O                                    4 '0112'
O                                   20 '     ON N N  N  '
O                         CSFLAG    21
O                                   29 'ITEM# - '
O                         ITM#      52
O                         NLINE     53
O                                   61 'QTY 1 - '
O                         QTY1      67
O                         NLINE     68
O                                   76 'QTY 2 - '
O                         QTY2      82
O                         NLINE     83
O                                   91 'QTY 3 - '
O                         QTY3      97
O                         NLINE     98
O                                  106 'QTY 4 - '
O                         QTY4     112
O                         END1     113
O                         END11    114
O                         END2     115
O                         END22    116
O        E        03 99
O                                   K8 '$$SENDNI'
O                                    4 '0039'
O                                   20 '     ON N N  N  '
O                         CSFLAG    21
O                                   39 '*** ITEM NOT FOUND'
O                         END1      40
O                         END11     41
O                         END2      42
O                         END22     43
```

Send the data or the ITEM NOT FOUND message to the 3601.

Finance Subsystem Return Codes

This part of Chapter 13 describes all the return codes that are valid for the Finance 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 four-digit value; the first two digits contain the major code, and the last two 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).

Note: In the return code descriptions, *your program* refers to the local System/34 application program that initiates the operation and receives the return code from the subsystem. The *remote program* refers to the application program in the remote (or host) system with which the System/34 application program is communicating through SSP-ICF.

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 Finance subsystem. Although all the operations shown are valid for Finance, their validity also depends on the logical sequence of communications events occurring between the System/34 and the remote system.

Input Operations to Your Program	**Output Operations from Your Program**	**Combined Operations in Your Program**
Accept input		
	Acquire[1]	
	End of session	
Get Get attributes[2]		
Invite		
	Put Put end of chain	Put then get[3] Put then invite
	Put FMH	Put FMH then get[3] Put FMH then invite
	Release	
	Set timer[4]	

[1] Normally, 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.

[2] Valid only in assembler and COBOL languages.

[3] Valid only in assembler language.

[4] For BASIC and RPG II programs, the set 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.

Major Codes 00-03 – Operation completed successfully.

General Description: The input or output operation issued by your program was completed successfully. The operation was one of the following:

- 0000 – The successful operation simply sent or received some data.
- 0100 – The successful operation was with a new requester. The new requester is a program on a *remote* system that initiated a session with your program by sending to the *local* system a procedure start request. The procedure start request was sent in the form of an *EXEC procedure start statement.
- 0200 – The operation was successful, but a stop system request or a disable subsystem request is now pending. Therefore, no new sessions using the subsystem can be initiated.
- 0300 – A set timer operation was performed successfully, but no data was sent or received.

General Considerations: Your program should continue with the next operation.

Code **Indication/Action**

0000 **Normal Indication:** For *input* operations performed by your program, some data was received successfully. For *output* operations performed by your program, the last output operation was completed successfully.

Normal Action: For both input and output operations, your program can issue an input, output, or end of session operation.

0100 **Normal Indication:** A procedure start request, sent by a remote program, initiated a new session with your 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.

If the first operation just performed by your program was an *input* operation, the operation was completed successfully and your program may have received some data from the new requester. If any data was received from the remote program, it was included in the incoming procedure start request statement. (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.)

If the first operation just performed by your program was an *output* operation and your program is an SRT program, the operation sent some data to the new requester. However, although the operation did complete successfully, if the procedure start statement also included data for your program, that data is lost.

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[1] for the new requester, and issue an input or output operation.

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. This return code also indicates that some data was received successfully by your program.

Normal Action: 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. Your program can now issue an input, output, or end of session operation.

0310 **Normal Indication:** The time interval specified by a set timer operation in your program has expired. (No data was sent or received with the operation.)

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.

[1] For some situations, no record keeping for the session is necessary. In other situations, you should record the session ID of the new requester. You may also want to keep a table containing the IDs of all active requesters, or to maintain a history log of all requests.

Major Codes 04-34 – Miscellaneous program errors.

Error Description: The operation just attempted by your program failed, or an output exception occurred.

- An operation may have failed because it was issued at the wrong time or because a data record was too long.
- An output exception may have occurred because your program attempted to send output when it should be receiving the output that has already been sent by the remote program.

Recovery Action: Refer to the individual return code descriptions for the appropriate recovery actions.

Code Indication/Action

0412 **Normal (Exception) Indication:** 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 remote program. Your program's output was not sent and should be sent later, after the remote program's data (still waiting in the subsystem input buffer) has been received.

Note: If your program issues another output operation, an error return code of 831C will be received.

Normal Action: Issue an input operation to receive the data waiting in the subsystem input buffer.

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 a new requester) 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 requesting program have been completed.

3401 **Error Indication:** This input operation was rejected because the record length of the data sent by the remote 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/34 Messages Guide* 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 (in COBOL and assembler only) a set timer operation, or by using the TIMER intrinsic function (in BASIC only); (2) continue local processing; or (3) terminate.
- If the session should be started again after the subsystem is enabled, it must be reacquired by your program or restarted by the remote 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 either in this subsystem or in the interrupt handler.

Recovery Action: This subsystem has been disabled; it must be enabled again before communications can resume. Your program can continue local processing, wait[1] to reissue the acquire operation, or terminate.

If more than one subsystem was active when the error occurred, all subsystems that were active when the error occurred should be disabled. (Note that all other active Finance subsystems are automatically disabled when the error occurs; all other types of active subsystems must be manually disabled.)

If all subsystems (of all types) on the system are *not* disabled to recover from the processor check, the common queue space used by the failing subsystem *cannot* be freed. And if it is not freed, that space is wasted, and an indication of insufficient common queue space being available can occur. The indication can occur as a message when the failing subsystem is reenabled or when a different subsystem is enabled. The indication can also occur as an 8315 return code for any subsystem that is performing an output operation in a session.

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 remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing; it can wait[1] until the subsystem has been reenabled, then reissue the acquire operation; or it can terminate.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

8083 **Error Indication:** An MLCA (multiline communications adapter) controller check occurred on an *output* operation. The subsystem has been disabled.

Recovery Action: Communications with the remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, wait[1] until the subsystem is enabled again, or terminate.

8084 **Error Indication:** An MLCA (multiline communications adapter) controller check occurred on an *input* operation. The subsystem has been disabled.

Recovery Action: Communications with the remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, wait[1] until the subsystem is enabled again, or terminate.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

Major Code 81 – Permanent (nonrecoverable) session error.

Error Description: A nonrecoverable error has occurred in the session; the session cannot be continued and 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/34 Messages Guide* 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 session must be acquired again before communications can resume.

General Recovery Actions: The following general actions can be taken for each 81xx return code. Other specific actions are given in each return code description.

- Several return codes indicate that an error condition must be corrected by changing a value in the subsystem configuration record or in the SESSION statement for your program.
  - To change a parameter value in the subsystem configuration being used by your program, disable the subsystem before making the change in the subsystem's configuration record, and enable the subsystem again to make the change effective.
  - To change a parameter value in the SESSION statement associated with your program, you must terminate your program only.

  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.

- If the session should be started again, it must be reacquired by your program or restarted by the remote program before communications can resume.

- An end of session (EOS or $$EOS) operation should be issued for the session that has terminated. Your program can also continue local processing, or it can terminate.

Note: If the session is started again, it starts from the beginning, not at the point where the session error occurred.

Code Indication/Action

8183 **Error Indication:** An MLCA (multiline communications adapter) controller check occurred on an *output* operation; data may have been lost. The session has been terminated.

Recovery Action: If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

8184 **Error Indication:** An MLCA (multiline communications adapter) controller check occurred on an *input* operation; data may have been lost. The session has been terminated.

Recovery Action: If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

8187 **Error Indication:** Your program was not synchronized with the remote program; that is, both programs may have been trying to send at the same time or receive at the same time. Or, one program may have tried to release or end the session before it should have. The session has been terminated.

Recovery Action: Check your program for a logic error and correct it. (If your program did not cause the error and it continues to occur, notify the remote location and have the remote program's logic checked.) Then if your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

8195 **Error Indication:** A subsystem disk I/O error occurred while the communications work area on the System/34 disk was being used. The session has been terminated.

Recovery Action: If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

81A3 **Error Indication:** Synchronization was lost between the subsystem and the remote system communications equipment; the session has been terminated.

Recovery Action: If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

81BA **Error Indication:** On an *input* operation, an indication was received that a data record is too long. The record, received from the remote program, exceeds the subsystem input buffer length of 512 bytes. The session has been terminated.

Recovery Action: Notify the remote location that it tried to send a record that was too long; the Finance subsystem cannot handle records longer than 512 bytes. Then, if your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

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/34 Messages Guide* 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, first disable the subsystem, make the change in the subsystem's configuration record, and 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, (and depending on what the return code description says):

   a. Notify the remote location that a change is required on that end to correct the error received.

   b. Simply reissue the acquire operation. It could be successful if the error occurred because there was not enough common queue space available to support a new session, because an isolated line error occurred, or because the remote system was not active at the time.

   c. If the acquire operation is again unsuccessful, retry it only a limited number of times. (The limit for retries should be specified in your program.)

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 operation is valid only in an *active* session and cannot, therefore, be issued after an 82xx return code is received. (This restriction does not apply to COBOL and assembler programs, or to the TIMER intrinsic function in BASIC, which also can be used to wait for a specified time interval.)

Code Indication/Action

821E **Error Indication:** The acquire operation attempted by your program (BASIC programs only) was unsuccessful because there was no SESSION statement specified between the LOAD and RUN statements for your program. The method used to issue the acquire operation is not supported by the Finance subsystem. The session was not started.

Recovery Action: Issue a SESSION statement that specifies, in the SYMID parameter, the identifier of the session to be acquired. The same identifier must be specified in the ID parameter of the OPEN statement.

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, respecify 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 either in this subsystem or in the interrupt handler.

Recovery Action: This subsystem has been disabled; it must be enabled again before communications can resume. Your program can continue local processing, wait[1] to reissue the acquire operation, or terminate.

If more than one subsystem was active when the error occurred, all subsystems that were active when the error occurred should be disabled. (Note that all other active Finance subsystems are automatically disabled when the error occurs; all other types of active subsystems must be manually disabled.)

If all subsystems (of all types) on the system are *not* disabled to recover from the processor check, the common queue space used by the failing subsystem *cannot* be freed. And if it is not freed, that space is wasted, and an indication of insufficient common queue space being available can occur. The indication can occur as a message when the failing subsystem is reenabled or when a different subsystem is enabled. The indication can also occur as an 8315 return code for any subsystem that is performing an output operation in a session.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

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 remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing; it can wait[1] until the subsystem has been reenabled, then reissue the acquire operation; or it can terminate.

8288 **Error Indication:** The acquire operation just performed was not successful because the remote location for which the session was being acquired is not active. The session was not started.

Recovery Action: Call the remote location and request that the device controller be made active. Then reissue the acquire operation. Otherwise, your program can continue local processing, wait[1] 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 100 sessions can be active in the System/34 at one time. The session was not started.

Recovery Action: Your program can wait[1] 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 to be enabled is identified by the location parameter in the SESSION statement. That location name must also be specified in the subsystem configuration record (shown on display 3.0 of the subsystem configuration planning charts). The session was not started.

Recovery Action: Verify that the subsystem name was specified correctly on the LOCATION parameter of the SESSION statement. If the correct name was specified, contact the System/34 system operator and request that the specified subsystem be enabled by executing the ENABLE procedure command at the system console. Then reissue the acquire operation. Otherwise, your program can continue local processing, wait[1] to reissue the acquire operation, or terminate.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

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[1] until the subsystem has been enabled, and 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[1] until the subsystem is enabled again, and then reissue the acquire operation. Otherwise, your program can continue local processing, or it can terminate.

82B4 **Error Indication:** The acquire operation was not successful because all of the resources needed for the session could not be allocated from the assign/free area of the system. All available resources are already being used in the system. The session was not started.

Recovery Action: Wait[1] for the needed resources to become available, then reissue the acquire operation. Otherwise, continue local processing or terminate.

82BB **Error Indication:** The acquire operation was not successful because the logical work station specified by the LWSID parameter on the SESSION statement is not available. Either it is already in use, or it is not active at this time. The session was not started.

Recovery Action: Either wait[1] for the specified logical work station to become available and reissue the acquire operation, or call the remote location and request that the work station be made ready. Otherwise, continue local processing or terminate.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

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/34 Messages Guide* 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, first disable the subsystem, make the change in the subsystem's configuration record, and 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.

   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.

   Note: If the session is started again, it starts from the beginning, not at the point where the session error occurred.

3. If no change is needed in your program or in the subsystem, (and depending on what the return code description says):
   a. Retry the operation, if possible. It could be successful if the error occurred because there was not enough common queue space available at the time, because an isolated line error occurred, or because the subsystem was not enabled at the time.
   b. If another operation is not successful, retry it only a limited number of times. (The limit for retries should be specified in your program.)
   c. Issue a set timer operation in your program so it can wait for a specified time interval before reissuing the operation.

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 an 81xx (session was terminated) or 82xx (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 an 81xx or 82xx return code is handled properly. If you want your program to recover from an improperly handled error condition, issue another acquire operation.

8313 **Error Indication:** On an *output* operation, a queue space error condition was detected. The output operation could not be performed because no *subsystem* queue space was available at the time.

Recovery Action: Your program can issue a set timer operation and wait for a period of time, then reissue the output operation. If an unacceptable number of queue space errors occur, you can disable the subsystem and change the subsystem configuration by specifying a larger subsystem queue space size in the subsystem queue space parameter. After the subsystem is enabled, reissue the acquire operation to restart the session.

8315 **Error Indication:** On an evoke operation, a queue space error condition was detected. The evoke operation could not be performed because no *common* queue space was available at the time.

Recovery Action: Your program can issue a set timer operation and wait for a period of time, and then reissue the evoke operation. If an unacceptable number of queue space errors occur, you can disable all the subsystems and change the subsystem configuration by specifying a larger common queue space size in the SSP-ICF common queue space parameter. After the subsystem is enabled, reissue the acquire operation to restart the session.

831C **Error Indication:** The output operation issued before this output operation received a return code of 0412 (indicating that the remote program sent 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 data.

Recovery Action: Issue an input operation to receive the 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. 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 before attempting to communicate with the remote 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 record cannot be longer than 256 bytes. The session is still active.

Recovery Action: If you want your program to recover dynamically, reissue the output operation with a smaller output length. Otherwise, you can either change the record length in your program and recompile it, or you can change the value specified for the maximum user record length parameter in the subsystem configuration. 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 erroneously followed by a release 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 other program.

8329 **Error Indication:** An invalid evoke operation was detected in this session. Your program was evoked by an incoming procedure start request and cannot, therefore, issue any evoke operations in this session.

Recovery Action: If you want your program to dynamically 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.

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.

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.

Chapter 14. The SNA Peer Subsystem

The SNA Peer subsystem provides distributed data processing support to users of the System/34 SSP in conjunction with other System/34s using the SNA Peer subsystem. The Peer subsystem provides an interface between application programs on different System/34s that can be batch or interactive in nature. The application programs can be written in BASIC, COBOL, or RPG II.

The Peer subsystem allows System/34 application programs to initiate procedures on other System/34s. The System/34 security options are supported.

The Peer subsystem provides support for communicating with another System/34 on a point-to-point switched or nonswitched line or a multipoint line. The communications occurs as peer communications, therefore, neither location is considered to be the host system. One of the systems, however, must be configured to use primary SDLC support, and the other must use secondary SDLC support.

The Peer subsystem allows up to 64 concurrent sessions between any two locations. Up to 100 total sessions are possible on a multipoint line. Up to 32 remote locations can be defined per multipoint line or switched line.

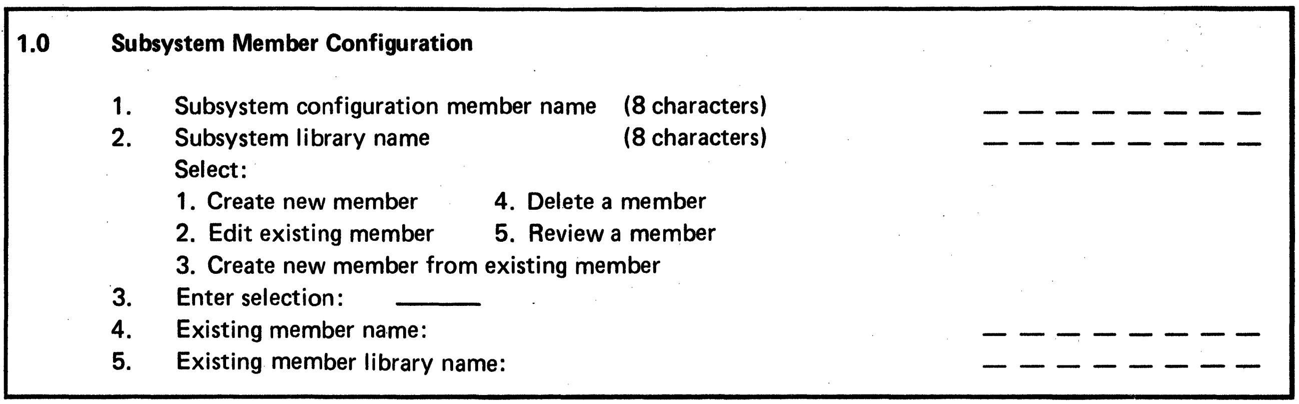
SETTING UP THE PEER SUBSYSTEM

The SSP procedures CNFIGSSP and INSTALL are used to include the interactive communications feature and Peer subsystem support on the System/34. The general interactive communications and line control support is included when it is requested on the appropriate CNFIGSSP prompt. The Peer subsystem support is copied to the system library when the appropriate responses to the INSTALL procedure prompts are taken. The CNFIGSSP and INSTALL procedures, with their displays and related responses, are described in the *Installation and Modification Reference Manual.*

Note: If the Peer subsystem, the Finance subsystem, the station link tests, and remote work station support (or any combination of three) are to run concurrently on the same line, that line must be specified as multipoint during microcode configuration.

After the Peer subsystem has been installed, the CNFIGICF procedure is used to tailor the subsystem support to an existing or proposed network. The operation of the CNFIGICF procedure is also explained in the *Installation and Modification Reference Manual*. Before running the CNFIGICF procedure, however, you should fill out a planning chart for each subsystem configuration that you want to define. Copies of the planning chart for each subsystem are available in Appendix F of this manual and in the *Installation and Modification Reference Manual*. The following sections explain how to fill out the planning chart for the Peer subsystem.

Display 1.0 Subsystem Member Configuration



```
1.0     Subsystem Member Configuration

        1.   Subsystem configuration member name  (8 characters)      — — — — — — — — —
        2.   Subsystem library name               (8 characters)      — — — — — — — — —
             Select:
             1. Create new member       4. Delete a member
             2. Edit existing member    5. Review a member
             3. Create new member from existing member
        3.   Enter selection:  ______
        4.   Existing member name:                                    — — — — — — — — —
        5.   Existing member library name:                            — — — — — — — — —
```

Subsystem configuration member name: Specify a name for this configuration of the subsystem. This name is used to store the member in a library, and is referenced in the ENABLE and DISABLE procedures.

Library name: Specify the name of the library in which the configuration member is stored or to be stored. The default is #LIBRARY.

Enter selection: Specify one of the five options: (1) create a new member, (2) edit an existing member, (3) create a new member from an existing member, (4) delete a member, or (5) review a member without changing it.

Existing member name: This prompt appears if option 3 was selected. Specify the name of the existing subsystem configuration member that is to be used to create the new member. The existing member remains unchanged.

Existing member library name: This prompt appears if option 3 was selected. Specify the library name where the existing member resides.

Display 2.0 Common SSP-ICF Parameters for Each Subsystem

```
2.0     Common SSP-ICF Parameters for Each Subsystem

        1.  SSP-ICF common queue space: (2 - 42 K)                      _ _
        2.  Define the subsystem type:                                  _ 7
            1  Intra                  2  BSC IMS/IRSS
            3  BSCEL                  4  BSC CICS
            5  BSC CCP                6  SNA Upline
            7  SNA Peer               8  BSC 3270
            9  SNA 3270              10  Finance
```

SSP-ICF common queue space: Specify the size, in multiples of 2 K bytes, of the common queue space. The common queue space requirements for each configuration of the Peer subsystem enabled are:

$$C = W + 188X + 80Y$$

where:

C = number of bytes required for common queue space

W = 655 bytes for the first time the subsystem is enabled; 370 bytes for each additional configuration enabled

X = total number of sessions that will be active concurrently for this configuration (including preestablished, acquired, and remotely started sessions)

Y = number of remote locations

The common queue space value specified for the first subsystem that is enabled becomes the size of the common queue space. *Be sure that the value specified for common queue space size takes into account the requirements of any other subsystem that might be active concurrently.*

The size of the common queue space plus the total subsystem queue space of all the enabled Peer subsystems cannot exceed 42 K bytes.

The default common queue space size is 4 K bytes.

Define the subsystem type: Specify a 7 for the SNA Peer subsystem.

Display 3.0 General Subsystem Parameters

3.0 General Subsystem Parameters

1. Location name: (8 characters) — — — — — — — —
2. Subsystem queue space: (0-40 K) — —
3. Subsystem support swappable: (0-No 1-Yes) —
4. Maximum user record length: (1-4075) — — — —

Location name: Specify up to 8 characters for the name of the location associated with this configuration. This location name will be referred to, in the SESSION OCL statement, by other stations on the communications line when referring to this station. Both systems exchange the location name to verify the validity of the physical connection. The location name is used in some of the displayed message texts. The default is the subsystem configuration member name.

Subsystem queue space: Specify the size, in multiples of 2 K bytes, of the subsystem queue space. No subsystem queue space is required; however, improved performance may be obtained by using a subsystem queue space. If you specify a subsystem queue space, specify a queue space that equals the maximum user record length times the number of sessions you expect to be active at the same time.

Whenever a record is to be sent or received, it must be moved to an intermediate buffer between the application program and the SDLC task. The Peer subsystem attempts to assign an intermediate buffer out of the subsystem queue space. If no subsystem queue space is available, a disk work area is used for the buffer. When the disk work area is used, main storage requirements for the subsystem queue space are reduced, but the amount of time required to move data between the application program and the intermediate buffer is greatly increased.

Factors affecting subsystem queue space requirements are:

- The number of concurrent sessions. Each active session competes for intermediate buffers from subsystem queue space. The more session competing, the greater the possibility of a failure to obtain an intermediate buffer.

- The size of records. The amount of storage required for an intermediate buffer is equal to the size of the record to be transmitted or received.

- Frequency of transmission. Sessions that transmit or receive records frequently may be more likely to obtain an intermediate buffer from disk than those that send and receive records infrequently.

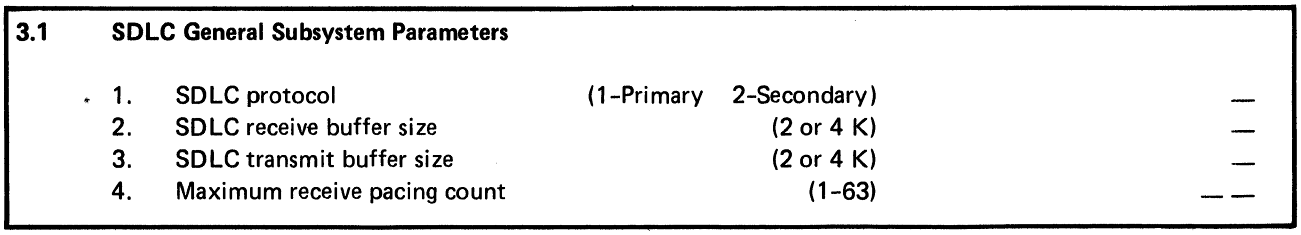
- Response requirements. Interactive programs that transmit and receive infrequently may be able to accept the slower response times that occur when intermediate buffers are on disk.

The default subsystem queue space size is 4 K bytes.

Subsystem support swappable: Specify whether you want the subsystem to be swappable. Consider the total system performance, the size of the subsystem, and the amount of user main storage when determining whether you want the subsystem to be swappable. Specifying nonswappable should make the subsystem operate more efficiently for batch activity, but could degrade performance for the rest of the system. Interactive applications should run with the subsystem swappable. If multiple Peer configurations are enabled, the swappable status of the first configuration enabled is used. The default is 0 (no).

Maximum user record length: Specify the maximum record length (1 through 4075 bytes) to be sent or received by a System/34 program using this subsystem configuration. The default is 256 bytes. This parameter affects the size of the communications work area. See *Disk Space Requirements* later in this chapter for details.

Display 3.1 SDLC General Subsystem Parameters



3.1 SDLC General Subsystem Parameters

1. SDLC protocol (1-Primary 2-Secondary) _
2. SDLC receive buffer size (2 or 4 K) _
3. SDLC transmit buffer size (2 or 4 K) _
4. Maximum receive pacing count (1-63) _ _

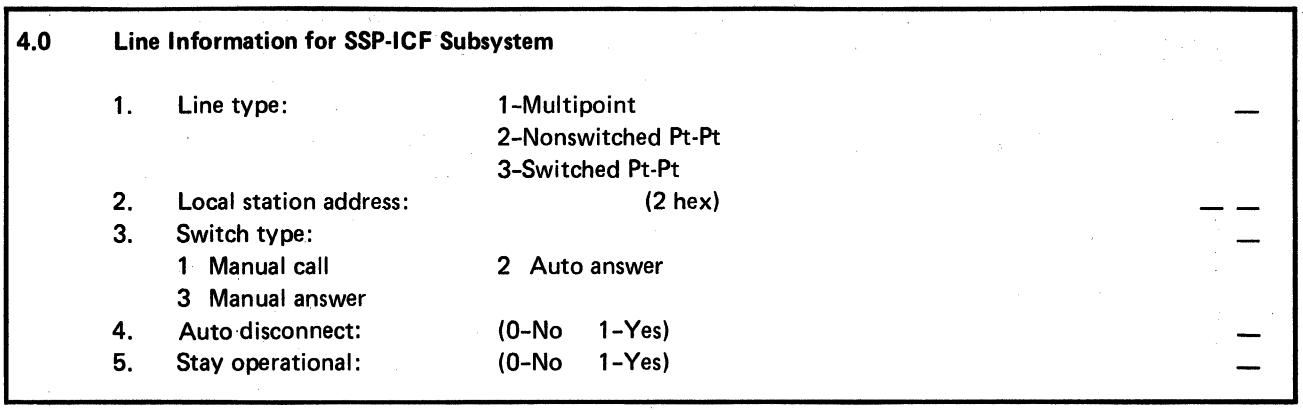
SDLC protocol: Specify the SDLC protocol to be used. Specify 1 (primary) or 2 (secondary).

SDLC receive buffer size: Specify the amount of storage for SDLC receive buffers. Specify either 2 K (7 buffers) or 4 K (15 buffers) bytes. This storage is allocated as nonswappable. The default is 2 K bytes.

SDLC transmit buffer size: Specify the amount of storage for SDLC transmit buffers. Specify either 2 K (7 buffers) or 4 K (15 buffers) bytes. This storage is allocated as nonswappable. The default is 2 K.

Maximum receive pacing count: Specify the receive pacing count (1 to 63) to be used. This parameter affects the amount of application program data the remote system can send before waiting for a pacing response. The pacing count also affects the size of the communications work area (see *Disk Space Requirements* later in this chapter). The default value is 3 and is used for interactive data exchange. If the remote system will send batch data, a larger value should allow the data to be transmitted more efficiently. A value of 7 should be sufficient for batch data.

Display 4.0 Line Information for SSP-ICF Subsystem



```
4.0    Line Information for SSP-ICF Subsystem

       1.   Line type:                 1-Multipoint                                  _
                                       2-Nonswitched Pt-Pt
                                       3-Switched Pt-Pt
       2.   Local station address:              (2 hex)                            _ _
       3.   Switch type:                                                             _
            1  Manual call             2  Auto answer
            3  Manual answer
       4.   Auto disconnect:           (0-No    1-Yes)                               _
       5.   Stay operational:          (0-No    1-Yes)                               _
```

Line type: Specify the line type that is suitable to your communications environment. There is no default line type. Multipoint line type is not displayed if secondary SDLC protocol was selected.

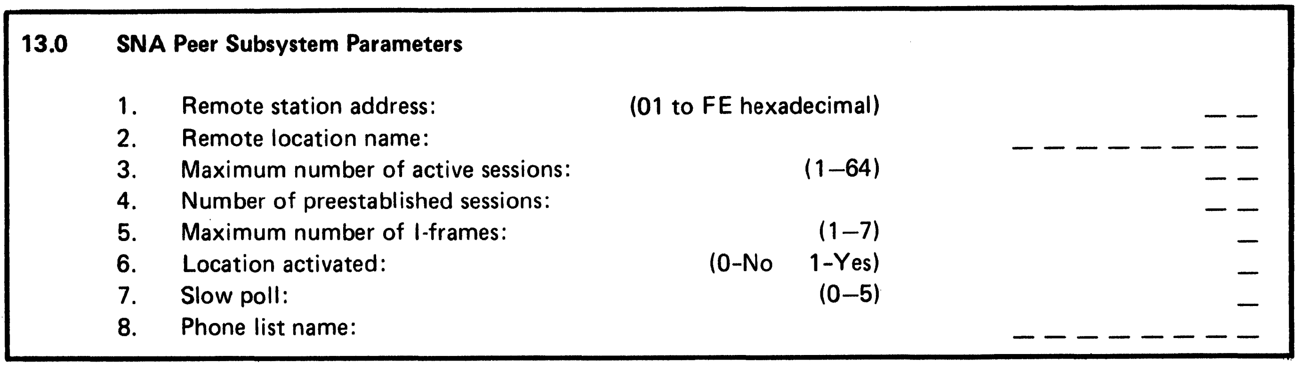
Local station address: This prompt is displayed only when secondary SDLC protocol is selected. Specify the System/34 SDLC station address identified for this configuration. This value must be specified as two hexadecimal characters in the range of 01 to FE. The address must be the same as the remote station address configured for this location by the station using primary SDLC protocol. If remote work stations and Peer are active on the same line concurrently, this address must not be the same as any other station address configured for a remote work station.

Switch type: This prompt is displayed only if switched pt-pt line was specified. Specify the switch type you want for the line. Specify 1 for manual call. If you are using autocall or switched X.21, specify 2 (auto answer). See *Switched Line Considerations* for additional information.

Auto-disconnect: This prompt is displayed only if switched pt-pt line was selected. Specify whether you want this configuration to initiate disconnection of the communications line when the last session is released.

Stay operational: This prompt is displayed only if switched pt-pt line was selected. Specify whether you want the Peer subsystem to remain active following an automatic line disconnect. Specifying 1 (yes) causes the Peer subsystem to remain active when a disable is initiated by the remote station (DISABLE command or an automatic disconnect). Specifying 0 (no) causes the subsystem to terminate when a disable is initiated by the remote station.

Display 13.0 SNA Peer Subsystem Parameters



```
13.0    SNA Peer Subsystem Parameters

        1.  Remote station address:            (01 to FE hexadecimal)              __
        2.  Remote location name:                                           ________
        3.  Maximum number of active sessions:                  (1–64)             __
        4.  Number of preestablished sessions:                                     __
        5.  Maximum number of I-frames:                          (1–7)              _
        6.  Location activated:                    (0-No    1-Yes)                  _
        7.  Slow poll:                                           (0–5)              _
        8.  Phone list name:                                                ________
```

Remote station address: This prompt is displayed only when primary SDLC protocol is selected. Specify the station address of the remote location. Specify the address as two hexadecimal digits between 01 and FE. This address must be the same as the address configured by the secondary location as local station address. If any combination of SNA Peer support, remote work stations, and/or Finance support are active on the same line concurrently, this address must be unique.

Remote location name: Specify a name for the remote location. The name must be from one to eight characters and must be the same as the location name configured by the remote system.

Maximum number of active sessions: Specify the maximum number of concurrent sessions that can be active with this location. Specify a value from 1 to 64. The default is 8. This parameter affects the size of the communications work area. See *Disk Space Requirements* later in this chapter for details. The same value should be specified at the remote location.

Number of preestablished sessions: This prompt is not displayed if a switched pt-pt line is selected. Specify the number of sessions to be activated when the subsystem is enabled. Establishing sessions during enable provides a pool of sessions that are available to application programs (on this system only) and can improve the response time for acquiring sessions from the application program. Each established session does, however, require common queue space, even if the session is not in use. Specify a decimal value from 0 to the maximum number of active sessions. The default is 0.

Maximum number of I-frames: Specify the maximum number of data records (I-frames) to be sent by SDLC before requiring an acknowledgement from the remote system. Generally, any form of batch transmission will run more efficiently with 7 specified for this parameter. Interactive applications might run more efficiently with a smaller number specified. Specify a decimal number from 1 to 7. The default is 7.

Location activated: Specify whether the remote location should be activated when the subsystem is enabled. The default is 1 (yes).

Slow poll: This prompt is displayed only when primary SDLC protocol is selected and the line type is not switched. Specify whether the secondary remote location should be slow polled by SDLC when the secondary disables. Slow polling allows a secondary station to be inactive without a time-out occurring. Specify a decimal value from 0 to 5. 0 indicates that the remote location will not be slow polled. The values from 1 through 5 indicate the relative polling frequency, with 1 being the most frequent. The default is 5. For the first enable of a secondary remote station from the primary station, slow polling is always used until the connection is established.

Phone list name: Specify the name of the load member that contains the list of phone numbers to be called by the autocall feature or the list of numbers for the public data network to be called by the X.21 feature. The list is created using the DEFINEPN or DEFINX21 procedure. These procedures are described in the *System Support Reference Manual.* The list must be in the current user library or in the same library as the configuration record. Also specify 2, auto answer, on display 4.0 of the CNFIGICF procedure.

Disk Space Requirements

The part of the user disk space used by the Peer subsystem is referred to as the communications work area. This area is not available to the user when the Peer subsystem is enabled. Three user-specified factors that affect the size of the communications work area are the maximum number of active sessions, the maximum receive pacing count, and the maximum user record length. Use the following steps to determine the size of the communications work area.

1. Multiply the maximum receive pacing count by 4.

2. Divide the maximum user record length by 256 and round up to the nearest whole number. Add this to the total from step 1.

3. Add 3 to the total from step 2.

4. Multiply the total from step 3 by the maximum number of sessions, divide the result by 10, and round up to the nearest whole number. This total is the total number of blocks used for the communications work area.

For example, during CNFIGICF you specified the following for the Peer subsystem:

- The maximum receive pacing count is 5.

- The maximum user record length is 256.

- The maximum number of active sessions is 4.

The number of blocks used for the communications work area would be 10:

1. 5 X 4 = 20

2. 256 ÷ 256 = 1
   1 + 20 = 21

3. 21 + 3 = 24

4. 24 X 4 = 96
   96 ÷ 10 = 9.6 blocks, rounded up to 10 blocks

Switched Line Considerations

When switched lines are used with the Peer subsystem (3, *switched pt-pt*, specified on display 4.0 of the CNFIGICF procedure), the connection with the remote location can be established in one of the following ways:

- If the switch type is manual call, the message OPERATOR DIAL REQUIRED is displayed when the subsystem is enabled.[1]
- If the switch type is manual answer, the message OPERATOR ANSWER REQUIRED is displayed when the subsystem is enabled.[1]
- If the switch type is auto answer, the System/34 answers a call from a remote location without operator intervention. If an application program attempts to acquire a session before the line connection is established, the System/34 temporarily overrides the auto answer switch type to manual call and displays the message OPERATOR CALL THIS LOCATION and the message OPERATOR DIAL REQUIRED on the system console.[1]
- If the switch type is auto answer and a phone list name is specified on display 13.0, the system uses the autocall or switched X.21 feature to call the remote location (specified by the acquire operation) without operator intervention. The line being used must be an autocall or switched X.21 line.

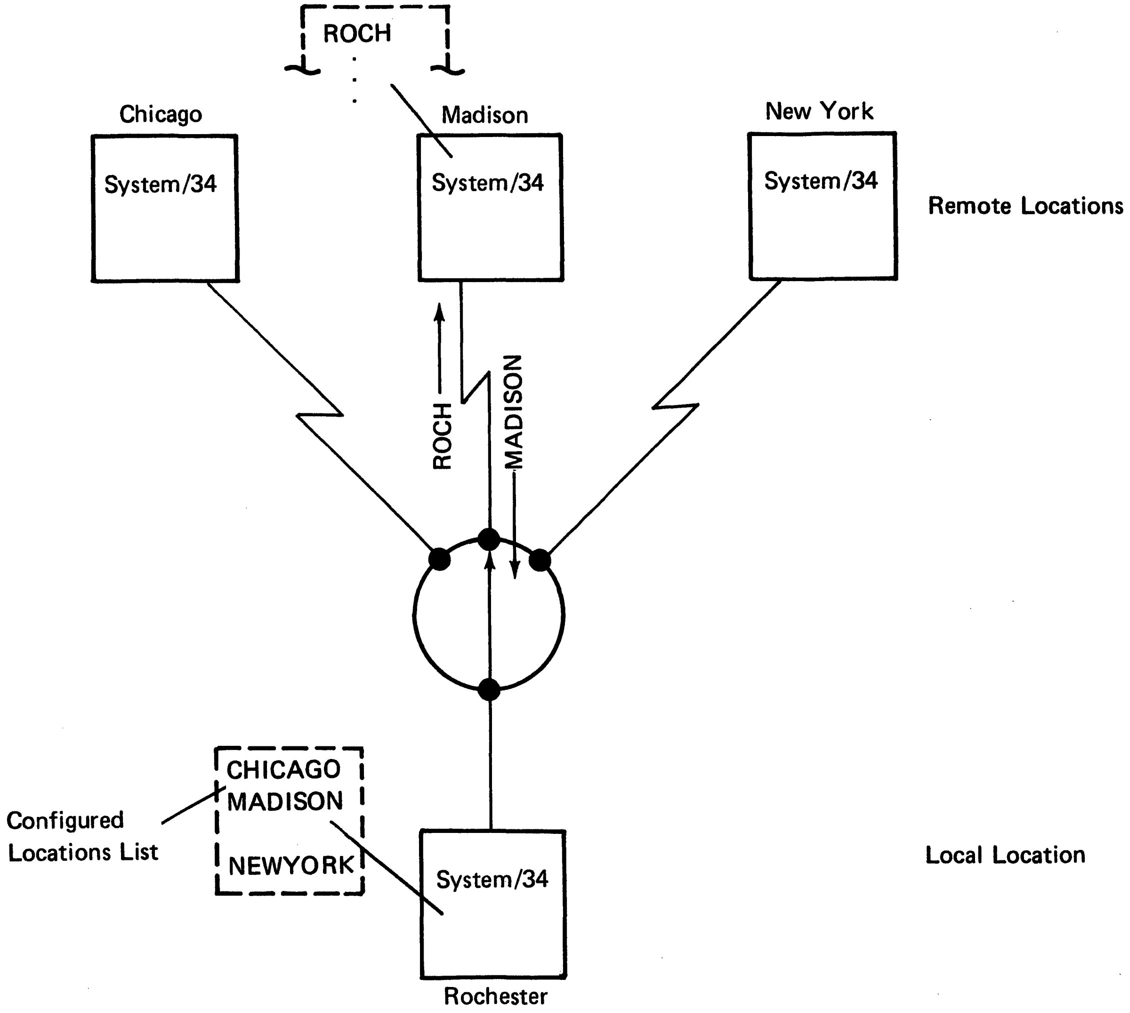
  When the system calls the remote location automatically, the message LOCATION BEING AUTOMATICALLY CALLED IS DISPLAYED. If the operation is unsuccessful, the message AUTOCALL TO THIS LOCATION FAILED is displayed.

Multiple Remote Locations

The Peer subsystem allows more than one location to be enabled on a switched line. This allows an application program to communicate with multiple locations (one at a time) without disabling one configuration and enabling another.

[1] If you are using the switched X.21 feature, you cannot call or answer manually. When you specify auto answer and enter a phone list name on display 13.0 of the CNFIGICF procedure, the System/34 calls and answers automatically. If you do not specify auto answer and enter a list name, the message ENABLE NOT COMPLETE is displayed on the system console when the operator attempts to enable the subsystem.

The following illustration shows a local System/34 and three possible remote System/34s with which the local system can communicate on switched lines.



In this example, the local System/34 at the location in Rochester is connected to the system at the location in Madison. Before data can be exchanged between the stations, the location names are exchanged and each location checks the name to determine that the correct station is called or is calling. For example, when the local System/34 calls the remote system at the location in Madison and a connection is established, the local system sends its location name (Rochester) to the system in Madison and the remote system sends its location name (Madison) to the Rochester location. The Peer subsystems at both locations ensure that the name received has been configured. If either system determines that the location name has not been configured, the message INCORRECT OR UNDEFINED LOCATION CALLED is displayed, the line is disconnected, and the calling station assumes its original enabled state.

Note: The location names are also exchanged in the same manner on nonswitched lines. If a location name is invalid, the message INCORRECT OR UNDEFINED LOCATION CALLED is displayed, and communications with the invalid location is terminated.

The following sequence shows how the subsystem establishes a connection when an acquire operation is performed and multiple locations are configured:

Is a location connected?

Yes No

No: A connection is established
with the location specified on the
SESSION OCL statement,
and the acquire operation continues.

Yes: Is the location connected the same as
the location specified on the
SESSION statement?

Yes No

No: The acquire operation is terminated,
and return code 82A7 (*line not available*)
is passed to the application
program.

Yes: The acquire operation continues as requested.

A connection with a remote location is terminated as follows:

- The operator disables the location using the DISABLE procedure. Once disabled, the location cannot be used again until it is enabled.

- The remote system disconnects the line. If *stay operational-yes* was specified at the local System/34, the location is not disabled and can be used again without being enabled.

- The application program releases the last session and auto-disconnect is specified as yes. If stay operational is specified as yes, the configuration can be used again without reenabling the subsystem.

STARTING AND ENDING THE PEER SUBSYSTEM

The ENABLE procedure starts the Peer subsystem and/or activates communication with a remote location. The DISABLE procedure ends the subsystem and/or de-activates communication with a remote location. The following sections describe the ENABLE and DISABLE procedures.

ENABLE Procedure

The ENABLE procedure can be used to activate the Peer subsystem or to activate a particular location on a multipoint line. Enable performs the following functions:

- Ensures compatibility between the configuration and the communication hardware.
- Determines whether the line requested is available.
- Loads and attaches the subsystem if it is not already active.
- Loads and attaches the SDLC task if it is not already active.
- Assigns storage for required data areas and buffers.

If the configured line type does not correspond to the actual configuration, a message is issued and the enable is terminated. You can use the SPECIFY procedure to change the line type. The SPECIFY procedure is described in the *SSP Reference Manual.*

Enable also ensures that all remote location names associated with this configuration are unique on the system. If a subsystem is active and one of the location names matches a remote location name in the Peer configuration being enabled, a message is issued and the enable is terminated.

When the ENABLE procedure is used to activate a particular location on a multipoint line and the subsystem is already active, enable ensures that the subsystem is on the specified line, and the communication begins with the specified location. If the first enable specifies a location name, enable performs the functions required to activate the subsystem.

The format of the ENABLE procedure command for the Peer subsystem is:

$$\text{ENABLE name, } \begin{bmatrix}\underline{\text{\#LIBRARY}}\\ \text{library name}\end{bmatrix}, \ [\text{line number}], \begin{bmatrix}\text{SHOW}\\ \underline{\text{NOSHOW}}\end{bmatrix}, \ [\text{location}]$$

name: Specifies the name of the subsystem configuration to be enabled.

library name: Specifies the name of the library that contains the specified subsystem configuration. The default is #LIBRARY.

line number: Specifies the communications line for which this subsystem is to be enabled. If this is not the first enable of this subsystem configuration, the line number need not be specified.

SHOW/NOSHOW: Specifies whether subsystem configuration parameters are to be displayed before the subsystem is enabled. If SHOW is specified, they are displayed, and some of the configuration parameters can be overridden. The overrides can only be performed on the first enable, and the subsystem configuration is not permanently altered. NOSHOW is the default.

location: Specifies the name of the remote location to be enabled.

Following completion of the enable, a message (SYS-8241) is displayed on the system console for each location that is activated. No application program operations are allowed until this message is issued. (See the acquire operation description for a possible exception for switched lines.)

DISABLE Procedure

The DISABLE procedure can be used to terminate the Peer subsystem, to terminate communications with a particular remote location on a multipoint line (from the primary SDLC station), or to terminate communication with all remote locations in a configuration.

When a disable is requested to de-activate a subsystem configuration or location, the following functions are performed:

- If no active sessions are active for the configuration or location being disabled, the disable is performed, and the main storage being used is freed. If no locations are active for a configuration, the entire configuration is disabled. When the last Peer configuration is disabled, the subsystem is terminated; the appropriate SDLC task is also terminated if it is not currently in use.

- If sessions are active for the Peer configuration being disabled, a message is issued to the system operator with the following options:
  - Hold the disable. New sessions and transactions are prevented from being started and, when all sessions complete, a normal disable occurs. Each successful operation receives a 02xx return code, indicating that a disable is pending.
  - Retry the disable. Check again for any active sessions.
  - Cancel active sessions and the disable. Active sessions are immediately terminated and the disable is performed.
  - Ignore the disable request. The DISABLE procedure is cancelled, and must be run again when a disable is desired.

The results of a disable on a switched line depends on the stay operational parameter defined during subsystem configuration. See the description of the stay operational parameter in display 4.0 for more information.

The format of the DISABLE procedure command for the Peer subsystem is:

```
DISABLE name,[location]
```

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

location: Specifies the name of the remote location to be disabled.

A subsystem configuration or location can also be disabled automatically by specifying auto-disconnect as yes and stay operational as no on a switched line. The subsystem configuration is then disabled when the last session is released.

STARTING PEER SUBSYSTEM APPLICATIONS

System/34 Peer subsystem applications can be started by a display station operator entering a procedure command or by a request from a remote System/34. Procedures that are started by a System/34 operator must have a SESSION OCL statement for each session to be started. The following sections describe the SESSION statement and the incoming procedure start requests.

SESSION OCL Statement

The format of the SESSION statement for the Peer subsystem is:

```
// SESSION LOCATION-name,SYMID-sessionid
```

LOCATION: Specifies the name of the remote location associated with the session. This name must have been configured as a remote location name for the configuration being used. The remote station must have this name configured as its local location name.

SYMID: Specifies the symbolic ID of the session with which this SESSION statement is associated. The symbolic ID must be two characters, with the first character numeric (0 through 9) and the second character alphabetic (A through Z, #, $, or @). This is the same ID that the application program uses when referring to this session. This ID is the equivalent of the symbolic display station ID as specified on the WORKSTN OCL statement. This parameter has no default.

Note: If the application program is a BASIC program, the SESSION statement is not required. See Chapter 4 for more information.

Incoming Procedure Requests

For remote applications to initiate procedures on the System/34, the remote System/34 application must issue an evoke operation to the Peer subsystem. Considerations for writing System/34 procedures to be remotely initiated are contained in Chapter 2.

OPERATION CONSIDERATIONS

The following operations are supported by the Peer subsystem. A complete chart of all interactive communications operations and the subsystems that support them is in the appropriate language chapter. The chart also shows the keyword or format name used to code the operation. More information about how an operation is coded is also described in the appropriate programming language chapter.

Whether an operation completes successfully or unsuccessfully, a return code is given to the application program. All of the return codes that are valid for this subsystem are described at the end of this chapter. A summary chart in Appendix A lists all the return codes and subsystems for which they are valid.

Acquire Operation

The acquire operation is issued by the application program to allocate a session. The effect of the acquire operation depends on whether any sessions were established when the subsystem was enabled.

If sessions were established during enable and one is available, the session is assigned to the issuing program. If no sessions were established during enable or if none of the established sessions are available, a new session is established and is assigned to the issuing program. An acquire operation can be used to initiate the connection on a switched line. To accomplish this, both stations must be configured with a switch type of automatic answer. Auto-disconnect and stay operational should both be configured as yes. The first acquire causes an operator dial message (SYS-3401) to be displayed; subsequent acquires are rejected until communications is established. Following the release of the last session, the link is disconnected, and both stations resume the original enabled state. Another acquire operation can be issued to establish the link again.

Evoke Operation

The evoke ($$EVOKNI), evoke then invite ($$EVOK), and evoke then get (assembler only) operations initiate a transaction. The output length of any evoke operation cannot exceed 120 bytes. If the evoke includes a procedure name, the actual output length is the length of the data plus the length of the procedure name plus 1. The password and user ID must be included when the program is on a System/34 with security active; password and user ID are ignored by a System/34 without security.

The evoke operation sends a procedure start request to the remote system. The remote Peer subsystem processes the request and initiates a remote application that communicates with the local application. The Peer subsystem waits for a response to the procedure start request. If a successful response is received, the remote applicaton is considered to be running, and the transaction is initiated. If an error is received in response to the procedure start request, a return code is passed to the application program indicating that the evoke failed.

The evoke end of transaction operation ($$EVOKET) initiates a remote procedure that does not communicate with the local System/34 application program. The evoke end of transaction operation is not completed until a response is received from the remote system. If a successful response is received, a normal return code is given to the application program. If an error is received, the application program receives a return code indicating that the evoke failed.

Put Operations

The put operation is used to pass data from the local System/34 application to the remote application. The put operation can be issued alone ($$SENDNI) or combined with an input operation ($$SEND).

If the put operation is combined with an input operation, the record, followed by a change direction is sent to the remote system. Then a record is received by the subsystem. In the case of a BASIC, COBOL, or RPG II program, the input data is made available to the application program on the next accept (READ) operation. Assembler users can choose to wait for the input data by using a put then get operation.

The current put operation is not started until the previous put operation is complete. If the previous put operation failed, the current put is not performed, and the application program is notified via the appropriate return code. The put end of file operation ($$SENDE) is used to terminate a series of put operations that do not request input.

Also available is the put end of transaction operation ($$SENDET). The put end of transaction operation is issued by the locally initiated program to terminate a transaction or by the remotely initiated program to terminate a session. Because only one transaction can be issued at a time, the active transaction must end before a new transaction can begin. A transaction is considered terminated when an end of transaction operation is successfully issued by either application program.

Input Operations

The input operations for the Peer subsystem are invite, get, and accept. The invite operation can be issued only as a combined operation with a put or evoke operation ($$SEND, $$EVOK) in BASIC, COBOL, and RPG II. Assembler language users can issue an invite operation explicitly. Either a get or invite operation signals the subsystem to obtain data on the session for the application program. A get operation causes the application program to wait for the data to be available. When a program issues an invite operation, it receives the data with a subsequent accept operation. The accept operation allows input from any previously invited session.

Request to Change Direction Operation

The Peer subsystem allows a request to change direction operation ($$RCD) only during a transaction and only when the issuing program is not transmitting. If the issuing program is receiving data, the request to change direction operation results in the request being sent to the remote system as the response to the next data record received. If the issuing program is not receiving and is not transmitting, the operation has no effect. Request to change direction is always accompanied by a get or invite operation.

Fail Operation

The fail operation ($$FAIL) indicates to the remote program that an abnormal condition has occurred. The fail operation can be issued while the program is sending or receiving. If a program issues a fail operation while sending, it indicates that the data just sent was in error. All data sent before the fail operation is transmitted to the remote application, and a return code indicating the fail operation is then given to the receiving program. If a program issues a fail operation while receiving, it indicates that the data received was in error. The subsystem discards all subsequent data until the transmitting subsystem acknowledges the receipt of the fail operation. In either case, the program that issued the fail operation should transmit, and the program that receives the fail return code must receive. If both programs issue a fail operation simultaneously, the program that was receiving will be successful and must transmit. The program that was transmitting will receive an unsuccessful return code and must receive. No data can accompany the fail operation.

Release Operation

The release operation is issued by the application program to terminate a session it acquired or by a remotely initiated MRT program to pass the session on to the next job step.

If the session was established during enable, the session remains available. If the session was established during the acquire, the session is terminated and must be reestablished to be used again. In either case, an acquire operation is necessary to begin using the session again.

If the session is on a switched line and no other sessions are active, the results depend on the automatic disconnect and stay operational parameters. See the parameter description for display 13.0 of subsystem configuration for more information.

End of Session Operation

The end of session operation ($$EOS) always results in a normal completion return code. The session is always terminated by the end of session operation. If the session is still communicating when the end of session operation is issued, the transmission is abnormally terminated by the Peer subsystem, and abnormal termination of the remote Peer application program could result.

Get Attributes Operation

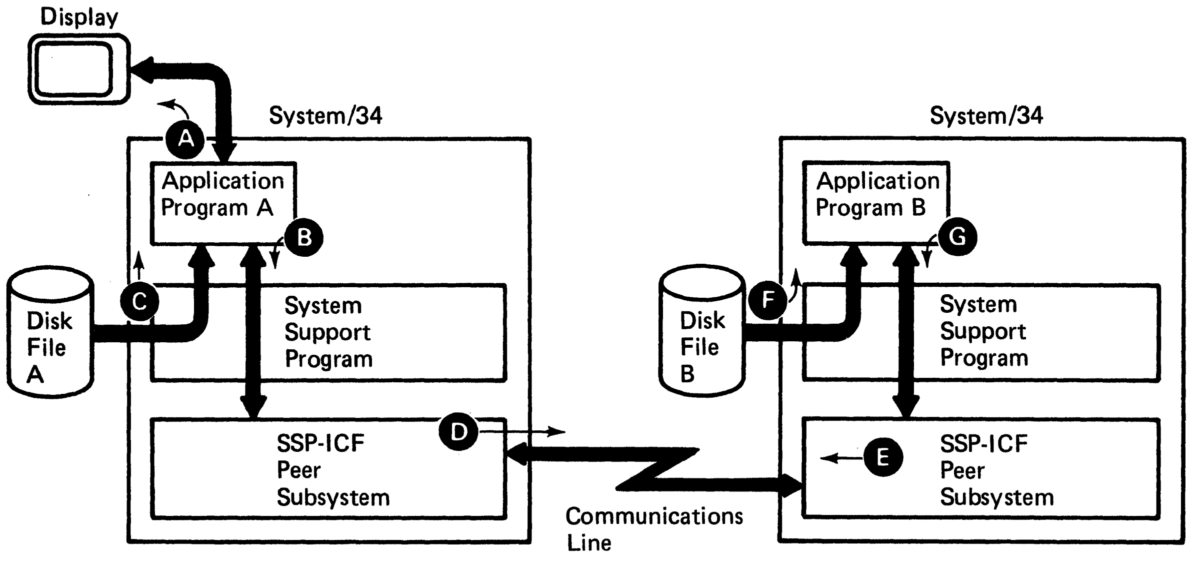
The get attributes operation (assembler only) can be issued at any time to determine the status of a session.

Set Timer Operation

The set timer operation ($$TIMER) results in a timer expired return code (0310) after a specific time interval in hours, minutes, and seconds has expired.

How to Write Programs that Use the Peer Subsystem

The following inquiry application is used in the programming examples:



1. Application program A (in the local System/34) displays a prompt asking an operator to enter an item number requesting the stock status for the item A.

2. When the operator enters the item number, program A reads the number and searches file A (the local file) for the item C.

3. If the item is found in the local file, program A displays the stock status on the screen A.

4. If the item is *not* in the local file, program A uses the Peer subsystem to send the item number to the remote System/34 B and D.

5. Program B (in the remote system) uses the item number to search the remote file for the item F.

6. If the item is in the remote file, program B sends the stock status to program A G and E. If the item is not in the remote file, program B sends the characters ***.

7. If program A receives the stock status, it displays it. If it receives the characters ***, it displays the message ITEM NOT FOUND A.

Programs A and B in this example are the programs described in Chapter 7 for the Intra subsystem. If you have not read the description of these programs in Chapter 7, see *How to Write Programs that Use the Intra Subsystem* in that chapter. The configuration and OCL examples in Chapter 7 are for the Intra subsystem only; you do not need to read those. Following are the configuration parameters and OCL statements for this subsystem.

Configuration Parameters

The following configuration parameters are used for this example. For a description of the configuration parameters, see *Setting up the Peer Subsystem* at the beginning of this chapter.

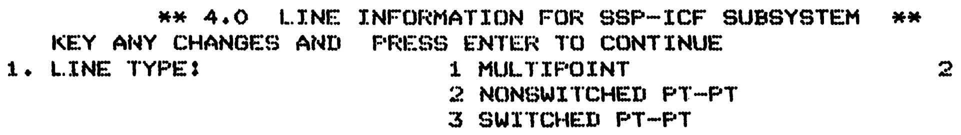
```
CREATE/EDIT                  ** 1.0 SUBSYSTEM MEMBER CONFIGURATION **
     1. SUBSYSTEM CONFIGURATION MEMBER NAME :                 PEERPRI
     2. SUBSYSTEM LIBRARY NAME :                              ICFLIBR
         1 CREATE NEW MEMBER                      4 DELETE A MEMBER
         2 EDIT EXISTING MEMBER                   5 REVIEW A MEMBER
         3 CREATE NEW MEMBER FROM EXISTING MEMBER
     3. ENTER SELECTION :      2
```

```
          ** 2.0 COMMON SSP-ICF PARAMETERS FOR EACH SUBSYSTEM **
   KEY ANY CHANGES AND  PRESS ENTER TO CONTINUE
1. SSP-ICF COMMON QUEUE SPACE              (2 - 42K)      02
2. DEFINE THE SUBSYSTEM TYPE                                7
     1 INTRA                              2 BSC IMS/IRSS
     3 BSCEL                              4 BSC CICS
     5 BSC CCP                            6 SNA UPLINE
     7 SNA PEER                           8 BSC 3270
     9 SNA 3270                           10 FINANCE
```

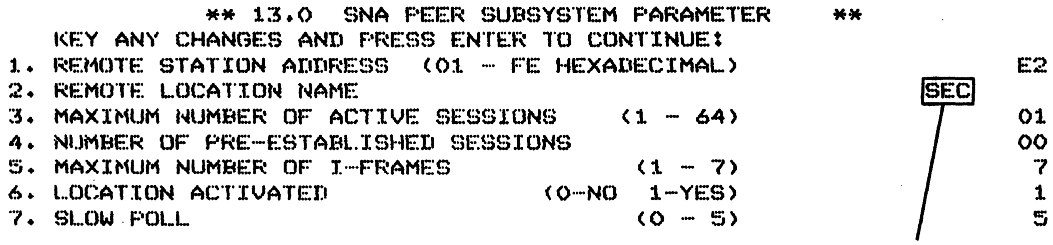
```
          ** 3.0  GENERAL SUBSYSTEM PARAMETERS   **
KEY ANY CHANGES AND PRESS ENTER TO CONTINUE:
     1. LOCATION NAME                                        PRI
     2. SUBSYSTEM QUEUE SPACE                      (0-40K)          00
     3. SUBSYSTEM SUPPORT SWAPPABLE?      (0-NO  1-YES)             1
     4. MAXIMUM USER RECORD LENGTH             (1 - 4075)         0256
```

```
             ** 3.1  SDLC GENERAL SUBSYSTEM PARAMETERS   **
   KEY ANY CHANGES AND PRESS ENTER TO CONTINUE:

1. SDLC PROTOCOL          (1-PRIMARY  2-SECONDARY)                    1
2. SDLC RECEIVE BUFFER SIZE             (2 OR 4K)                     2
3. SDLC TRANSMIT BUFFER SIZE            (2 OR 4K)                     2
4. MAXIMUM RECEIVE PACING COUNT          (1 - 63)                    03
```

Configuration Parameters (continued)



```
          ** 4.0  LINE INFORMATION FOR SSP-ICF SUBSYSTEM  **
   KEY ANY CHANGES AND  PRESS ENTER TO CONTINUE
1. LINE TYPE:                  1 MULTIPOINT                     2
                               2 NONSWITCHED PT-PT
                               3 SWITCHED PT-PT
```



```
            ** 13.0  SNA PEER SUBSYSTEM PARAMETER     **
   KEY ANY CHANGES AND PRESS ENTER TO CONTINUE:
1. REMOTE STATION ADDRESS  (01 - FE HEXADECIMAL)                  E2
2. REMOTE LOCATION NAME                                   SEC
3. MAXIMUM NUMBER OF ACTIVE SESSIONS    (1 - 64)                  01
4. NUMBER OF PRE-ESTABLISHED SESSIONS                             00
5. MAXIMUM NUMBER OF I-FRAMES              (1 - 7)                 7
6. LOCATION ACTIVATED               (0-NO  1-YES)                  1
7. SLOW POLL                               (0 - 5)                 5
```

Also specified on the SESSION OCL statement.

OCL Statements

The following OCL statements are used for the BASIC example:

```
BASICR ITEMABAS,ITEMBAS,30,,PEERSESS

// SESSION LOCATION-SEC,SYMID-1S
```

The BASICR procedure includes the SESSION statement.

The location name (SEC) is also specified on display 13.0 of the CNFIGICF procedure.

The SYMID (session ID) is explained in the description of the program.

The following OCL statements are used for the COBOL example:

```
// LOAD ITEMAC
// FILE NAME-FILEA
// SESSION LOCATION-SEC,SYMID-1S
// RUN
```

The location name (SEC) is also specified on display 13.0 of the CNFIGICF procedure.

The SYMID (session ID) is explained in the description of the program.

The following OCL statements are used for the RPG II example:

```
// LOAD ITEMAR
// FILE NAME-ITEMA
// SESSION LOCATION-SEC,SYMID-1S
// RUN
```

The location name (SEC) is also specified on display 13.0 of the CNFIGICF procedure.

The SYMID (session ID) is explained in the description of the program.

Peer Subsystem Return Codes

This part of Chapter 14 describes all the return codes that are valid for the Peer 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 four-digit value; the first two digits contain the major code, and the last two 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).

Note: In the return code descriptions, *your program* refers to the local System/34 application program that initiates the operation and receives the return code from the subsystem. The *remote program* refers to the application program in the remote (or host) system with which the System/34 application program is communicating through SSP-ICF.

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 Peer subsystem. Although all the operations shown are valid for Peer, their validity also depends on the logical sequence of communications events occurring between the System/34 and the remote system.

Input Operations to Your Program	Output Operations from Your Program	Combined Operations in Your Program
Accept input		
	Acquire[1]	
	End of session	
	Evoke Evoke end of transaction	Evoke then get[2] Evoke then invite
	Fail	
Get Get attributes[3]		
Invite		
	Put Put end of chain Put end of transaction	Put then get[2] Put then invite
	Release	
		Request to change direction then get[2] Request to change direction then invite
	Set timer[4]	

[1] Normally, 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.

[2] Valid only in assembler language.

[3] Valid only in assembler and COBOL languages.

[4] For BASIC and RPG II programs, the set 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.

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, or it received a message from the remote system.

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 remote 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 This Output Operation Was:	Then (in This Session):
Initiated the transaction (evoked the remote program)	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 a remote procedure start request)	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.

[1] An evoked program (started by a procedure start request) 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.

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 code of xx00 (a change direction indication, which allows your program to send data), or xx08 (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 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.

0008 **Normal Indication:** An end of transaction indication was received with the last of the data on a successful input operation. Communications have ended with the remote *program*, but the session with the remote *system* is still active.

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 remote program on a successful *output* operation for your program; the remote program wants to send data as soon as possible. (If the remote system is another System/34, it has issued a *request to change direction then get* operation or a *request to change direction then invite* operation.) You should allow the remote program to send its data.

Normal Action: Issue an input operation as soon as possible.

0028 **Normal Indication:** An end of transaction indication was received with a remote 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. Communications have ended with the remote *program*, but the session with the remote *system* is still active.

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.

0038 **Normal Indication:** An end of transaction indication was received with a *truncated* remote 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. Communications have ended with the remote *program,* but the session with the remote *system* is still active.

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, either issue an end of session operation or terminate.

Major Code 01 – Successful operation with a new requester.

The new requester is a program on a remote system that initiated a session with your program by sending a procedure start request to the local system. 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 procedure start request was initiated by the remote program with an evoke operation (EVK or $$EVOKNI). The request may have included some data for your program.

Normal Description: Each of the following return codes indicates either that the *input* operation issued by your program and responded to by a new requester completed successfully, or that the *output* operation issued by your program in response to a new requester completed successfully.

If the operation was an *input* operation, your program may have received some data from the requester. Any data that was received from the remote system was included in the incoming procedure start request statement.

If your program is an SRT program that was evoked by an incoming procedure start request 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 procedure start request statement 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 some data may have been received with the request. The remote 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[1] for the new requester, and issue an input operation or an output 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 code of xx00 (a change direction indication) or xx08 (an end of transaction indication).

Normal Action: Handle any data passed with the request, perform any necessary record keeping[1] 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.

0103 **Normal Indication:** On a successful input operation from a new requester, a procedure start request and an end of chain indication were received with some data; a complete chain has been received by your program.

Note: This return code is returned only in a multiple-program procedure, and only to each one of the *succeeding* programs in that procedure (not to the first program). Also, it is returned only for the first operation in each of those programs.

Normal Action: Handle the data passed with the request, perform any necessary record keeping[1] for the new requester, and issue another input operation to receive the next chain.

[1] For some situations, no record keeping for the session is necessary. In other situations, you should record the session ID of the new requester. You may also want to keep a table containing the IDs of all active requesters, or to maintain a history log of all requests.

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 remote program. Its communications have ended with your program; however, the session is still active between the local and remote systems.)

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 incoming 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[1] for the new requester of the transaction that has ended. Then, either issue an end of session operation or terminate your program.

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 remote 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 incoming 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[1] for the new requester of the transaction that has ended. Then, because your program was evoked, issue an end of session operation or terminate.

[1] For some situations, no record keeping for the session is necessary. In other situations, you should record the session ID of the new requester. You may also want to keep a table containing the IDs of all active requesters, or to maintain a history log of all requests.

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, or it received a message from the remote system. 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 remote 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 code of xx00 (a change direction indication) or xx08 (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 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.

0208 **Normal Indication:** An end of transaction indication was received with the last of the data on a successful input operation. Although communications have ended with the remote program, the session with the remote system is still active. 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 (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.

0228 **Normal Indication:** An end of transaction indication was received with a remote 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. Although communications have ended with the remote program, the session with the remote system is still active. 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 your program's input buffer (display it, for example). If 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 your program was evoked, either issue an end of session operation or terminate.

0230 **Normal Indication:** A truncated message from the remote system and a change direction indication were received on a successful input operation. The remote system now wants to receive some data. (The message was truncated because it was too long for your program's input buffer.) 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 (possibly display it) in the input buffer, and issue an output operation.

0238 **Normal Indication:** An end of transaction indication was received with a *truncated* remote 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. Although communications have ended with the remote program, the session with the remote system is still active. 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 (to start another program), it can issue a release operation (to perform local processing), or it can terminate. If your program was evoked, either issue an end of session operation or terminate.

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 code of xx00 (a change direction indication, which allows your program to send data), or xx08 (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 remote program has issued a fail operation to indicate, for example, that the previous data 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. Although communications have ended with the remote program, the session with the remote system is still active.

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.

Major Codes 04-34 – Miscellaneous program errors.

Error Description: The operation just attempted by your program failed, or an output exception occurred.

- An operation may have failed because it was issued at the wrong time or because a data record was too long.
- An output exception may have occurred because your program attempted to send output when it should be receiving the output that has already been sent by the remote program.

Recovery Action: Refer to the individual return code descriptions for the appropriate recovery actions.

Code Indication/Action

0412 **Normal (Exception) Indication:** 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 remote program. Your program's output was not sent and should be sent later, after the remote program's data (still waiting in the subsystem input buffer) has been received.

Note: If your program issues another output operation, an error return code of 831C will be received.

Normal Action: Issue an input operation to receive the data waiting in the subsystem input buffer.

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 a new requester) 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 requesting program have been completed.

3401 **Error Indication:** This input operation was rejected because the record length of the data sent by the remote 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/34 Messages Guide* 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 (in COBOL and assembler only) a set timer operation, or by using the TIMER intrinsic function (in BASIC only); (2) continue local processing; or (3) terminate.
- If the session should be started again after the subsystem is enabled, it must be reacquired by your program or restarted by the remote 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 either in this subsystem or in the interrupt handler.

Recovery Action: This subsystem has been disabled; it must be enabled again before communications can resume. Your program can continue local processing, wait[1] to reissue the acquire operation, or terminate.

If more than one subsystem was active when the error occurred, all subsystems that were active when the error occurred should be disabled. (Note that all other active Peer subsystems are automatically disabled when the error occurs; all other types of active subsystems must be manually disabled.)

If all subsystems (of all types) on the system are *not* disabled to recover from the processor check, the common queue space used by the failing subsystem *cannot* be freed. And if it is not freed, that space is wasted, and an indication of insufficient common queue space being available can occur. The indication can occur as a message when the failing subsystem is reenabled or when a different subsystem is enabled. The indication can also occur as a return code to your program for any subsystem that is starting a *new* session (code 8215) or performing an output operation in any *existing* session (code 8315).

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 remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, it can wait[1] until the subsystem has been reenabled and reissue the acquire operation, or it can terminate.

8083 **Error Indication:** An MLCA (multiline communications adapter) controller check occurred on an *output* operation. The subsystem has been disabled.

Recovery Action: Communications with the remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, wait[1] until the subsystem is enabled again, or terminate.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

8084 **Error Indication:** An MLCA (multiline communications adapter) controller check occurred on an *input* operation. The subsystem has been disabled.

Recovery Action: Communications with the remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, wait[1] until the subsystem is enabled again, or terminate.

Major Code 81 – Permanent (nonrecoverable) session error.

Error Description: A nonrecoverable error has occurred in the session; the session cannot be continued and 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/34 Messages Guide* 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 session must be acquired again before communications can resume.

General Recovery Actions: The following general actions can be taken for each 81xx return code. Other specific actions are given in each return code description.

- Several return codes indicate that an error condition must be corrected by changing a value in the subsystem configuration record or in the SESSION statement for your program.
  - To change a parameter value in the subsystem configuration being used by your program, disable the subsystem before making the change in the subsystem's configuration record, and enable the subsystem again to make the change effective.
  - To change a parameter value in the SESSION statement associated with your program, terminate your program only.

  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.
- If the session should be started again, it must be reacquired by your program or restarted by the remote program before communications can resume.
- An end of session (EOS or $$EOS) operation should be issued for the session that has terminated. Your program can also continue local processing, or it can terminate.

Note: If the session is started again, it starts from the beginning, not at the point where the session error occurred.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

Code **Indication/Action**

8183 **Error Indication:** An MLCA (multiline communications adapter) controller check occurred on an *output* operation; data may have been lost. The session has been terminated.

Recovery Action: If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

8184 **Error Indication:** An MLCA (multiline communications adapter) controller check occurred on an *input* operation; data may have been lost. The session has been terminated.

Recovery Action: If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

8185 **Error Indication:** An attempt by this subsystem to automatically call one or more remote locations using the autocall or X.21 feature was not successful. All available numbers in the list of phone numbers or in the list of public data network numbers were called, but no connection was established. The session has been terminated.

Recovery Action: The list has been reinitialized. If the session should be started again, reissue the acquire operation; calling will begin with the first number in the list. Otherwise, your program can continue local processing or terminate.

8191 **Error Indication:** A permanent line error occurred on an *output* operation, and the system operator has taken a recovery option in response to the line error message. (You can find out what type of line error occurred by asking the system operator.) The session has been terminated.

Recovery Action: If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

8195 **Error Indication:** A subsystem disk I/O error occurred while the communications work area on the System/34 disk was being used. The session has been terminated.

Recovery Action: If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again, continue local processing, or terminate.

8196 **Error Indication:** An SNA unbind command was received from the remote system. The session has been terminated.

Recovery Action: If the session should be started again, reissue the acquire operation. Otherwise, your program can continue local processing or terminate.

819D **Error Indication:** On an input or output operation, unexpected data was received from the remote system either after an end of transaction indication was received or before an evoke operation was issued by your program. The session has been terminated.

Recovery Action: Check that your program did not issue an end of transaction operation before the transaction was completed. Also check to see if the remote system sent a procedure start request while your session was still active. If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

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/34 Messages Guide* 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, first disable the subsystem, make the change in the subsystem's configuration record, and 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, (and depending on what the return code description says):
   a. Simply reissue the acquire operation. It could be successful if the error occurred because there was not enough common queue space available to support a new session, or because the communications line was already in use at the time.
   b. If the acquire operation is again unsuccessful, retry it only a limited number of times. (The limit for retries should be specified in your program.)
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 operation is valid only in an *active* session and cannot, therefore, be issued after an 82xx return code is received. (This restriction does not apply to COBOL and assembler programs, or to the TIMER intrinsic function in BASIC, which also can be used to wait for a specified time interval.)

Code	**Indication/Action**
8215	**Error Indication**: On an unsuccessful acquire operation, a queue space error condition was detected. The session cannot be started because the size of the *common* queue space, specified during subsystem configuration, is too small. **Recovery Action**: Your program can wait[1] for a period of time, then reissue the acquire operation. If an unacceptable number of queue space errors occur, you can disable all the subsystems that are active in the system, and change the subsystem configuration by specifying a larger common queue space size in the SSP-ICF common queue space parameter. After the subsystem is enabled, reissue the acquire operation to start the session.
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, respecify 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 either in this subsystem or in the interrupt handler. **Recovery Action**: This subsystem has been disabled; it must be enabled again before communications can resume. Your program can continue local processing, wait[1] to reissue the acquire operation, or terminate. If more than one subsystem was active when the error occurred, all subsystems that were active when the error occurred should be disabled. (Note that all other active Peer subsystems are automatically disabled when the error occurs; all other types of active subsystems must be manually disabled.) If all subsystems (of all types) on the system are *not* disabled to recover from the processor check, the common queue space used by the failing subsystem *cannot* be freed. And if it is not freed, that space is wasted, and an indication of insufficient common queue space being available can occur. The indication can occur as a message when the failing subsystem is reenabled or when a different subsystem is enabled. The indication can also occur as a return code to your program for any subsystem that is starting a *new* session (code 8215) or performing an output operation in any *existing* session (code 8315).

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

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 remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, it can wait[1] until the subsystem has been reenabled and reissue the acquire operation, or it can terminate.

8285 **Error Indication:** An acquire operation, for which the subsystem used the autocall or X.21 feature to automatically call one or more remote locations, was not successful. All available numbers in the list of phone numbers or in the list of public data network numbers were called, but no connection was established. The session was not started.

Recovery Action: The list has been reinitialized. Your program can reissue the acquire operation, continue local processing, or terminate. (Calling will begin with the first number in the list.)

82A6 **Error Indication:** On an unsuccessful acquire operation, the subsystem received an SNA bind command from the remote system that was not in the correct format. The session was not started.

Recovery Action: Determine the error in the format of the received bind command (given in the topic *VTAM/NCP Considerations* in this chapter), and contact the VTAM programmer at the remote location to have the format corrected. Then reissue the acquire operation to start the session.

82A7 **Error Indication:** The acquire operation was unsuccessful because the specified communications line was already in use. The session was not started.

Recovery Action: Your program can wait[1] for the line to become available, then reissue the acquire operation. Otherwise, it can continue local processing or terminate.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

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 100 sessions can be active in the System/34 at one time. The session was not started.

Recovery Action: Your program can wait[1] 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 to be enabled is identified by the location parameter in the SESSION statement. That location name must also be specified in the subsystem configuration record (shown on display 3.0 of the subsystem configuration planning charts). The session was not started.

Recovery Action: Verify that the subsystem name was specified correctly on the LOCATION parameter of the SESSION statement. If the correct name was specified, contact the System/34 system operator and request that the specified subsystem be enabled by executing the ENABLE procedure command at the system console. Then reissue the acquire operation. Otherwise, your program can continue local processing, wait[1] 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[1] until the subsystem has been enabled, and 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[1] until the subsystem is enabled again, and then reissue the acquire operation. Otherwise, your program can continue local processing, or it can terminate.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

82B3 **Error Indication:** The acquire operation was not successful because all of the sessions specified in the subsystem configuration are already in use. The session was not started.

Recovery Action: Wait[1] for one of the sessions in the subsystem to become available, then reissue the acquire operation. Otherwise, continue local processing or terminate.

82B4 **Error Indication:** The acquire operation was not successful because all of the resources needed for the session could not be allocated from the assign/free area of the system. All available resources are already being used in the system. The session was not started.

Recovery Action: Wait[1] for the needed resources to become available, then reissue the acquire operation. Otherwise, continue local processing or terminate.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

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/34 Messages Guide* 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, first disable the subsystem, make the change in the subsystem's configuration record, and 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.

   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.

   Note: If the session is started again, it starts from the beginning, not at the point where the session error occurred.

3. If no change is needed in your program or in the subsystem, (and depending on what the return code description says):
   a. Notify the remote location that a change is required on that end to correct the error received.
   b. Retry the operation, if possible. It could be successful if the error occurred because there was not enough common queue space available at the time, because an isolated line error occurred, or because the remote system was not active at the time.
   c. If another operation is not successful, retry it only a limited number of times. (The limit for retries should be specified in your program.)
   d. Issue a set timer operation in your program so it can wait for a specified time interval before reissuing the operation.

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 an 81xx (session was terminated) or 82xx (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 an 81xx or 82xx return code is handled properly. If you want your program to recover from an improperly handled error condition, issue another acquire operation.

8313 **Error Indication:** On an *output* operation, a queue space error condition was detected. The output operation could not be performed because no *subsystem* queue space was available at the time.

Recovery Action: Your program can issue a set timer operation and wait for a period of time, then reissue the output operation. If an unacceptable number of queue space errors occur, you can disable the subsystem and change the subsystem configuration by specifying a larger subsystem queue space size in the subsystem queue space parameter. After the subsystem is enabled, reissue the acquire operation to restart the session.

8314 **Error Indication:** On an *input* operation, a queue space error condition was detected. The input operation could not be performed because no *subsystem* queue space was available at the time.

Recovery Action: Your program can issue a set timer operation and wait for a period of time, then reissue the input operation. If an unacceptable number of queue space errors occur, you can disable the subsystem and change the subsystem configuration by specifying a larger subsystem queue space size in the subsystem queue space parameter. After the subsystem is enabled, reissue the acquire operation to restart the session.

8315 **Error Indication:** On an evoke operation, a queue space error condition was detected. The evoke operation could not be performed because no *common* queue space was available at the time.

Recovery Action: Your program can issue a set timer operation and wait for a period of time, and then reissue the evoke operation. If an unacceptable number of queue space errors occur, you can disable all the subsystems and change the subsystem configuration by specifying a larger common queue space size in the SSP-ICF common queue space parameter. After the subsystem is enabled, reissue the acquire operation to restart the session.

8316 **Error Indication:** The evoke operation just performed was not successful because the program being evoked could not be found.

Recovery Action: Verify that the program name was specified correctly on the evoke operation statement in your program. If it was, call the remote location to determine if the program is still in the remote system.

831A **Error Indication:** An evoke operation failed to complete successfully, or the remote program terminated abnormally. A message from the remote system describing why it failed is waiting in the subsystem input buffer. The evoke operation could have been 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 and continue local processing, or it can terminate.

831C **Error Indication:** The output operation issued before this output operation received a return code of 0412 (indicating that the remote program sent 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 data.

Recovery Action: Issue an input operation to receive the 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. 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 before attempting to communicate with the remote 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: If you want your program to recover dynamically, reissue the output operation with a smaller output length. Otherwise, you can either change the record length in your program and recompile it, or you can change the value specified for the maximum user record length parameter in the subsystem configuration. 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 erroneously followed by a *request to change direction then get* operation, a *request to change direction then invite* operation, or a release 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 other 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 indication). The session is still active.

Recovery Action: Before attempting to communicate with another other program, correct the error in your 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 remote 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 remote program. The session is still active.

Recovery Action: If you want your program to dynamically 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 incoming procedure start request and cannot, therefore, issue any evoke operations in this session.

Recovery Action: If you want your program to dynamically 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 a release operation. If a coding error in your program caused the error, correct your program.

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.

8334 **Error Indication:** On an evoke operation issued by your program, no procedure name was included with the operation.

Recovery Action: Correct the evoke operation statement by supplying the correct procedure name, and reissue the operation.

8397 **Error Indication:** An abort transmission sequence was received from the remote system on an *output* operation; the remote system is terminating the line transmission abnormally because it could not or did not want to continue communicating with your program. The session, however, is still active.

Recovery Action: If your program started the session, issue another evoke operation to start a different transaction. If your program was evoked, it can continue local processing, or it can terminate.

83B0 **Error Indication:** The 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; this session, however, is still active.

Recovery Action: Your program can wait (using the set timer operation) until the subsystem is enabled again, and then reissue the acquire operation. Otherwise, your program can continue local processing or terminate.

Chapter 15. The SNA Upline Facility Subsystem

The SNA upline facility (SNUF) subsystem provides distributed data processing support to users of the System/34 in conjunction with a System/370 using CICS/VS or IMS/VS. The SNUF subsystem provides an interactive and batch interface between System/34 application programs on a CICS/VS system or an IMS/VS system. The SNUF subsystem can support multiple application programs concurrently communicating with CICS/VS and IMS/VS.

The SNUF subsystem allows System/34 application programs to initiate tasks on the CICS/VS or IMS/VS system. System/34 security options as well as CICS/VS and IMS/VS security options are supported.

When communicating with IMS/VS, the subsystem performs message resynchronization and retransmission for failures other than those on the System/34. The SNUF subsystem isolates the application program from SNA protocol requirements. A pass-through function, however, allows the application programmer to write programs that send and receive the SNA protocols. The pass-through support is described in Appendix B.

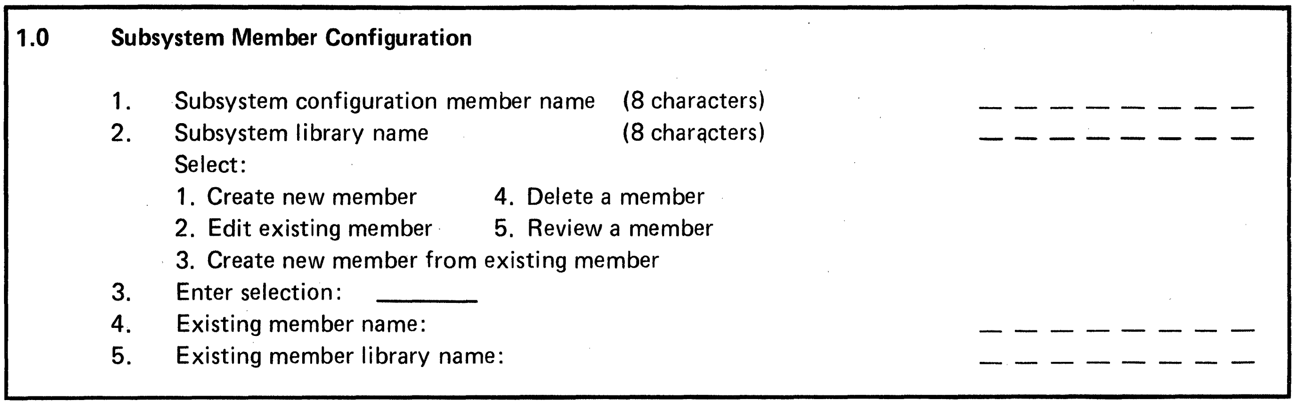
The SNUF subsystem supports up to 32 concurrent sessions, two of which are reserved for incoming procedure start requests.

SETTING UP THE SNUF SUBSYSTEM

The SSP procedures CNFIGSSP and INSTALL are used to include the interactive communications feature and SNUF subsystem support on the System/34. The general interactive communications and line control support is included when it is requested on the appropriate CNFIGSSP prompt. The SNUF subsystem support is then copied to the system library when the appropriate responses to the INSTALL procedure prompts are taken. The CNFIGSSP and INSTALL procedures, with their displays and related responses, are described in the *Installation and Modification Reference Manual.*

After the SNUF subsystem has been installed, the CNFIGICF procedure is used to tailor the subsystem support to an existing or proposed network. The operation of the CNFIGICF procedure is also explained in the *Installation and Modification Reference Manual.* Before running the CNFIGICF procedure, however, you should fill out a planning chart for each subsystem that you want to define. Copies of the planning chart for each subsystem are available in Appendix F of this manual and in the *Installation and Modification Reference Manual.* The following sections explain how to fill out the planning chart for the SNUF subsystem.

Display 1.0 Subsystem Member Configuration



```
1.0     Subsystem Member Configuration

        1.   Subsystem configuration member name  (8 characters)          _ _ _ _ _ _ _ _
        2.   Subsystem library name               (8 characters)          _ _ _ _ _ _ _ _
             Select:
             1. Create new member       4. Delete a member
             2. Edit existing member    5. Review a member
             3. Create new member from existing member
        3.   Enter selection: ____
        4.   Existing member name:                                        _ _ _ _ _ _ _ _
        5.   Existing member library name:                                _ _ _ _ _ _ _ _
```

Subsystem configuration member name: Specify a name for this configuration of the subsystem. This name is used to store the member in a library and is referenced in the ENABLE and DISABLE procedures.

Library name: Specify the name of a library in which the configuration is stored or to be stored. The default is #LIBRARY, however, you should probably store the configuration in a user library.

Enter selection: Specify one of the five options: (1) create a new member, (2) edit an existing member, (3) create a new member from an existing member, (4) delete a member, or (5) review a member without changing it.

Existing member name: This prompt appears only if option 3 was selected. Specify the name of the existing subsystem configuration member that is to be used to create the new member. The existing member remains unchanged.

Existing member library name: This prompt appears if option 3 was selected. Specify the library name where the existing member resides.

Display 2.0 Common SSP-ICF Parameters for Each Subsystem

```
2.0   Common SSP-ICF Parameters for Each Subsystem

      1.  SSP-ICF common queue space: (2 - 42 K)                  _ _
      2.  Define the subsystem type:                              _ 6
          1  Intra              2  BSC IMS/IRSS
          3  BSCEL              4  BSC CICS
          5  BSC CCP            6  SNA Upline
          7  SNA Peer           8  BSC 3270
          9  SNA 3270          10  Finance
```

SSP-ICF common queue space: Specify the size, in multiples of 2 K bytes, of the common queue space. The common queue space requirements for each configuration of the SNUF subsystem enabled are:

$$C = 180 + 3M + 2X + 220(Y + Z)$$

where:
C = number of bytes required for the common queue space
M = maximum number of sessions
X = number of transmit buffers
Y = number of acquired sessions that will be active concurrently
Z = number of incoming sessions (0, 1, or 2) that will be active concurrently

The common queue space value specified for the first subsystem that is enabled becomes the size of the common queue space. Be sure that the value specified for common queue space size takes into account the requirements of any other subsystem that might be active concurrently.

The size of the common queue space plus the total subsystem queue space sizes of all enabled SNUF subsystems cannot exceed 42 K bytes.

The default common queue space size is 4 K bytes.

Define subsystem type: Specify a 6 for the SNA Upline subsystem.

Display 3.0 General Subsystem Parameters

3.0 General Subsystem Parameters

1. Location name: (8 characters) _ _ _ _ _ _ _ _
2. Subsystem queue space: (0-40 K) _ _
3. Subsystem support swappable: (0-No 1-Yes) _
4. Maximum user record length: (1-4075) _ _ _ _

Location name: Specify up to 8 characters for the name of the location associated with this configuration. The location name is used in some of the displayed message texts. If no remote locations are defined, the location name must be coded on the SESSION OCL statement. The default is the subsystem configuration member name.

Subsystem queue space: Specify the size, in multiples of 2 K bytes, of the subsystem queue space. The subsystem queue space is only required for pass-through sessions. For each acquired pass-through session, 328 bytes are required. For sessions other than pass-through, no subsystem queue space is required; however, improved performance may be obtained by using subsystem queue space.

Whenever a record is to be sent or received, it must be moved to an intermediate buffer between the application program and the SDLC task. The SNUF subsystem attempts to assign an intermediate buffer out of the subsystem queue space. If no queue space is available, a disk work area is used for the buffer. When the disk work area is used, main storage requirements for the subsystem queue space are reduced, but the amount of time required to move data between the application program record area and the intermediate buffer is greatly increased.

Factors affecting subsystem queue space requirements are:

- The number of concurrent sessions: Each active session competes for intermediate buffers from subsystem queue space. The more sessions competing, the greater the possibility of a failure to obtain an intermediate buffer.

- The size of records: The amount of storage required for an intermediate buffer is equal to the size of the record to be transmitted. For records to be received, the amount of storage is equal to the maximum user record length.

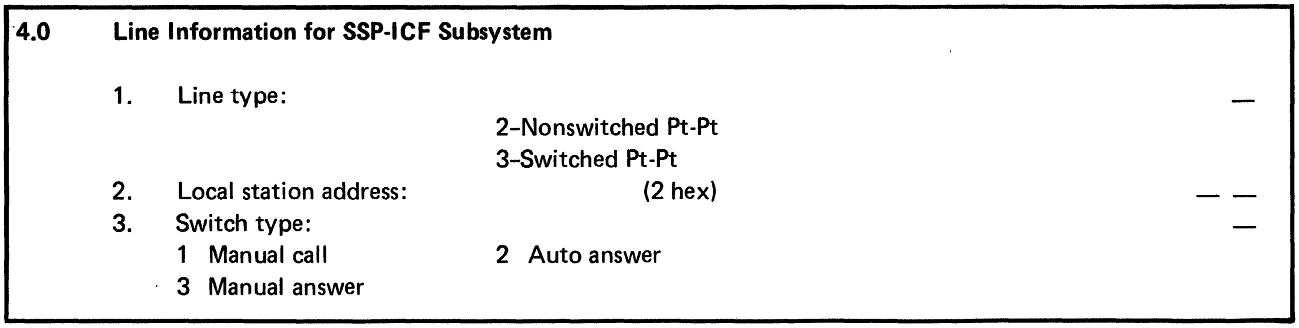
- Frequency of transmission: Sessions that transmit or receive records infrequently are more likely to successfully obtain an intermediate buffer than sessions that frequently send and receive records.

- Response requirements: Interactive programs that transmit and receive infrequently may be able to accept the slower response times that occur when intermediate buffers are on disk.

Subsystem support swappable: Specify whether you want the SNA task and the subsystem to be swappable. Consider the total system performance, the size of the subsystem, and the amount of user main storage when determining whether you want the subsystem to be swappable. The total amount of main storage required by the SNUF subsystem and the SNA task is 20 K bytes.

Maximum user record length: Specify the maximum record length (1 through 4075 bytes) to be sent or received by a System/34 program using this subsystem configuration. The default is 1024 bytes. This parameter affects the size of the communications work area. See *Disk Space Requirements* later in this chapter for details.

Display 4.0 Line Information for SSP-ICF Subsystem



```
4.0     Line Information for SSP-ICF Subsystem

        1.   Line type:                                                        _
                                   2-Nonswitched Pt-Pt
                                   3-Switched Pt-Pt
        2.   Local station address:            (2 hex)                        _ _
        3.   Switch type:                                                      _
             1  Manual call        2  Auto answer
             3  Manual answer
```

Line type: Specify the line type that is suitable to your communications environment. If the System/34 is connected to a multipoint line, specify 2 (nonswitched point-to-point). The default is 2 (nonswitched point-to-point).

Local station address: Specify the System/34 secondary SDLC station address identified for this configuration. This value must be specified as two hexadecimal characters in the range of 01 to FE.

Switch type: This prompt is displayed only if switched point-to-point line type was selected. Specify the switch type you want for the line. If the switch type is defined as auto answer, the adapter is initially enabled for automatic answer. If a System/34 application program attempts an acquire operation before the line is connected, the switch type will be dynamically changed by the subsystem to allow the System/34 to automatically make the call or to allow the System/34 operator to make the call. See *Switched Line Considerations* in this section for additional information.

Note: If the SNUF subsystem is to be run concurrently with SNA 3270 device emulation, the values from this display specified in the two configurations must agree. If the values are not the same, an error occurs when the second subsystem is enabled.

Display 7.0 Subsystem Inactive Destination Messages

7.0 **Subsystem Inactive Destination Messages**

1. Subsystem procedure name: (8 characters) _ _ _ _ _ _ _ _
2. Subsystem procedure library name: (8 characters) _ _ _ _ _ _ _ _

Subsystem procedure name: Specify the name of a user-written procedure to be started when data or messages are received for an inactive session (uninvited data). This can happen when the SNUF subsystem receives data on the program start session that does not begin with *EXEC or *EXEX. When uninvited data is received, the procedure is run to receive the data. The program within this procedure cannot send data on this session. The program can save the data in a file, process the data as it is received, or discard the data.

Subsystem library name: Specify the name of the library containing the inactive destination messages procedure.

Display 8.0 SNA Upline Subsystem Parameters I

8.0 **SNA General Subsystem Parameters I**

1. SDLC Buffer pool size: (2-8 K) _
2. Number of transmit buffers: (1-15) _ _
3. Maximum number of sessions: (1-32) _ _
4. Maximum receive pacing count: (1-63) _ _
5. Local ID: (Hexadecimal) _ _ _ _ _
6. LU configuration table library name: _ _ _ _ _ _ _ _
7. LU configuration table member name: _ _ _ _ _ _ _ _

SDLC buffer pool size: Specify the amount of main storage, in multiples of 2 K bytes, to be used for SDLC buffers.

The following table shows the number of buffers available for each possible value specified:

Pool Size	**Buffers**
2 K	7
4 K	15
6 K	22
8 K	30

The default is 4 K. The total active secondary SDLC buffer pool size (including all users of secondary SDLC) cannot exceed 48 K.

Number of transmit buffers: Specify the number of transmit buffers you require. Specify a number from 1 to 15. All transmit buffers will be in main storage. If enough main storage is not specified for the number of transmit buffers and receive buffers, an error message is displayed. For example, if you specified 2 K of main storage for the SDLC buffer pool, you would have 7 buffers available. Three buffers are required to be receive buffers; therefore, you would be able to specify up to 4 transmit buffers. The number of receive buffers in main storage is equal to the total number of buffers available minus the number of transmit buffers. The default number of transmit buffers is 7.

Maximum number of active sessions: Specify the maximum number of sessions that can be active at one time. You can specify from 1 to 32 active sessions. Two sessions are reserved for incoming procedure requests, so if sessions are to be acquired, a value of at least 3 must be specified. The default is 8. This parameter affects the size of the communications work area. See *Disk Space Requirements* later in this chapter for details.

Maximum receive pacing count: Each logical unit of the host system contains a receive pacing count. This count directly affects the amount of application program data that can be queued by System/34 SNA. Specify the maximum receive pacing count that was determined on the host system. You can specify a value from 1 to 63. If large quantities of data records are to be sent, a larger receive pacing count could allow them to be sent more efficiently. If only one or a few records will be sent at a time, a smaller receive pacing count will be sufficient and will require less storage for receive buffers. The default is 3. This parameter affects the size of the communications work area. See *Disk Space Requirements* later in this chapter for details.

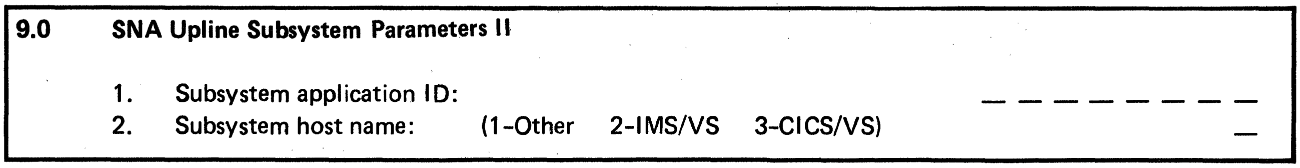
Local ID: Specify 5 hexadecimal digits to be used as the SDLC exchange of identification for the System/34. The default is 00000. This ID must be the same as the host specified remote ID. (The host specifies the ID in the IDNUM parameter of the PU macro for VTAM switched line definition.)

LU configuration table library name: Specify the name of the library in which the LU configuration table is stored. This table is required if the SNUF subsystem and SNA 3270 device emulation are to be run concurrently on the same line. If this parameter is not specified, no configuration table is assumed to exist.

LU configuration table member name: Specify the name of the member containing the LU configuration table. This name corresponds to the name assigned to the table when it was built using the DEFINELU procedure (see Appendix G). This table is required if the SNUF subsystem and SNA 3270 device emulation are to be run concurrently on the same line. If this parameter is not specified, no configuration table is assumed to exist.

Note: If the SNUF subsystem is to be run concurrently on the same line with SNA 3270 device emulation, the values for parameters 1, 2, 5, 6, and 7 from this display specified in the two configurations must agree. If the values are not the same, an error occurs when the second subsystem is enabled.

Display 9.0 SNA Upline Subsystem Parameters II



9.0 SNA Upline Subsystem Parameters II

1. Subsystem application ID: _ _ _ _ _ _ _ _
2. Subsystem host name: (1-Other 2-IMS/VS 3-CICS/VS) _

Application ID: Specify the VTAM application ID to be sent on the logon message. The APPLID parameter on the SESSION OCL statement overrides this parameter.

Host name: Specify the protocol to be used for the session:

1. Other specifies that the pass-through support will be used.

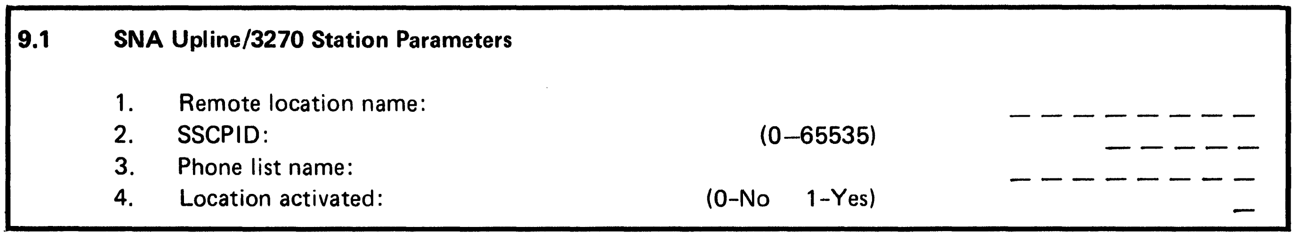
2. IMS/VS specifies that the IMS/VS protocol will be used.

3. CICS/VS specifies that the CICS/VS protocol will be used.

The HOSTNAME parameter on the SESSION statement overrides this parameter.

Display 9.1 SNA Upline/3270 Station Parameters

The following prompt is displayed only if you specified *switched pt-pt* on display 4.0. You can configure up to 32 remote locations. If you do not specify a remote location, the location name you specified on display 3.0 is used, and the SSCPID for the remote location is not checked.



```
9.1       SNA Upline/3270 Station Parameters

          1.   Remote location name:                                          _ _ _ _ _ _ _ _
          2.   SSCPID:                          (0–65535)                             _ _ _ _ _
          3.   Phone list name:                                               _ _ _ _ _ _ _ _
          4.   Location activated:           (0-No    1-Yes)                                 _
```

Remote location name: Specify up to 8 characters for the name of the remote system location associated with this configuration. The remote location name is used in some displayed messages. The name must also be specified on the SESSION OCL statement.

SSCPID: Specify the 5-digit decimal number that identifies the SSCP for this location. This value must be the same as the parameter in the ACF/VTAM start procedure at the remote location. See the *Advanced Communications Function for VTAM (ACF/VTAM) System Programmer's Guide* for more information about the SSCPID parameter.

Phone list name: Specify the name of the load member that contains the list of phone numbers to be called by the autocall feature or the list of numbers for the public data network to be called by the X.21 feature. The list is created using the DEFINEPN or DEFINX21 procedure. These procedures are described in the *System Support Reference Manual.* The list must be in the current user library or #LIBRARY. If you specify a list, also specify 2 (auto answer) on display 4.0 of the CNFIGICF procedure.

Location activated: Specify 1 (yes) if you want this location enabled when the configuration is enabled. Specify 0 (no) if you want this location enabled with a separate ENABLE command. The location must be enabled before it can be used.

Disk Space Requirements

The part of the user disk space used by the SNUF subsystem is referred to as the communications work area. This area is not available to the user when the SNUF subsystem is enabled. Three user specified factors that affect the size of the communications work area are the maximum number of active sessions, the maximum receive pacing count, and the maximum user record length. The following formula determines the size of the communications work area:

$$B = 1 + \frac{X}{10}\left(4Y + \frac{Z}{256} + 2\right)$$

where:

B = Number of blocks used for the communications work area (round up to the next whole block).
X = Maximum number of active sessions.
Y = Maximum receive pacing count.
Z = Maximum user record length.

For example, during CNFIGICF you specified the following for the SNUF subsystem:

- The maximum number of active sessions was 4.
- The maximum receive pacing count was 3.
- The maximum user record length was 512.

The number of blocks used for the communications work area would be 8:

$$1 + \frac{4}{10}\left(12 + \frac{512}{256} + 2\right) = 7.4 \text{ round up to } 8$$

SETTING UP THE HOST SYSTEM

Host system generation must occur before communications with the System/34 can take place. This could include IMS/VS, CICS/VS, VTAM, and NCP generation or any combination necessary to set up the network. See the specific generation considerations for each of these products later in this chapter.

STARTING AND ENDING THE SNUF SUBSYSTEM

To start the SNUF subsystem, the ENABLE procedure must be run. The ENABLE procedure performs the following functions for SNUF:

- Determines whether the line requested is available
- Loads and attaches the SNUF subsystem if it is not already active
- Loads and attaches the SNA and SDLC tasks if they are not already active
- Assigns storage for required data areas and buffers
- Allocates the disk file for the communications work area
- Initializes program start logical units

To de-activate the SNUF subsystem, the DISABLE procedure must be run. When a disable is performed for the SNUF subsystem, the subsystem waits for the host system to de-activate the sessions and send an SNA disconnect command. If something prevents the host from doing either (for example, a machine check at the host), the subsystem remains active until a link error occurs, an IPL is performed at the System/34, or a second disable is requested.

STARTING SNUF SUBSYSTEM APPLICATIONS

System/34 SNUF subsystem applications can be started by a display station operator entering a procedure command, or by a request from the host system. Procedures that are started by the System/34 operator must contain a SESSION OCL statement for each session to be started. Requests from the host system to start a procedure must be in a special format. The following sections describe the SESSION statement and the incoming procedure requests.

Session OCL Statement

The format of the SESSION statement for the SNUF subsystem is:

$$\text{// SESSION LOCATION-name ,SYMID-session-id}$$

$$\left[\text{, APPLID-application-id}\right] \quad \left[\text{, RECL-nnnn}\right]$$

$$\left[\text{,HOSTNAME-}\left\{\begin{array}{l}\text{IMSRTR}\\ \text{IMS}\\ \text{CICS}\\ \text{OTHER}\end{array}\right\}\right] \quad \left[\text{,FMHI-}\left\{\begin{array}{l}\text{YES}\\ \underline{\text{NO}}\end{array}\right\}\right]$$

$$\left[\text{, MSGPROT-}\left\{\begin{array}{l}\underline{\text{YES}}\\ \text{NO}\end{array}\right\}\right] \quad \left[\text{, BATCH-}\left\{\begin{array}{l}\text{YES}\\ \underline{\text{NO}}\end{array}\right\}\right]$$

LOCATION: Specifies the location name associated with this session. The location name is defined during subsystem configuration and refers to the name of the location with which communications is to take place.

SYMID: Specifies the symbolic ID of the session with which this SESSION statement is associated. The symbolic ID must be two characters, with the first character numeric (0 through 9) and the second character alphabetic (A through Z, #, $, or @). This is the same ID that the application program uses when referring to this session. This ID is the equivalent of the symbolic display station ID as specified on the WORKSTN OCL SESSION statement. This parameter has no default.

APPLID: Specifies the VTAM application ID to be sent with the logon message. If this parameter is not specified, the ID specified during subsystem configuration or during enable is used.

RECL: Specifies the maximum record length that will be transmitted or received for this session. The record length can be any decimal value from 1 through 4075 bytes. For evoke operations, the record length must include the length of the transaction ID plus 1 byte, plus the length of any accompanying data. If an IMS/VS password is included, its length plus 1 must also be added.

HOSTNAME: Specifies the type of host system protocol to be used for this session. CICS indicates to use the CICS/VS protocol; IMS indicates to use the IMS/VS protocol; OTHER indicates that the pass-through support will be used; IMSRTR indicates that the IMS/VS protocol will be used with the ready to receive option. (See *IMS/VS Ready to Receive Option* later in this chapter for more information.)

FMHI: Specifies whether received function management headers should be passed to the application program. YES indicates that they should be passed. NO indicates that the SNUF subsystem will remove function management headers before passing data to the application program.

MSGPROT: Specifies whether message protection is to be used for this session. YES indicates that message protection will be used. The SNUF subsystem will save messages until they are acknowledged, and attempt resynchronization if errors occur. See *Session Restart and Recovery* later in this chapter for more details. YES is the default, and is only valid with BATCH-NO. NO indicates that message protection will not be used.

BATCH: Specifies whether this session will be used for batch activity. YES indicates that batch activity will occur, and that the SNUF subsystem will not assemble physical records into logical records. NO indicates that batch activity will not occur. For more information on the BATCH parameter, see *Put Operations* and *Input Operations* later in this chapter. If BATCH-YES is specified, MSGPROT-NO must also be specified.

Incoming Procedure Start Requests

For CICS/VS or IMS/VS applications to initiate programs on the System/34, the host application must transmit an *EXEC or an *EXEX request. The format of the *EXEC and *EXEX requests is in Chapter 2.

To facilitate procedure start, the SNUF subsystem accepts procedure start requests on sessions that have been reserved for this purpose. Logical unit numbers 1 and 2 are reserved for incoming procedure start requests.

The begin bracket and first of chain indicators must accompany the procedure start request. If the procedure start request is for end of transaction (*EXEX), an end bracket indicator must accompany the request.

The program start logical units must be started by the host system sending a bind command. This can be accomplished by using the VARY command with the LOGON option, the LOGAPPL parameter in the VTAM definition, or the appropriate host subsystem (IMS/VS or CICS/VS) procedures. The end of session operation can be issued after an end of transaction return code to release the session so that other procedure start requests can be handled.

Communicating programs started by the host system (via *EXEC) are treated as if they specified BATCH-YES on the SESSION OCL statement. This means that each input request from the System/34 application program is satisfied with only one element of a chain instead of the entire chain. The end of chain return code (xx03 or xx07) is set when the last element of the chain is received, if not overridden by the end of transaction return code (xxx8).

When a program start session is terminated by the host system, a message is displayed on the System/34 system console indicating that the logical unit is inactive. If the session can be restarted, a message is displayed indicating that the logical unit is waiting for a bind command. If the restart of the session involves resynchronization (with the STSN command), the message that caused the failure may be resent. Operator intervention at the host system may be required to remove the message from the output queue.

If an attempt to start a procedure is not successful, the subsystem sends a negative response to the host system with a system sense code of hex 0812. The user sense code contains the number of the message (as hex digits) associated with the error. For example, if a procedure start request specifies an unrecognizable procedure name (message SYS 1305), a negative response is sent with a sense code of hex 08121305.

More specific information on what must be done by the host system application is in the IMS/VS or CICS/VS considerations later in this chapter.

OPERATION CONSIDERATIONS

The following sections describe the operations supported by the SNUF subsystem. A complete chart of all interactive communications operations and the subsystems that support them is in each language chapter. The chart also shows the keyword or format name used to code the operation. More information about how an operation is coded is also described in the appropriate programming language chapter.

Whether an operation completes successfully or unsuccessfully, a return code is given to the application program. All of the return codes that are valid for this subsystem are described at the end of this chapter. A summary chart in Appendix A lists all the return codes and the subsystems for which they are valid.

Acquire Operation

The acquire operation is used to start a session. The acquire operation causes the following initialization sequence to take place:

- If the session will use a switched pt-pt line, the line connection is established (if it is not already established).
- The SNUF subsystem sends a logon to the host system.
- The SNUF subsystem examines the bind command parameters from the host system and sends a positive response if the parameters are acceptable.
- When the host system is ready to begin the session, the subsystem gives the application program a normal return code, which indicates that the session was started successfully.

Evoke Operation

The evoke operation ($$EVOK, $$EVOKNI, or $$EVOKET) is used to send a transaction ID to the host system and optionally to include data.

The evoke end of transaction operation ($$EVOKET) indicates that the System/34 application program does not expect an immediate reply from the host application program. The protocols used for this operation are different for CICS/VS than for IMS/VS. See the appropriate host considerations for details.

Put Operations

The put operations ($$SEND, $$SENDNI, $$SENDE, and $$SENDET) allow the application program to send data and/or control information to the host system. A put operation permits the application program to send data to a communicating transaction partner. A put operation can also be used to send commands, such as the IMS /SET command, to the host system.

The manner in which logical records are sent depends on the BATCH parameter of the SESSION OCL statement. If BATCH-NO is chosen, each logical record is sent as a complete chain. The SNUF subsystem automatically divides records that are greater than 256 bytes into elements of a chain. If BATCH-YES is chosen, logical records are divided as required, but are not chained.

The put end of chain operation ($$SENDE) allows an application program to break a stream of data into smaller units. This causes the SNUF subsystem to set the end of chain indicator and request a definite response. The application program receives control when a positive response is received. Put end of chain is only valid if BATCH-YES was specified. The put end of chain operation can be sent with or without data. If no data is to be sent, the record length must be set to zero.

The put end of transaction operation ($$SENDET) is used to ask for termination of a transaction. The SNUF subsystem sends the end of chain and change direction indicators. If an end bracket with no data is the reply from the host system, a normal return code is set; otherwise, the session is abnormally terminated. Put end of transaction can be sent with or without data. If no data is to be sent, the record length must be set to zero.

The put function management header operation (assembler only) indicates to the SNUF subsystem that a function management header is in the data buffer. A function management header contains information concerning the data that follows. A function management header is valid only with the first record in a chain; that is, after issuing a put end of chain, the first put operation, or after receiving a change direction or end of chain.

Input Operations

The input operations for the SNUF subsystem are invite, get, and accept. The invite operation can be issued only as a combined operation with a put or evoke ($$SEND, $$EVOK) in BASIC, COBOL and RPG II. Assembler language users can issue an invite operation explicitly. Either a get or invite operation signals the subsystem to obtain data on the session for the application program. A get operation causes the application program to wait for the data to be available. When a program issues an invite operation, it receives the data with the next accept operation. The accept operation allows input from any previously invited session.

The record that the application program receives depends on the BATCH parameter of the SESSION OCL statement. If BATCH-NO was specified, each physical record is assembled into a logical record until the end of chain indicator is reached. For more information on chains, see *Chains* later in this chapter.

Data is not always passed to the application program after an input request. In certain instances, only a return code is set to indicate a change in the operating state of a session.

Request to Change Direction Operation

The request to change direction operation ($$RCD) indicates that the System/34 application program wants to transmit data. This operation can only be issued if the System/34 program is receiving. After issuing the request to change direction operation, the System/34 program should continue to receive data until it receives a return code indicating that the host application program is ready to receive data. The System/34 program can then begin sending data.

Release Operation

The release operation breaks the logical connection between the application program and the SNUF subsystem. When the release operation is issued, the SNUF subsystem ends communication with the application program, and makes the session available for use.

Negative Response Operation

The negative response operation ($$NRSP or $$NRSPNI) allows an application program to send sense information with an SNA negative response. To send sense information, the application program must put the sense codes into the logical record buffer. The system sense code is the first four characters in the record buffer, and the next four characters are the user sense code. The system sense code must be 10xx, 08xx, or 0000. If the system sense code is not one of these forms, the operation is rejected. If no sense code is supplied by the application program, the default system sense code of 0811 (break) is used. A complete list of supported SNA sense codes is contained in the *Systems Network Architecture Reference Summary*.

After issuing the negative response operation ($$NRSPNI), the application program must issue an input operation to determine the action taken by the host system. The negative response then get operation (assembler only) allows the application program to combine the two operations. Also available is a negative response then invite operation ($$NRSP), which allows the application program to perform other processing while the host system prepares and sends the reply.

Cancel Operation

The cancel operation ($$CANL or $$CANLNI) is used to terminate a partially sent group (chain) of records. This operation results in an error return code if a cancel operation was issued on a session that was specified as BATCH-NO on the SESSION OCL statement or if issued between chains. See *Chains* in this chapter for a complete description of chains.

A typical use of the cancel operation would be as follows:

> A System/34 program reads and edits records from a local file and sends them to a host application. If, after sending several records to the host, a bad record is found, a cancel operation can be issued by the System/34 program to terminate the chain and tell the host to ignore the previous records in the chain.

The host system performs error recovery upon receipt of a cancel. The System/34 application program should, after sending the cancel, perform an input operation to determine the error recovery taken by the host. The cancel then get operation (assembler only) allows the application program to combine these two operations. The cancel then invite operation ($$CANL) allows the application program to perform other processing while the host system prepares and sends the response.

End of Session Operation

The end of session operation ($$EOS) always results in a normal completion return code. The session is always terminated by the end of session operation. If the session is still communicating when the end of session operation is issued, the transmission is abnormally terminated by the SNUF subsystem.

Get Attributes Operation

The get attributes operation (assembler only) can be issued at any time to determine the status of a session.

Set Timer Operation

The set timer operation ($$TIMER) results in a timer expired return code (0310) after a specific time interval in hours, minutes, and seconds has elapsed.

Switched Line Considerations

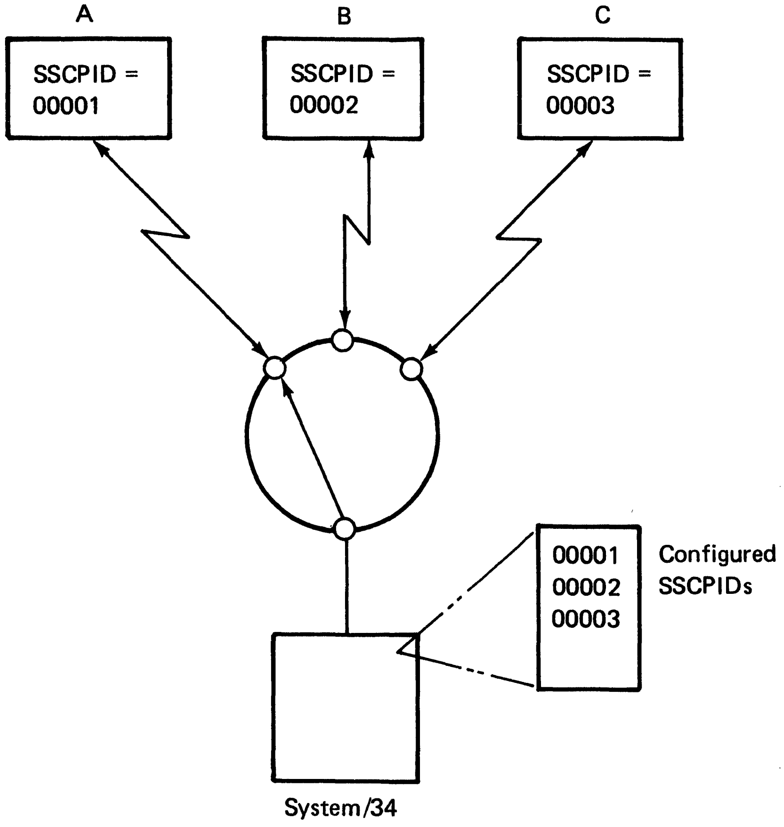
When switched lines are used with the SNUF subsystem (*switched pt-pt* specified on display 4.0 of the CNFIGICF procedure), the connection with the remote location can be established in one of the following ways:

- If the switch type parameter is manual call, the message OPERATOR DIAL REQUIRED is displayed when the subsystem is enabled.
- If the switch type parameter is manual answer, the message OPERATOR ANSWER REQUIRED is displayed when the subsystem is enabled.
- If the switch type parameter is auto answer, the System/34 answers a call from a remote location without operator intervention.
- If the switch type parameter is auto answer and a phone list name is specified on display 9.1, the System/34 calls the remote location without operator intervention. If the autocall feature is not available, the message OPERATOR DIAL REQUIRED is displayed on the system console when an acquire operation is performed.

Multiple Remote Locations

The SNUF subsystem allows more than one location to be enabled on a switched line. This allows an application program to communicate with multiple locations (one at a time) without disabling one configuration and enabling another.

The following illustration shows the System/34 and three possible locations with which the System/34 can communicate on switched lines. In this example, the System/34 is connected to Location A. If the System/34 calls the remote location, the SNUF subsystem uses the SSCPID (00001 for Location A in this example) to ensure that the correct location has been called. If a remote location calls the System/34, the subsystem uses the SSCPID to determine which location called.



Note: The subsystem assumes that each location has a unique SSCPID. If the SSCPIDs are not unique, the System/34 cannot determine which location is connected.

The following sequence shows how the subsystem establishes a connection when an acquire operation is performed and multiple locations are configured:

Is a location connected?

Yes No

A connection is established
with the location specified on the
SESSION OCL statement,
and the acquire operation continues.

Is the location connected the same as
the location specified on the
SESSION OCL statement?

Yes No

The acquire operation is terminated,
and return code 82A7
(*line not available*) is passed to the
application program.

The acquire operation continues as requested.

A connection with a remote location is disconnected in one of the following ways:

- The subsystem disables the location using the DISABLE procedure. Once disabled, the location cannot be used again until it is enabled.

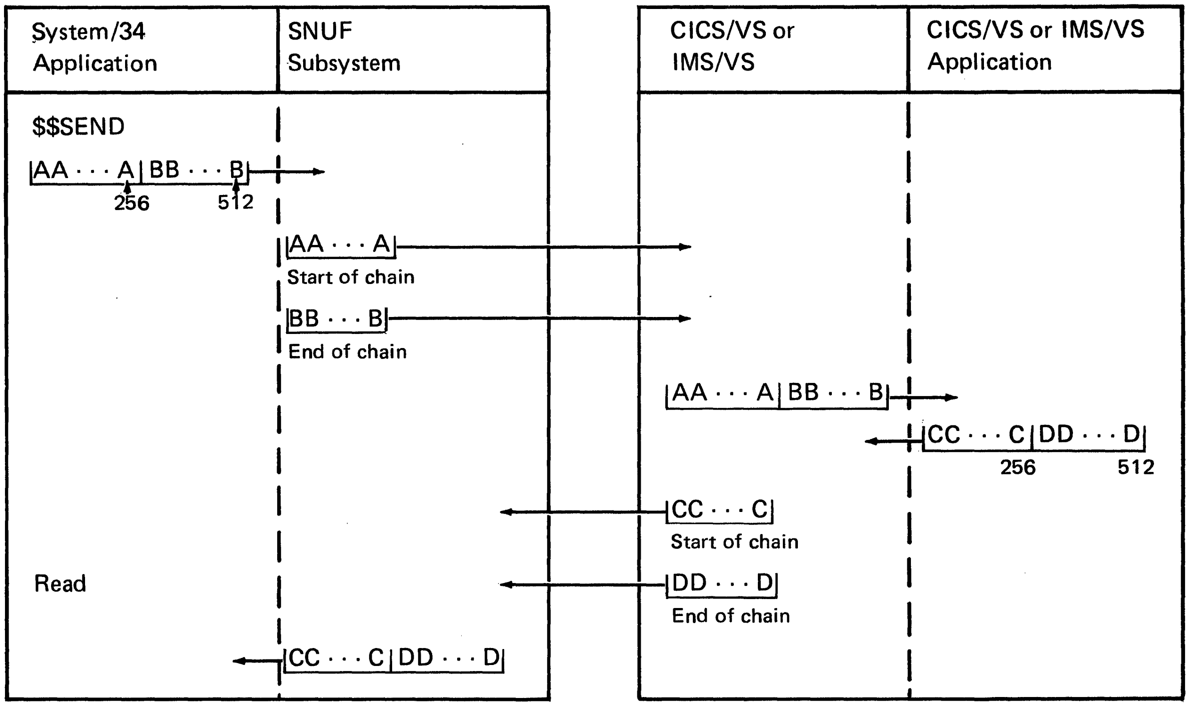
- The remote system disconnects the line. The location is not disabled and can be used again without being enabled.

- The location is automatically disconnected when a session is complete if DISCNT = YES is specified on the PU statement during the ACF/VTAM generation.

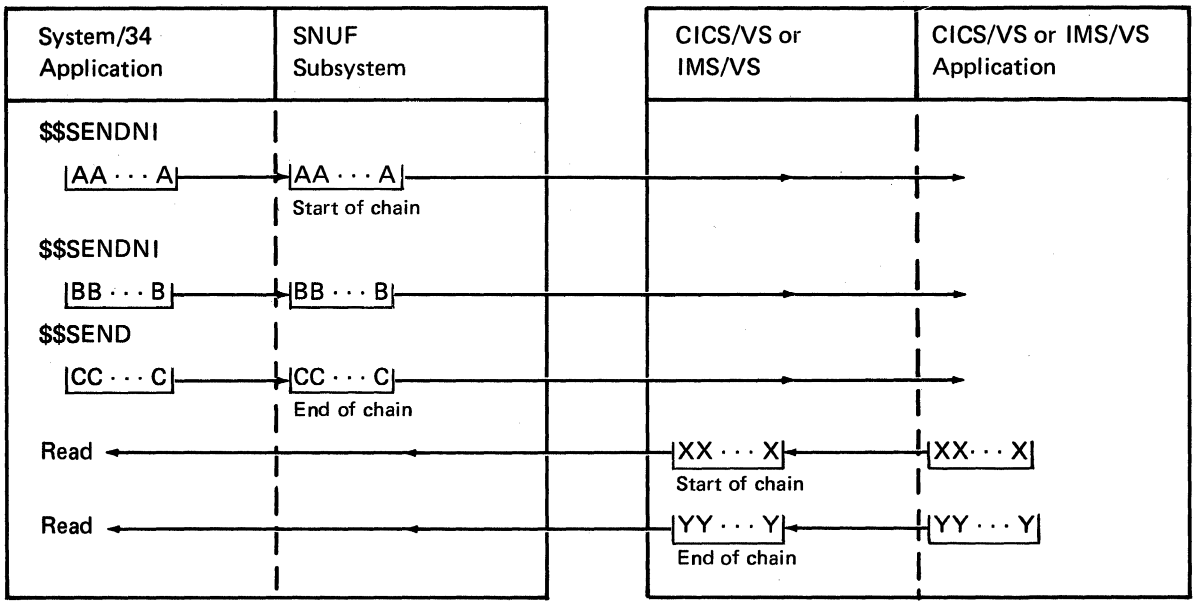
PROGRAMMING CONSIDERATIONS

Chains

Chains are used in either of two ways, depending on the BATCH parameter of the SESSION OCL statement. If BATCH-NO is specified, each application program logical record is a separate chain. For input operations, this means that the SNUF subsystem assembles elements of a chain into one logical record until end of chain is reached or maximum record length is reached. If the maximum record length is reached before the end of chain is reached, an error return code is set. For output operations, each record is sent as a chain. If an application program sends a record greater than 256 bytes, the subsystem automatically divides the record and sends the segments as a chain.



If BATCH-YES is specified, the SNUF subsystem does not attempt to distinguish logical records. For input operations, this means that each record received is passed to the application program, which must determine the logical records. While operating with BATCH-YES, the application program should check for the end of chain return code.



Normally an end of chain is sent when the application program is doing puts and then performs an input operation, or when an application program is doing puts and then ends the transaction. In these cases, the subsystem automatically ends the chain. The application program, operating with BATCH-YES might want to send end of chain without sending change direction or ending the transaction. In this case, the program can issue a put end of chain operation ($$SENDE). Whenever the SNUF subsystem sends an end of chain (if BATCH-YES was specified) without a change of direction, it asks for a definite response. The application program can use the end of chain operation to break data streams into smaller, recoverable units.

Function Management Headers

A function management header is a special record or portion of a record that contains control information for the data that is to follow. The first byte of the record defines the length of the header portion of the record. The length is in hexadecimal and includes the length byte. The header portion immediately follows the length byte.

When a function management header is received, the SNUF subsystem action depends on the FMHI parameter of the SESSION OCL statement. If FMHI-NO was specified, the function management header is removed and any user data is placed in the record area. If FMHI-YES was specified, the function management header is placed in the record area with the characters FMH preceding the header; the return code indicates that a function management header was received. If data accompanied the header, the application program must issue a second input operation to obtain the data.

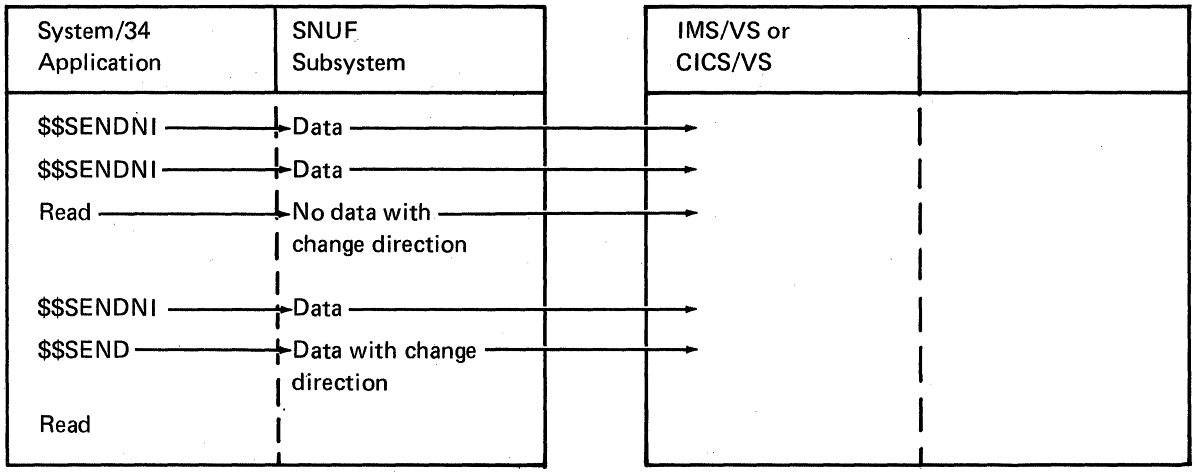
The SNUF subsystem examines any function management header that accompanies a program start request for a hex 0542000001. This function management header is the standard IMS/VS header; it does not contain function management information. The SNUF subsystem treats these headers as if none were received.

If a function management header other than the standard IMS/VS header is received with a procedure start request, the function management header is passed to the application program as its initial input. A second input operation is required to obtain any additional data that accompanied the procedure start request.

Half-Duplex Flip-Flop Protocols

Communication between a SNUF application program and an SNA host application is in half-duplex flip-flop mode; that is, one program sends at a time. When the sender wants to become the receiver, it sends the change direction indicator.

While transmitting, the application program can perform a change direction by issuing an input operation. The subsystem interprets the input operation and sends the change direction indicator. The application program can use the put then invite or put then get operation to make more efficient use of the line.



If the application program is receiving and must send data, the request to change direction operation can be used. This operation causes the SNUF subsystem to request the current sender to transmit a change direction as soon as possible. If the application program is sending and receives a request to change direction return code, the program should perform an input operation as soon as possible.

Session Restart and Recovery

Sessions that are defined as protected sessions can be restarted after line failures. MSGPROT-YES specified (or defaulted to) on the SESSION OCL statement defines a session as protected.

If a recoverable line error occurs, message STS 8217 (RECOVERABLE SDLC ERROR) with option 1 and 2 is displayed on the system console. If option 2 is selected, the subsystem is disabled, and all sessions are abnormally terminated. If option 1 is selected, the line is reenabled, and all recoverable sessions are restarted. Part of the session restart includes exchanging information with the host system regarding the last messages sent and received. The SNUF subsystem uses this information to determine if any data must be retransmitted. For session restart to complete correctly, the host system session and transaction must also be defined as protected. For CICS/VS host systems, the TYPE=OPTGRP parameter should be specified on the DFHPCT macro. For IMS/VS host systems, the INQUIRY parameter should be specified on the TRANSACT macro.

VTAM/NCP CONSIDERATIONS

The System/34 must be defined during VTAM/NCP generation. Each System/34 line is represented as a physical unit in the VTAM generation, therefore, each System/34 line that the SNUF subsystem uses requires a physical unit definition in the generation. The following parameters on the physical unit definition apply to the SNUF subsystem:

PUTYPE = 2
The physical unit type must be 2.

ADDR = xx
The ADDR parameter specifies the SDLC station address. This parameter must be the same as the local station address specified during subsystem configuration (display 4.0).

ISTATUS = ACTIVE/INACTIVE
The ISTATUS parameter specifies whether the physical unit should be activated when its major node is activated.

MAXDATA = 265
The MAXDATA parameter specifies the maximum amount of data, including the transmission header and request/response header that the physical unit can receive. System/34 accepts a maximum of 265 bytes.

MAXOUT = 7
The MAXOUT parameter specifies the number of records that NCP will send to the System/34 before sending a response. For best performance, 7 should be specified.

DISCNT = YES / NO
The DISCNT parameter specifies whether VTAM is to disconnect the physical unit when the last logical unit session is ended. DISCNT=NO allows the System/34 to remain active when no sessions are active; the physical unit is de-activated when the last subsystem on the line is disabled. DISCNT=YES disconnects the System/34 when the last session terminates; the SNUF subsystem remains active until a disable is performed. DISCNT=YES also causes VTAM to ignore the System/34 disable request. If switched lines and multiple locations are configured, specify DISCNT=YES.

IDBLK = 00E
IDNUM = number
The IDBLK and IDNUM parameters make up the SDLC exchange ID. These parameters are specified only for a switched line. The IDBLK must be specified as 00E for a System/34. The IDNUM must be the same as the local ID parameter specified during subsystem configuration (display 8.0).

SSCPFM = USSSCS

Specifying SSCPFM=USSSCS indicates that the System/34 logical units associated with this physical unit use character-coded messages for communication with VTAM. The System/34 requires character-coded messages.

USSTAB = name

The USSTAB parameter specifies the name of a USS definition table. The SNUF subsystem requires that the IBM-supplied definition table be used; therefore, do not specify this parameter.

Each SNUF session corresponds to an SNA logical unit. A logical unit definition in the VTAM generation is required for each session. The following parameters on the logical unit definition apply to the SNUF subsystem:

LOCADDR = address

The LOCADDR parameter specifies the local address of the session. The local address is equivalent to a logical unit number. The SNUF subsystem uses logical units 1 and 2 for incoming procedure start requests. Any number of logical units can be defined; however, the number that can be active concurrently is limited by the maximum number of active sessions specified during System/34 subsystem configuration (display 8.0).

ENCR = NONE

The ENCR parameter specifies the type of encryption to be used. Encryption is not supported by the System/34, so NONE must be specified.

ISTATUS = ACTIVE / INACTIVE

The ISTATUS parameter specifies whether the logical unit is to be activated when the physical unit is activated. The number of logical units activated should be equal to the maximum number of sessions value specified during System/34 subsystem configuration (display 8.0).

PACING = count

The PACING parameter specifies the way pacing is to be handled between NCP and the logical unit. This value should agree with the maximum receive pacing count specified during System/34 subsystem configuration (display 8.0).

VTAM is also responsible for sending the bind command to the SNUF subsystem. Receipt of a correctly formatted bind command is required before a session can be started successfully. The bind command received from the host system must be in the following format:

Byte:	0	1	2	3	4	5
Value: (in hex)	31	01	04	04	B1	B1 or B0

The following is a sample VTAM/NCP definition with the parameters that correspond to SNUF subsystem configuration highlighted.

```
        G1          GROUP      LNCTL=SDLC,
                               TYPE=NCP,
                               PACING=(3,1),
                               ISTATUS=ACTIVE,
                               DSCNT=NO,
                               PASSLIM=7,
                               MAXOUT=7,
                               VPACING=(7,1),
                               CUTYPE=SDLC1,
                               DUPLEX=HALF,
                               SPEED=9600,
                               PAUSE=0.1,
                               CLOCKING=EXT,
                               RETRIES=(5,1,2),
                               POLLED=YES,
                               MAXDATA=265,
                               SSCPFM=USSSCS
        L023        LINE       ADDRESS=028,
                               NRZI=YES,
                               SPEED=7200,
                               PACING=(7,1),
                               ISTATUS=ACTIVE
                  1 SERVICE    ORDER=(PU023A,PU023B,PU023C)          3
        PU023A      PU         ADDR=C1,PACING=(7,1),DISCNT=NO
      { LU023A1     LU         LOCADDR=01,ISTATUS=ACTIVE
      { LU023A2     LU         LOCADDR=02,ISTATUS=ACTIVE
      { LU023A3     LU         LOCADDR=03,ISTATUS=ACTIVE
      { LU023A4     LU         LOCADDR=04,ISTATUS=ACTIVE
    2 { LU023A5     LU         LOCADDR=05,ISTATUS=ACTIVE
      { LU023A6     LU         LOCADDR=06,ISTATUS=ACTIVE
      { LU023A7     LU         LOCADDR=07,ISTATUS=ACTIVE
      { LU023A8     LU         LOCADDR=08,ISTATUS=ACTIVE
        LU023A9     LU         LOCADDR=09,ISTATUS=INACTIVE
        LU023A10    LU         LOCADDR=10,ISTATUS=INACTIVE

        VTAM Application Definition:
        CICSAP      APPL       AUTH=(ACQ,PASS),BUFFACT=25,VPACING=7
            4
```

VTAM Switched Line Definition:

```
SWS34A     VBUILD    TYPE=SWNET,SUBAREA=3,MAXNO=5,MAXGRP=5      5
PU024A     PU        ADDR=C1,PUTYPE=2,IDBLK=00E,IDNUM=00EC1,
                     MAXPATH=2,MAXDATA=265,PACING=(3,1),
                     BATCH=YES,VPACING=0,DISCNT=YES,
                     ISTATUS=ACTIVE,PASSLIM=7,MAXOUT=7,
                     SSCPFM=USSSCS
PATH1      PATH      DIALNO=3306,PID=1,GID=1,GRPNM=G2
LU024A1    LU        LOCADDR=1
LU024A2    LU        LOCADDR=2,ISTATUS=INACTIVE
LU024A3    LU        LOCADDR=3,ISTATUS=INACTIVE
LU024A4    LU        LOCADDR=4,ISTATUS=INACTIVE
```

1 This value corresponds to the local station address specified on display 4.0 of CNFIGICF.

2 The number defines as active correponds to the maximum number of active sessions specified on display 8.0 of CNFIGICF.

3 This value corresponds to the maximum receive pacing count specified on display 8.0 of CNFIGICF.

4 This name is the same as the name specified as subsystem application ID on display 9.0 of CNFIGICF.

5 This parameter is the same as that specified for the local ID on display 8.0 of CNFIGICF.

VTAM Start Option:

```
                  6
SSCPID = 00001
```

6 This value corresponds to the SSCPID value specified on display 9.1 of CNFIGICF.

IMS/VS CONSIDERATIONS

IMS/VS Generation Considerations

Each session must be defined during IMS/VS system generation. The TERMINAL and NAME macros are used for this purpose.

The NAME macro defines the logical terminal (LTERM) name of the session.

The TERMINAL macro defines session parameters to IMS/VS. Each session is defined as an SLUTYPEP terminal; the following parameters apply to SNUF sessions:

- NAME specifies the VTAM node name from VTAM/NCP generation.
- MSGDEL specifies which message types IMS/VS should discard for this logical terminal. The SYSINFO option is required for the program start logical units (local addresses 1 and 2). The NONIOPCB option is recommended for the other sessions. The NONIOPCB option prevents the following types of messages from being sent to System/34 application programs:
  - Message switches
  - Messages inserted by an application program to an alternate PCB
  - Broadcast messages
  - Terminal status messages

  If the application requires any of these types of messages to be sent or received, the NOTERM option can be specified.
- COMPT1, COMPT2, COMPT3, and COMPT4 define up to four separate components for each session. The SNUF subsystem provides no explicit support for multiple components per session, therefore, only one component per session should be defined. Additional options include component protection and blocking. The SNUF subsystem provides no explicit support for distributed presentation or SNA character string processing, so these options should not be used unless the System/34 application can handle them.

- OPTIONS specifies other parameters to be used. The following options apply to SNUF sessions:
  - Response mode specifies the response mode for this session. See *Response Modes* later in this chapter for more details.
  - MFS/NOMFS specifies whether message format services is to be provided for this session. If message format services is specified, the System/34 application must be prepared to handle the data streams.
  - ACK/OPTACK specifies the type of responses required. The SNUF subsystem requires that OPTACK be selected. If the ACK option is selected, the session will be terminated when a recoverable or update transaction is attempted.
  - BID/NOBID specifies whether the VTAM BID command will be used. The BID option is recommended for SNUF sessions.
  - OPNDST/NOPNDST specifies whether sessions can be started with the /OPNDST command. Program start sessions (local addresses 1 and 2) may have to be started from the host with this command, so OPNDST should be selected for those sessions.

- OUTBUF specifies the maximum request unit size for the bind command. The SNUF subsystem rejects any bind with a maximum request unit size of greater than 256 bytes.

The following is a portion of an IMS/VS definition with parameters highlighted to show how they correspond to SESSION OCL statement parameters.

```
                                              1
COMM        RECANY=(4,2000),APPLID=IMS,
            OPTIONS=(TIMESTAMP,2000,FMTMAST)
TYPE        UNITYPE=SLUTYPEP
TERMINAL    NAME=LU021A1,MSGDEL=SYSINFO,OUTBUF=256,
            OPTIONS=(NORESP,PAGDEL,OPTACK,BID)
       NAME  PGMST1                           2
TERMINAL    NAME=LU021A2,MSGDEL=SYSINFO,OUTBUF=256,
            OPTIONS=(NORESP,PAGDEL,OPTACK,BID)
       NAME  PGMST2
TERMINAL    NAME=LU021A3,MSGDEL=NONIOPCB,OUTBUF=256,
            OPTIONS=(NORESP,PAGDEL,OPTACK,NOBID)
       NAME  LTERMA                             3
```

1 This name corresponds to the APPLID parameter on the SESSION statement.

2 This option should be used when a session is defined as HOSTNAME-IMS on the SESSION statement.

3 This option should be used when a session is defined as HOSTNAME-IMSRTR on the SESSION statement.

Terminal Response Mode

Terminal response mode is a method of operation that is defined for transactions and terminals attached to IMS/VS. When an IMS/VS logical terminal (LTERM) is operating in terminal response mode, each transaction evoked must have a reply before the next transaction can be evoked.

The terminal response mode can be defined as one of the following:

- *Negated* – No transactions use terminal response mode.

- *Forced* – Every transaction from the LTERM is in terminal response mode.

- *Transaction* – The terminal response mode is defined separately for each transaction.

Operating in *negated terminal response mode* allows a System/34 application program to evoke several transactions before receiving a reply to any one of them. The application program does, however, require more processing, because it must correlate replies from IMS/VS to the transactions that generated them.

For example, if the application issues evoke then get operations or evoke then invite operations followed by accepts, the return code indicates no data and end of transaction or data (a reply) and end of transaction. In either case, another transaction can be evoked, but the session should not end until all replies have been received. After all transactions have been evoked, input operations can be issued until all the replies are received.

If the program releases the session or terminates before all replies are received, data remains on the IMS/VS output queue. Unrecoverable data left on the IMS/VS output queue may be lost during session termination. If data is put on the queue after the session is released, the data becomes the first input record received when another System/34 application program acquires the session.

Operating in *forced terminal response mode* might require less processing by application programs, because IMS/VS does not allow input on a session until the reply is sent to the previously evoked transaction. An application that evokes a transaction on a session using terminal response mode cannot use the same session until the reply is received. If an error prevents transmission of a reply, the session will wait for data until the session is abnormally terminated. Some conditions that might prevent transmission of a reply message are:

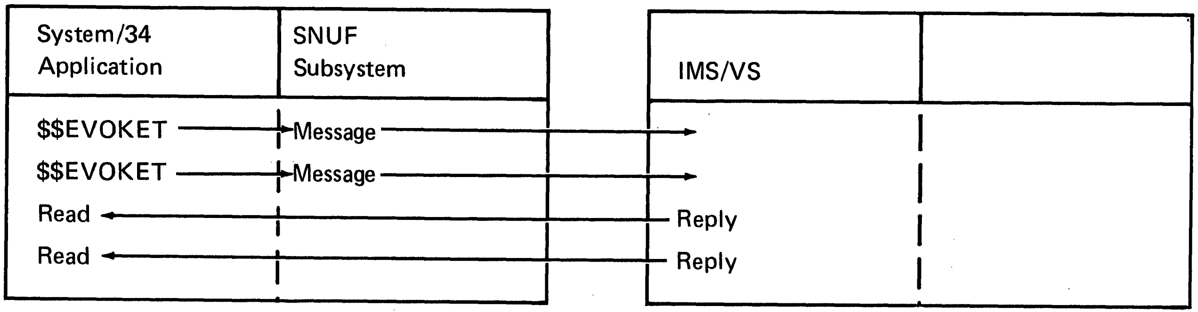
- LTERM stopped

- IMS/VS is unable to schedule the message processing program

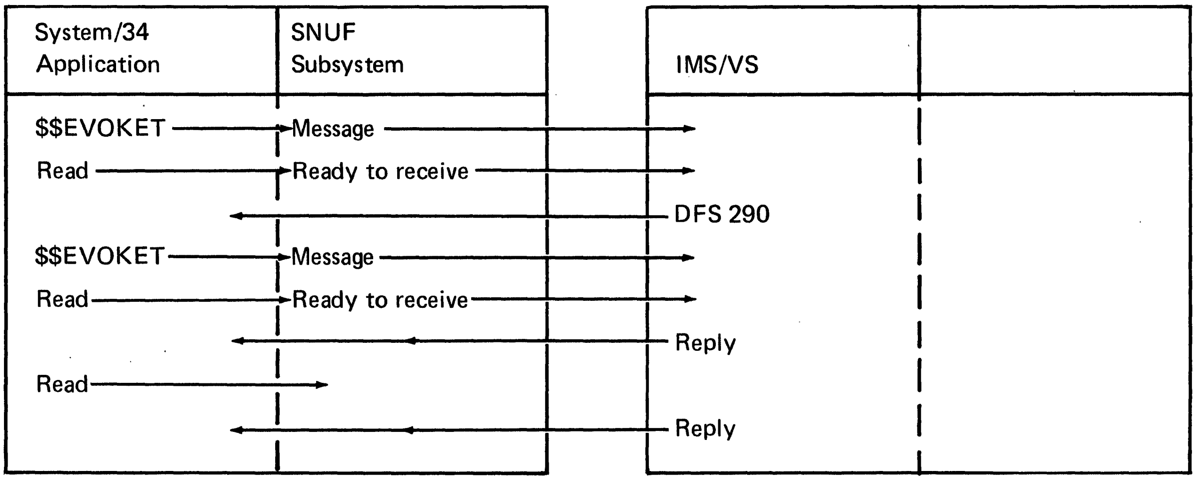
- A message processing program logic error causes no message to be sent

Evoke End of Transaction

The evoke end of transaction operation ($$EVOKET) allows a SNUF application program to send a complete message to IMS/VS without the change direction indicator. This capability allows the SNUF application program to send several messages before receiving a reply from IMS/VS. Evoke end of transaction also allows the program to send a message whose reply is sent to another session (the program start session, for example).



The processing may differ slightly if HOSTNAME-IMSRTR was specified.



IMS/VS Ready to Receive Option

The SNA ready to receive command can be used to request any messages waiting on the IMS/VS output queue for this session. When HOSTNAME-IMSRTR is specified on the SESSION OCL statement, the SNUF subsystem uses the ready to receive command.

The SNUF subsystem sends a ready to receive command whenever an input operation is requested between transactions. If IMS/VS has a message on the output queue for this session, IMS/VS sends the message. If there are no messages on the output queue, IMS/VS sends a system message (DFS290) indicating that no messages are on the queue. The message is passed to the System/34 application program with a return code indicating a system message with end of transaction (0028). If more data is expected, the System/34 application program can continue with other processing and retry the input operation later.

If HOSTNAME-IMS is specified on the SESSION statement, the System/34 application program must wait until output is available before the input operation completes.

IMS/VS Message Format Services

Message format services can be used to give IMS/VS application programs independence of terminal requirements and to improve online performance. The LTERMs must be defined to use message format services (if they want it) during IMS/VS generation. Message format services processing begins when one of the following occurs:

- The System/34 program requests message format services by sending a function management header which contains message identifier (MID name).

- The System/34 program sends a // `midname` before sending a message.

Output messages from IMS/VS that are processed by message format services are sent with a function management header. If the System/34 program wants to process these headers, FMH-YES must be specified on the SESSION statement.

A complete description of message format services is in the *IMS/VS Message Format Users Guide*.

IMS/VS Message Headers

If the System/34 application program specifies that it can process function management headers (FMHI-YES on the SESSION OCL statement), IMS/VS message headers are passed to the application program in its input area. The application program can also send message headers by using the function management operations.

Message headers can be used to pass message descriptors, control information, and component identification. Additional information on message headers can be found in the *IMS/VS Advanced Function for Communications*.

Procedure Start

The SNUF subsystem accepts procedure start requests only on sessions reserved for this purpose. Logical units to be used for sending procedure start requests to the System/34 must use the MSGDEL = SYSINFO option on the TERMINAL macro.

The two logical units reserved for procedure start requests must be started from the host system by using the VARY command, the LOGAPPL parameter in the VTAM definition, or the IMS/VS /OPNDST command.

IMS/VS applications that use the procedure start function must have defined a modifiable alternative program control block in the program specification block. For example, the IMS/VS application program that wants to start a procedure on a System/34 via LTERM=S34A must first do a change call (CHNG) to set the destination of the alternative program control block to S34A; then the program must do an insert call (ISRT) to include the name of the procedure to be started and any security information or parameters required. The application can then do ISRTs to build an output message.

Because the IMS/VS application cannot receive data through the alternative program control block, the program started on the System/34 cannot reply to the message it received from IMS/VS through the same logical unit it was received, but it can acquire another session and evoke a transaction on that session to send a reply.

IMS/VS Commands

The System/34 application program can send IMS/VS commands by using the put operation and placing the command at the beginning of the logical record buffer. Commands can be sent only when the session is between transactions. The System/34 application program can send any command it is authorized to send. Commands that alter the status of the logical unit (such as the /ASSIGN command) should not be sent by a program because unpredictable results can occur.

IMS/VS Security

If IMS/VS requires password security from the System/34 application program, the System/34 program can supply the password in the evoke parameter list. The SNUF subsystem sends the password in the proper position on behalf of the user. For additional security, the password should be supplied to the System/34 application program by an external source, such as a terminal operator.

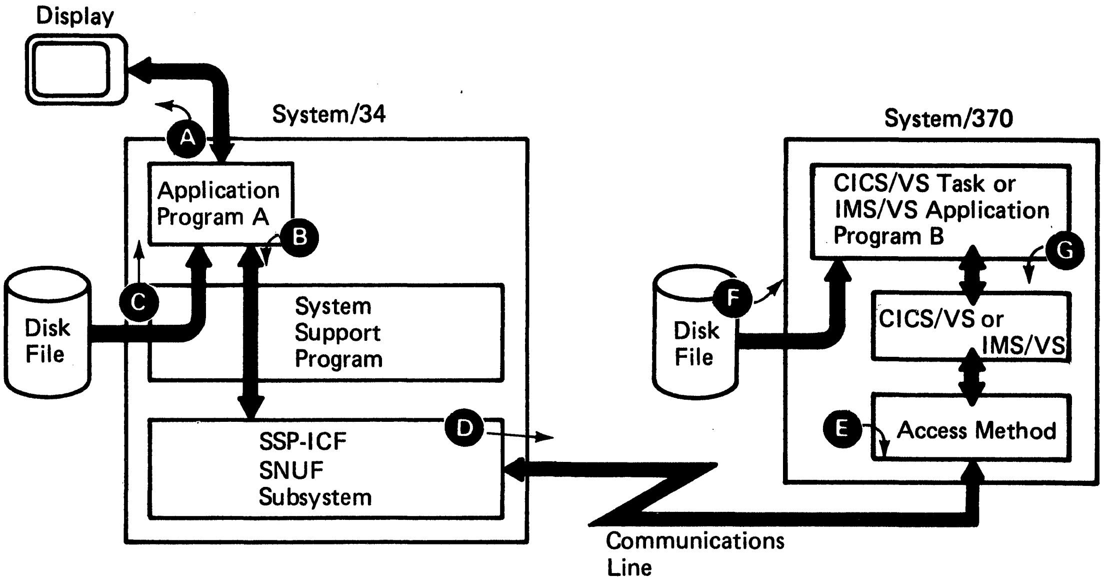
Error Handling

When IMS/VS detects an error on a received message, it returns an exception response with sense data. The System/34 application program is notified that an error has occurred and that sense information is available. The System/34 application program should then do a get or invite to receive the sense data and the status of the session.

The system sense bytes will be either hex 0800 or hex 0826. The user sense fields contain the IMS/VS error message number in displayable form. For example, if the System/34 application program evokes an invalid transaction ID, IMS/VS returns sense bytes of hex 08000040. This is converted and placed in the user buffer as the characters DFS0064.

How to Write Programs that Use the SNUF Subsytem

The following inquiry application is used in the programming examples:



1. Application program A (in the System/34) displays a prompt asking an operator to enter an item number requesting the stock status for the item Ⓐ.

2. When the operator enters the item number, program A reads the number and searches file A (the local file) for the item Ⓒ.

3. If the item is found in the local file, program A displays the stock status on the screen Ⓐ.

4. If the item is *not* in the local file, program A uses the SNUF subsystem to send the item number to the host system Ⓑ and Ⓓ.

5. Program B (in the remote system) uses the item number to search the remote file for the item Ⓕ.

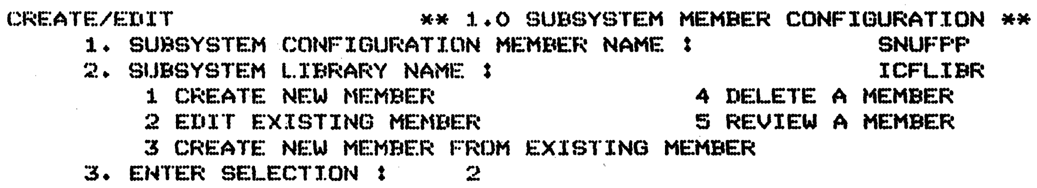
6. If the item is in the remote file, program B sends the stock status to program A Ⓖ and Ⓔ. If the item is not in the remote file, program B sends the characters ***.

7. If program A receives the stock status, it displays it. If it receives the characters ***, it displays the message ITEM NOT FOUND Ⓐ.

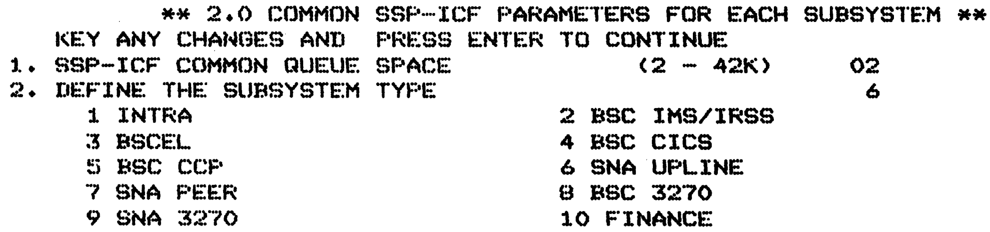
Except for a few minor changes, program A in this example is the program described in Chapter 7 for the Intra subsystem. The changes to the program required for both IMS and CICS are shown in this chapter following the configuration and OCL examples. If you have not read the description of program A in Chapter 7, see *How to Write Programs that Use the Intra Subsystem* in Chapter 7. The application, configuration, and OCL examples in Chapter 7 are for the Intra subsystem only; you do not need to read those. Following are the configuration parameters and OCL statements for this subsystem.

Configuration Parameters

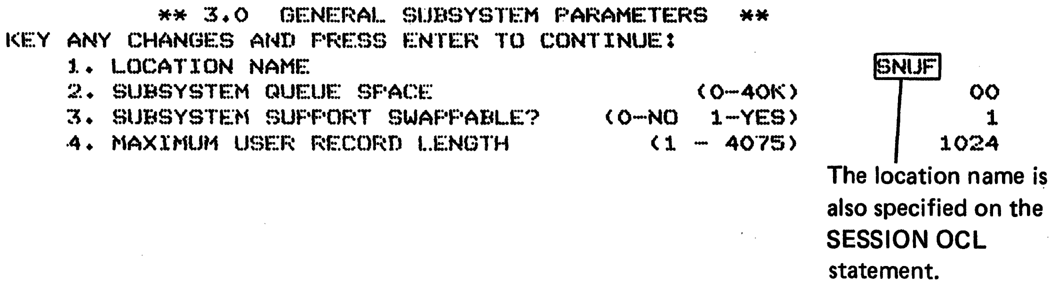
The following configuration parameters are used for this example. For a description of the configuration parameters, see *Setting up the SNUF Subsystem* at the beginning of this chapter.



```
CREATE/EDIT                    ** 1.0 SUBSYSTEM MEMBER CONFIGURATION **
     1. SUBSYSTEM CONFIGURATION MEMBER NAME :                  SNUFPP
     2. SUBSYSTEM LIBRARY NAME :                               ICFLIBR
         1 CREATE NEW MEMBER                   4 DELETE A MEMBER
         2 EDIT EXISTING MEMBER                5 REVIEW A MEMBER
         3 CREATE NEW MEMBER FROM EXISTING MEMBER
     3. ENTER SELECTION :      2
```

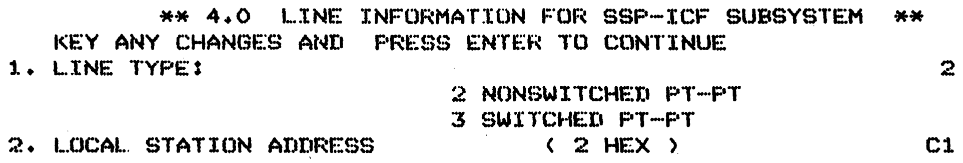


```
          ** 2.0 COMMON SSP-ICF PARAMETERS FOR EACH SUBSYSTEM **
   KEY ANY CHANGES AND  PRESS ENTER TO CONTINUE
1. SSP-ICF COMMON QUEUE SPACE               (2 - 42K)      02
2. DEFINE THE SUBSYSTEM TYPE                                6
     1 INTRA                          2 BSC IMS/IRSS
     3 BSCEL                          4 BSC CICS
     5 BSC CCP                        6 SNA UPLINE
     7 SNA PEER                       8 BSC 3270
     9 SNA 3270                       10 FINANCE
```



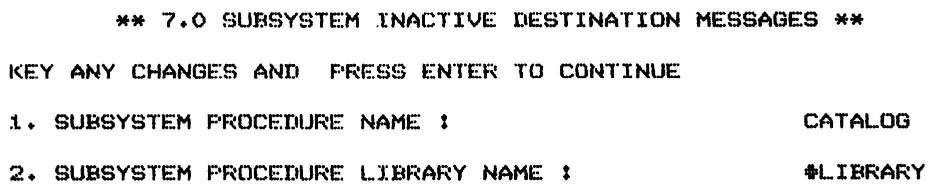
```
          ** 3.0  GENERAL SUBSYSTEM PARAMETERS  **
KEY ANY CHANGES AND PRESS ENTER TO CONTINUE:
     1. LOCATION NAME                                        SNUF
     2. SUBSYSTEM QUEUE SPACE                    (0-40K)           00
     3. SUBSYSTEM SUPPORT SWAPPABLE?      (0-NO  1-YES)             1
     4. MAXIMUM USER RECORD LENGTH            (1 - 4075)         1024
```

The location name is also specified on the SESSION OCL statement.

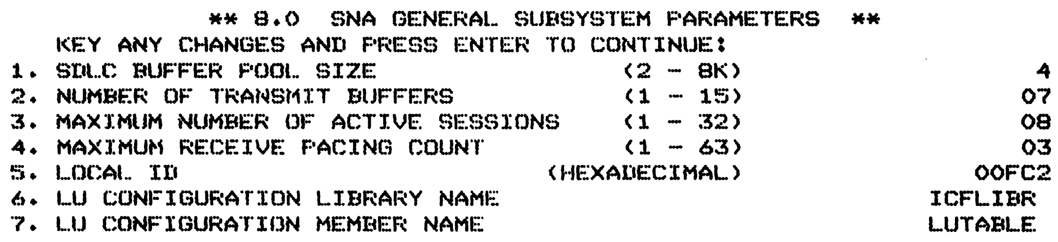


```
          ** 4.0  LINE INFORMATION FOR SSP-ICF SUBSYSTEM  **
   KEY ANY CHANGES AND  PRESS ENTER TO CONTINUE
1. LINE TYPE:                                                  2
                                  2 NONSWITCHED PT-PT
                                  3 SWITCHED PT-PT
2. LOCAL STATION ADDRESS              ( 2 HEX )               C1
```

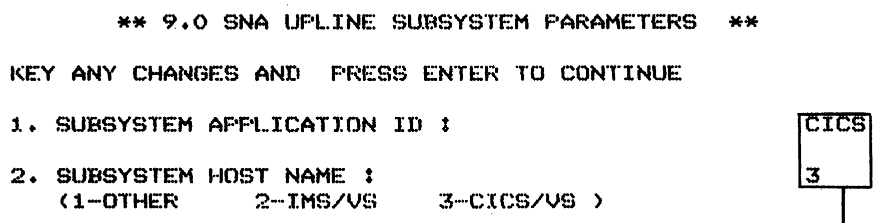
Configuration Parameters (continued)



```
        ** 7.0 SUBSYSTEM INACTIVE DESTINATION MESSAGES **

KEY ANY CHANGES AND  PRESS ENTER TO CONTINUE

1. SUBSYSTEM PROCEDURE NAME :                          CATALOG

2. SUBSYSTEM PROCEDURE LIBRARY NAME :                  #LIBRARY
```



```
              ** 8.0  SNA GENERAL SUBSYSTEM PARAMETERS  **
   KEY ANY CHANGES AND PRESS ENTER TO CONTINUE:
1. SDLC BUFFER POOL SIZE                   (2 - 8K)                4
2. NUMBER OF TRANSMIT BUFFERS              (1 - 15)               07
3. MAXIMUM NUMBER OF ACTIVE SESSIONS       (1 - 32)               08
4. MAXIMUM RECEIVE PACING COUNT            (1 - 63)               03
5. LOCAL ID                            (HEXADECIMAL)           00FC2
6. LU CONFIGURATION LIBRARY NAME                             ICFLIBR
7. LU CONFIGURATION MEMBER NAME                              LUTABLE
```



```
        ** 9.0 SNA UPLINE SUBSYSTEM PARAMETERS  **

KEY ANY CHANGES AND  PRESS ENTER TO CONTINUE

1. SUBSYSTEM APPLICATION ID :                          CICS

2. SUBSYSTEM HOST NAME :                               3
   (1-OTHER      2-IMS/VS     3-CICS/VS )
```

Although CICS is shown in this configuration example, both IMS and CICS programming changes are shown following the OCL examples.

OCL Statements

The following OCL statements are used for the BASIC example:

```
BASICR IMSSNUF,ITEMBAS,30,,SNUFIMS

// SESSION LOCATION-SNUF,SYMID-1S,HOSTNAME-IMS,APPLID-IMS,BATCH-NO,RECL-1000,
// MSGPROT-NO
```

The BASICR procedure includes the SESSION statement.

The location name is also specified on display 3.0 of the CNFIGICF procedure.

The SYMID (session ID) is explained in the description of the program.

```
BASICR CICSSNUF,ITEMBAS,30,,SNUFSESS

// SESSION LOCATION-SNUF,SYMID-1S,BATCH-NO,MSGPROT-YES,RECL-256,APPLID-CICS,
// HOSTNAME-CICS
```

The BASICR procedure includes the SESSION statement.

The host name CICS or IMS) is also specified on display 9.0 of the CNFIGICF procedure.

The following OCL statements are used for the COBOL example:

The SYMID (session ID) is explained in the description of the program.

```
// LOAD COBCIC
// FILE NAME-FILEA
// SESSION LOCATION-SNUF,SYMID-1S,BATCH-NO,MSGPROT-YES,RECL-0256,
// APPLID-CICS,HOSTNAME-CICS
// RUN
```

The location name is also specified on display 3.0 of the CNFIGICF procedure.

The host name (CICS or IMS) is also specified on display 9.0 of the CNFIGICF procedure.

```
// LOAD COBISN
// FILE NAME-FILEA
// SESSION LOCATION-SNUF,SYMID-1S,HOSTNAME-IMS,APPLID-IMS,
// BATCH-NO,MSGPROT-NO,RECL-528
// RUN
```

The SYMID (session ID) is explained in the description of the program.

The following OCL statements are used for the RPG II example:

The SYMID (session ID) is explained in the description of the program.

```
// LOAD RPGCIC
// FILE NAME-FILEA
// SESSION LOCATION-SNUF,SYMID-IS,BATCH-NO,MSGPROT-YES,RECL-0256,
// APPLID-CICS,HOSTNAME-CICS
// RUN
```

The location name is also specified on display 3.0 of the CNFIGICF procedure.

The host name (CICS or IMS) is also specified on display 9.0 of the CNFIGICF procedure.

```
// LOAD RPGISN
// FILE NAME-FILEA
// SESSION LOCATION-SNUF,SYMID-IS,HOSTNAME-IMS,APPLID-IMS,
// BATCH-NO,RECL-1000
// RUN
```

The SYMID (session ID) is explained in the description of the program.

Changes for the Screen Format

```
SFORM1               Y                                G
DSSPICF  00200119Y                  Y          C   S S P - I C F
DITEMINQ 00200219Y                  Y          C    ITEM INQUIRY
DITEMNUM 00180415Y                             CITEM NUMBER.......
DITM#    00230434Y Y    Y      Y   03
DWH1     00180615Y                             CWARE HOUSE 1......
DQTY1    00060634Y             Y
DWH2     00180715Y                             CWARE HOUSE 2......
DQTY2    00060734Y             Y
DWH3     00180815Y                             CWARE HOUSE 3......
DQTY3    00060834Y             Y
DWH4     00180915Y                             CWARE HOUSE 4......
DQTY4    00060934Y             Y
DMSG     00801101Y             Y   01
DICF-MSG 00451215Y             Y   02          CCHECK ICF REFERENCE MANX
DUAL. FOR RETURN CODE OF
DRTCODE  00041261Y             Y   02
DREASON  00301315Y             Y   02
DFL0015  00201346Y             Y   02          CPRESS ENTER TO RETRY
DFL0016  00201502Y                             CCMD 7: END PROGRAM
```

The item number parameters have been changed. For the example in Chapter 7, the item number sent to program B was right-adjusted. However, the item number sent to CICS or IMS must be left-adjusted.

Changes for the Programming Example for CICS

The following examples show the changes required for program A, described in Chapter 7, to permit communications with CICS.

Changes for BASIC

See *Changes for the Screen Format* earlier in this example.

```
00060 DIM ITEM$*23,MES$*80,REASON$*30
00070 OPEN #1: "WS,NAME=FMITEM,RECL=161,LIBR=ICFLIBR"
00080 OPEN #2: "NAME=FILEA,SHR,RECL=50,KEYL=23,KEYP=1,RANDOM",KEYED.INPUT
00090 INDIC$(1:2)="11"
```

The program ICII is evoked at the host system. The password, user ID, and library name are not required in this example.

```
00410 !*---------------------------------------------------------------*
00420 !*                     EVOKE PROCEDURE 'ICII'                    *
00430 !*---------------------------------------------------------------*
00440 EVOK:OPCODE$="EVOK"
00450   WRITE #3,USING 460,FORMAT "$$EVOK": "ICII",ITEM$ IOERR ICFERR
00460   FORM C 8,X 24,C 23
00470 !*---------------------------------------------------------------*
00480 !*      GET INPUT FROM 'ICII'. IF ITEM NUMBER IS FOUND,          *
00490 !*             DISPLAY THE INVENTORY INFORMATION                 *
00500 !*---------------------------------------------------------------*
00510 OPCODE$="GET"
00520 READ #3,USING 530: DATA$ IOERR ICFERR
00530 FORM C 3
00540 IF DATA$="***" THEN NOITEM
00550   REREAD #3,USING 560: ITEM$,QTY1$,QTY2$,QTY3$,QTY4$ IOERR ICFERR
00560   FORM X 32,C 23,X 145,4*C 6
00570   INDIC$(1:2)="11"
00580   GOTO DISPLY
```

Input is from ICII at the host system.

Note: This program does not send an end of transaction ($$SENDET). The host system ends the transaction. The program does receive a return code when the end of transaction is received.

Changes for COBOL

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.

    SELECT TRANSACTION-FILE
       ASSIGN TO WORKSTATION-FMITEM-01,
       ORGANIZATION IS TRANSACTION,
       FILE STATUS IS WS-FS, ICF-FS,
       CONTROL-AREA IS WS-CONTROL-AREA.

    SELECT FILEA-FILE ASSIGN TO DISK-FILEA,
       ORGANIZATION IS INDEXED, ACCESS IS RANDOM,
       RECORD KEY IS FILEA-NUMBER.

    SELECT PRINT-FILE ASSIGN TO PRINTER-PRINTER.
```

See *Changes for the Screen Format* earlier in this example.

```
01  EVOKE-RECORD.
    03  PROCEDURE-NAME           PIC X(8) VALUE 'ICII'.
    03  PASSWORD                 PIC X(8).
    03  USER-ID                  PIC X(8).
    03  LIBRARY-NAME             PIC X(8).
    03  FILLER                   PIC X(20).
    03  DATA-LENGTH              PIC XXXX VALUE '0023'.
    03  ICF-ITEM-NUMBER-OUT      PIC X(23).
```

The program ICII is evoked at the host system.

These parameters are not required in this example.

```
*---------------------------------------------------------------*
*    EVOKE 'ICII' AT HOST                                       *
*---------------------------------------------------------------*
     MOVE ITEM-NUMBER TO ICF-ITEM-NUMBER-OUT.
     WRITE TRANSACTION-RECORD FROM EVOKE-RECORD
        FORMAT IS '$$EVOK',  TERMINAL IS ICF-SESSION.
     MOVE 'EVOK' TO OPCODE.
     PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
     IF IO2 = B'O'
        PERFORM SEND-EOS,
        GO TO ITEM-INQUIRY.
*---------------------------------------------------------------*
*    GET INPUT FROM HOST
*---------------------------------------------------------------*
     MOVE SPACES TO ICF-RECORD-IN.
     READ TRANSACTION-FILE RECORD INTO ICF-RECORD-IN,
       TERMINAL IS ICF-SESSION.
     MOVE 'GET' TO OPCODE.
     PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
     IF IO2 = B'O'
        PERFORM SEND-EOS,
        GO TO ITEM-INQUIRY.
*---------------------------------------------------------------*
*    RELEASE SESSION                                            *
*---------------------------------------------------------------*
     DROP ICF-SESSION FROM TRANSACTION-FILE.
     MOVE 'DROP' TO OPCODE.
     PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
     IF IO2 = B'O'
       PERFORM SEND-EOS.
```

Input is from the host system.

Program A does not send $$SENDET (end of transaction). The host system ends the transaction. The program does receive a return code when the end of transaction is received.

See *Changes for the Screen Format* earlier in this example.

```
F/EJECT
FWSFILE  CD        256              WORKSTN
F                                                KNUM         2
F                                                KFMTS  FMITEM
F                                                KINFDS INFDS
F                                                KID    ID
F                                                KINFSR INFSR
FFILEA   IC  F  50  50R23AI     1 DISK
```

```
O        E        05N07
O                                    K6 '$$EVOK'
O                                     4 'ICII'
O                                    56 '0023'
O                         ITM#       79
```

The program ICII is evoked at the host system. User ID, password, and library name are not required in this example.

```
C*-------------------------------------------------------------*
C*       EVOKE 'ICII' AT HOST                                  *
C*-------------------------------------------------------------*
C                     MOVE '1S'    ID
C                     SETON                     05
C                     EXCPT
C                     SETOF                     05
C*
C*-------------------------------------------------------------*
C*       GET INFORMATION SENT BY 'ICII'.                       *
C*-------------------------------------------------------------*
C           '1S'      NEXT WSFILE
C                     READ WSFILE
C*
C*-------------------------------------------------------------*
C*       RELEASE ICF SESSION '1S' AND GO TO ITMINQ.            *
C*-------------------------------------------------------------*
C           '1S'      REL  WSFILE
C   10                SETOF                     01
C   10                SETON                     03
C                     GOTO ITMINQ
```

Input is from ICII at the host system.

Program A does not send an end of transaction ($$SENDET); therefore, indicator 06 is not used. The host system ends the transaction. Program A does receive an end of transaction return code when the end of transaction is received.

Changes for the Programming Example for IMS

The following examples show the changes required for Program A, described in Chapter 7, to permit communications with IMS.

Changes for BASIC

See *Changes for the Screen Format* eariler in this example.

```
00060 DIM ITEM$*23,ITEMNO$*23,MES$*80,REASON$*30
00070 OPEN #1: "WS,NAME=FMITEM,RECL=161,LIBR=ICFLIBR"
00080 OPEN #2: "NAME=FILEA,SHR,RECL=50,KEYL=23,KEYP=1,RANDOM",KEYED.INPUT
00090 INDIC$(1:2)="11"
```

Input is from the program MSCGT005 at the host system. The password, user ID, and library name are not required in this example.

```
00340 !*-----------------------------------------------------------------*
00350 !*                  ACQUIRE ICF-SESSION (1S)                        *
00360 !*-----------------------------------------------------------------*
00370 ICF:IF INDIC$(4:4)="1" THEN EVOK
00380   OPCODE$="ACQ"
00390   OPEN #3: "WS,ID=1S,RECL=522" IOERR ICFERR
00400   INDIC$(4:4)="1"
00410 !*-----------------------------------------------------------------*
00420 !*                 EVOKE PROCEDURE 'MSCGT005'                       *
00430 !*-----------------------------------------------------------------*
00440 EVOK:OPCODE$="EVOK"
00450   WRITE #3,USING 460,FORMAT "$$EVOK": "MSCGT005",ITEM$ IOERR ICFERR
00460   FORM C 8,X 24,C 23
00470 !*-----------------------------------------------------------------*
00480 !*                  GET INPUT FROM 'MSCGT005'                       *
00490 !*-----------------------------------------------------------------*
00500 OPCODE$="GET"
00510 READ #3,USING 520: DATA$,ITEMNO$,QTY1$,QTY2$,QTY3$,QTY4$ IOERR ICFERR,&
     &EOF 530
00520 FORM C 3,X 29,C 23,X 145,4*C 6
00530 IF RETCODE$<>"0308" THEN 600
00540 READ #3,USING 520: DATA$,ITEMNO$,QTY1$,QTY2$,QTY3$,QTY4$ IOERR ICFERR
00550 !*-----------------------------------------------------------------*
00560 !*    IF ITEM NUMBER IS NOT FOUND, DISPLAY MESSAGE 'ITEM            *
00570 !*    NOT FOUND' TO THE SCREEN. IF THE ITEM IS FOUND,               *
00580 !*    DISPLAY THE INVENTORY INFORMATION.                            *
00590 !*-----------------------------------------------------------------*
00600 INDIC$(1:2)="11"
00610 READ #3: IOERR ICFERR,EOF 620
00620 IF DATA$="***" THEN NOITEM
00630   ITEM$=ITEMNO$
00640   GOTO DISPLY
00650 NOITEM:MES$="ITEM NUMBER "&ITEM$&" NOT FOUND"
00660   INDIC$(1:1)="0":INDIC$(3:3)="1"
00670   GOTO DISPLY
00680 !*-----------------------------------------------------------------*
00690 !*                     CLSFILE ROUTINE                              *
```

This read operation checks for an end of file condition that may have been caused by a previous read operation.

When the host system ends the transaction, return code 0308 is returned to the program. If a 0308 return code is received, another read operation is needed to get the data.

Note: This program does not send an end of transaction ($$SENDET). The host system ended the transaction.

Changes for COBOL

```
SELECT TRANSACTION-FILE
   ASSIGN TO WORKSTATION-FMITEM-01,
   ORGANIZATION IS TRANSACTION,
   FILE STATUS IS WS-FS, ICF-FS,
   CONTROL-AREA IS WS-CONTROL-AREA.
```

See *Changes for the Screen Format* earlier in this example.

```
DATA DIVISION.
FILE SECTION.
FD  TRANSACTION-FILE, LABEL RECORDS ARE OMITTED.
01  TRANSACTION-RECORD              PIC X(528).
```

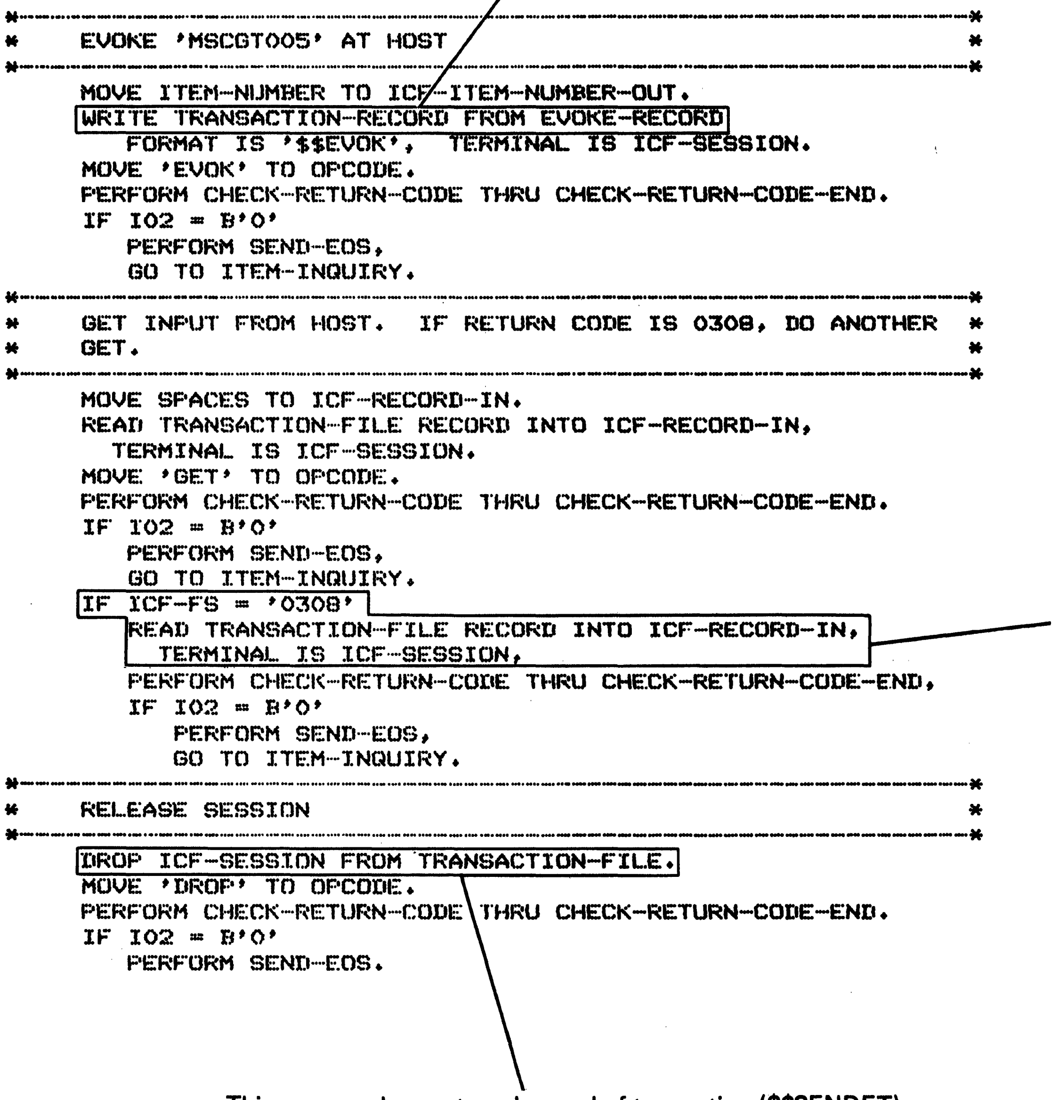
These parameters are not required for this example.

```
01  EVOKE-RECORD.
    03  PROCEDURE-NAME              PIC X(8) VALUE 'MSCGT005'.
    03  PASSWORD                    PIC X(8).
    03  USER-ID                     PIC X(8).
    03  LIBRARY-NAME                PIC X(8).
    03  FILLER                      PIC X(20).
    03  DATA-LENGTH                 PIC XXXX VALUE '0023'.
    03  ICF-ITEM-NUMBER-OUT         PIC X(23).

01  ICF-RECORD-IN.
    03  ICF-RECORD-CHECK.
      05  FIRST-3-CHARACTERS        PIC X(3).
      05  REST-OF-DATA              PIC X(525).
    03  ICF-RECORD-OK REDEFINES ICF-RECORD-CHECK.
      05  FILLER                    PIC X(32).
      05  ICF-ITEM-NUMBER-IN        PIC X(23).
      05  FILLER                    PIC X(145).
      05  ICF-QTY-1                 PIC 9(6).
      05  ICF-QTY-2                 PIC 9(6).
      05  ICF-QTY-3                 PIC 9(6).
      05  ICF-QTY-4                 PIC 9(6).
      05  FILLER                    PIC X(304).
```

The record length was increased because the record from IMS was longer than 256 bytes.

Changes for COBOL (continued)

Program MSCGT005 is evoked at the host system.



```
*------------------------------------------------------------------------*
*    EVOKE 'MSCGT005' AT HOST                                            *
*------------------------------------------------------------------------*
     MOVE ITEM-NUMBER TO ICF-ITEM-NUMBER-OUT.
     WRITE TRANSACTION-RECORD FROM EVOKE-RECORD
        FORMAT IS '$$EVOK',   TERMINAL IS ICF-SESSION.
     MOVE 'EVOK' TO OPCODE.
     PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
     IF IO2 = B'O'
        PERFORM SEND-EOS,
        GO TO ITEM-INQUIRY.
*------------------------------------------------------------------------*
*    GET INPUT FROM HOST.  IF RETURN CODE IS 0308, DO ANOTHER            *
*    GET.                                                                *
*------------------------------------------------------------------------*
     MOVE SPACES TO ICF-RECORD-IN.
     READ TRANSACTION-FILE RECORD INTO ICF-RECORD-IN,
       TERMINAL IS ICF-SESSION.
     MOVE 'GET' TO OPCODE.
     PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
     IF IO2 = B'O'
        PERFORM SEND-EOS,
        GO TO ITEM-INQUIRY.
     IF ICF-FS = '0308'
        READ TRANSACTION-FILE RECORD INTO ICF-RECORD-IN,
          TERMINAL IS ICF-SESSION,
        PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END,
        IF IO2 = B'O'
           PERFORM SEND-EOS,
           GO TO ITEM-INQUIRY.
*------------------------------------------------------------------------*
*    RELEASE SESSION                                                     *
*------------------------------------------------------------------------*
     DROP ICF-SESSION FROM TRANSACTION-FILE.
     MOVE 'DROP' TO OPCODE.
     PERFORM CHECK-RETURN-CODE THRU CHECK-RETURN-CODE-END.
     IF IO2 = B'O'
        PERFORM SEND-EOS.
```

When the host system ends the transaction, the program receives an end of transaction return code (0308); however, data is still at the host system. Another READ is used to get the data.

This program does not send an end of transaction ($$SENDET). The host system ends the transaction.

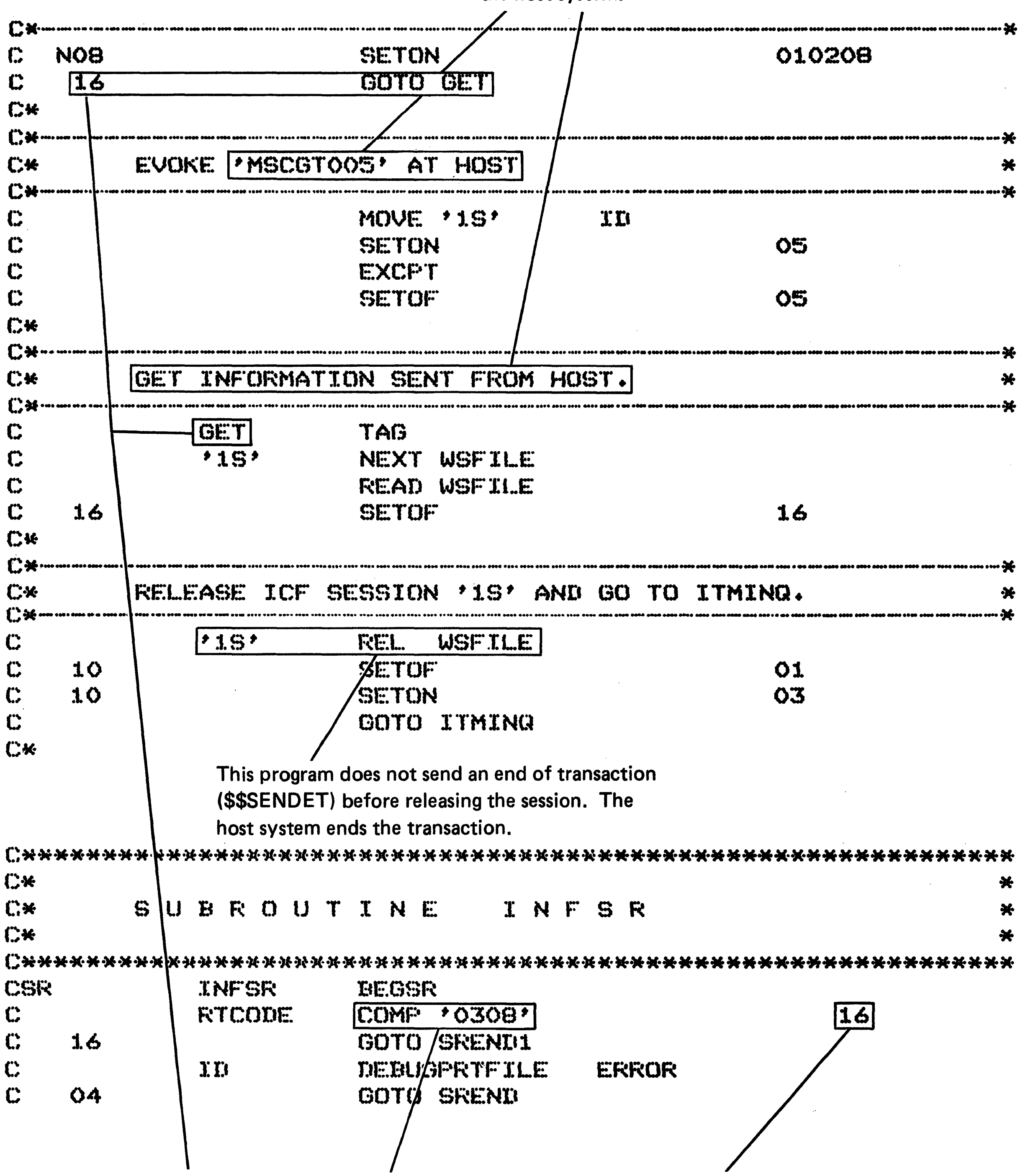
See *Changes for the Screen Format* earlier in this example.

```
FWSFILE  CD        528            WORKSTN
F                                              KNUM        2
F                                              KFMTS  FMITEM
F                                              KID    ID
F                                              KINFDS INFDS
F                                              KINFSR INFSR
FFILEA   IC  F  50  50R23AI     1 DISK
```

The record length was increased because of the longer record from IMS.

```
O*----------------------------------------------------------*
O*   OUTPUT $$EVOK SENDING ITEM NUMBER                      *
O*----------------------------------------------------------*
O        E        05N07
O                                   K6 '$$EVOK'
O                                    8 'MSCGT005'
O                                   56 '0023'
O                         ITM#      79
```

The program MSCGT005 is evoked at the host system. The password, user ID, and library name are not required for this example.



Input is from MSCGT005 at the host system.

```
C*-------------------------------------------------------*
C  NO8               SETON                   010208
C  16                GOTO GET
C*
C*-------------------------------------------------------*
C*     EVOKE 'MSCGT005' AT HOST                          *
C*-------------------------------------------------------*
C                    MOVE '1S'     ID
C                    SETON                   05
C                    EXCPT
C                    SETOF                   05
C*
C*-------------------------------------------------------*
C*     GET INFORMATION SENT FROM HOST.                   *
C*-------------------------------------------------------*
C          GET       TAG
C          '1S'      NEXT WSFILE
C                    READ WSFILE
C  16                SETOF                   16
C*
C*-------------------------------------------------------*
C*     RELEASE ICF SESSION '1S' AND GO TO ITMINQ.        *
C*-------------------------------------------------------*
C          '1S'      REL  WSFILE
C  10                SETOF                   01
C  10                SETON                   03
C                    GOTO ITMINQ
C*
```

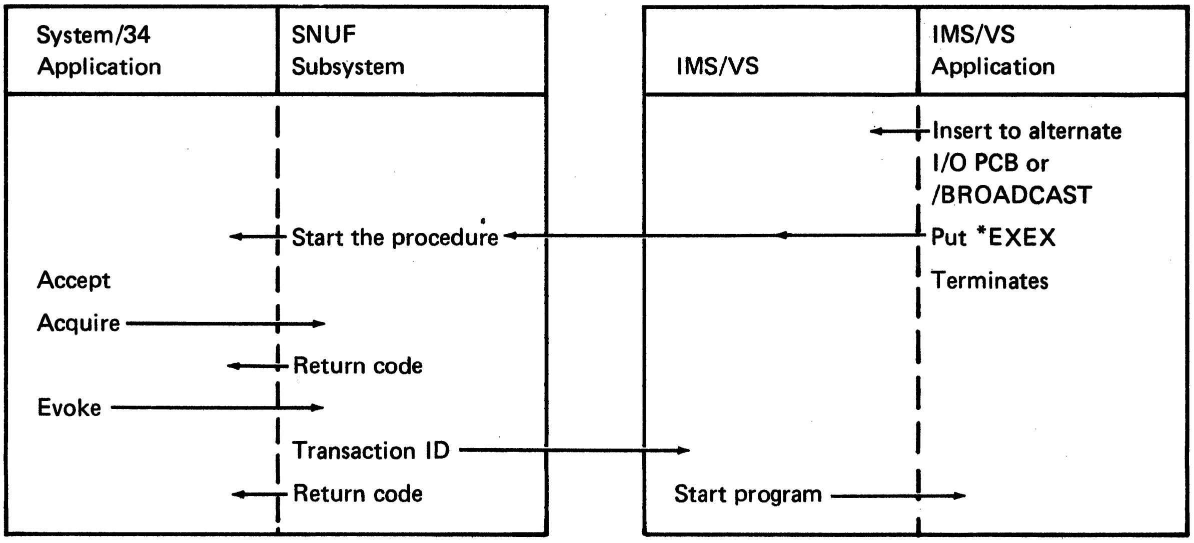
This program does not send an end of transaction ($$SENDET) before releasing the session. The host system ends the transaction.

```
C*********************************************************
C*                                                       *
C*     S U B R O U T I N E     I N F S R                 *
C*                                                       *
C*********************************************************
CSR        INFSR     BEGSR
C          RTCODE    COMP '0308'                 16
C  16                GOTO SREND1
C          ID        DEBUGPRTFILE  ERROR
C  04                GOTO SREND
```

When the host system ends the transaction, the program receives an end of transaction return code (0308). However, data is still at the host system. Another READ operation is used to get the data.

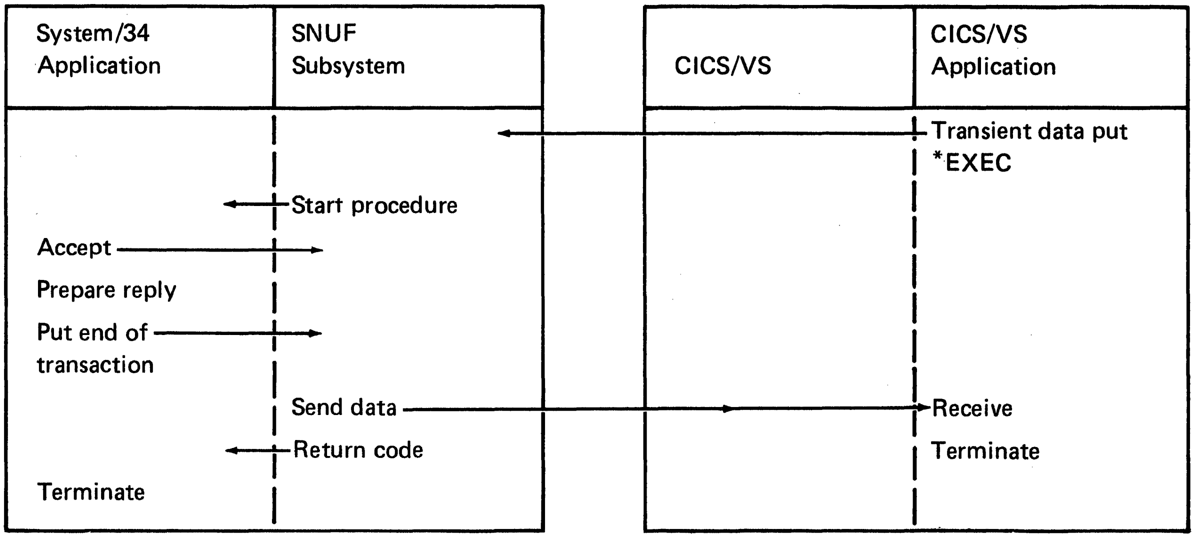
Remote Procedure Start Request Example for IMS

The following sample application flow shows an IMS/VS program starting a System/34 procedure, and the System/34 program attaching to the IMS/VS application program that originally performed the procedure request.



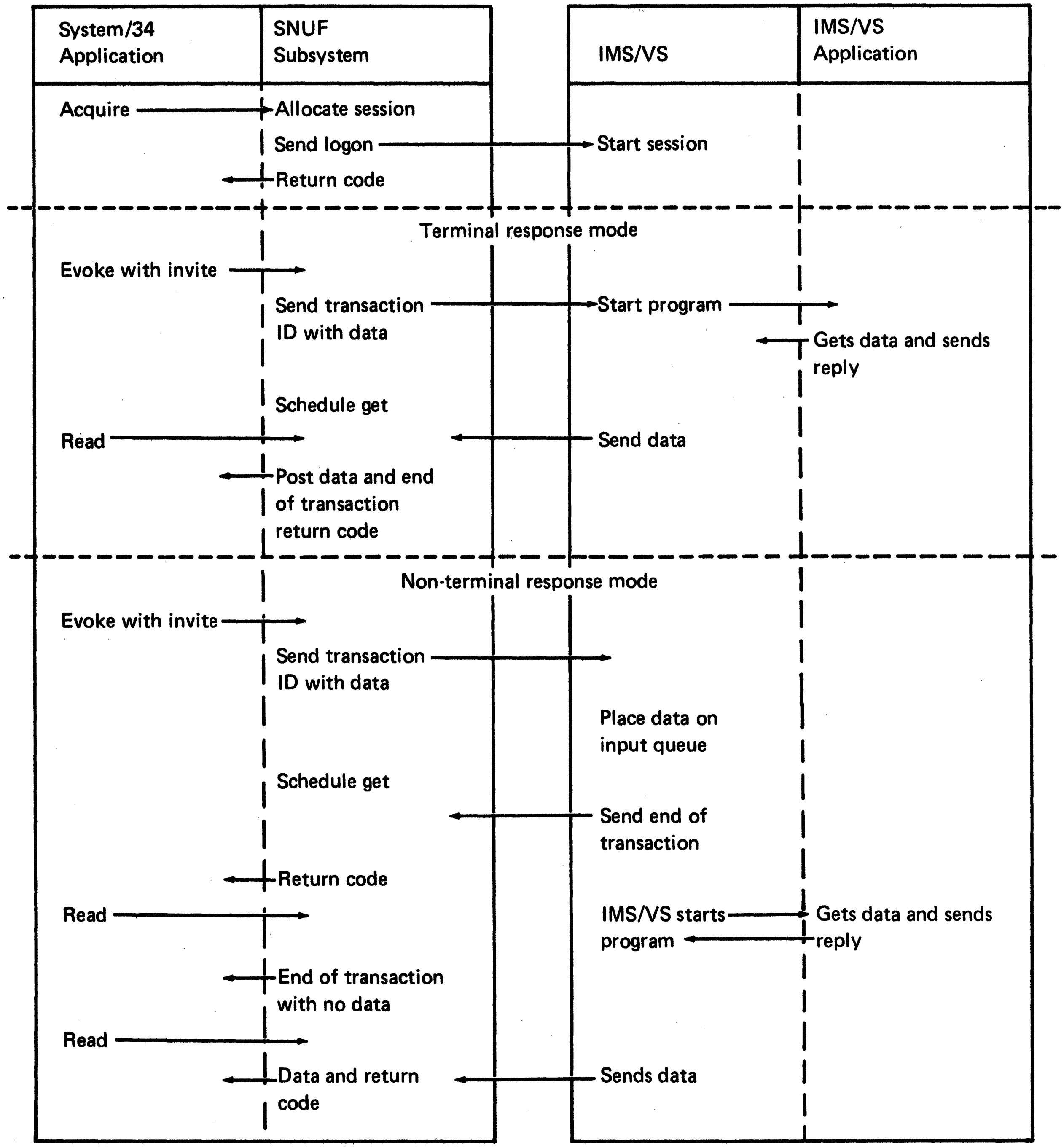
Remote Procedure Start Request Example for CICS

The following sample application flow shows a System/34 application program being started from a CICS/VS program.



Terminal Response and Non-Terminal Response Mode Example for IMS

The following sample application flow shows an inquiry program on the System/34 starting an inquiry program on the IMS/VS system. The example shows two alternative methods; one for terminal reponse mode, the second for non-terminal response mode.



This page is intentionally left blank.

CICS/VS CONSIDERATIONS

CICS/VS System Programming

The CICS/VS programmer defines the SNUF subsystem logical units by coding SESTYPE = USERPROG in addition to TRMTYPE = 3790 in DFHTCT TYPE = TERMINAL macro.

If the application programs are to use function management headers, they must be defined as such in the program control table.

The SNUF subsystem assumes that the CICS/VS system will read from the System/34 program until the change direction indicator is sent. To ensure that this happens, the read ahead queuing option (RAQ operand of the DFHSG PROGRAM=TCP macro) should be used.

The SNUF logical units should be defined in the CICS/VS terminal control table as 3790 full function logical units. The following are examples of the table entries generated by the DFHTCT macro for SNUF logical units:

CICS/VS Stage 1:

```
DFHSG    PROGRAM=TCP,ACCMETH=(SAM,VTAM,BTAM),
         DEVICE=(CRLP,DASD,TAPE),EODI=E0,  1
         VTAMDEV=(3270,3770B,INTLU,3790,LUTYPE2,LUTYPE3,SCSPRT),
         BTAMDEV=(L3277,SYS/3,SYS/3D,3277,3286,3600,3740,3740D),
         FEATURE=(AUTOANSW,AUTOPOLL,TRANSPARENCY),UCTRAN=EBCDIC,
         ANSWRBK=(TERMINAL,AUTOMATIC,EXIDVER),INITRL=YES,
         BSCODE=(EBCDIC,ASCII),WRAPLST=YES,COMPAT=NO,
         LOGREC=YES,RAQ=YES,AUTOTRN=YES,CHNASSY=YES
                          2
```

CICS/VS Terminal Control Table:

```
                                                       3
DFHTCT   TYPE=INITIAL,ACCMETH=(NONVTAM,VTAM),APPLID=CICS,   4
         RAMAX=512,RAMIN=512,SUFFIX=OK

DFHTCT   TYPE=TERMINAL,TRMIDNT=L101,TRMTYPE=3790,CHNASSY=YES,
         SESTYPE=USERPROG,TRMSTAT=TRANSCEIVE,TIOAL=(512,4096),
         BRACKET=YES,ACCMETH=VTAM,NETNAME=LU021A2,BUFFER=256
```

CICS/VS Destination Control Table:

```
  { DFHDCT   TYPE=INTRA,DESTID=L101,TRIGLEV=1,TRANSID=PGST,
5 {          DESTFAC=TERMINAL
  { DFHDCT   TYPE=INTRA,DESTID=L202,TRIGLEV=1,TRANSID=PGST,
  {          DESTFAC=TERMINAL
```

1 SNUF sessions are defined to CICS/VS as VTAM 3790 devices.

2 RAQ=YES permits CICS/VS to accept null request units with the change direction indicator. SNUF protocols assume that RAQ=YES has been specified.

3 The APPLID parameter on the SESSION OCL statement must match the one specified on the INITIAL DFHTCT macro.

4 CHNASSY=YES allows the CICS/VS program to receive records exactly as the System/34 program does with BATCH-NO on the SESSION OCL statement.

5 Two intrapartition destinations are used for sending procedure start requests to the System/34.

Evoke End of Transaction

The evoke end of transaction operation ($$EVOKET) indicates that the System/34 application program does not expect to communicate with the CICS/VS application program that was started. To perform the evoke end of transaction, the SNUF subsystem sends a change direction indicator with the transaction ID and expects an end bracket indicator without data from CICS/VS. If the end bracket is not received or if no data accompanies it, the session is abnormally terminated. The CICS/VS application program can control the sending of the end bracket indicator by using the LAST parameter on the EXEC CICS send command or the DFHTCT TYPE=WRITE macro.

Security

Two special transactions can be evoked through the SNUF subsystem when communicating with CICS/VS. These are the sign-on (CSSN) and sign-off (CSSF) transactions. The CSSN and CSSF transactions are used only to start a secured CICS/VS transaction; that is, a transaction with security defined during CICS/VS generation.

The rules for the use of the CSSN and CSSF evoke transactions are as follows:

- The application program data buffers associated with the CSSN evoke must contain the nonblank password (PS-password) and the name (NAME-name) separated by a comma. The password is 1 to 4 characters, and the name is 1 to 20 characters.

- The CSSN evoke always results in a response from CICS/VS. The SNUF subsystem examines the response and issues the appropriate return code.

- After a successful CSSF evoke, the application program can again issue a CSSN evoke to the same session or to another session. A different password and name can be used each time.

SNUF Subsystem Return Codes

This part of Chapter 15 describes all the return codes that are valid for the SNUF 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 four-digit value; the first two digits contain the major code, and the last two 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).

Note: In the return code descriptions, *your program* refers to the local System/34 application program that initiates the operation and receives the return code from the subsystem. The *remote program* refers to the application program in the remote (or host) system with which the System/34 application program is communicating through SSP-ICF.

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 SNUF subsystem. Although all the operations shown are valid for SNUF, their validity also depends on the logical sequence of communications events occurring between the System/34 and the remote system.

Input Operations to Your Program	Output Operations from Your Program	Combined Operations in Your Program
Accept input		
	Acquire[1]	
	Cancel	Cancel then get[2] Cancel then invite
	End of session	
	Evoke[3] Evoke end of transaction[3]	Evoke then get[2,3] Evoke then invite[3]
Get Get attributes[4]		
Invite		
	Negative response	Negative response then get[2] Negative response then invite
Pass-through invite	Pass-through put[2]	Pass-through put then invite
	Put Put end of chain Put end of transaction	Put then get[2] Put then invite
	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]	

[1] Normally, 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.

[2] Valid only in assembler language.

[3] Evoke operations in assembler can have OPM-FMH specified with the $WSIO macro.

[4] Valid only in assembler and COBOL languages.

[5] For BASIC and RPG II programs, the set 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.

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, or it received a message from the remote system.

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 remote 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 This Output Operation Was:	Then (in This Session):
Initiated the transaction (evoked the remote program)	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 a remote procedure start request)	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.

[1] An evoked program (started by a procedure start request) 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.

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 code of xx00 (a change direction indication, which allows your program to send data), or xx08 (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 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 remote 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 code of xx00 (a change direction indication) or xx08 (an end of transaction indication).

Normal Action: Process the function management header and issue another input operation.

0007 **Normal Indication:** A function management header and an end of chain indication were received with some data on a successful input operation; the last record in the chain has been received.

Normal Action: Process the function management header and issue another input operation to receive the next chain.

0008 **Normal Indication:** An end of transaction indication was received with the last of the data on a successful input operation. Communications have ended with the remote *program*, but the session with the remote *system* is still active.

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. Communications have ended with the remote *program*, but the session with the remote *system* is still active.

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 remote program on a successful *output* operation for your program; the remote program wants to send data as soon as possible. You should allow the remote program to send its data.

Normal Action: Issue an input operation as soon as possible.

0020 **Normal Indication:** A message from the remote system and a change direction indication were received on a successful input operation. The message, now in your program's input buffer, describes why the previous operation was rejected. The remote system now wants to receive some data.

Normal Action: Handle the message in the input buffer (possibly display it). Your program now has control of the session; issue an output operation or another input operation.

0021 **Normal Indication:** A message was received from the remote system on a successful input operation. (The message is now in your program's input buffer.) Your program must continue to receive input.

Normal Action: Handle the message in the input buffer (possibly display it), 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.

0028 **Normal Indication:** An end of transaction indication was received with a remote 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. Communications have ended with the remote *program*, but the session with the remote *system* is still active.

Normal Action: Handle the message in your program's 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.

0030 **Normal Indication:** A truncated message from the remote system and a change direction indication were received on a successful input operation. The message, truncated because it was too long for your program's input buffer, describes why the previous operation was rejected. The remote system now wants to receive some data.

Normal Action: Handle the truncated message (possibly display it) in the input buffer, and issue an output operation.

0038 **Normal Indication:** An end of transaction indication was received with a *truncated* remote 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. Communications have ended with the remote *program*, but the session with the remote *system* is still active.

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, either issue an end of session operation or terminate.

Major Code 01 – Successful operation with a new requester.

The new requester is a program on a remote system that initiated a session with your program by sending a procedure start request to the local system. 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 procedure start request, initiated by the remote program, was sent by the remote system in the form of an *EXEC or *EXEX procedure start statement. The request may have included, for your program, data from the remote *program* or a system message from the remote *system.*

Normal Description: Each of the following return codes indicates either that the *input* operation issued by your program and responded to by a new requester completed successfully, or that the *output* operation issued by your program in response to a new requester completed successfully.

If the operation was an *input* operation, your program received some data, no data, or a system message from the requester. If any data was received, it was included in the incoming procedure start request statement. If a system message was sent by the remote system, it is now in the subsystem input buffer, waiting to be passed to your program when the next input operation is issued.

If your program is an SRT program that was evoked by an incoming procedure start request 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 procedure start request statement 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 or message 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, either data or a message 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 some data may have been received with the request. The remote 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[1] 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 code of xx00 (a change direction indication) or xx08 (an end of transaction indication).

Normal Action: Handle any data passed with the request, perform any necessary record keeping[1] 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.

0103 **Normal Indication:** On a successful input operation from a new requester, a procedure start request and an end of chain indication were received with some data; a complete chain has been received by your program.

Note: This return code is returned only in a multiple-program procedure, and only to each one of the *succeeding* programs in that procedure (not to the first program). Also, it is returned only for the first operation in each of those programs.

Normal Action: Handle the data passed with the request, perform any necessary record keeping[1] for the new requester, and issue another input operation to receive the next chain.

[1] For some situations, no record keeping for the session is necessary. In other situations, you should record the session ID of the new requester. You may also want to keep a table containing the IDs of all active requesters, or to maintain a history log of all requests.

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 remote 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[1] for the new requester, and issue an output operation to the remote 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 code of xx00 (a change direction indication) or xx08 (an end of transaction indication).

Normal Action: Process the function management header, handle the data passed with the request, perform any necessary record keeping[1] for the new requester, and issue an input operation.

0107 **Normal Indication:** On a successful input operation from a new requester, a procedure start request, a function management header, and an end of chain indication were received with some data; the last record in the chain has been received.

Normal Action: Process the function management header, handle the data passed with the request, perform any necessary record keeping[1] for the new requester, and issue another input operation to receive the next chain.

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 remote program. Its communications have ended with your program; however, the session is still active between the local and remote systems.)

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 incoming 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.

[1] For some situations, no record keeping for the session is necessary. In other situations, you should record the session ID of the new requester. You may also want to keep a table containing the IDs of all active requesters, or to maintain a history log of all requests.

Note: Return code 0108 is returned only to each one of the *succeeding* programs in a multiple-program procedure (and only for the first operation in each program).

Normal Action: Perform any necessary record keeping[1] 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 remote program. Its communications have ended with your program; however, the session is still active between the local and remote systems.)

Normal Action: Process the function management header, handle the data passed with the request, and perform any necessary record keeping[1] for the new requester. Then, either issue an end of session operation or terminate your program.

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, or it received a message from the remote system. 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 remote program now wants to receive some data; your program must send it.

Normal Action: Issue an output operation.

[1] For some situations, no record keeping for the session is necessary. In other situations, you should record the session ID of the new requester. You may also want to keep a table containing the IDs of all active requesters, or to maintain a history log of all requests.

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 code of xx00 (a change direction indication) or xx08 (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 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 remote 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 code of xx00 (a change direction indication) or xx08 (an end of transaction indication).

Normal Action: Process the function management header and issue another input operation.

0207 **Normal Indication:** A function management header and an end of chain indication were 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: Process the function management header and issue another input operation to receive the next chain.

0208 **Normal Indication:** An end of transaction indication was received with the last of the data on a successful input operation. Although communications have ended with the remote program, the session with the remote system is still active. 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 (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. Although communications have ended with the remote program, the session with the remote system is still active. 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 (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.

0220 **Normal Indication:** A message from the remote system and a change direction indication were received on a successful input operation. The message, now in your program's input buffer, describes why the previous operation was rejected. The remote system 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: Handle the message in your program's input buffer (possibly display it), and issue an output operation or another input operation.

0221 **Normal Indication:** A message was received from the remote system on a successful input operation. (The message is now in your program's input buffer.) 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.

Normal Action: Handle the message in your program's input buffer (possibly display it), 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.

0228 **Normal Indication:** An end of transaction indication was received with a remote 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. Although communications have ended with the remote program, the session with the remote system is still active. 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 your program's input buffer (display it, for example). If 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 your program was evoked, either issue an end of session operation or terminate.

0230 **Normal Indication:** A truncated message from the remote system and a change direction indication were received on a successful input operation. The remote system now wants to receive some data. (The message, truncated because it was too long for your program's input buffer, describes why the previous operation was rejected.) 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 (possibly display it) in your program's input buffer, and issue an output operation.

0238 **Normal Indication:** An end of transaction indication was received with a *truncated* remote 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. Although communications have ended with the remote program, the session with the remote system is still active. 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 (to start another program), it can issue a release operation (to perform local processing), or it can terminate. If your program was evoked, either issue an end of session operation or terminate.

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 code of xx00 (a change direction indication, which allows your program to send data), or xx08 (an end of transaction indication).

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. Although communications have ended with the remote program, the session with the remote system is still active.

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.

Major Codes 04-34 – Miscellaneous program errors.

Error Descripton: The operation just attempted by your program failed, or an output exception occurred.

- An operation may have failed because it was issued at the wrong time or because a data record was too long.
- An output exception may have occurred because your program attempted to send output when it should be receiving the output that has already been sent by the remote program.

Recovery Action: Refer to the individual return code descriptions for the appropriate recovery actions.

Code **Indication/Action**

0412 **Normal (Exception) Indication:** 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 remote program. Your program's output was not sent and should be sent later, after the remote program's data (still waiting in the subsystem input buffer) has been received.

Note: If your program issues another output operation, an error return code of 831C will be received.

Normal Action: Issue an input operation to receive the data waiting in the subsystem input buffer.

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 a new requester) 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 requesting program have been completed.

3401 **Error Indication:** This input operation was rejected because the record length of the data sent by the remote 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 Descripton: 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/34 Messages Guide* 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 (in COBOL and assembler only) a set timer operation, or by using the TIMER intrinsic function (in BASIC only); (2) continue local processing; or (3) terminate.
- If the session should be started again after the subsystem is enabled, it must be reacquired by your program or restarted by the remote 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 either in this subsystem or in the interrupt handler.

Recovery Action: This subsystem has been disabled; it must be enabled again before communications can resume. Your program can continue local processing, wait[1] to reissue the acquire operation, or terminate.

If more than one subsystem was active when the error occurred, all subsystems that were active when the error occurred should be disabled. (Note that all other active SNUF subsystems are automatically disabled when the error occurs; all other types of active subsystems must be manually disabled.)

If all subsystems (of all types) on the system are *not* disabled to recover from the processor check, the common queue space used by the failing subsystem *cannot* be freed. And if it is not freed, that space is wasted, and an indication of insufficient common queue space being available can occur. The indication can occur as a message when the failing subsystem is reenabled or when a different subsystem is enabled. The indication can also occur as a return code to your program for any subsystem that is starting a *new* session (code 8215) or performing an output operation in any *existing* session (code 8315).

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 remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, it can wait[1] until the subsystem has been reenabled and reissue the acquire operation, or it can terminate.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

Major Code 81 – Permanent (nonrecoverable) session error.

Error Descripton: A nonrecoverable error has occurred in the session; the session cannot be continued and 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/34 Messages Guide* 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 session must be acquired again before communications can resume.

General Recovery Actions: The following general actions can be taken for each 81xx return code. Other specific actions are given in each return code description.

- Several return codes indicate that an error condition must be corrected by changing a value in the subsystem configuration record or in the SESSION statement for your program.
  - To change a parameter value in the subsystem configuration being used by your program, disable the subsystem before making the change in the subsystem's configuration record, and enable the subsystem again to make the change effective.
  - To change a parameter value in the SESSION statement associated with your program, terminate your program only.

  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.

- If the session should be started again, it must be reacquired by your program or restarted by the remote program before communications can resume.

- An end of session (EOS or $$EOS) operation should be issued for the session that has terminated. Your program can also continue local processing, or it can terminate.

Note: If the session is started again, it starts from the beginning, not at the point where the session error occurred.

Code Indication/Action

8191 **Error Indication:** A permanent line error occurred on an input or output operation, and the system operator has taken a recovery option in response to the line error message. (You can find out what type of line error occurred by asking the system operator.) The session has been terminated.

Recovery Action: If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

8195 **Error Indication:** A subsystem disk I/O error occurred while the communications work area on the System/34 disk was being used. The session has been terminated.

Recovery Action: If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

8196 **Error Indication:** An SNA unbind command was received from the remote system. The session has been terminated.

Recovery Action: If the session should be started again, reissue the acquire operation. Otherwise, your program can continue local processing or terminate.

819C **Error Indication:** On an *input* operation, the length of the data block sent by the remote system exceeded the length of the subsystem line buffer. The session has been terminated.

Recovery Action: Check that the value for the maximum user record length (RECL) parameter is correct in the subsystem configuration record or in the SESSION statement. Then, if your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

819D **Error Indication:** On an input or output operation, unexpected data was received from the remote system either after an end of transaction indication was received or before an evoke operation was issued by your program. The session has been terminated.

Recovery Action: Check that your program did not issue an end of transaction operation before the transaction was completed. Also check to see if the remote system sent a procedure start request while your session was still active. If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

81B5 **Error Indication:** On an *input* operation, an indication was received that the maximum pacing count was exceeded; the host system sent more records than the subsystem was configured to handle. The session has been terminated.

Recovery Action: The subsystem must be reconfigured before your program can communicate with the host system. The maximum receive pacing count parameter must be respecified with the value that was determined by the host system. After the subsystem has been reconfigured and enabled, the session can be started again. If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

81B6 **Error Indication:** On a previous operation, an indication was received that the host system has quiesced the session (stopped all requests to the host) by issuing a QUIESCE AT END OF CHAIN command. The session has been terminated.

Recovery Action: If the session should be started again, reissue the acquire operation. Otherwise, your program can continue local processing or terminate. If your program was evoked and it is an MRT program, it can also wait[1] to be evoked again by a new requester.

[1] For BASIC and RPG II, the set timer operation is not valid at this time because the session is not active. (It *is* valid in COBOL and assembler, and the TIMER intrinsic function can be used in BASIC.)

Major Code 82 – Acquire operation failed.

Error Descripton: 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/34 Messages Guide* 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, first disable the subsystem, make the change in the subsystem's configuration record, and 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, (and depending on what the return code description says):
   a. Notify the remote location that a change is required on that end to correct the error received.
   b. Simply reissue the acquire operation. It could be successful if the error occurred because there was not enough common queue space available to support a new session, because an isolated line error occurred, or because the remote system was not active at the time.
   c. If the acquire operation is again unsuccessful, retry it only a limited number of times. (The limit for retries should be specified in your program.)
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 operation is valid only in an *active* session and cannot, therefore, be issued after an 82xx return code is received. (This restriction does not apply to COBOL and assembler programs, or to the TIMER intrinsic function in BASIC, which also can be used to wait for a specified time interval.)

Code Indication/Action

8213 **Error Indication:** On an unsuccessful acquire operation, a queue space error condition was detected. The session could not be started because no *subsystem* queue space was available at the time.

Recovery Action: Your program can wait[3] for a period of time, then reissue the acquire operation. If an unacceptable number of queue space errors occur, you can disable the subsystem and change the subsystem configuration by specifying a larger subsystem queue space size in the subsystem queue space parameter. After the subsystem is enabled, reissue the acquire operation to start the session.

8215 **Error Indication:** On an unsuccessful acquire operation, a queue space error condition was detected. The session cannot be started because the size of the *common* queue space, specified during subsystem configuration, is too small.

Recovery Action: Your program can wait[1] for a period of time, then reissue the acquire operation. If an unacceptable number of queue space errors occur, you can disable all the subsystems that are active in the system, and change the subsystem configuration by specifying a larger common queue space size in the SSP-ICF common queue space parameter. After the subsystem is enabled, reissue the acquire operation to start the session.

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, respecify 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.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

8281 **Error Indication:** On an unsuccessful acquire operation, an SSP-ICF error condition was detected. The error caused a processor check either in this subsystem or in the interrupt handler.

Recovery Action: This subsystem has been disabled; it must be enabled again before communications can resume. Your program can continue local processing, wait[1] to reissue the acquire operation, or terminate.

If more than one subsystem was active when the error occurred, all subsystems that were active when the error occurred should be disabled. (Note that all other active SNUF subsystems are automatically disabled when the error occurs; all other types of active subsystems must be manually disabled.)

If all subsystems (of all types) on the system are *not* disabled to recover from the processor check, the common queue space used by the failing subsystem *cannot* be freed. And if it is not freed, that space is wasted, and an indication of insufficient common queue space being available can occur. The indication can occur as a message when the failing subsystem is reenabled or when a different subsystem is enabled. The indication can also occur as a return code to your program for any subsystem that is starting a *new* session (code 8215) or performing an output operation in any *existing* session (code 8315).

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 remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, it can wait[1] until the subsystem has been reenabled and reissue the acquire operation, or it can terminate.

8296 **Error Indication:** On an unsuccessful acquire operation, an SNA unbind command was received from the remote system. The session was not started.

Recovery Action: Your program can reissue the acquire operation, continue local processing, or terminate.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

82A1 **Error Indication:** On an unsuccessful acquire operation, the logon portion of the acquire operation failed. The host subsystem might not be active, or the name of the remote application program specified in the APPLID parameter of the SESSION statement might be incorrect. The session was not started.

Recovery Action: Verify that the name of the remote application program is correctly specified in the SESSION statement. If the program name was specified correctly, call the remote location and request that the host system be made active. Then reissue the acquire operation. Otherwise, your program can continue local processing, wait[1] to reissue the acquire operation, or terminate.

82A5 **Error Indication:** On an unsuccessful acquire operation, an invalid combination of parameter values was detected. The SESSION statement was specified incorrectly. Both the MSGPROT and BATCH parameters were specified with a value of *yes*. The session was not started.

Recovery Action: Change the value of either the MSGPROT or BATCH parameters in the SESSION statement to *no*, then reissue the acquire operation.

82A6 **Error Indication:** On an unsuccessful acquire operation, the subsystem received an SNA bind command from the remote system that was not in the correct format. The session was not started.

Recovery Action: Determine the error in the format of the received bind command (given in the topic *VTAM/NCP Considerations* in this chapter), and contact the VTAM programmer at the remote location to have the format corrected. Then reissue the acquire operation to start the session.

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 100 sessions can be active in the System/34 at one time. The session was not started.

Recovery Action: Your program can wait[1] for another session to end and then reissue the acquire operation. Otherwise, your program can continue local processing or terminate.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

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 to be enabled is identified by the location parameter in the SESSION statement. That location name must also be specified in the subsystem configuration record (shown on display 3.0 of the subsystem configuration planning charts). The session was not started.

Recovery Action: Verify that the subsystem name was specified correctly on the LOCATION parameter of the SESSION statement. If the correct name was specified, contact the System/34 system operator and request that the specified subsystem be enabled by executing the ENABLE procedure command at the system console. Then reissue the acquire operation. Otherwise, your program can continue local processing, wait[1] 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[1] until the subsystem has been enabled, and 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[1] until the subsystem is enabled again, and then reissue the acquire operation. Otherwise, your program can continue local processing, or it can terminate.

82B3 **Error Indication:** The acquire operation was not successful because all of the sessions specified in the subsystem configuration are already in use. The session was not started.

Recovery Action: Wait[1] for one of the sessions in the subsystem to become available, then reissue the acquire operation. Otherwise, continue local processing or terminate.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

82B4 **Error Indication:** The acquire operation was not successful because all of the resources needed for the session could not be allocated from the assign/free area of the system. All available resources are already being used in the system. The session was not started.

Recovery Action: Wait[1] for the needed resources to become available, then reissue the acquire operation. Otherwise, continue local processing or terminate.

Major Code 83 – Session error occurred.

Error Descripton: 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/34 Messages Guide* 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, first disable the subsystem, make the change in the subsystem's configuration record, and 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.

   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.

   Note: If the session is started again, it starts from the beginning, not at the point where the session error occurred.
3. If no change is needed in your program or in the subsystem, (and depending on what the return code description says):
   a. Retry the operation, if possible. It could be successful if the error occurred because there was not enough common queue space available at the time, or because the subsystem was not enabled at the time.
   b. If another operation is not successful, retry it only a limited number of times. (The limit for retries should be specified in your program.)
   c. Issue a set timer operation in your program so it can wait for a specified time interval before reissuing the operation.

[1] For Basic and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

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 an 81xx (session was terminated) or 82xx (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 an 81xx or 82xx return code is handled properly. If you want your program to recover from an improperly handled error condition, issue another acquire operation.

8315 **Error Indication:** On an evoke operation, a queue space error condition was detected. The evoke operation could not be performed because no *common* queue space was available at the time.

Recovery Action: Your program can issue a set timer operation and wait for a period of time, and then reissue the evoke operation. If an unacceptable number of queue space errors occur, you can disable all the subsystems and change the subsystem configuration by specifying a larger common queue space size in the SSP-ICF common queue space parameter. After the subsystem is enabled, reissue the acquire operation to restart the session.

8319 **Error Indication:** A negative response to the previous output operation was issued by the remote (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.

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 of 0412 (indicating that the remote program sent 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 data.

Recovery Action: Issue an input operation to receive the 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 remote 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: If you want your program to recover dynamically, reissue the output operation with a smaller output length. Otherwise, you can either change the record length in your program and recompile it, or you can change the value specified for the maximum user record length parameter in the subsystem configuration or in the SESSION statement. 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 erroneously followed by a *request to change direction then get* operation, a *request to change direction then invite* operation, a negative response operation, or a release 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 other 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 indication). The session is still active.

Recovery Action: Before attempting to communicate with another other program, correct the error in your program.

8324 **Error Indication:** On an output operation, a function management header was detected by the subsystem. Your program erroneously sent a function management header with a record that was not the first record in the chain. A header is valid only at the beginning of a chain, not within the chain. The session is still active.

Recovery Action: Change your program so that it sends a function management header only with the first record in a chain.

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 remote 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 remote 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 remote program. The session is still active.

Recovery Action: If you want your program to dynamically 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 incoming procedure start request and cannot, therefore, issue any evoke operations in this session.

Recovery Action: If you want your program to dynamically 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.

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 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 remote 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 remote 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 code of xx00 (a change direction indication) or xx08 (an end of transaction indication).

Recovery Action: Issue another input operation.

8332 **Error Indication:** On an input operation performed by this program, a cancel operation was received with an end of transaction indication. The last of the data was sent, but the remote program canceled the transaction (possibly because it detected an error in the data). The session is still active.

Recovery Action: If your program started the session, it can issue another evoke operation to start another transaction, it can issue an end of session operation to continue local processing, or it can terminate. If your program was evoked, it can wait[1] to be evoked again (MRT programs only), continue local processing, or terminate.

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.

839B **Error Indication:** On an input or output operation, the subsystem detected a block size error before it detected that both your program and the remote system were attempting to send data at the same time. The remote system sent data, but the length of the data block exceeded the length of the subsystem's line buffer. The session, however, is still active.

Recovery Action: Check that the value for the maximum user record length (RECL) parameter is correct in the subsystem configuration record or in the SESSION statement. Then, if your program started the session, reissue the output operation.

[1] For BASIC and RPG II, the set timer operation cannot be issued if the session was not acquired. See item 4 in the boxed description of major code 82.

Appendix A. Summary Charts

This appendix contains the following summary charts:

- A return code summary chart. The chart lists all of the communications-related return codes, and indicates, for each code, all of the subsystems supported by SSP-ICF that can issue that code to a program.

- An input/output operations summary chart. The chart shows all of the input, output, and combined input/output operations that can be issued in communications programs. It also indicates which subsystems support each operation.

RETURN CODE SUMMARY CHART

The following chart lists all of the return codes that are valid for any of the nine subsystems supported by SSP-ICF. Each subsystem whose name is shown on the same line as a return code can issue that code to a communications program that is using that subsystem.

Code	Subsystems								
0000	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
0001	Intra	BSCEL	CCP	CICS	IMS			Peer	SNUF
0003	Intra							Peer	SNUF
0004	Intra								SNUF
0005	Intra								SNUF
0007									SNUF
0008	Intra	BSCEL			IMS			Peer	SNUF
000C	Intra								SNUF
0010	Intra	BSCEL	CCP	CICS				Peer	SNUF
0012						3270			
0020		BSCEL	CCP			3270			SNUF
0021		BSCEL		CICS					SNUF
0028	Intra	BSCEL			IMS			Peer	SNUF
0030		BSCEL				3270			SNUF
0031		BSCEL		CICS					
0038	Intra	BSCEL			IMS			Peer	SNUF
0100	Intra	BSCEL	CCP		IMS		Finance	Peer	SNUF
0101	Intra	BSCEL	CCP	CICS	IMS			Peer	SNUF
0103								Peer	SNUF
0104	Intra								SNUF
0105	Intra								SNUF
0107									SNUF
0108	Intra	BSCEL			IMS			Peer	SNUF
010C	Intra								SNUF
0118	Intra	BSCEL			IMS			Peer	
0200	Intra						Finance	Peer	SNUF
0201	Intra	BSCEL	CCP	CICS	IMS			Peer	SNUF
0203	Intra							Peer	SNUF
0204	Intra								SNUF
0205	Intra								SNUF
0207									SNUF
0208	Intra	BSCEL			IMS			Peer	SNUF
020C	Intra								SNUF
0212						3270			
0220		BSCEL	CCP			3270			SNUF
0221		BSCEL		CICS					SNUF

Code	Subsystems								
0228	Intra	BSCEL			IMS			Peer	SNUF
0230		BSCEL	CCP			3270		Peer	SNUF
0231		BSCEL		CICS					
0238	Intra	BSCEL			IMS			Peer	SNUF
0300	Intra	BSCEL	CCP	CICS	IMS			Peer	SNUF
0301	Intra	BSCEL						Peer	SNUF
0302	Intra							Peer	
0303	Intra							Peer	SNUF
0308	Intra	BSCEL	CCP	CICS	IMS			Peer	SNUF
0310	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
0402	Intra								
0411	Intra	BSCEL	CCP		IMS				
0412	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
0800	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
1100	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
2800	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
3401	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
8081	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
8082	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
8083			CCP		IMS	3270	Finance	Peer	
8084			CCP		IMS	3270	Finance	Peer	
80BD			CCP	CICS					
8136		BSCEL							
8137		BSCEL							
8183		BSCEL	CCP		IMS		Finance	Peer	
8184		BSCEL	CCP		IMS		Finance	Peer	
8185		BSCEL	CCP					Peer	
8186		BSCEL	CCP						
8187							Finance		
8191		BSCEL	CCP					Peer	SNUF
8192		BSCEL	CCP						
8193		BSCEL	CCP	CICS					
8194		BSCEL	CCP	CICS					
8195							Finance	Peer	SNUF
8196								Peer	SNUF
8197		BSCEL							
8198		BSCEL							
8199		BSCEL	CCP						
819A		BSCEL	CCP						

Code	Subsystems								
819B		BSCEL	CCP						
819C		BSCEL	CCP						SNUF
819D		BSCEL						Peer	SNUF
819E			CCP						
819F			CCP						
81A3							Finance		
81B5									SNUF
81B6									SNUF
81B8		BSCEL							
81B9		BSCEL							
81BA							Finance		
81BC		BSCEL	CCP	CICS					
820A		BSCEL							
820D					IMS				
8213		BSCEL	CCP	CICS	IMS	3270			SNUF
8215		BSCEL	CCP	CICS		3270		Peer	SNUF
821E		BSCEL		CICS	IMS		Finance		
8233	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
8236		BSCEL	CCP						
8281	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
8282	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
8283		BSCEL	CCP						
8285		BSCEL	CCP					Peer	
8286		BSCEL	CCP						
8288							Finance		
8289		BSCEL							
828A		BSCEL							
828B		BSCEL							
828C		BSCEL							
828D		BSCEL							
828E		BSCEL							
828F		BSCEL							
8290		BSCEL							
8291		BSCEL	CCP						
8293		BSCEL	CCP						
8296									SNUF
8297		BSCEL							
829B			CCP						
829F			CCP						

Code	Subsystems								
82A0		BSCEL							
82A1									SNUF
82A2			CCP						
82A5									SNUF
82A6								Peer	SNUF
82A7		BSCEL	CCP	CICS				Peer	
82A8	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
82A9					IMS				
82AA	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
82AB	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
82AC					IMS				
82AD					IMS				
82AE					IMS	3270			
82AF					IMS				
82B0	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
82B1			CCP	CICS	IMS	3270			
82B2			CCP	CICS					
82B3					IMS	3270		Peer	SNUF
82B4	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
82BB							Finance		
82BC		BSCEL	CCP						
830B	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
830D					IMS				
830E					IMS				
8313	Intra				IMS		Finance	Peer	
8314	Intra							Peer	
8315	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
8316								Peer	
8317					IMS				
8319	Intra								SNUF
831A	Intra	BSCEL	CCP		IMS			Peer	
831B	Intra								SNUF
831C	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
831D					IMS				
831E	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
831F	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
8320					IMS				
8322	Intra	BSCEL	CCP		IMS		Finance	Peer	SNUF
8323	Intra							Peer	SNUF

Code	Subsystems								
8324									SNUF
8326	Intra								SNUF
8327	Intra	BSCEL			IMS			Peer	SNUF
8329	Intra	BSCEL	CCP	CICS	IMS		Finance	Peer	SNUF
832A	Intra							Peer	
832B			CCP	CICS	IMS				
832C	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
832D	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
832E			CCP						
832F	Intra	BSCEL	CCP			3270		Peer	SNUF
8330	Intra								SNUF
8331	Intra								SNUF
8332									SNUF
8333	Intra	BSCEL	CCP	CICS	IMS	3270	Finance	Peer	SNUF
8334		BSCEL			IMS	3270		Peer	
8336				CICS					
8338						3270			
8339						3270			
833C		BSCEL							
8383			CCP	CICS					
8384			CCP	CICS					
8385				CICS					
8386				CICS					
8391				CICS	IMS				
8392				CICS	IMS				
8397				CICS				Peer	
8398				CICS					
8399				CICS					
839A				CICS					
839B				CICS					SNUF
839C				CICS		3270			
83A7			CCP						
83B0								Peer	

INPUT/OUTPUT OPERATIONS SUMMARY CHART

The following chart lists all of the operations that can be issued in a communications program which uses a subsystem to communicate with another program on a remote system. All the input operations are listed (alphabetically) in the first column, the output operations are in the second column, and the operations that perform both input and output are in the third column.

Unless a footnote reference follows the name of an operation, the operation is supported for all nine subsystem types. The footnotes for the other operations give the names of the subsystems for which the operation is or is not valid.

Input Operations to Your Program	Output Operations from Your Program	Combined Operations in Your Program
Accept input		
	Acquire	
	Cancel[1]	Cancel then get[1,x] Cancel then invite[1]
	End of session	
	Evoke[6] Evoke end of transaction[7]	Evoke then get[6,x] Evoke then invite[6]
	Fail[2]	
Get Get attributes[y]		
Invite		
	Negative response[1]	Negative response then get[1,x] Negative response then invite[1]
Pass-through invite[1]	Pass-through put[1,x]	Pass-through put then invite[1]
	Put[5] Put end of file/chain[4] Put end of transaction[7]	Put then get[x] Put then invite
	Put FMH[3]	Put FMH then get[3,x] Put FMH then invite[3]
	Release	
		Request to change direction then get[8,x] Request to change direction then invite[8]
	Set timer	

Subsystem Restrictions

[1] Valid only for Intra and SNUF subsystems.
[2] Valid only for Intra and Peer subsystems.
[3] Valid only for Intra, Finance, and SNUF subsystems.
[4] Not valid for IMS subsystem.
[5] Not valid for 3270 subsystem.
[6] Not valid for Finance subsystem.
[7] Not valid for CCP, 3270, or Finance subsystems.
[8] Not valid for CICS, IMS, 3270, or Finance subsystems.

Language Restrictions

[x] Valid only in assembler language.
[y] Valid only in assembler and COBOL languages.

Appendix B. SNA Pass-Through Support

The SNA pass-through support is a function of the SNUF subsystem that allows a program to control the SNA commands and protocols. This support gives the user program increased capabilities and greater flexibility when communicating with the host system. The pass-through user, and therefore the reader of this chapter, is assumed to have previous knowledge of SNA.

The SNA pass-through support allows communication with applications on a System/370 with VTAM or TCAM that adhere to transmission services profile 3 or 4 and function management profile 3 or 4. The pass-through user program can have multiple concurrent sessions, each with a unique session ID.

The pass-through user program communicates to the SNUF subsystem through an SNA input/output area and the interactive communications DTF.

The pass-through program must define and use the SNA input/output area. The pass-through user program must modify the SNA input/output area to reflect the specific SNA operation desired.

When control is returned to the user program, the return code in the DTF indicates whether the operation was received by the SNUF subsystem. Only the interactive communications error return codes and the normal completion (00) code can be received. If the request was received by the SNUF subsystem, the SNA completion code field in the SNA input/output area indicates the status of the operation. (The SNA completion codes are listed later in this chapter.)

BASIC, COBOL AND RPG II PASS-THROUGH PROGRAMMING

Two formats ($$PTINV and $$PTPUT) are provided for issuing pass-through operations in BASIC, COBOL, and RPG II. When either of these formats are issued, the fields in the SNA input/output area must be included. These fields and their locations are described in *SNA Input/Output Area* later in this chapter.

$$PTINV

The $$PTINV format is used to invite input for a particular session. After receiving data from an invited session, via a subsequent input operation, SNA fills in the following fields:

- The logical unit number indicating the number of the logical unit from which the data was received.
- The operation code with the characters, 80, indicating that this is an SNA get operation. (See *SNA Operations* later in this chapter for a complete list of operation codes.)
- The completion code field indicating the completion code from SNA. (Note that the completion code from interactive communications data management is contained in the INFDS and should be checked first.) (A complete list of SNA completion codes is given later in this chapter.)
- All receive control byte fields.
- System sense and user sense fields if sense data was included by the remote system.
- The inbound sequence number.
- The record length field indicating the total length of the data that follows.
- The user data that was received.

$$PTPUT

The $$PTPUT format is used to put data, SNA commands, or responses to a particular session. The receive control bytes and the inbound and outbound sequence numbers should not be given values. The remainder of the fields should be updated as follows:

- The logical unit number with the number of the logical unit to which this data is to be sent.

- The operation code indicating which type of put to perform (See *SNA Operations* later in this chapter for a complete list of operation codes.)

- The transmit control bytes identifying the indicators to be used for this record.

- The system sense and user sense fields if sense data is included with this request.

- The record length field specifying the length of the data to be sent (maximum of 256 bytes).

- The user data to be sent.

When control is returned to the user program, the completion code from interactive communications data management should be checked. If that code shows a normal completion, the SNA completion code should be checked. To obtain the SNA completion code, another input operation must be issued to receive the updated SNA input/output area. If the $$PTPUT was issued during detail output, the accept done during the normal RPG cycle would make the updated SNA input/output area available. If the $$PTPUT was done as exception output, a NEXT and READ can be done to obtain the updated information.

BASIC ASSEMBLER PASS-THROUGH PROGRAMMING

A $SNIO macroinstruction is provided to use and modify the SNA input/output area. The $SNIO macroinstruction can have one of six forms:

- Define
- Get
- Invite
- Accept
- Put
- Command

Define

The define form of the $SNIO macroinstruction generates the storage required for the SNA input/output area and/or the labels, field constants, and field equates. The format of the define $SNIO macroinstruction is:

$$[\text{label}]\ \text{\$SNIO TYPE-DEF ,V-}\left\{\begin{array}{l}\underline{\text{EQU}}\\ \text{DC}\\ \text{ALL}\end{array}\right\}$$

TYPE: Specifies the type of $SNIO macroinstruction. TYPE-DEF identifies this as a define macroinstruction.

V: Specifies what this macroinstruction is to generate. DC indicates to generate only the storage required for the area. EQU indicates to generate only the equates for the field displacements, labels, and constants for the area. ALL indicates to generate both the storage and the equates.

A define $SNIO macroinstruction that specifies the equates must occur exactly once and can be anywhere within the program; however, it must be branched around if it is placed in the executable portion of the program.

Get

The get form of the $SNIO macroinstruction performs a get operation. The format of the get $SNIO macroinstruction is:

```
[label] $SNIO TYPE-GET [,SYMID-id]
                [,SNIOA-address] [,DTF-address]
                [,ERROR-address]
```

TYPE: Specifies the type of $SNIO macroinstruction. TYPE-GET identifies this as a get macroinstruction.

SYMID: Specifies the 2-character symbolic session ID for this operation. If this parameter is omitted, the value in the DTF remains unchanged.

SNIOA: Specifies the address of the leftmost byte of the SNA input/output area used for this operation. The value should be the label on the define $SNIO macroinstruction. If this parameter is omitted, the value in index register 1 is used.

DTF: Specifies the address of the leftmost byte of the DTF used for this operation. The value should be the label on the $DTFW macroinstruction. If this parameter is omitted, the value in index register 2 is used.

ERROR: Specifies the address to which control is passed if the return code in the DTF is not normal; that is, if the return code is hex 34 or is greater than hex 80. If this parameter is omitted, the next instruction is executed.

The get operation is issued to a specific session, and the issuing program waits for the data. When data is received, a data record is placed in the buffer of the SNA input/output area, and the length is placed in the prefix area. The receive control byte contains the request header received with the data.

If a response is received, the completion code field in the prefix area of the SNA input/output area indicates the type of response. Sense data, which accompanies a negative response, is also contained in the prefix.

If an SNA command is received, a completion code indicating the type of command is set. Any sense data received with the command is placed in sense area of the prefix.

Invite

The invite form of the $SNIO macroinstruction performs an invite operation. The format of the invite $SNIO macroinstruction is:

```
[label] $SNIO TYPE-INVITE [,SYMID-id]
                    [,DTF-address] [,ERROR-address]
```

TYPE: Specifies the type of $SNIO macroinstruction. TYPE-INVITE identifies this as an invite macroinstruction.

SYMID: Specifies the 2-character symbolic session ID for this operation. If this parameter is omitted, the value in the DTF remains unchanged.

DTF: Specifies the address of the DTF used for this operation. The value should be the label on the $DTFW macroinstruction. If this parameter is omitted, the value in index register 2 is used.

ERROR Specifies the address to which control is passed if the return code in the DTF is not normal; that is, if the return code is hex 34 or is greater than hex 80. If this parameter is omitted, the next instruction is executed.

The fields in the SNA input/output area remain unchanged after the invite operation.

Accept

The accept form of the $SNIO macroinstruction performs an accept operation. The format of the accept $SNIO macroinstruction is:

```
[label] $SNIO TYPE-ACCEPT [,DTF-address]

                  [,SNIOA-address] [,ERROR-address]

                  [,NEWREQ-address]
```

TYPE: Specifies the type of $SNIO macroinstruction. TYPE-ACCEPT identifies this as an accept macroinstruction.

DTF: Specifies the address of the DTF used for this operation. The value should be the label on the $DTFW macroinstruction. If this parameter is omitted, the value in index register 2 is used.

SNIOA: Specifies the address of the SNA input/output area used for this operation. The value should be the label on the define $SNIO macroinstruction. If this parameter is omitted, the value in index register 1 is used.

ERROR: Specifies the address to which control is passed if the return code in the DTF is not normal; that is, if the return code is hex 34 or is greater than hex 80. If this parameter is omitted, the next instruction is executed.

NEWREQ: Specifies the address to which control is passed if the return code in the DTF indicates a new requestor. If this parameter is omitted, the next instruction is executed.

The accept operation places the ID of the session that provided the input in the DTF. If more than one invite was issued before the accept, the program should check this ID. The record address field in the DTF points to the leftmost byte of the SNA input/output area that contains the data. After any pass-through request, except invite, index register 1 points the leftmost byte of the SNA input/output area.

Put

The put form of the $SNIO macroinstruction transmits data to the remote system. The format of the put $SNIO macroinstruction is:

$$\left[\text{label}\right]\ \text{\$SNIO TYPE-PUT}\ \left[\text{,SYMID-id}\right]\ \left[\text{,SNIOA-address}\right]\ \left[\text{,DTF-address}\right]$$

$$\left[\text{,RCAD-address}\right]\ \left[\text{,RECL-length}\right]\ \left[\text{,RECLAD-address}\right]$$

$$\left[\text{,FMH-}\begin{Bmatrix}\text{Y}\\ \underline{\text{N}}\end{Bmatrix}\right]\ \left[\text{,CODE-}\begin{Bmatrix}\underline{0}\\ 1\end{Bmatrix}\right]\left[\text{,CD-}\begin{Bmatrix}\underline{\text{Y}}\\ \text{N}\end{Bmatrix}\right]$$

$$\left[\text{,CHAIN-}\begin{Bmatrix}\text{FIRST}\\ \text{MID}\\ \text{LAST}\\ \underline{\text{ONLY}}\end{Bmatrix}\right]\ \left[\text{,BRAK-}\begin{Bmatrix}\text{BEGIN}\\ \text{END}\\ \text{BOTH}\\ \underline{\text{NONE}}\end{Bmatrix}\right]\ \left[\text{,DR1-}\begin{Bmatrix}\underline{\text{Y}}\\ \text{N}\end{Bmatrix}\right]$$

$$\left[\text{,DR2-}\begin{Bmatrix}\text{Y}\\ \underline{\text{N}}\end{Bmatrix}\right]\ \left[\text{,ERI-}\begin{Bmatrix}\underline{\text{Y}}\\ \text{N}\end{Bmatrix}\right]$$

TYPE: Specifies the type of $SNIO macroinstruction. TYPE-PUT identifies this as a put macroinstruction.

SYMID: Specifies the 2-character symbolic session ID for this operation. If this parameter is omitted, the value in the DTF remains unchanged.

SNIOA: Specifies the address of the SNA input/output area used for this operation. The value should be the label specified of the define $SNIO macroinstruction. If this parameter is omitted, the value in index register 1 is used.

DTF: Specifies the address of the DTF used for this operation. The value should be the label on the $DTFW macroinstruction. If this parameter is omitted, the value in index register 2 is used.

RCAD: Specifies the address of the leftmost byte of the record to be sent. The record is moved from this address into the SNA input/output area. If the record is already there, do not code this parameter.

RECL: Specifies the length, in decimal, of the data record to be sent. This length cannot be greater than 256 and cannot be specified if the RECLAD parameter is used. If neither RECL nor RECLAD is specified, the DTF remains unchanged.

RECLAD: Specifies the address of the leftmost byte of a two-byte field containing the length, in binary, of the data record to be sent. The value in the field cannot be greater than 256 bytes. This parameter cannot be used if RECL is used. If neither RECL nor RECLAD is specified, the DTF remains unchanged.

FMH: Specifies whether a function management header is included with this request. Y indicates that a function management header is present; N indicates that a function management header is not present and is the default. The function management header must be at the beginning of the record area, and its length must be included when specifying record length (RECL or RECLAD). FMH is valid only with a CHAIN of FIRST or ONLY.

CODE: Specifies the setting of the code selection indicator for this request. 0 indicates to use the standard code (EBCDIC) and is the default; 1 indicates that this is a special code.

CD: Specifies whether change direction should be indicated on this request. Y indicates to request a change direction; N indicates to not request a change direction and is the default. CD is valid only with a CHAIN of LAST or ONLY.

CHAIN: Specifies the status of chaining for this operation. FIRST identifies this record as the beginning of the chain. MID identifies this record as neither the beginning nor the end of the chain. LAST identifies this record as the end of the chain. ONLY identifies this record as both the first and the last record in the chain and is the default.

BRAK: Specifies the status of brackets for this operation. BEGIN identifies this as the first chain in the bracket. END identifies this as the last chain in the bracket. BOTH identifies this as the only chain in the bracket. NONE indicates that this chain is not the first or last chain in the bracket and is the default. BRAK values of BEGIN, END, and BOTH can only be specified with a CHAIN of FIRST or ONLY.

DR1: Specifies whether a type 1 definite response is requested for this operation. Y indicates that a type 1 definite response is requested, and is the default; N indicates that a type 2 response is not requested. If DR1-Y is specified, the next operation should be a get or accept to receive the response.

DR2: Specifies whether a type 2 definite response is requested for this operation. Y indicates that a type 2 definite response is requested; N indicates that a type 2 response is not requested and is the default. If DR2-Y is specified, the next operation should be a get or accept to receive the response.

ERI: Specifies whether an exception response is requested for this operation. Y indicates an exception response is requested and is the default; N indicates that an exception response is not requested.

Command

The command form of the $SNIO macroinstruction is used to send SNA commands and responses to the remote system. The format of the command $SNIO macroinstruction is:

```
[label] $SNIO TYPE-COMMAND [,SYMID-id] [,SNIOA-address]

                  [,DTF-address] [,ERROR-address] [,CTYPE-command]

                  [,USENSE-sense] [,SSENSE-sense]

                  [,BRAK- {END }] [,CD- {Y}]
                          {NONE}        {N}

                  [,LUNUM-number]
```

TYPE: Specifies the type of $SNIO macroinstruction. TYPE-COMMAND identifies this as a command macroinstruction.

SYMID: Specifies the 2-character symbolic session ID for this operation. If this parameter is omitted, the value in the DTF remains unchanged.

SNIOA: Specifies the address of the SNA input/output area used for this operation. The value should be the label on the define $SNIO macroinstruction. If this parameter is omitted, the value in index register 1 is used.

DTF: Specifies the address of the DTF used for this operation. The value should be the label on the $DTFW macroinstruction. If this parameter is omitted, the value in index register 2 is used.

ERROR: Specifies the address to which control is passed if the return code in the DTF is not normal; that is, if the return code is hex 34 or hex 80 or greater. If this parameter is omitted, the next instruction is executed.

CTYPE: Specifies the type of SNA command or response that is to be sent. The following values are allowed (references to other parameters refer to those whose descriptions follow this one):

- POSRSP indicates to send a positive response. No further parameters are required.

- NEGRSP indicates to send a negative response. Only the USENSE and SSENSE parameters are required.

- CANCEL indicates to send a cancel command. BRAK-END and CD-Y can also be specified.

- SIGNAL indicates to send a signal command. A signal code can optionally be supplied using the USENSE and SSENSE parameters. The default signal code is hex 0001, request to change direction.

- LUSTAT indicates to send a logical unit status command. A status code or user information must be supplied using the USENSE and SSENSE parameters. BRAK-END and CD-Y can also be specified.

- RSHUTD indicates to send a request shutdown command. No further parameters are required.

- SHUTC indicates to send a shutdown complete command. No further parameters are required.

- CHASE indicates to send a chase command. BRAK-END and CD-Y can also be specified.

- RTR indicates to send a ready to receive command. No further parameters are required.

- INIT indicates to perform an initialize without logon operation. The LUNUM parameter specifies the logical unit number to which the operation is performed.

- LOGON indicates to perform an initialize with logon operation. The LUNUM parameter specifies the logical unit number to log on to.

- LOGOFF indicates to perform a terminate with logoff operation. No further parameters are required.

- QEC indicates to send a quiesce at end of chain command. No further parameters are required.

- QC indicates to send a quiesce complete command. No further parameters are required.

- RELQ indicates to send a release quiesce command. No further parameters are required.

- RQR indicates to send a request recovery command. No further parameters are required.

- UNFSSCP indicates to send unformatted data on an LU-SSCP session.

USENSE: Specifies a 4-character hexadecimal number to be sent as the user sense bytes. USENSE is valid only with a CTYPE of NEGRSP, SIGNAL, or LUSTAT. If this parameter is not specified when one of those CTYPE values is specified, the value in the SNA input/output area is used. If USENSE is specified with any other CTYPE, the value is ignored and the SNA input/output area is not modified.

SSENSE: Specifies a 4-character hexadecimal number to be sent as the system sense bytes. SSENSE is valid only with a CTYPE of NEGRSP, SIGNAL, or LUSTAT. If this parameter is not specified when one of these CTYPE values is specified, the value in the SNA input/output area is used. If SSENSE is specified with any other CTYPE, the value is ignored, and the SNA input/output area is not modified.

BRAK: Specifies the status of the brackets. For a command $SNIO macroinstruction, only BRAK-END is allowed. BRAK-END can be specified to terminate any active bracket with a CTYPE of CANCEL, CHASE, or LUSTAT. BRAK values of BEGIN and BOTH are invalid.

CD: Specifies whether a change direction should be requested with this operation. For a command $SNIO macroinstruction, only CD-Y is allowed. CD-Y can be used with a CTYPE of CANCEL, CHASE, or LUSTAT.

LUNUM: Specifies the 3-digit decimal number identifying the logical unit session to be used. LUNUM is valid only with a CTYPE of INIT or LOGON. If LUNUM is not specified with either of these CTYPE values, the first available session is used. If LUNUM is specified with any other session, the value is ignored.

SNA INPUT/OUTPUT AREA

The SNA input/output area is used by the pass-through program to communicate to SNA the operation desired as well as any appropriate control information. All the fields are character fields; for example, a record length of 128 bytes would be represented as a character 128 (hex F1F2F8). The receive control byte and transmit control byte fields have a value of character 1 (hex F1) or character 0 (hex F0). The 1 indicates that the description applies to this record. For example, if the begin bracket indicator field contains a 1, this record is the beginning of a bracket.

The following is the format of the SNA input/output area:

Field Name	**Location**	**Description**
$SFLUNUM	1-3	SNA logical unit number
$SFOPC	4-5	SNA operation code
$SFCMPC	6-7	SNA completion code
Receive Control Bytes		
$SFRCFMH	8	Function management header included
$SFRCSDI	9	Sense data included
$SFRCBCI	10	Begin chain indicator
$SFRCECI	11	End chain indicator
$SFRCDR1	12	Definite response type 1 indicator
$SFRCDR2	13	Definite response type 2 indicator
$SFRCERI	14	Exception response indicator
$SFRCBBI	15	Begin bracket indicator
$SFRCEBI	16	End bracket indicator
$SFRCCDI	17	Change direction indicator
$SFRCCSI	18	Code selection indicator
	19-20	Reserved

Field Name	Location	Description
Transmit Control Bytes		
$SFTCFMH	21	Function management header included
$SFTCSDI	22	Sense data included
$SFTCBCI	23	Begin chain indicator
$SFTCECI	24	End chain indicator
$SFTCDR1	25	Definite response type 1 indicator
$SFTCDR2	26	Definite response type 2 indicator
$SFTCERI	27	Exception response indicator
$SFTCBBI	28	Begin bracket indicator
$SFTCIBI	29	End bracket indicator
$SFTCCDI	30	Change direction indicator
$SFTCCSI	31	Code selection indicator
	32-33	Reserved
$SFSSNS	34-37	System sense bytes
$SFUSNS	38-41	User sense bytes
$SFSSEQ	42-46	Inbound sequence number
$SFOSEQ	47-51	Outbound sequence number
	52-57	Reserved
$SFRUBL	58-60	Record length
	61-316	User data

SNA OPERATIONS

When using the pass-through support with RPG II, the operation code field must be filled in to indicate the type of operation desired. The following lists the operation codes, and describes the other fields that must be used when issuing the operations:

- Put data (40): Transmit control bytes must be used and the record length must be the length of the data to send.

- Put positive response (50): The record length must be 000.

- Put negative response (51): A negative response must include the system sense and user sense data. The record length must be 000.

- Put cancel (60): The record length should be 000. The end bracket indicator and the change direction indicator fields in the transmit control bytes can be set to 1.

- Put chase (61): The record length must be 000. The end bracket indicator and the change direction indicator fields in the transmit control bytes can be set to 1.

- Put logical unit status (62): The record length should be 000. The user sense and system sense fields must also be filled in. The end bracket indicator and change direction indicator fields in the transmit control bytes can be set to 1.

- Put ready to receive (63): The record length must be 000.

- Put quiesce complete (64): The record length must be 000.

- Put signal (70): The record length must be 000. A signal code should be supplied in the system sense and user sense fields. The default code of 00 01, request change direction, is used if none is specified.

- Put request shutdown (71): The record length must be 000.

- Put shutdown complete (72): The record length must be 000.

- Put quiesce at end of chain (73): The record length must be 000.

- Put release quiesce (74): The record length must be 000.

- Put request recovery (75): The record length must be 000.

- Get (80): No other parameters are required.

- Initialize with logon (C0): The logical unit number should contain the number of the logical unit to log on to. The record length should be the length of the logon message, which should be in the user data field.

- Initialize without logon (C1): The logical unit number should contain the number of the logical unit to log on to. The record length must be 000.

- Terminate with logoff (C2): The record length must be 000.

- Send unformatted LU-SSCP message (C3): The record length should be the length of the message. The message should be in the user data field.

A description of the SNA commands is given later in this appendix.

SNA COMPLETION CODES

The following completion codes can be returned from SNA. The completion code value is in hexadecimal.

Completion Code	Description
40	Successful completion
41	Permanent line error
42	Logical unit session reset; one of the following commands was received: • Unbind • De-activate logical unit • Activate logical unit (cold) • De-activate physical unit • Activate physical unit (cold)
43	Invalid SNA request; one of the following occurred: • The initialize operation was not the first request of SNA • The put response for the bind was not the second request of SNA • An invalid operation code was issued
45	Bind command rejected by SNA; bind parameters available in the buffer
46	Logon failed; unformatted message available in the buffer
47	Logon failed; no message available
48	Unbind received
50	Quiesce at end of chain received
51	Start data traffic received
52	Bind received
53	Cancel received
54	Chase received
55	Bid received
56	Signal received
57	Shutdown received
58	Negative response received on normal flow
59	Negative response received on expedited flow

Completion Code	Description
5A	Positive response received on expedited flow
5B	Exception request received; put negative response required
5C	Transmit control usage error
5D	Purging chain state exited
5E	Quiesce complete received
5F	Release quiesce received
61	Set and test sequence numbers received; data available in the buffer
62	Logical unit status received
63	Disk buffer queue overflow
64	Permanent disk error for SNA
80	Response exception; one of the following has occurred: • Response operation issued, but no response required • Response not allowed • Positive response and negative response required
81	Get or put operation issued and a response is required
82	Protocol state error; one of the following has occurred: • Chaining error; middle of chain or end of chain specified and not in chains, or begin chain specified and already in chains • Bracket error; end bracket specified while between brackets or secondary station cannot send end bracket, begin bracket specified and already in brackets, or begin bracket not specified on a request issued between brackets • Put invalid after shutdown complete sent • Put invalid during error recovery • Format indicator invalid with middle of chain or end chain request • Get data operation invalid while sending a chain • Put data operation invalid while receiving a chain

Completion Code **Description**

82 (Continued)

- Change direction indicator invalid with beginning of chain or middle of chain
- Begin bracket or end bracket specified with middle of chain or end of chain
- Definite response specified with beginning of chain or middle of chain or session requires exception response chains
- Exception response specified with end of chain and session requires definite response chains

83 Put operation issued and response required from host system

SNA COMMANDS

The following SNA commands can be issued or received by an SNA pass-through user program:

- Bid
- Cancel
- Chase
- Logical unit status
- Quiesce at end of chain
- Quiesce complete
- Ready to receive
- Release quiesce
- Request recovery
- Request shutdown
- Set and test sequence numbers
- Shutdown
- Shutdown complete
- Signal

Bid

The bid command is the host system request to start a bracket. Only the host can send the bid command. When the user program receives a bid command, it should send either a positive response or a negative response. A positive response indicates to the host that the System/34 program will not begin another bracket but will wait for the host to begin one. A negative response denies the host permission to begin a bracket, but the user program can grant permission later by sending a ready to receive command.

Cancel

The cancel command indicates to the receiver that the remainder of the current chain will not be sent. Both System/34 and the host can send a cancel command. If the user program cancels a chain, it must then retransmit the entire chain. If the System/34 user program receives a cancel command, it must acknowledge it by sending a positive response. You must then ignore the part of the chain you have received so far and prepare to receive the entire chain again.

Chase

The chase command requests the receiver to send all outstanding responses. Both System/34 and the host system can send a chase command. If you receive a chase command, send all outstanding responses and then send a positive response to the chase command.

Logical Unit Status

The logical unit status command, using 4 bytes of data, indicates the status of the sending logical unit. The first two bytes of sense data are system data, and the last two bytes are user data. Valid sense byte values are described in the *SNA Reference Summary.*

Quiesce at End of Chain

The quiesce at end of chain command indicates to the receiver that it should stop sending data at the end of this chain. The receiver can respond with either a negative response or a quiesce complete command. The negative response indicates that the receiver will continue to send. The quiesce complete command indicates that the receiver has quiesced.

Quiesce Complete

The quiesce complete command indicates to the receiver that the remote system has quiesced. While quiesced, the program must receive any data or commands and respond accordingly. Data cannot be sent until a release quiesce command is received.

Ready to Receive

The ready to receive command indicates to the receiver that it can now begin a bracket. Only the secondary can send a ready to receive command. If you have previously sent a negative response to a bid command, send ready to receive when you are ready to accept a bracket from the host system.

Release Quiesce

The release quiesce command indicates to the receiver that it can begin to send. The release quiesce command is sent after a station has been quiesced by the quiesce at end of chain command.

Request Recovery

The request recovery command indicates to the receiver that it should begin error recovery. The request recovery command can be sent by only the secondary.

Request Shutdown

The request shutdown command indicates that the sender is ready to have the session de-activated. Only the secondary can send a request shutdown command. You should send request shutdown when you have completed sending information to the host system. The host system sends a positive response but might continue to send data. When the host system is ready to end the session it sends an unbind command, which SNA responds to, and the session is de-activated.

Set and Test Sequence Numbers

The set and test sequence numbers command indicates to the receiver that resynchronization of the sequence numbers is required. Only the primary can send the set and test sequence numbers command. Included with the command are the sequence numbers to be checked; these are placed in the buffer with no conversion to decimal being done. The response should indicate whether the numbers have been accepted (positive response) or rejected (negative response); a positive response must include 5 bytes of data (hexadecimal characters) in the buffer.

Shutdown

The shutdown command signals the receiver to stop sending when it is ready to end the session. Only the host system can send a shutdown command. If you receive a shutdown command, send a positive response and then continue to send data until you have finished. When you finish sending data, send the shutdown complete command.

Shutdown Complete

The shutdown complete command indicates that the sender, in response to the shutdown command, is ready to end the session and will send no more data. Only the secondary can send a shutdown complete command. Because shutdown complete is an expedited command, the host might receive it before receiving some of the data you have already sent. To avoid this situation, request a definite response on your last request and wait for the response before sending shutdown complete. You must still be prepared to send responses to the host system.

Signal

The signal command, using 4 bytes of information, requests an action from the receiver. Both the System/34 and the host can send a signal command. The only request currently supported for the signal command is a request to send (a signal field of hex 0001 0000). If you receive a signal command and it is a request to send, reply with either a positive response (granting the host permission to send) or a negative response (denying the host permission to send).

Appendix C. BSC to SNA Migration

IMS/VS CONSIDERATIONS

The following items must be considered when writing interactive communications programs that communicate with IMS/VS:

- Uninvited data
- End of transaction operations
- BATCH parameter on the SESSION OCL 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 on 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. During configuration, a procedure can be specified that begins executing when uninvited data is received. The subsystem passes the data to the program, which can then process the data, save the data in a file, or discard the data.

Uninvited data can be received for any of the following reasons:

- A message switch or broadcast message is received.
- A System/34 application program releases a session with 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 on the session. The System/34 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 issued to the IMS/IRSS subsystem, the subsystem indicates to IRSS that the message is ended.

When an end of transaction operation is issued to the SNUF subsystem, the subsystem sends a change direction indicator and expects to receive a null record terminating the transaction from IMS/VS. If anything but the null record is received, the session is abnormally terminated and an error return code is given to the System/34 application. Something other than the null record can be received in any of the following cases:

- The IMS/VS application has generated output to be sent on this session.
- An output message was placed on the IMS/VS output queue by an IMS/VS application other than the one evoked by the System/34 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 to IMS/VS.

BATCH Parameter on the SESSION OCL Statement

The BATCH parameter on the SESSION OCL statement determines how the SNUF and IMS/IRSS subsystems handle input and output operations.

When BATCH-NO is specified, the IMS/IRSS subsystem accumulates each record as it is received from the System/34 application. The records are then sent as one message when the application program indicates the message is complete (either by an end of transaction 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-NO is specified, the SNUF subsystem handles each record from the System/34 application as a complete chain. IMS/VS requires that when a chain ends, a change direction must also be sent. This means that the System/34 application cannot issue multiple output operations consecutively. To assure that the change direction is sent, each put or evoke operation must be accompanied by an invite, get, or end of transaction modifier. Note that an evoke operation without a modifier followed by a get or invite operation is not acceptable because the change direction would not have accompanied the evoke operation, and IMS/VS would, therefore, have rejected it.

When BATCH-YES is specified, the IMS/IRSS subsystem transmits each record when it is received from the System/34 application. Each record is transmitted as a segment of a message until an end of transaction or input operation is issued. A session specified as BATCH-YES cannot be acquired while another session is active because of the delays in line turnaround that might be experienced. BATCH-YES should be specified when large amounts of data must be sent without intervening responses from the host system.

When BATCH-YES is specified, the SNUF subsystem sends each record from the System/34 application as an element of a chain. This allows the application to perform consecutive output operations; however, a change 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.

Converting IMS/IRSS Programs to SNUF Programs

If the following guidelines are followed, programs that communicate to 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 OCL statement.

- If a stand-alone get operation (one that is not a modifier of an evoke or put operation) is used, the BATCH-YES parameter must be specified on the SESSION OCL 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.

- Minor return codes for permanent errors (major return codes of hex 80 or greater or *STATUS values greater than 99) returned by the SNUF subsystem are different from those returned by the IMS/IRSS subsystem.

SYSTEM/34 CONVERSION CONSIDERATIONS

An interactive communications program currently running using the BSCEL subsystem can use the Peer subsystem. The following paragraphs describe considerations for converting the application program.

Configuration parameters for the Peer subsystem are different from those for the BSCEL subsystem. See the appropriate subsystem chapters for a complete description of configuration parameters.

Parameters on the SESSION OCL statement for the Peer subsystem are different from those for the BSCEL subsystem. See the Peer and BSCEL chapters for a description of each type of SESSION statement.

The application program using the BSCEL subsystem should be running with a partner attribute of NORM.

The put end of file operation ($$SENDE) executed using the BSCEL subsystem is treated as a put end of chain operation by the Peer subsystem, but these are functionally the same. See Chapter 2 for a description of these operations.

Minor return codes returned by the BSCEL subsystem might be different from those returned by the Peer subsystem. If the application program checks minor return codes, changes might have to be made to handle the Peer minor return codes. If the program checks only major return codes, no changes in return code checking are required.

An application program using the BSCEL subsystem must be synchronized with the application program on the remote system. That is, in situations when either application program can perform input or output, one application program must perform input and the other must perform output. A program using the Peer subsystem can achieve synchronization with the remote application 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 A contains a summary chart that shows which return codes are used by each subsystem.

CICS/VS CONSIDERATIONS

Application programs written using the CICS subsystem can be run without change using the SNUF subsystem if no minor return codes are checked. The minor return codes issued by the two subsystems differ somewhat, especially in indicating an end of transaction, so 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.

Appendix D. Debugging Interactive Communications Programs

When debugging interactive communications programs, the same facilities normally available to the programming language (DEBUG operation in RPG II, for example) can be used. See the appropriate programming language chapter for a description of the debugging facilities available in each language.

In addition, status displays and a debug facility are provided for debugging interactive communications programs issued.

STATUS DISPLAYS

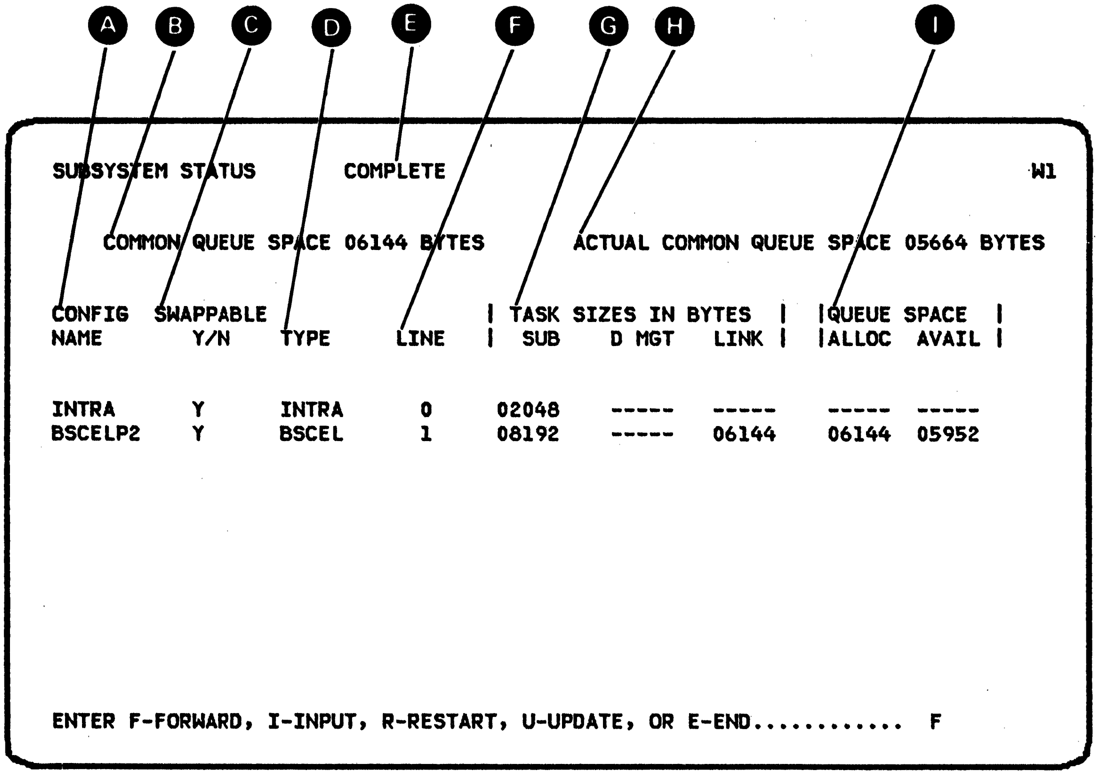
Two status displays are available for displaying interactive communications information: a subsystem status display and a session status display. These displays are available to the system operator using the STATUS operator command. Complete information on the STATUS command is in the *Operator's Guide*.

Subsystem Status

The status of active interactive communications subsystems can be displayed by entering:

```
STATUS    SUBSYS
D         I
```

The following is a sample subsystem status display:



```
SUBSYSTEM STATUS       COMPLETE                                       W1

  COMMON QUEUE SPACE 06144 BYTES      ACTUAL COMMON QUEUE SPACE 05664 BYTES

CONFIG   SWAPPABLE                | TASK SIZES IN BYTES  |  |QUEUE SPACE  |
NAME        Y/N    TYPE      LINE |  SUB    D MGT   LINK |  |ALLOC  AVAIL |


INTRA       Y      INTRA      0     02048   -----   -----    -----  -----
BSCELP2     Y      BSCEL      1     08192   -----   06144    06144  05952







ENTER F-FORWARD, I-INPUT, R-RESTART, U-UPDATE, OR E-END............  F
```

Ⓐ CONFIG NAME shows a list of subsystem configuration names currently enabled or being enabled.

Ⓑ COMMON QUEUE SPACE shows the total amount of common queue space allocated.

Ⓒ SWAPPABLE indicates whether the subsystem is swappable. Y indicates it is swappable; N indicates it is not.

Ⓓ TYPE indicates the type of subsystem being used.

Ⓔ COMPLETE appears on the last page of a status display. Any page but the last has blanks in this position. * indicates a noncommunicating Peer session (slow poll only).

Ⓕ LINE indicates the line number being used by the subsystem. (The Intra subsystem indicates line 0 or no line.)

Ⓖ TASK SIZES IN BYTES shows the sizes of all the tasks involved. SUB shows the size of the subsystem task. D MGT shows the size of the subsystem data management task (the SNA task for the SNUF subsystem, for example). LINK shows the size of the interrupt handler for the subsystem. If this interrupt handler is SDLC, this value includes line buffers.

Ⓗ ACTUAL COMMON QUEUE SPACE shows the amount of common queue space being used.

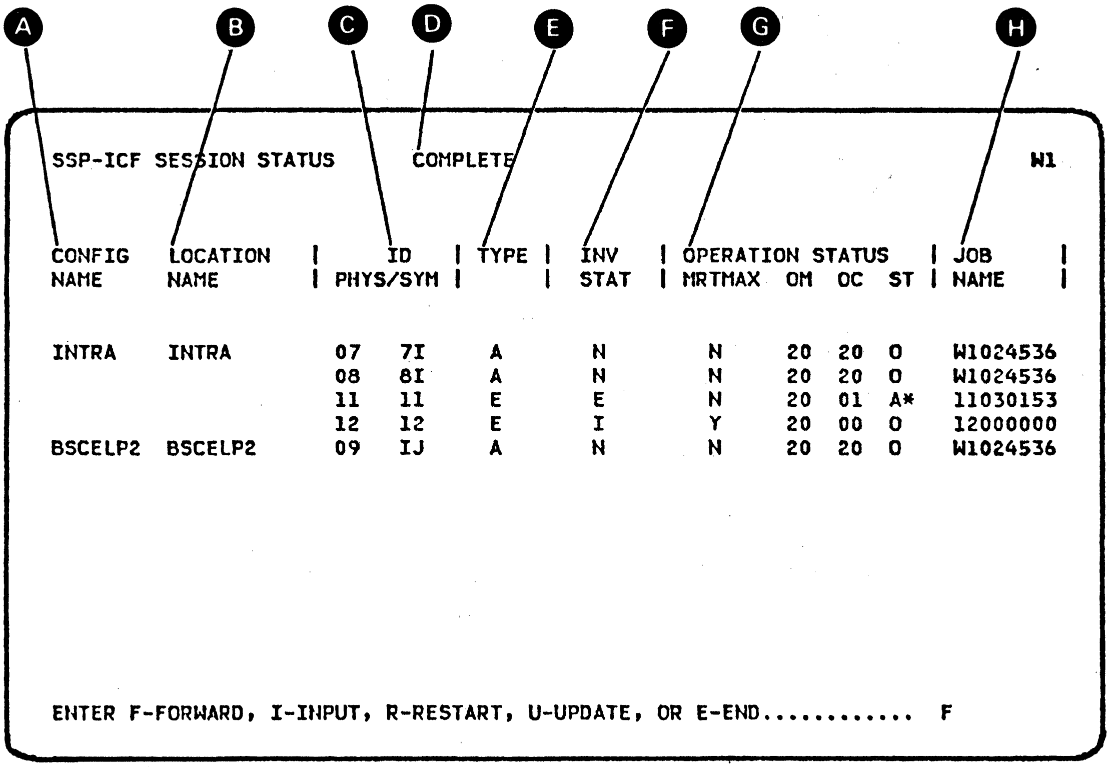
Ⓘ QUEUE SPACE shows the status of the subsystem queue space. ALLOC shows the amount of the subsystem queue space allocated. AVAIL shows the amount of subsystem queue space available.

Session Status

The status of active interactive communications sessions can be displayed by entering:

```
STATUS    SUBSESS
D         N
```

The following is a sample session status display:



```
SSP-ICF SESSION STATUS       COMPLETE                                    W1

CONFIG   LOCATION  |    ID    | TYPE |  INV  | OPERATION STATUS    | JOB       |
NAME     NAME      | PHYS/SYM |      |  STAT | MRTMAX  OM   OC  ST | NAME      |


INTRA    INTRA       07    7I     A       N       N     20   20  O    W1024536
                     08    8I     A       N       N     20   20  O    W1024536
                     11    11     E       E       N     20   01  A*   11030153
                     12    12     E       I       Y     20   00  O    12000000
BSCELP2  BSCELP2     09    IJ     A       N       N     20   20  O    W1024536









ENTER F-FORWARD, I-INPUT, R-RESTART, U-UPDATE, OR E-END............  F
```

Ⓐ CONFIG NAME shows a list of subsystem names currently enabled or being enabled.

Ⓑ LOCATION NAME shows a list of active location names.

Ⓒ ID is a list of active session IDs. PHYS shows the physical ID (assigned by data management). SYM shows the symbolic ID (SYMID on SESSION OCL statement for acquired session).

Ⓓ COMPLETE appears on the last page of a status display. Any page but the last has blanks in this position.

Ⓔ TYPE shows the type of session. A indicates the session is an acquired session. E indicates the session was started by an incoming procedure request. * indicates a noncommunicating Peer session (slow poll only).

Ⓕ INV STAT shows the invite status of the session. E indicates the session is invited. I indicates an accept has been issued for this session. N indicates the session is not invited. O indicates the session is invited and data is available.

Ⓖ OPERATION STATUS shows the status of the session. MRTMAX indicates if the session is waiting to attach to a MRT program that already has the maximum number of requestors. Y indicates the session is waiting; N indicates it is not. OM and OC show the current operation modifier and operation code (in hex). (A complete list of operation modifiers and codes is shown in the following table.) ST shows the status of the operation. A indicates the operation is active; O indicates the operation is complete. I indicates the operation is being issued to the subsystem. If the A, I, or O is followed by an *, the session is waiting for a reply from the remote system.

Ⓗ JOB NAME shows the name of the job to which the session is assigned.

Work Station Operation Code Modifiers

Modifier (In Hex)	Meaning
20	Session execute
21	Session fail
22	Session end of file or end of chain
24	Session function management header
28	Session pass-through
30	Session end of transaction

Work Station Operation Codes

Modifier (In Hex)	Meaning
01	Get data
02	Put data
03	Put data, then get data
04	Evoke data
05	Evoke then get data
08	Request to change direction
09	Request to change direction, then get data
10	Release session
20	Acquire session
40	Cancel
41	Cancel, then get data
80	Negative response
81	Negative response, then get data
99	Special acquire (used only by system programs)
E0	Normal end of step or end of job
F0	Abnormal end of step or end of job (used only by system programs)
FF	Assign failure occurred in subsystem queue space (used only by system programs)

DEBUG FACILITY

The debug facility is controlled by a procedure command (ICFDEBUG). When activated, the debug facility generates an entry in a disk file for each interactive communications operation issued. This entry includes such information as the operation, the return code, and the IDs of both sessions involved. The debug facility also allows these entries to be subsequently displayed or printed.

The debug facility runs as an SRT program. However, one copy of the program traces all application programs that issue interactive communications operations.

ICFDEBUG Procedure

The ICFDEBUG procedure command controls the execution of the debug facility. The command can be entered from any command capable display station. The format of the ICFDEBUG procedure command is:

```
          ┌ON              ┐
          │OFF             │
ICFDEBUG  │CRT             │
          │CRT,xxxxxxxx    │
          │PRINT           │
          └PRINT,xxxxxxxx  ┘
```

ON: Activates the debug facility. The debug facility begins recording interactive communications operations in a disk file. Any previous file created by the debug facility is deleted.

OFF: De-activates the debug facility. No further interactive communications operations will be traced. The disk file created by the debug facility is deleted.

CRT: Displays the file created by the debug facility. If a file exists and contains entries to be displayed, up to eight entries are displayed at one time on the display screen. The maximum of eight entries applies to both 960-character and 1920-character display screens. For the format of the display and a description of operator considerations, see *Displaying the Debug File* later in this appendix. CRT is the default.

CRT,xxxxxxxx: Displays the entries from the file created by the debug facility for a specific job. The xxxxxxxx specifies the job name for which entries are to be displayed. For the format of the display and a description of operator considerations, see *Displaying the Debug File* later in this appendix.

PRINT: Prints the file created by the debug facility. All entries in the file are printed on the printer associated with the display station that entered this command. For the format of the printed output, see *Printing the Debug File* later in this chapter.

PRINT,xxxxxxxx: Prints the entries from the file created by the debug facility for a specific job. The xxxxxxxx specifies the job name for which entries are to be printed. For the format of the printed output, see *Printing the Debug File* later in this chapter.

Note that whenever a debug file is printed or displayed, a new file is created to record all subsequent interactive communications operations. After a file is displayed or printed, it is deleted.

Debug File Information

The debug file consists of 64-byte entries. Up to 256 of these entries can be written in the file. When the file is filled, new entries are written over the oldest entries in the file, so that the file always contains entries for the most recent 256 interactive communications operations. The entries from the file can be either printed or displayed, and are presented in chronological order from the oldest entry to the newest entry.

An entry is made in the file when the return code is given to the application program that issued an interactive communications operation, so the entries might not be in the order of execution. (The operations issued to a display station are not included in the debug file.) Each entry contains the following items:

Job name: This field is an 8-character field giving the job name of the job that issued the operation.

Procedure name: This field is an 8-character field giving the name of the first level procedure containing the program that issued the operation.

Program name: This field is an 8-character field giving the name of the application program that issued the operation.

Location name: This field is an 8-character field giving the location name (LOCATION parameter of the SESSION OCL statement) associated with the session that issued the operation.

Format name: This field is an 8-character field giving the format name used to issue this operation. The format name begins with $$ and is the name used in an RPG II or COBOL program to issue the operation. If no format name was used, this field contains blanks.

Symbolic ID: This field is a 5-character field giving the symbolic session ID of the session that issued the operation. If the Intra subsystem is being used, this field also gives the ID of the session being communicated with. The first two characters are the ID of the session that issued the operation. If the session is not using the Intra subsystem, the remaining three characters are blank. If the session is using the Intra subsystem, the third character is a slash (/) followed by the ID of the session with which communication is occurring.

Return code: This field is a 4-character field giving the major and minor return code received by the application program following this operation. The first two characters are the major return code, and the second two characters are the minor return code. See Appendix A for a list of all the return codes.

Operation code: This field is a 4-character field that identifies the operation that was issued. The characters are the same as those coded to issue the operation in assembler. See Chapter 3 for a list of these symbolic operations.

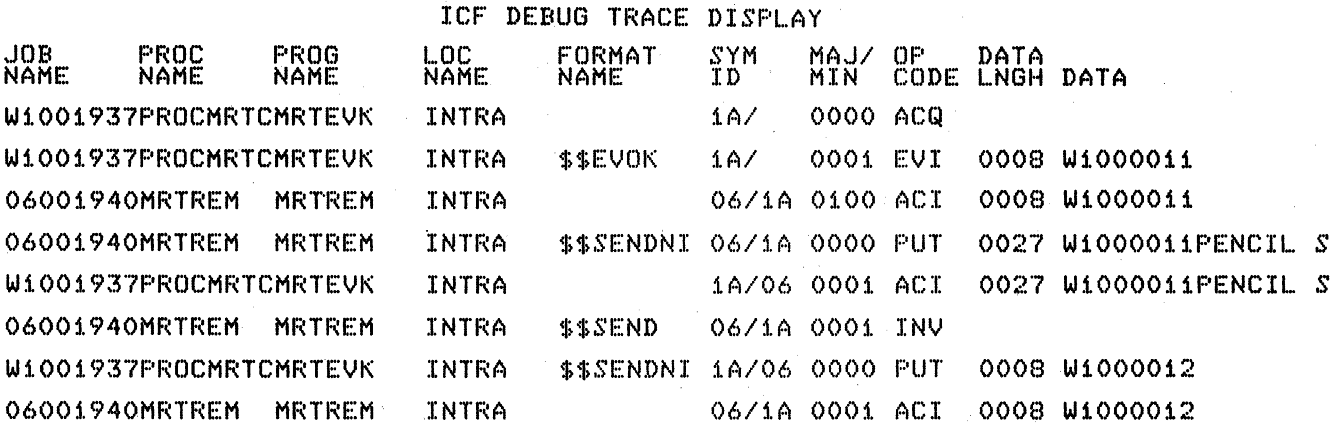
Data length: This field is a 4-character field giving the length of the data sent or received by the program. The length is displayed as a decimal value. If no data is associated with this operation, the length is displayed as blanks.

Data: This field is a 16-character field giving the first 16 characters of data associated with this operation. Characters with a value less than hex 40 are displayed as blanks. If no data is associated with this operation, blanks are displayed.

Printing the Debug File

To print the debug file, specify the PRINT option on the ICFDEBUG procedure command. Either the entire file or just the entries for a specific job number can be printed. Headings are printed at the top of each page to identify the fields, and entries are printed from oldest to newest. After the file has been printed, it is deleted.

The following is a sample printout of a debug file:



```
                          ICF DEBUG TRACE DISPLAY
JOB     PROC    PROG     LOC     FORMAT   SYM   MAJ/ OP   DATA
NAME    NAME    NAME     NAME    NAME     ID    MIN  CODE LNGH DATA
W1001937PROCMRTCMRTEVK   INTRA            1A/   0000 ACQ
W1001937PROCMRTCMRTEVK   INTRA   $$EVOK   1A/   0001 EVI  0008 W1000011
06001940MRTREM  MRTREM   INTRA            06/1A 0100 ACI  0008 W1000011
06001940MRTREM  MRTREM   INTRA   $$SENDNI 06/1A 0000 PUT  0027 W1000011PENCIL S
W1001937PROCMRTCMRTEVK   INTRA            1A/06 0001 ACI  0027 W1000011PENCIL S
06001940MRTREM  MRTREM   INTRA   $$SEND   06/1A 0001 INV
W1001937PROCMRTCMRTEVK   INTRA   $$SENDNI 1A/06 0000 PUT  0008 W1000012
06001940MRTREM  MRTREM   INTRA            06/1A 0001 ACI  0008 W1000012
```

Displaying the Debug File

To display the debug file, the CRT option must be specified on the ICFDEBUG procedure command. Either the entire file or just the entries for a specific job number can be displayed. Headings are displayed to identify the fields, and the entries are displayed from oldest to newest.

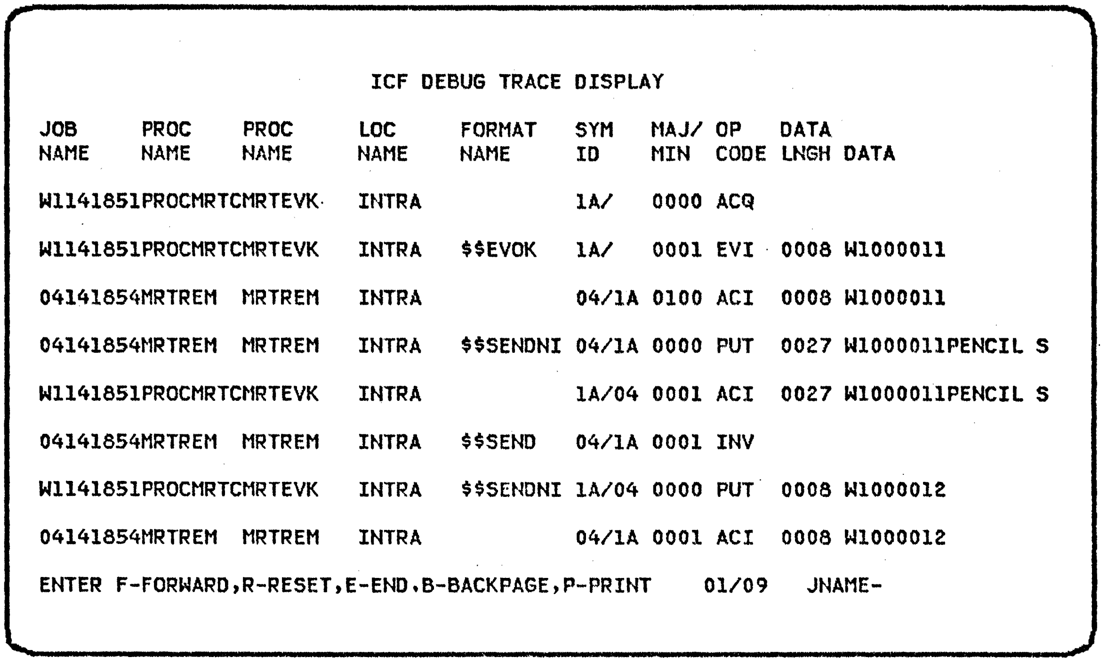
Only eight entries can be displayed at one time, but the operator can page forward and backward through the file as well as return to the beginning of the file and print the file. The operator can enter one of the following characters to control the displaying of the file:

- F to display the next eight entries
- B to display the previous eight entries
- R to return to the first eight entries
- P to print all the entries to be displayed and to return to the first eight entries
- E to end the display functions of the debug facility (the file being displayed is deleted)

Also on the display are the page number identifying the current page of eight entries and the total number of pages to be displayed. These two numbers are separated by a slash (/).

If the operator requests only the entries for a specific job (CRT, xxxxxxxx), a job name (JNAME) is displayed to indicate the job name that was requested. If the operator requests the entries for all jobs (CRT), the JNAME field displays eight blanks. The operator can enter a job name in a blank JNAME field to display entries for all jobs or enter a different job name in the JNAME field to display entries for a specific job. The same options (F, B, R, P, and E) are available. If a requested job name has no entries, only the headings are displayed or printed. The operator can also blank out the JNAME field to display entries for all jobs.

The following is a sample display of a debug file requested by ICFDEBUG CRT:



```
                          ICF DEBUG TRACE DISPLAY

JOB     PROC    PROC     LOC     FORMAT   SYM   MAJ/ OP   DATA
NAME    NAME    NAME     NAME    NAME     ID    MIN  CODE LNGH DATA

W1141851PROCMRTCMRTEVK   INTRA            1A/   0000 ACQ

W1141851PROCMRTCMRTEVK   INTRA   $$EVOK   1A/   0001 EVI  0008 W1000011

04141854MRTREM  MRTREM   INTRA            04/1A 0100 ACI  0008 W1000011

04141854MRTREM  MRTREM   INTRA   $$SENDNI 04/1A 0000 PUT  0027 W1000011PENCIL S

W1141851PROCMRTCMRTEVK   INTRA            1A/04 0001 ACI  0027 W1000011PENCIL S

04141854MRTREM  MRTREM   INTRA   $$SEND   04/1A 0001 INV

W1141851PROCMRTCMRTEVK   INTRA   $$SENDNI 1A/04 0000 PUT  0008 W1000012

04141854MRTREM  MRTREM   INTRA            04/1A 0001 ACI  0008 W1000012

ENTER F-FORWARD,R-RESET,E-END.B-BACKPAGE,P-PRINT    01/09   JNAME-
```

Appendix E. SSP-ICF Installation Verification

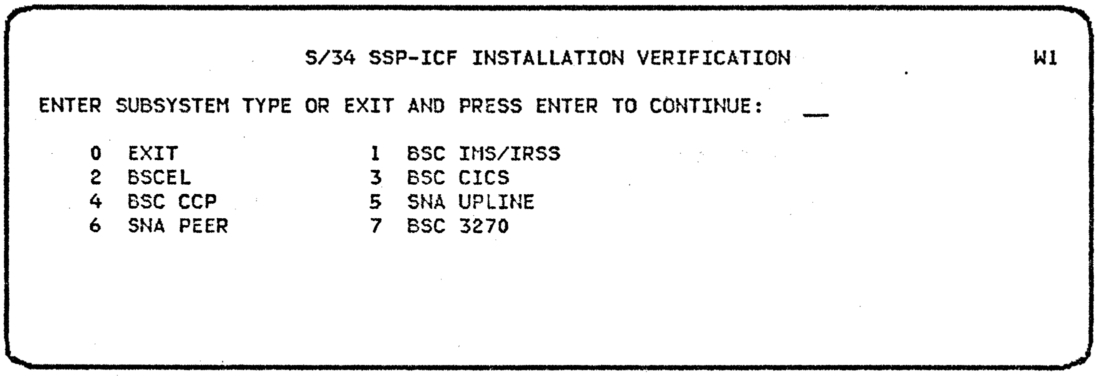
The SSP-ICF installation verification package is provided with the IBM System/34 Interative Communications feature and is automatically loaded when an SSP-ICF subsystem is installed during INSTALL. This package contains procedures, screen formats, and programs to facilitate the verification process. It also contains a procedure named ICDROP which, when run, removes the entire package from disk. The ICDROP procedure must be run from the system console and no other programs can be running.

The SSP-ICF installation verification package can be used to verify that a communications link is operational between:

- A BSCEL subsystem on a local System/34 and a BSCEL subsystem on a remote System/34
- A CCP subsystem on a local System/34 and a CCP system on a remote System/3
- A CICS subsystem on a local System/34 and a CICS/VS host system
- An IMS/IRSS subsystem on a local System/34 and an IMS/VS host system
- A BSC 3270 subsystem on a local System/34 and a remote host system
- A Peer subsystem on a local System/34 and a Peer subsystem on a remote System/34
- A SNUF subsystem on a local System/34 and an IMS/VS or CICS/VS host system

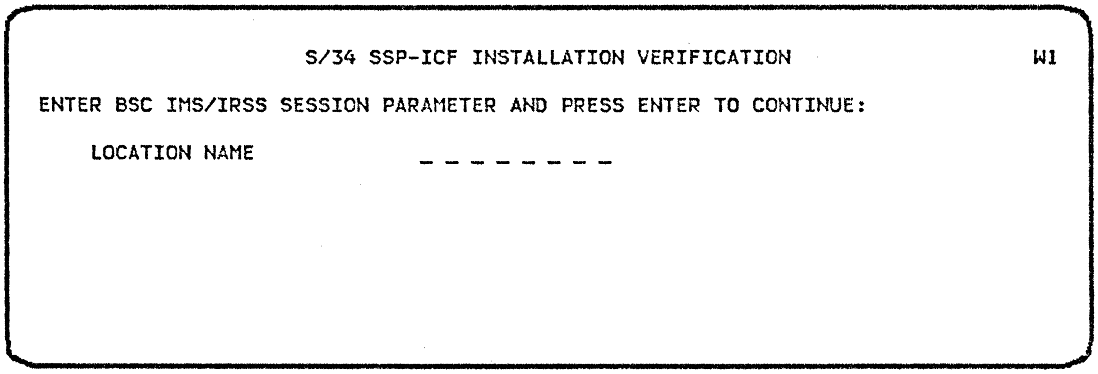
The operation of the communications link can be verified once one of these subsystems is configured and enabled on the local System/34, its communications partner activated on the remote system, and the communications line started.

You verify the operation of a communications link by running the ICVERIFY procedure. This procedure displays the following screen which prompts you to select the subsystem to be used in the verification test:

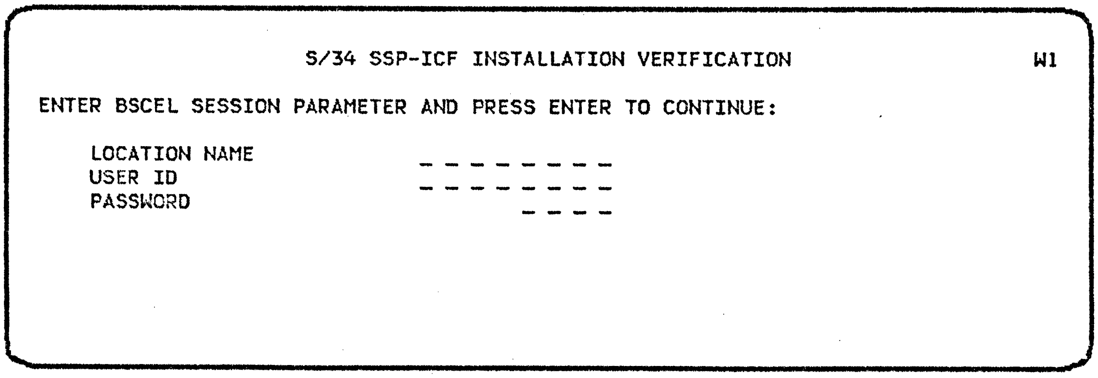


```
                     S/34 SSP-ICF INSTALLATION VERIFICATION                 W1

ENTER SUBSYSTEM TYPE OR EXIT AND PRESS ENTER TO CONTINUE:  _

    0  EXIT                   1  BSC IMS/IRSS
    2  BSCEL                  3  BSC CICS
    4  BSC CCP                5  SNA UPLINE
    6  SNA PEER               7  BSC 3270
```

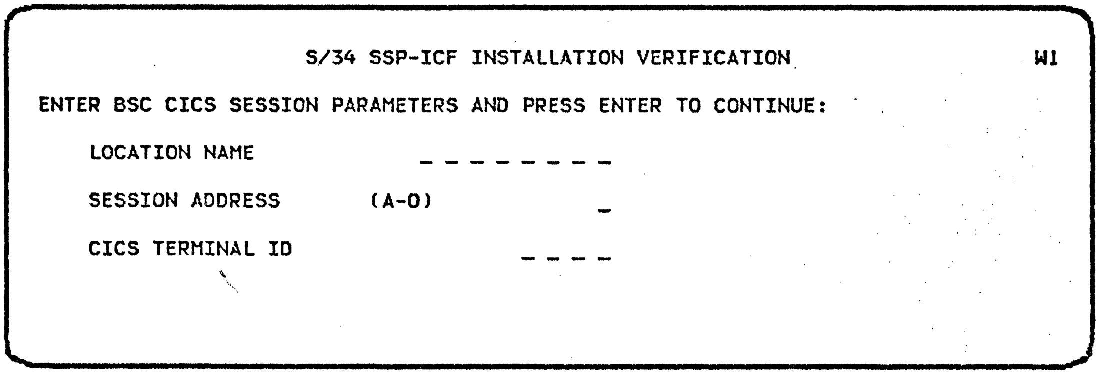
Once you select a subsystem, the ICVERIFY procedure displays one of the following subsystem dependent screens:



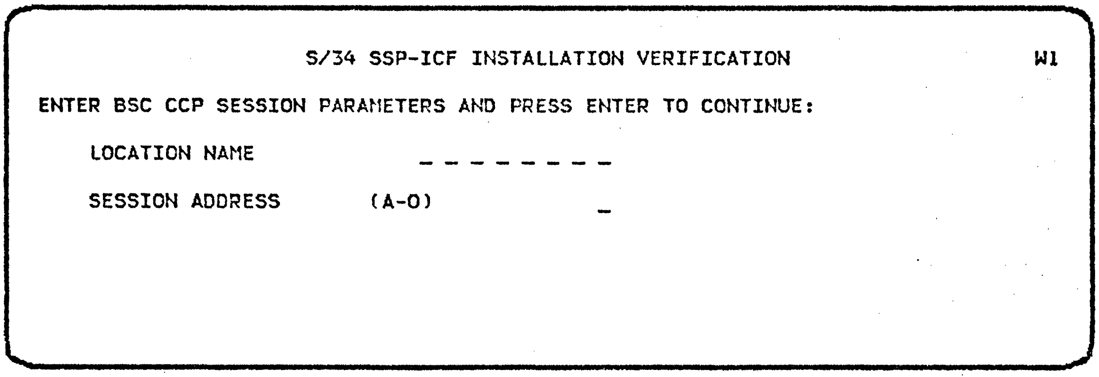
```
                     S/34 SSP-ICF INSTALLATION VERIFICATION                 W1

ENTER BSC IMS/IRSS SESSION PARAMETER AND PRESS ENTER TO CONTINUE:

    LOCATION NAME            _ _ _ _ _ _ _ _
```



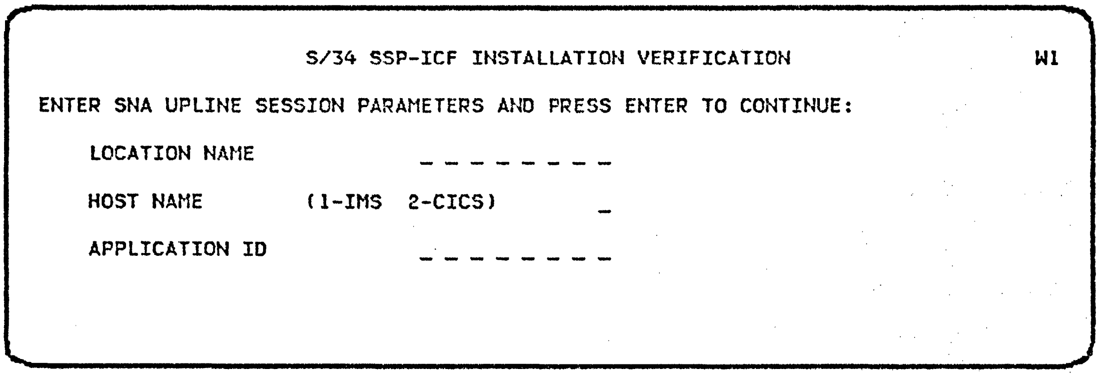
```
                     S/34 SSP-ICF INSTALLATION VERIFICATION                 W1

ENTER BSCEL SESSION PARAMETER AND PRESS ENTER TO CONTINUE:

    LOCATION NAME            _ _ _ _ _ _ _ _
    USER ID                  _ _ _ _ _ _ _ _
    PASSWORD                         _ _ _ _
```



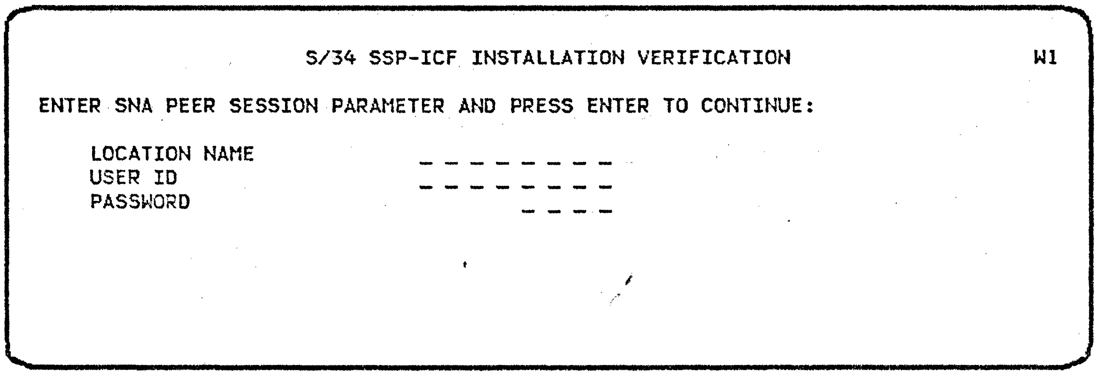
```
                    S/34 SSP-ICF INSTALLATION VERIFICATION                 W1

ENTER BSC CICS SESSION PARAMETERS AND PRESS ENTER TO CONTINUE:

    LOCATION NAME                      _ _ _ _ _ _ _ _

    SESSION ADDRESS        (A-O)                     _

    CICS TERMINAL ID                           _ _ _ _
```



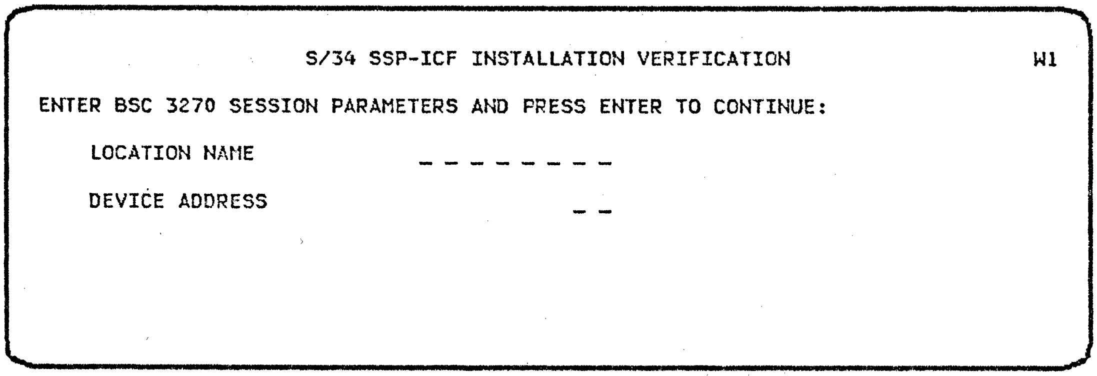
```
                    S/34 SSP-ICF INSTALLATION VERIFICATION                 W1

ENTER BSC CCP SESSION PARAMETERS AND PRESS ENTER TO CONTINUE:

    LOCATION NAME                      _ _ _ _ _ _ _ _

    SESSION ADDRESS        (A-O)                     _
```



```
                    S/34 SSP-ICF INSTALLATION VERIFICATION                 W1

ENTER SNA UPLINE SESSION PARAMETERS AND PRESS ENTER TO CONTINUE:

    LOCATION NAME                      _ _ _ _ _ _ _ _

    HOST NAME          (1-IMS  2-CICS)                _

    APPLICATION ID                     _ _ _ _ _ _ _ _
```

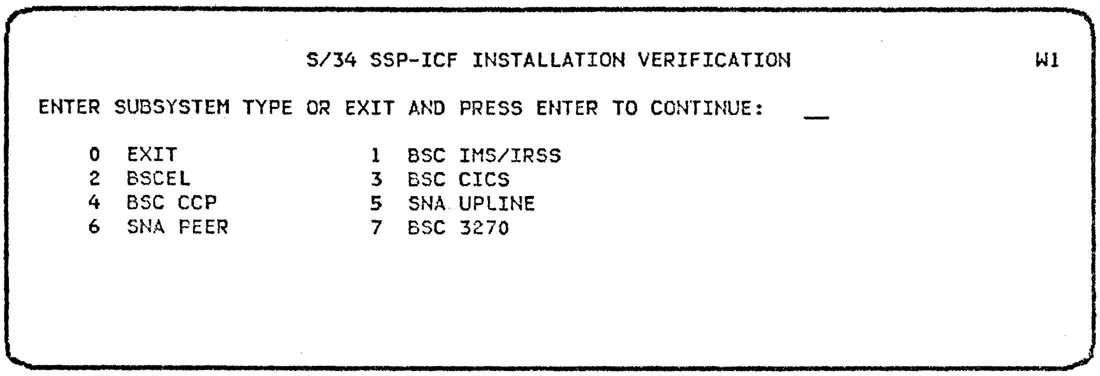


```
                   S/34 SSP-ICF INSTALLATION VERIFICATION                 W1

ENTER SNA PEER SESSION PARAMETER AND PRESS ENTER TO CONTINUE:

    LOCATION NAME             _ _ _ _ _ _ _ _
    USER ID                   _ _ _ _ _ _ _ _
    PASSWORD                          _ _ _ _
```



```
                   S/34 SSP-ICF INSTALLATION VERIFICATION                 W1

ENTER BSC 3270 SESSION PARAMETERS AND PRESS ENTER TO CONTINUE:

    LOCATION NAME             _ _ _ _ _ _ _ _

    DEVICE ADDRESS                        _ _
```

These screens prompt you to enter parameters associated with the SESSION OCL statement for the specified subsystem. Once you enter the session parameters the verification is attempted and you are notified of the results. If successful, the ICVERIFY procedure again displays the following screen:



```
                    S/34 SSP-ICF INSTALLATION VERIFICATION                W1

ENTER SUBSYSTEM TYPE OR EXIT AND PRESS ENTER TO CONTINUE:  _

    0  EXIT                 1  BSC IMS/IRSS
    2  BSCEL                3  BSC CICS
    4  BSC CCP              5  SNA UPLINE
    6  SNA PEER             7  BSC 3270
```

You can verify another configured subsystem at this time or exit (0 option) from the procedure.

Session Parameter Descriptions

When running the ICVERIFY procedure, you are required to enter certain session parameters based on the type of subsystem being verified. A description of these session parameters follows:

Location Name: Specify the location name to be used by the verification program. This location name must be the same as the location name assigned during the CNFIGICF procedure for the respective subsystem. This parameter must be supplied for all subsystem types. For Peer, this is the remote location name.

User ID: If security is active on the remote System/34, specify a user ID that exists on that remote System/34 and its corresponding password. A user ID and a password do not need to be entered if security is not active on the remote System/34.

Password: If security is active on the remote System/34, specify the password associated with the user ID specified. A user ID and a password do not need to be entered if security is not active on the remote System/34.

Session Address: This parameter is used for only the CCP and CICS subsystems. This parameter must be specified when you want to use a specific session address. If not specified, the subsystem will specify an address from the address pool.

CICS Terminal ID: This parameter is used only by the CICS subystem. This parameter must be specified when a switched line is used to communicate with CICS/VS on the System/370. The value specified must be the 1 to 4 character TERMIDNT specified for your terminal in the CICS/VS Terminal Control Table. Leave this parameter blank if running on other than a switched BSC line.

Host Name: This parameter is used for the SNUF subsystem only. Specify the type of subsystem from the host system to be used for the session. Specify either a 1 (IMS/VS) or a 2 (CICS/VS).

Application ID: This parameter is used for the SNUF subsystem only. Specify the VTAM application ID of the CICS/VS or IMS/VS subsystem used on the host system. This application ID is used for the session.

Device Address: This parameter is used only by the BSC 3270 subsystem. This parameter specifies the physical address by which the remote system identifies the session. The address specified should correspond to a device capable of receiving data at the remote system when the procedure is run. If this parameter is not specified, the first available address, configured for programs in the active SSP-ICF configuration record, is used.

Appendix F. Planning Charts For Configuring Subsystems

The following planning charts are to be filled out before running the CNFIGICF procedure. A separate planning chart is included for each subsystem type. For instructions on how to fill out the planning chart, see the appropriate subsystem chapter.

Intra Subsystem Planning Chart

1.0 Subsystem Member Configuration

1. Subsystem configuration member name (8 characters) _ _ _ _ _ _ _ _
2. Subsystem library name (8 characters) _ _ _ _ _ _ _ _
   Select:
   1. Create new member   4. Delete a member
   2. Edit existing member   5. Review a member
   3. Create new member from existing member
3. Enter selection: ______
4. Existing member name: _ _ _ _ _ _ _ _
5. Existing member library name: _ _ _ _ _ _ _ _

2.0 Common SSP-ICF Parameters for Each Subsystem

1. SSP-ICF common queue space: (2 - 42 K) _ _
2. Define the subsystem type: _ 1
   1 Intra   2 BSC IMS/IRSS
   3 BSCEL   4 BSC CICS
   5 BSC CCP   6 SNA Upline
   7 SNA Peer   8 BSC 3270
   9 SNA 3270   10 Finance

3.0 General Subsystem Parameters

1. Location name: (8 characters) _ _ _ _ _ _ _ _
2. Subsystem queue space: (0-40 K) _ _
3. Subsystem support swappable: (0-No 1-Yes) _

BSCEL Subsystem Planning Chart

1.0 Subsystem Member Configuration

1. Subsystem configuration member name (8 characters) _ _ _ _ _ _ _ _
2. Subsystem library name (8 characters) _ _ _ _ _ _ _ _

   Select:

   1. Create new member 4. Delete a member

   2. Edit existing member 5. Review a member

   3. Create new member from existing member
3. Enter selection: ______
4. Existing member name: _ _ _ _ _ _ _ _
5. Existing member library name: _ _ _ _ _ _ _ _

2.0 Common SSP-ICF Parameters for Each Subsystem

1. SSP-ICF common queue space: (2 - 42 K) _ _
2. Define the subsystem type: _ 3

   1 Intra 2 BSC IMS/IRSS

   3 BSCEL 4 BSC CICS

   5 BSC CCP 6 SNA Upline

   7 SNA Peer 8 BSC 3270

   9 SNA 3270 10 Finance

3.0 General Subsystem Parameters

1. Location name: (8 characters) _ _ _ _ _ _ _ _
2. Subsystem queue space: (0-40 K) _ _
3. Subsystem support swappable: (0-No 1-Yes) _
4. Maximum user record length: (1-4075) _ _ _ _

4.0 Line Information for SSP-ICF Subsystem

1. Line type: 1-Multipoint _

   2-Nonswitched Pt-Pt

   3-Switched Pt-Pt

3. Switch type: _

   1 Manual call 2 Auto answer

   3 Manual answer

BSCEL Subsystem Planning Chart

5.0 BSC General Subsystem Parameters I

1. EBCDIC/ASCII: (1-EBCDIC 2-ASCII) _
2. Local station address: (2 hex) _ _
3. Wait time: (1 - 999 seconds) _ _ _
4. Transparency: (0-No 1-Yes) _
5. Multiple remote IDs: (0-No 1-Yes) _
6. Remote ID:

   _

7. Local ID:

   _

5.1 BSC General Subsystem Parameters II

1. Phone list name: _ _ _ _ _ _ _ _
2. Refresh: (0-No 1-Yes) _
3. Block length: (0–4075) _ _ _ _
4. Record separator: (Hexadecimal) _ _
5. ITB mode: (0-No 1-Yes) _
6. Blank: (0-No, 1-Compression, 2-Truncation) _
7. 3740 Multiple files: (0-No 1-Yes) _

6.0 BSCEL Subsystem Parameters

1. Partner (1-NORM 2-ATTR) _

BSC CCP Subsystem Planning Chart

1.0 Subsystem Member Configuration

1. Subsystem configuration member name (8 characters) _ _ _ _ _ _ _ _
2. Subsystem library name (8 characters) _ _ _ _ _ _ _ _

Select:
1. Create new member 4. Delete a member
2. Edit existing member 5. Review a member
3. Create new member from existing member

3. Enter selection: ______
4. Existing member name: _ _ _ _ _ _ _ _
5. Existing member library name: _ _ _ _ _ _ _ _

2.0 Common SSP-ICF Parameters for Each Subsystem

1. SSP-ICF common queue space: (2 - 42 K) _ _
2. Define the subsystem type: _ 5

1 Intra 2 BSC IMS/IRSS
3 BSCEL 4 BSC CICS
5 BSC CCP 6 SNA Upline
7 SNA Peer 8 BSC 3270
9 SNA 3270 10 Finance

3.0 General Subsystem Parameters

1. Location name: (8 characters) _ _ _ _ _ _ _ _
2. Subsystem queue space: (0-40 K) _ _
3. Subsystem support swappable: (0-No 1-Yes) _
4. Maximum user record length: (1-4075) _ _ _ _

4.0 Line Information for SSP-ICF Subsystem

1. Line type 1-Multipoint _
   2-Nonswitched Pt-Pt
   3-Switched Pt-Pt

5.0 BSC General Subsystem Parameters I

1. EBCDIC/ASCII: (1-EBCDIC 2-ASCII) _
2. Local station address: (2 hex) _ _
3. Wait time: (1 - 999 seconds) _ _ _
4. Transparency: (0-No 1-Yes) _

6. Remote ID:

_ _

5.1 BSC General Subsystem Parameters II

Key any changes and press ENTER to continue:
1. Phone list name _ _ _ _ _ _ _ _
2. Refresh: (0-No 1-Yes) _

BSC CCP Subsystem Planning Chart

10.0 BSC Multipoint Session Addresses

Define session addresses:

0-Address not defined
1-Address in pool
2-Address reserved

Incoming - Specify 0 or 2 (Blank)____

Outgoing - Specify 0, 1, or 2

A____ B____ C____ D____ E____
F____ G____ H____ I____ J____
K____ L____ M____ N____ O____

11.0 BSC CCP Subsystem Parameters

1. (Disposition of unsolicited host messages: __
   (1-System console 2-History file 3-Ignored)
2. Data mode escape characters: (Hexadecimal) _ _ _ _ _ _ _ _ _ _ _ _
3. Sign on option: (1-Enable 2-Acquire) __
4. Queuing: (0-No 1-Yes) __
5. CCP password security: (0-No 1-Yes) __
6. Specify password: _ _ _ _ _ _
7. Requested local ID: (15 characters) _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
8. Requestor local ID: (15 characters) _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

BSC CICS Subsystem Planning Chart

1.0 Subsystem Member Configuration

1. Subsystem configuration member name (8 characters) _ _ _ _ _ _ _ _
2. Subsystem library name (8 characters) _ _ _ _ _ _ _ _

Select:
1. Create new member   4. Delete a member
2. Edit existing member   5. Review a member
3. Create new member from existing member

3. Enter selection: ____
4. Existing member name: _ _ _ _ _ _ _ _
5. Existing member library name: _ _ _ _ _ _ _ _

2.0 Common SSP-ICF Parameters for Each Subsystem

1. SSP-ICF common queue space: (2 - 42 K) _ _
2. Define the subsystem type: _ 4

1 Intra   2 BSC IMS/IRSS
3 BSCEL   4 BSC CICS
5 BSC CCP   6 SNA Upline
7 SNA Peer   8 BSC 3270
9 SNA 3270   10 Finance

3.0 General Subsystem Parameters

1. Location name: (8 characters) _ _ _ _ _ _ _ _
2. Subsystem queue space: (0-40 K) _ _
3. Subsystem support swappable: (0-No 1-Yes) _
4. Maximum user record length: (1-4075) _ _ _ _

4.0 Line Information for SSP-ICF Subsystem

1. Line type: _
1-Multipoint
2-Nonswitched Pt-Pt
3-Switched Pt-Pt

3. Switch type: _
1 Manual call   2 Auto answer
3 Manual answer

BSC CICS Subsystem Planning Chart

5.0 BSC General Subsystem Parameters I

1. EBCDIC/ASCII: (1-EBCDIC 2-ASCII) _
2. Local station address: (2 hex) _ _
3. Wait time: (1-999 seconds) _ _ _
4. Transparency: (0-No 1-Yes) _
5. Multiple remote IDs: (0-No 1-Yes) _
6. Remote ID:

_ _

7. Local ID:

_ _

5.1 BSC General Subsystem Parameters II

Key any changes and press ENTER to continue:

1. Phone list name _ _ _ _ _ _ _ _
2. Refresh: (0-No 1-Yes) _

7.0 Subsystem Inactive Destination Messages

1. Subsystem procedure name: (8 characters) _ _ _ _ _ _ _ _
2. Subsystem procedure library name: (8 characters) _ _ _ _ _ _ _ _

10.0 BSC Multipoint Session Addresses

Define session addresses:

0-Address not defined
1-Address in pool
2-Address reserved
Incoming - Specify 0 or 2 (Blank) ___
Outgoing - Specify 0, 1, or 2 A___ B___ C___ D___ E___
F___ G___ H___ I ___ J ___
K___ L___ M___ N___ O___

BSC IMS/IRSS Subsystem Planning Chart

1.0 Subsystem Member Configuration

1. Subsystem configuration member name (8 characters) _ _ _ _ _ _ _ _
2. Subsystem library name (8 characters) _ _ _ _ _ _ _ _

Select:
1. Create new member 4. Delete a member
2. Edit existing member 5. Review a member
3. Create new member from existing member

3. Enter selection: ______
4. Existing member name: _ _ _ _ _ _ _ _
5. Existing member library name: _ _ _ _ _ _ _ _

2.0 Common SSP-ICF Parameters for Each Subsystem

1. SSP-ICF common queue space: (2 - 40 K) _ _
2. Define the subsystem type: _ 2

1 Intra 2 BSC IMS/IRSS
3 BSCEL 4 BSC CICS
5 BSC CCP 6 SNA Upline
7 SNA Peer 8 BSC 3270
9 SNA 3270 10 Finance

3.0 General Subsystem Parameters

1. Location name: (8 characters) _ _ _ _ _ _ _ _
2. Subsystem queue space: (2-40 K) _ _
3. Subsystem support swappable: (0-No 1-Yes) _

5.0 BSC General Subsystem Parameters I

2. Local station address: (2 hex) _ _

7.0 Subsystem Inactive Destination Messages

1. Subsystem procedure name: (8 characters) _ _ _ _ _ _ _ _
2. Subsystem procedure library name: (8 characters) _ _ _ _ _ _ _ _

12.0 BSC IMS/IRSS Subsystem PTERMs

1. Subsystem remote program start PTERM: _ _ _ _
2. Subsystem local PTERMs:

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

BSC 3270 Subsystem Planning Chart

1.0 Subsystem Member Configuration

1. Subsystem configuration member name (8 characters) _ _ _ _ _ _ _ _
2. Subsystem library name (8 characters) _ _ _ _ _ _ _ _
   Select:
   1. Create new member 4. Delete a member
   2. Edit existing member 5. Review a member
   3. Create new member from existing member
3. Enter selection: ____
4. Existing member name: _ _ _ _ _ _ _ _
5. Existing member library name: _ _ _ _ _ _ _ _

2.0 Common SSP-ICF Parameters for Each Subsystem

1. SSP-ICF common queue space: (2 - 42 K) _ _
2. Define the subsystem type: _ 8

1	Intra	2	BSC IMS/IRSS
3	BSCEL	4	BSC CICS
5	BSC CCP	6	SNA Upline
7	SNA Peer	8	BSC 3270
9	SNA 3270	10	Finance

3.0 General Subsystem Parameters

1. Location name: (8 characters) _ _ _ _ _ _ _ _
2. Subsystem queue space: (0-40 K) _ _
3. Subsystem support swappable: (0-No 1-Yes) _
4. Maximum user record length: (0-4075) _ _ _ _

4.0 Line Information for SSP-ICF Subsystem

2. Local station address: (2 hex) _ _

14.0 3270 Subsystem General Parameters

1. Line buffer size: (256-4096) _ _ _ _
2. Delay count: (0-255) _ _ _

15.0 BSC 3270 Subsystem Device Parameters

1. Device address	40	C1	C2	C3	C4	C5	C6	C7	C8	C9	4A	4B	4C	4D	4E	4F
2. Device type* (1, 2, 3, or 5)	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
3. S/34 logical ID	_ _	_ _	_ _	_ _	_ _	_ _	_ _	_ _	_ _	_ _	_ _	_ _	_ _	_ _	_ _	_ _
4. Lower case (0-No 1-Yes)	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_

5. Device address	50	D1	D2	D3	D4	D5	D6	D7	D8	D9	5A	5B	5C	5D	5E	5F
6. Device type* (1, 2, 3, or 5)	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
7. S/34 logical ID	_ _	_ _	_ _	_ _	_ _	_ _	_ _	_ _	_ _	_ _	_ _	_ _	_ _	_ _	_ _	_ _
8. Lower case (0-No 1-Yes)	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_

* 1-Program 2-3277 Display 3-3277 with numeric lock 5-3288 Printer

Finance Subsystem Planning Chart

1.0 Subsystem Member Configuration

1. Subsystem configuration member name (8 characters) _ _ _ _ _ _ _ _
2. Subsystem library name (8 characters) _ _ _ _ _ _ _ _

   Select:
   1. Create new member 4. Delete a member
   2. Edit existing member 5. Review a member
   3. Create new member from existing member
3. Enter selection:
4. Existing member name: ________ _ _ _ _ _ _ _ _
5. Existing member library name: _ _ _ _ _ _ _ _

2.0 Common SSP-ICF Parameters for Each Subsystem

1. SSP-ICF common queue space: (2–42 K) _ _
2. Define the subsystem type: 1 0

   1 Intra 2 BSC IMS/IRSS
   3 BSCEL 4 BSC CICS
   5 BSC CCP 6 SNA Upline
   7 SNA Peer 8 BSC 3270
   9 SNA 3270 10 Finance

3.0 General Subsystem Parameters

1. Location name: (8 characters) _ _ _ _ _ _ _ _
2. Subsystem queue space: (0–40 K) _ _
3. Subsystem support swappable: (0-No 1-Yes) _

3.1 SDLC General Subsystem Parameters

2. SDLC receive buffer size (2 or 4 K) _
3. SDLC transmit buffer size (2 or 4 K) _

Finance Subsystem Planning Chart

4.0 Line Information for SSP-ICF Subsystem

1. Line type: 1-Multipoint _
   2-Nonswitched Pt-Pt
   3-Switched Pt-Pt

3. Switch type: _
   1 Manual call 2 Auto answer
   3 Manual answer

17.0 Finance Subsystem Parameters (Extra copies of this section are included.)

1. Remote station address: (01—FE hexadecimal) _ _
2. Remote location name: _ _ _ _ _ _ _ _
3. Number of logical work stations: (1—30) _ _
4. Delayed entry: (0-No 1-Yes) _
5. Automatic recovery: (0-No 1-Yes) _
6. Location activated: (0-No 1-Yes) _
7. Exchange ID: (5 hexadecimal) _ _ _ _ _
8. System monitor session: (0-No 1-Yes) _

Finance Subsystem Planning Chart

17.0 Finance Subsystem Parameters

1. Remote station address: (01–FE hexadecimal) _ _
2. Remote location name: _ _ _ _ _ _ _ _
3. Number of logical work stations: (1–30) _ _
4. Delayed entry: (0-No 1-Yes) _
5. Automatic recovery: (0-No 1-Yes) _
6. Location activated: (0-No 1-Yes) _
7. Exchange ID: (5 hexadecimal) _ _ _ _ _
8. System monitor session: (0-No 1-Yes) _

17.0 Finance Subsystem Parameters

1. Remote station address: (01–FE hexadecimal) _ _
2. Remote location name: _ _ _ _ _ _ _ _
3. Number of logical work stations: (1–30) _ _
4. Delayed entry: (0-No 1-Yes) _
5. Automatic recovery: (0-No 1-Yes) _
6. Location activated: (0-No 1-Yes) _
7. Exchange ID: (5 hexadecimal) _ _ _ _ _
8. System monitor session: (0-No 1-Yes) _

17.0 Finance Subsystem Parameters

1. Remote station address: (01–FE hexadecimal) _ _
2. Remote location name: _ _ _ _ _ _ _ _
3. Number of logical work stations: (1–30) _ _
4. Delayed entry: (0-No 1-Yes) _
5. Automatic recovery: (0-No 1-Yes) _
6. Location activated: (0-No 1-Yes) _
7. Exchange ID: (5 hexadecimal) _ _ _ _ _
8. System monitor session: (0-No 1-Yes) _

17.0 Finance Subsystem Parameters

1. Remote station address: (01–FE hexadecimal) _ _
2. Remote location name: _ _ _ _ _ _ _ _
3. Number of logical work stations: (1–30) _ _
4. Delayed entry: (0-No 1-Yes) _
5. Automatic recovery: (0-No 1-Yes) _
6. Location activated: (0-No 1-Yes) _
7. Exchange ID: (5 hexadecimal) _ _ _ _ _
8. System monitor session: (0-No 1-Yes) _

Finance Subsystem Planning Chart

17.0 Finance Subsystem Parameters

1. Remote station address: (01–FE hexadecimal) _ _
2. Remote location name: _ _ _ _ _ _ _ _ _ _
3. Number of logical work stations: (1–30) _ _
4. Delayed entry: (0-No 1-Yes) _
5. Automatic recovery: (0-No 1-Yes) _
6. Location activated: (0-No 1-Yes) _
7. Exchange ID: (5 hexadecimal) _ _ _ _ _
8. System monitor session: (0-No 1-Yes) _

17.0 Finance Subsystem Parameters

1. Remote station address: (01–FE hexadecimal) _ _
2. Remote location name: _ _ _ _ _ _ _ _ _ _
3. Number of logical work stations: (1–30) _ _
4. Delayed entry: (0-No 1-Yes) _
5. Automatic recovery: (0-No 1-Yes) _
6. Location activated: (0-No 1-Yes) _
7. Exchange ID: (5 hexadecimal) _ _ _ _ _
8. System monitor session: (0-No 1-Yes) _

17.0 Finance Subsystem Parameters

1. Remote station address: (01–FE hexadecimal) _ _
2. Remote location name: _ _ _ _ _ _ _ _ _ _
3. Number of logical work stations: (1–30) _ _
4. Delayed entry: (0-No 1-Yes) _
5. Automatic recovery: (0-No 1-Yes) _
6. Location activated: (0-No 1-Yes) _
7. Exchange ID: (5 hexadecimal) _ _ _ _ _
8. System monitor session: (0-No 1-Yes) _

17.0 Finance Subsystem Parameters

1. Remote station address: (01–FE hexadecimal) _ _
2. Remote location name: _ _ _ _ _ _ _ _ _ _
3. Number of logical work stations: (1–30) _ _
4. Delayed entry: (0-No 1-Yes) _
5. Automatic recovery: (0-No 1-Yes) _
6. Location activated: (0-No 1-Yes) _
7. Exchange ID: (5 hexadecimal) _ _ _ _ _
8. System monitor session: (0-No 1-Yes) _

SNA Peer Subsystem Planning Chart

1.0 Subsystem Member Configuration

1. Subsystem configuration member name (8 characters) _ _ _ _ _ _ _ _
2. Subsystem library name (8 characters) _ _ _ _ _ _ _ _

   Select:

   1. Create new member 4. Delete a member
   2. Edit existing member 5. Review a member
   3. Create new member from existing member
3. Enter selection: ____
4. Existing member name: _ _ _ _ _ _ _ _
5. Existing member library name: _ _ _ _ _ _ _ _

2.0 Common SSP-ICF Parameters for Each Subsystem

1. SSP-ICF common queue space: (2 - 42 K) _ _
2. Define the subsystem type: _ 7

   1 Intra 2 BSC IMS/IRSS
   3 BSCEL 4 BSC CICS
   5 BSC CCP 6 SNA Upline
   7 SNA Peer 8 BSC 3270
   9 SNA 3270 10 Finance

3.0 General Subsystem Parameters

1. Location name: (8 characters) _ _ _ _ _ _ _ _
2. Subsystem queue space: (0-40 K) _ _
3. Subsystem support swappable: (0-No 1-Yes) _
4. Maximum user record length: (1-4075) _ _ _ _

3.1 SDLC General Subsystem Parameters

1. SDLC protocol (1-Primary 2-Secondary) _
2. SDLC receive buffer size (2 or 4 K) _
3. SDLC transmit buffer size (2 or 4 K) _
4. Maximum receive pacing count (1-63) _

SNA Peer Subsystem Planning Chart

4.0 Line Information for SSP-ICF Subsystem

1. Line type: 1-Multipoint _
   2-Nonswitched Pt-Pt
   3-Switched Pt-Pt
2. Local station address: (2 hex) _ _
3. Switch type: _
   1 Manual call 2 Auto answer
   3 Manual answer
4. Auto disconnect: (0-No 1-Yes) _
5. Stay operational: (0-No 1-Yes) _

13.0 SNA Peer Subsystem Parameters

1. Remote station address: (01 to FE hexadecimal) _ _
2. Remote location name: _ _ _ _ _ _ _ _
3. Maximum number of active sessions: (1 - 64) _ _
4. Number of preestablished sessions: _ _
5. Maximum number of I-frames: (1 - 7) _
6. Location activated: (0-No 1-Yes) _
7. Slow poll: (0 - 5) _
8. Phone list name: _ _ _ _ _ _ _ _

SNA Peer Subsystem Planning Chart

13.0 SNA Peer Subsystem Parameters

1. Remote station address: (01 to FE hexadecimal) _ _
2. Remote location name: _ _ _ _ _ _ _ _
3. Maximum number of active sessions: (1–64) _ _
4. Number of preestablished sessions: _ _
5. Maximum number of I-frames: (1–7) _
6. Location activated: (0-No 1-Yes) _
7. Slow poll: (0–5) _
8. Phone list name: _ _ _ _ _ _ _ _

13.0 SNA Peer Subsystem Parameters

1. Remote station address: (01 to FE hexadecimal) _ _
2. Remote location name: _ _ _ _ _ _ _ _
3. Maximum number of active sessions: (1–64) _ _
4. Number of preestablished sessions: _ _
5. Maximum number of I-frames: (1–7) _
6. Location activated: (0-No 1-Yes) _
7. Slow poll: (0–5) _
8. Phone list name: _ _ _ _ _ _ _ _

13.0 SNA Peer Subsystem Parameters

1. Remote station address: (01 to FE hexadecimal) _ _
2. Remote location name: _ _ _ _ _ _ _ _
3. Maximum number of active sessions: (1–64) _ _
4. Number of preestablished sessions: _ _
5. Maximum number of I-frames: (1–7) _
6. Location activated: (0-No 1-Yes) _
7. Slow poll: (0–5) _
8. Phone list name: _ _ _ _ _ _ _ _

13.0 SNA Peer Subsystem Parameters

1. Remote station address: (01 to FE hexadecimal) _ _
2. Remote location name: _ _ _ _ _ _ _ _
3. Maximum number of active sessions: (1–64) _ _
4. Number of preestablished sessions: _ _
5. Maximum number of I-frames: (1–7) _
6. Location activated: (0-No 1-Yes) _
7. Slow poll: (0–5) _
8. Phone list name: _ _ _ _ _ _ _ _

SNA Peer Subsystem Planning Chart

13.0 SNA Peer Subsystem Parameters

1. Remote station address: (01 to FE hexadecimal) _ _
2. Remote location name: _ _ _ _ _ _ _ _ _ _
3. Maximum number of active sessions: (1–64) _ _
4. Number of preestablished sessions: _ _
5. Maximum number of I-frames: (1–7) _
6. Location activated: (0-No 1-Yes) _
7. Slow poll: (0–5) _
8. Phone list name: _ _ _ _ _ _ _ _ _ _

13.0 SNA Peer Subsystem Parameters

1. Remote station address: (01 to FE hexadecimal) _ _
2. Remote location name: _ _ _ _ _ _ _ _ _ _
3. Maximum number of active sessions: (1–64) _ _
4. Number of preestablished sessions: _ _
5. Maximum number of I-frames: (1–7) _
6. Location activated: (0-No 1-Yes) _
7. Slow poll: (0–5) _
8. Phone list name: _ _ _ _ _ _ _ _ _ _

13.0 SNA Peer Subsystem Parameters

1. Remote station address: (01 to FE hexadecimal) _ _
2. Remote location name: _ _ _ _ _ _ _ _ _ _
3. Maximum number of active sessions: (1–64) _ _
4. Number of preestablished sessions: _ _
5. Maximum number of I-frames: (1–7) _
6. Location activated: (0-No 1-Yes) _
7. Slow poll: (0–5) _
8. Phone list name: _ _ _ _ _ _ _ _ _ _

13.0 SNA Peer Subsystem Parameters

1. Remote station address: (01 to FE hexadecimal) _ _
2. Remote location name: _ _ _ _ _ _ _ _ _ _
3. Maximum number of active sessions: (1–64) _ _
4. Number of preestablished sessions: _ _
5. Maximum number of I-frames: (1–7) _
6. Location activated: (0-No 1-Yes) _
7. Slow poll: (0–5) _
8. Phone list name: _ _ _ _ _ _ _ _ _ _

SNA Upline Subsystem Planning Chart

1.0 Subsystem Member Configuration

1. Subsystem configuration member name (8 characters) _ _ _ _ _ _ _ _
2. Subsystem library name (8 characters) _ _ _ _ _ _ _ _

Select:
1. Create new member 4. Delete a member
2. Edit existing member 5. Review a member
3. Create new member from existing member

3. Enter selection: ____
4. Existing member name: _ _ _ _ _ _ _ _
5. Existing member library name: _ _ _ _ _ _ _ _

2.0 Common SSP-ICF Parameters for Each Subsystem

1. SSP-ICF common queue space: (2 - 42 K) _ _
2. Define the subsystem type: _ 6

1 Intra 2 BSC IMS/IRSS
3 BSCEL 4 BSC CICS
5 BSC CCP 6 SNA Upline
7 SNA Peer 8 BSC 3270
9 SNA 3270 10 Finance

3.0 General Subsystem Parameters

1. Location name: (8 characters) _ _ _ _ _ _ _ _
2. Subsystem queue space: (0-40 K) _ _
3. Subsystem support swappable: (0-No 1-Yes) _
4. Maximum user record length: (1-4075) _ _ _ _

4.0 Line Information for SSP-ICF Subsystem

1. Line type: _

2-Nonswitched Pt-Pt
3-Switched Pt-Pt

2. Local station address: (2 hex) _ _
3. Switch type: _

1 Manual call 2 Auto answer
3 Manual answer

7.0 Subsystem Inactive Destination Messages

1. Subsystem procedure name: (8 characters) _ _ _ _ _ _ _ _
2. Subsystem procedure library name: (8 characters) _ _ _ _ _ _ _ _

SNA Upline Subsystem Planning Chart

8.0 SNA General Subsystem Parameters

1. SDLC Buffer pool size: (2-8 K) _
2. Number of transmit buffers: (1-15) _ _
3. Maximum number of sessions: (1-32) _ _
4. Maximum receive pacing count: (1-63) _ _
5. Local ID: (Hexadecimal) _ _ _ _ _ _
6. LU configuration library name: _ _ _ _ _ _ _ _
7. LU configuration member name: _ _ _ _ _ _ _ _

9.0 SNA Upline Subsystem Parameters

1. Subsystem application ID: _ _ _ _ _ _ _ _
2. Subsystem host name: (1-Other 2-IMS/VS 3-CICS/VS) _

SNA Upline Subsystem Planning Chart

9.1 SNA Upline/3270 Station Parameters

1. Remote location name: _ _ _ _ _ _ _ _
2. SSCPID: (0–65535) _ _ _ _ _
3. Phone list name: _ _ _ _ _ _ _ _
4. Location activated: (0-No 1-Yes) _

9.1 SNA Upline/3270 Station Parameters

1. Remote location name: _ _ _ _ _ _ _ _
2. SSCPID: (0–65535) _ _ _ _ _
3. Phone list name: _ _ _ _ _ _ _ _
4. Location activated: (0-No 1-Yes) _

9.1 SNA Upline/3270 Station Parameters

1. Remote location name: _ _ _ _ _ _ _ _
2. SSCPID: (0–65535) _ _ _ _ _
3. Phone list name: _ _ _ _ _ _ _ _
4. Location activated: (0-No 1-Yes) _

9.1 SNA Upline/3270 Station Parameters

1. Remote location name: _ _ _ _ _ _ _ _
2. SSCPID: (0–65535) _ _ _ _ _
3. Phone list name: _ _ _ _ _ _ _ _
4. Location activated: (0-No 1-Yes) _

SNA Upline Subsystem Planning Chart

9.1 SNA Upline/3270 Station Parameters

1. Remote location name: _ _ _ _ _ _ _ _ _ _
2. SSCPID: (0–65535) _ _ _ _ _ _
3. Phone list name: _ _ _ _ _ _ _ _ _ _
4. Location activated: (0-No 1-Yes) _

9.1 SNA Upline/3270 Station Parameters

1. Remote location name: _ _ _ _ _ _ _ _ _ _
2. SSCPID: (0–65535) _ _ _ _ _ _
3. Phone list name: _ _ _ _ _ _ _ _ _ _
4. Location activated: (0-No 1-Yes) _

9.1 SNA Upline/3270 Station Parameters

1. Remote location name: _ _ _ _ _ _ _ _ _ _
2. SSCPID: (0–65535) _ _ _ _ _ _
3. Phone list name: _ _ _ _ _ _ _ _ _ _
4. Location activated: (0-No 1-Yes) _

9.1 SNA Upline/3270 Station Parameters

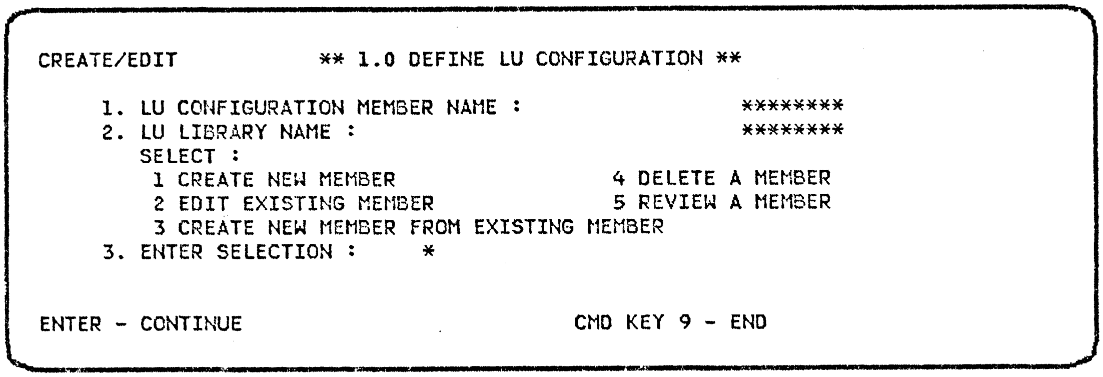
1. Remote location name: _ _ _ _ _ _ _ _ _ _
2. SSCPID: (0–65535) _ _ _ _ _ _
3. Phone list name: _ _ _ _ _ _ _ _ _ _
4. Location activated: (0-No 1-Yes) _

Appendix G. The DEFINELU Procedure

The SNUF subsystem and SNA 3270 device emulation can share a communications line from the System/34 to a System/370. To share the line, the logical unit addresses must be assigned to the subsystem. The DEFINELU procedure is used to assign the logical unit addresses.

The DEFINELU procedure produces a configuration table stored as a member in a library. The member name and library name are then specified during CNFIGICF for the SNUF subsystem or SNA 3270 device emulation.

To run the DEFINELU procedure, enter the procedure command DEFINELU. The following option display appears:



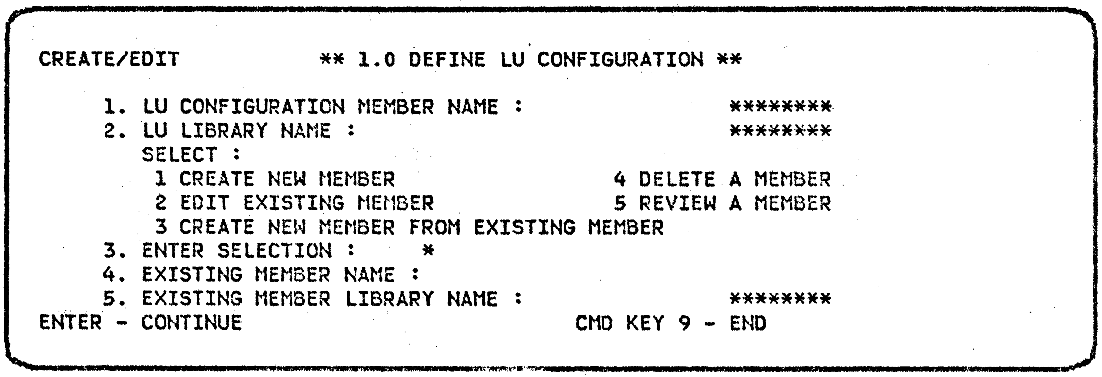
```
CREATE/EDIT          ** 1.0 DEFINE LU CONFIGURATION **

     1. LU CONFIGURATION MEMBER NAME :                   ********
     2. LU LIBRARY NAME :                                ********
        SELECT :
         1 CREATE NEW MEMBER                     4 DELETE A MEMBER
         2 EDIT EXISTING MEMBER                  5 REVIEW A MEMBER
         3 CREATE NEW MEMBER FROM EXISTING MEMBER
     3. ENTER SELECTION :      *

ENTER - CONTINUE                          CMD KEY 9 - END
```

LU configuration member name: Specify a name for this configuration of logical unit addresses. This name is used to store the member in a library and is referenced in the CNFIGICF prompts for the SNUF and SNA 3270 subsystems.

LU library name: Specify the name of a library in which the configuration is stored or is to be stored. The default is #LIBRARY, however, you should probably store the configuration in a user library.

Enter selection: Specify one of the five options: (1) create a new member, (2) edit an existing member, (3) create a new member from an existing member, (4) delete a member, or (5) review a member without changing it.

If you select option 3, the display reappears requesting two additional items:

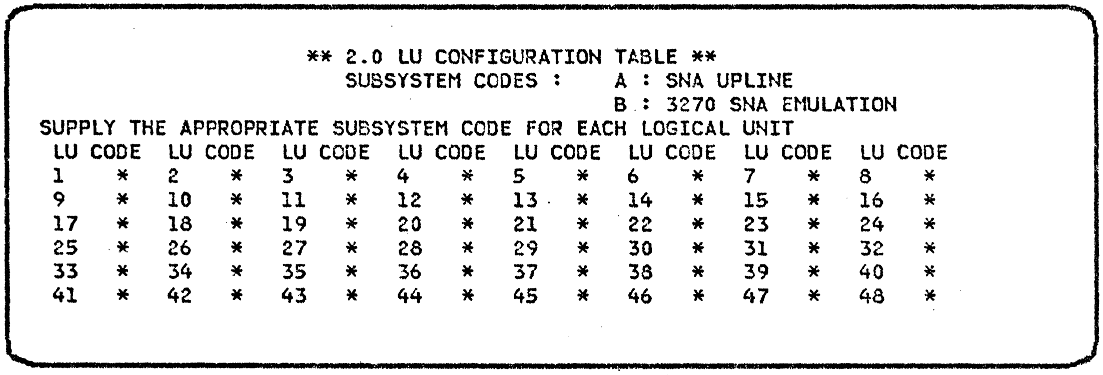


```
CREATE/EDIT           ** 1.0 DEFINE LU CONFIGURATION **

     1. LU CONFIGURATION MEMBER NAME :               ********
     2. LU LIBRARY NAME :                            ********
        SELECT :
         1 CREATE NEW MEMBER                   4 DELETE A MEMBER
         2 EDIT EXISTING MEMBER                5 REVIEW A MEMBER
         3 CREATE NEW MEMBER FROM EXISTING MEMBER
     3. ENTER SELECTION :    *
     4. EXISTING MEMBER NAME :
     5. EXISTING MEMBER LIBRARY NAME :               ********
ENTER - CONTINUE                          CMD KEY 9 - END
```

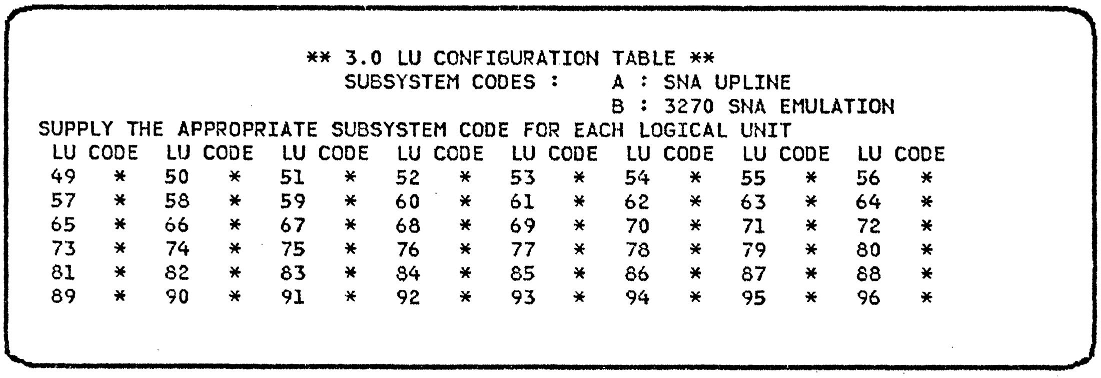
Existing member name: Specify the name of the existing logical unit configuration member that is to be used to create the new member. The existing member remains unchanged.

Existing member library name: Specifies the name of the library where the existing member resides.

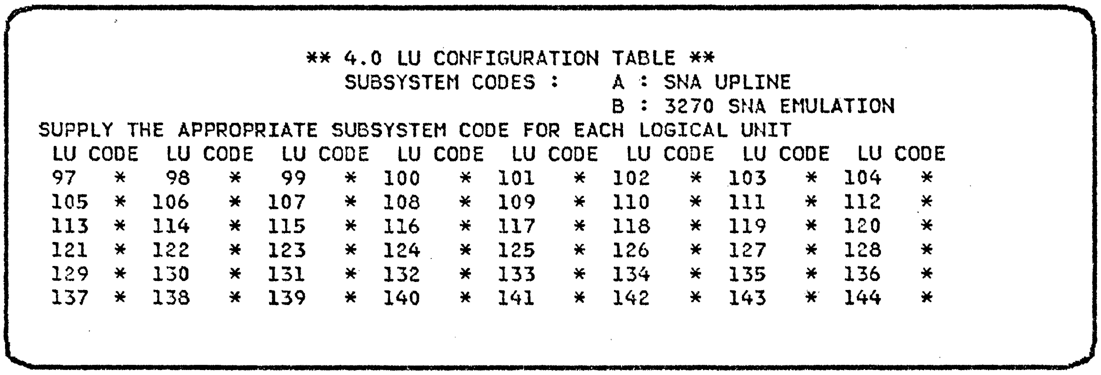
After you have entered all required information from display 1.0, the following sequence of displays occurs.



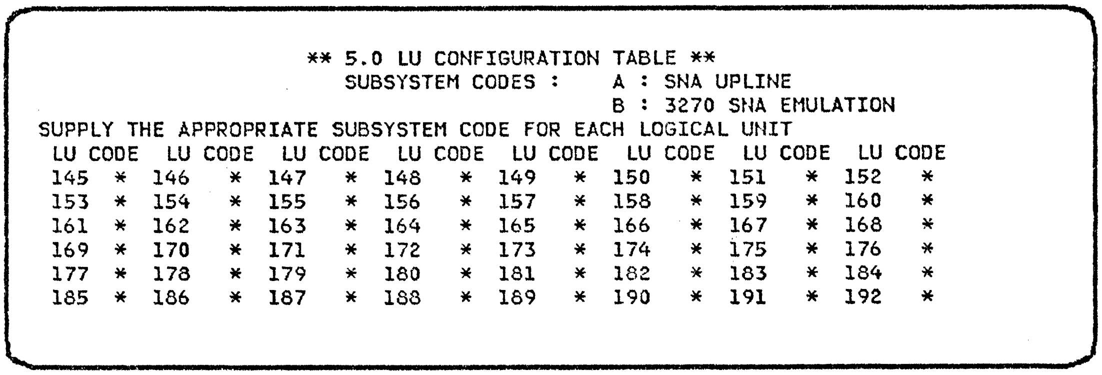
```
                     ** 2.0 LU CONFIGURATION TABLE **
                        SUBSYSTEM CODES :    A : SNA UPLINE
                                             B : 3270 SNA EMULATION
SUPPLY THE APPROPRIATE SUBSYSTEM CODE FOR EACH LOGICAL UNIT
LU CODE  LU CODE  LU CODE  LU CODE  LU CODE  LU CODE  LU CODE  LU CODE
 1    *   2    *   3    *   4    *   5    *   6    *   7    *   8    *
 9    *   10   *   11   *   12   *   13   *   14   *   15   *   16   *
 17   *   18   *   19   *   20   *   21   *   22   *   23   *   24   *
 25   *   26   *   27   *   28   *   29   *   30   *   31   *   32   *
 33   *   34   *   35   *   36   *   37   *   38   *   39   *   40   *
 41   *   42   *   43   *   44   *   45   *   46   *   47   *   48   *
```



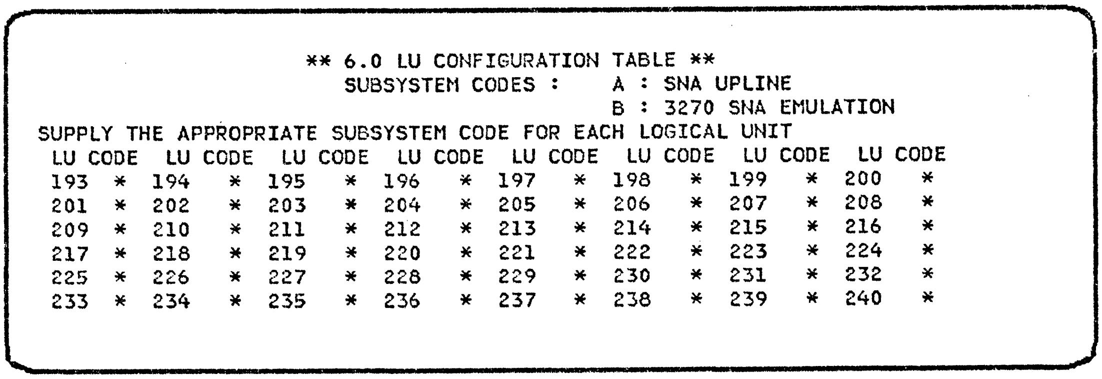
```
                      ** 3.0 LU CONFIGURATION TABLE **
                         SUBSYSTEM CODES :      A : SNA UPLINE
                                                B : 3270 SNA EMULATION
SUPPLY THE APPROPRIATE SUBSYSTEM CODE FOR EACH LOGICAL UNIT
 LU CODE  LU CODE  LU CODE  LU CODE  LU CODE  LU CODE  LU CODE  LU CODE
 49   *   50   *   51   *   52   *   53   *   54   *   55   *   56   *
 57   *   58   *   59   *   60   *   61   *   62   *   63   *   64   *
 65   *   66   *   67   *   68   *   69   *   70   *   71   *   72   *
 73   *   74   *   75   *   76   *   77   *   78   *   79   *   80   *
 81   *   82   *   83   *   84   *   85   *   86   *   87   *   88   *
 89   *   90   *   91   *   92   *   93   *   94   *   95   *   96   *
```



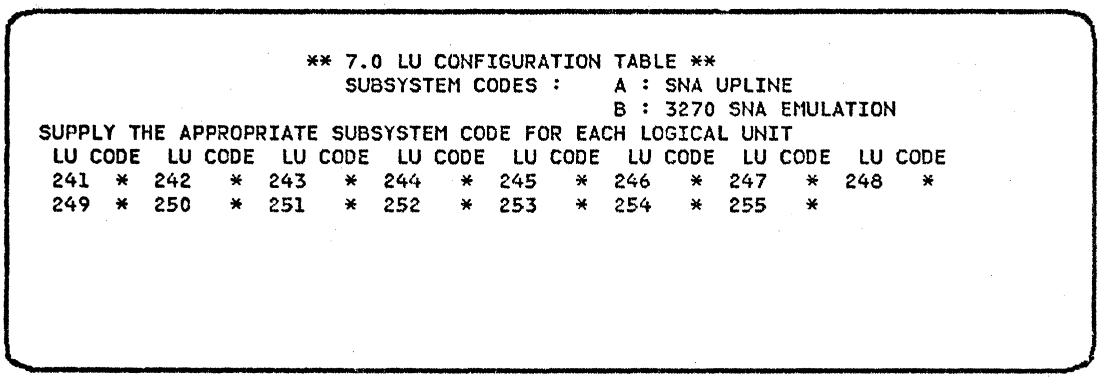
```
                      ** 4.0 LU CONFIGURATION TABLE **
                         SUBSYSTEM CODES :      A : SNA UPLINE
                                                B : 3270 SNA EMULATION
SUPPLY THE APPROPRIATE SUBSYSTEM CODE FOR EACH LOGICAL UNIT
 LU CODE  LU CODE  LU CODE  LU CODE  LU CODE  LU CODE  LU CODE  LU CODE
 97   *    98  *    99  *   100  *   101  *   102  *   103  *   104   *
 105  *   106  *   107  *   108  *   109  *   110  *   111  *   112   *
 113  *   114  *   115  *   116  *   117  *   118  *   119  *   120   *
 121  *   122  *   123  *   124  *   125  *   126  *   127  *   128   *
 129  *   130  *   131  *   132  *   133  *   134  *   135  *   136   *
 137  *   138  *   139  *   140  *   141  *   142  *   143  *   144   *
```



```
                      ** 5.0 LU CONFIGURATION TABLE **
                         SUBSYSTEM CODES :      A : SNA UPLINE
                                                B : 3270 SNA EMULATION
SUPPLY THE APPROPRIATE SUBSYSTEM CODE FOR EACH LOGICAL UNIT
 LU CODE  LU CODE  LU CODE  LU CODE  LU CODE  LU CODE  LU CODE  LU CODE
 145  *   146  *   147  *   148  *   149  *   150  *   151  *   152   *
 153  *   154  *   155  *   156  *   157  *   158  *   159  *   160   *
 161  *   162  *   163  *   164  *   165  *   166  *   167  *   168   *
 169  *   170  *   171  *   172  *   173  *   174  *   175  *   176   *
 177  *   178  *   179  *   180  *   181  *   182  *   183  *   184   *
 185  *   186  *   187  *   188  *   189  *   190  *   191  *   192   *
```



```
                      ** 6.0 LU CONFIGURATION TABLE **
                         SUBSYSTEM CODES :      A : SNA UPLINE
                                                B : 3270 SNA EMULATION
SUPPLY THE APPROPRIATE SUBSYSTEM CODE FOR EACH LOGICAL UNIT
 LU CODE  LU CODE  LU CODE  LU CODE  LU CODE  LU CODE  LU CODE  LU CODE
 193  *   194  *   195  *   196  *   197  *   198  *   199  *   200   *
 201  *   202  *   203  *   204  *   205  *   206  *   207  *   208   *
 209  *   210  *   211  *   212  *   213  *   214  *   215  *   216   *
 217  *   218  *   219  *   220  *   221  *   222  *   223  *   224   *
 225  *   226  *   227  *   228  *   229  *   230  *   231  *   232   *
 233  *   234  *   235  *   236  *   237  *   238  *   239  *   240   *
```



```
                        ** 7.0 LU CONFIGURATION TABLE **
                           SUBSYSTEM CODES :     A : SNA UPLINE
                                                 B : 3270 SNA EMULATION
 SUPPLY THE APPROPRIATE SUBSYSTEM CODE FOR EACH LOGICAL UNIT
  LU CODE  LU CODE  LU CODE  LU CODE  LU CODE  LU CODE  LU CODE  LU CODE
  241  *  242   *  243   *  244   *  245   *  246   *  247   *  248   *
  249  *  250   *  251   *  252   *  253   *  254   *  255   *
```

On each of these displays, enter the code for the subsystem that you want assigned to each logical unit address. Note that logical unit addresses 1 and 2 are used by the SNUF subsystem for program start sessions, therefore, they should be assigned to the SNUF subsystem.

After pressing the Enter/Rec Adv key for display 7.0, display 1.0 reappears. You can then create, change, delete, or review another configuration, or you can press command key 9 to end the DEFINELU procedure.

Glossary

accept: An operation that allows the issuing program to receive data from any previously invited session or work station, to allow new requesters, or to check if the timer has expired.

acquire: An operation that establishes a session between the issuing application program and the remote subsystem.

address pool: A collection of identifiers used to distinguish sessions on a multipoint line.

BSC: Binary synchronous communications

BSCEL: Binary synchronous communications equivalence link

BTAM: Basic telecommunications access method

cancel: An operation that sends a command to the remote system indicating to abnormally end and disregard previous records in this chain of data.

CCP: Communication Control Program

chain: A group of records logically linked, by SNA indicators, as a unit.

CICS: Customer Information Control System

disable: A function that terminates a subsystem.

DOS: Disk Operating System

enable: A function that starts a subsystem.

EP: Emulation program

evoke: An operation that starts an application program on the remote system.

function management header: A special record or portion of a record that contains control information for the data that is to follow.

IMS: Information Management System

invite: An operation that asks for input data from a display station or a session.

IRSS: Intelligent Remote Station Support

K: 1024 bytes

MLMP: Multiline/multipoint

MRT: Multiple requester terminal program

NCP: Network control program

NEP: Never ending program

negative response: A reply that indicates data was not received correctly or a command was incorrect or unacceptable.

OS: Operating System

partner: The remote application program or the remote system.

PEP: Partitioned emulation program

polling: A method of checking station(s) on a point-to-point or multipoint line to see if they have anything to send.

release: An operation that attempts to end a session or pass a session to the next step in the procedure.

requestor: The partner in a session that started (evoked) the other partner.

SDLC: Synchronous data link control

session: The communication between an application program and the remote subsystem.

SNA: Systems network architecture

SNUF: SNA Upline facility

SRT: Single requester terminal program

subsystem: A portion of the interactive communications feature that handles the requirements of the remote system, thereby isolating most system-dependent considerations from the application program.

TCAM: Telecommunications access method

transaction: The communication between the application program and a specific entity (usually another application program) at the remote system.

VS: Virtual storage

VTAM: Virtual telecommunications access method

Index

Please use this form only to identify publication errors or request changes to publications. Technical questions about IBM systems, changes in IBM programming support, requests for additional publications, etc, should be directed to your IBM representative or to the IBM branch office nearest your location.

Error in publication (typographical, illustration, and so on). **No reply.**

Page Number *Error*

Inaccurate or misleading information in this publication. Please tell us about it by using this postage-paid form. We will correct or clarify the publication, or tell you why a change is not being made, provided you include your name and address.

Page Number *Comment*

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Name ________________________________

Address ______________________________

____

● **No postage necessary if mailed in the U.S.A.**

IBM System/34
Interactive Communications Feature
Reference Manual

SC21-7751-3

SC21-7751-3

— Cut Along Line —

IBM System/34 Interactive Communications Feature Reference Manual (File No. S34-36) Printed in U.S.A. SC21-7751-3

Fold and tape   Please do not staple   Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS   PERMIT NO. 40   ARMONK, N. Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM CORPORATION
General Systems Division
Development Laboratory
Publications, Dept. 245
Rochester, Minnesota 55901

Fold and tape   Please do not staple   Fold and tape

IBM

International Business Machines Corporation

Please use this form only to identify publication errors or request changes to publications. Technical questions about IBM systems, changes in IBM programming support, requests for additional publications, etc, should be directed to your IBM representative or to the IBM branch office nearest your location.

Error in publication (typographical, illustration, and so on). **No reply.**

Page Number *Error*

Inaccurate or misleading information in this publication. Please tell us about it by using this postage-paid form. We will correct or clarify the publication, or tell you why a change is not being made, provided you include your name and address.

Page Number *Comment*

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

● **No postage necessary if mailed in the U.S.A.**

Name ____

Address ______________________________________

____

IBM System/34
Interactive Communications Feature
Reference Manual

SC21-7751-3

Cut Along Line

Fold and tape

Please do not staple

Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N. Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM CORPORATION
General Systems Division
Development Laboratory
Publications, Dept. 245
Rochester, Minnesota 55901

Fold and tape

Please do not staple

Fold and tape

IBM

International Business Machines Corporation

IBM System/34 Interactive Communications Feature Reference Manual (File No. S34-36) Printed in U.S.A. SC21-7751-3

Please use this form only to identify publication errors or request changes to publications. Technical questions about IBM systems, changes in IBM programming support, requests for additional publications, etc, should be directed to your IBM representative or to the IBM branch office nearest your location.

Error in publication (typographical, illustration, and so on). **No reply.**

Page Number *Error*

Inaccurate or misleading information in this publication. Please tell us about it by using this postage-paid form. We will correct or clarify the publication, or tell you why a change is not being made, provided you include your name and address.

Page Number *Comment*

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

- **No postage necessary if mailed in the U.S.A.**

Name ______________________________

Address ______________________________

____

IBM System/34
Interactive Communications Feature
Reference Manual

SC21-7751-3

Cut Along Line

Fold and tape | Please do not staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N. Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM CORPORATION
General Systems Division
Development Laboratory
Publications, Dept. 245
Rochester, Minnesota 55901

Fold and tape | Please do not staple | Fold and tape

IBM

International Business Machines Corporation

IBM

Technical Newsletter

This Newsletter No. SN21-8294

Date 8 October 1982

Base Publication No. SC21-7751-3

File No. S34-36

Previous Newsletters None

IBM System/34
Interactive Communications Feature
Reference Manual

This technical newsletter applies to Release 8 of the IBM System/34 System Support Program Product (Program 5726-SS1) and provides replacement pages for the subject publication. These replacement pages remain in effect for subsequent releases and modifications unless specifically altered. Pages to be inserted and/or removed are:

8-29, 8-30
8-59, 8-60

Changes to text and illustrations are indicated by a vertical line at the left of the change.

Summary of Amendments

- Receiving null records in the BSCEL subsystem

Note: Please file this cover letter at the back of the manual to provide a record of changes.

IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901

Printed in U.S.A.

IBM®

International Business Machines Corporation

IBM System/34 Interactive Communications Feature Reference Manual (File No. S34-36) Printed in U.S.A. SC21-7751-3