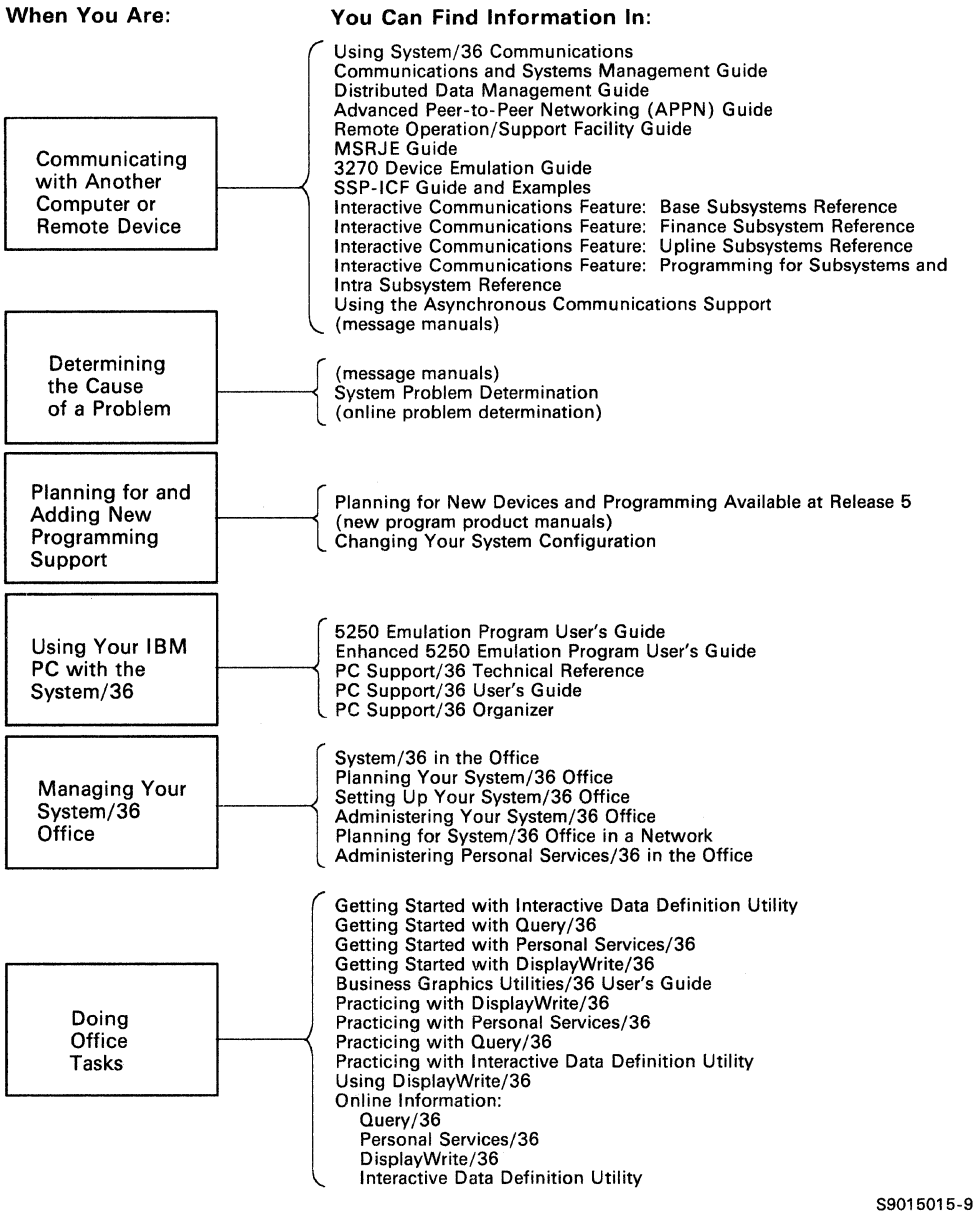
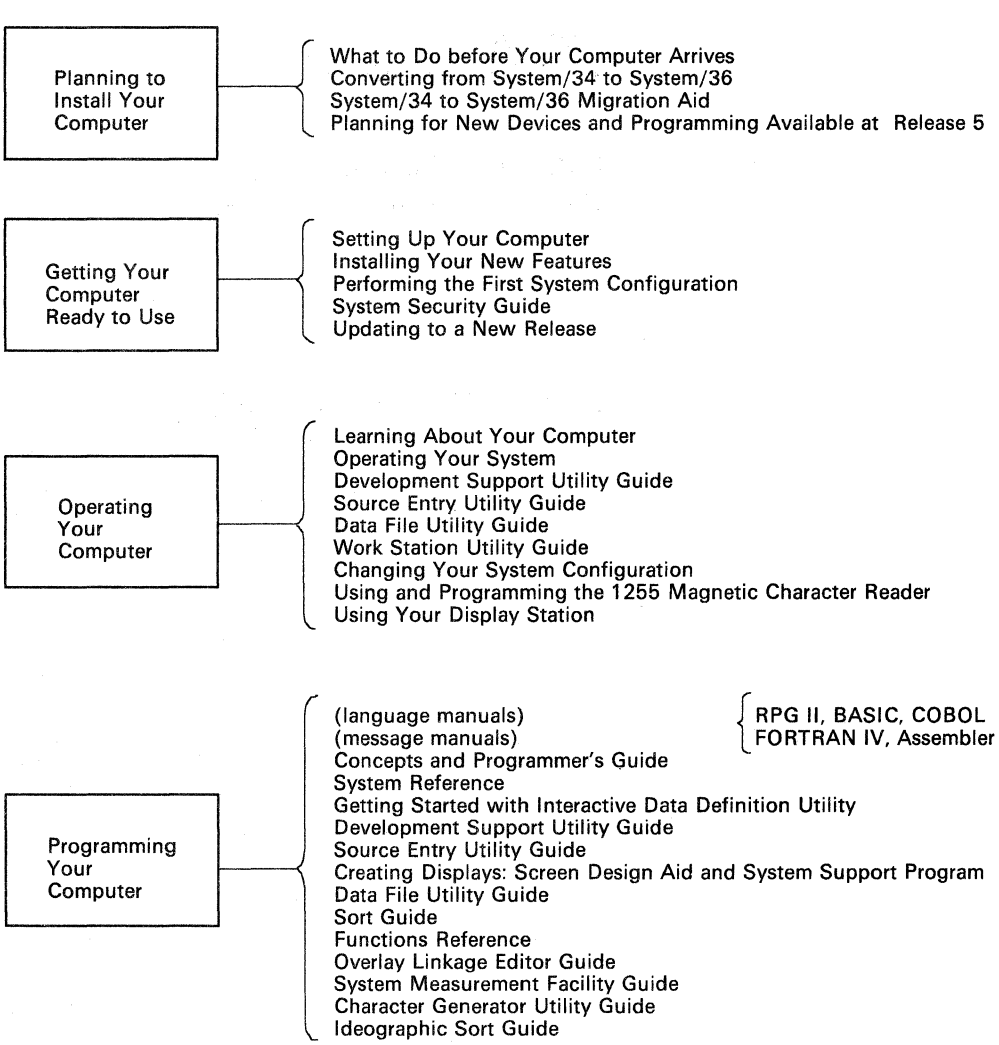
IBM System/36

# Using the Asynchronous Communications Support


When You Are:
You Can Find Information In:
Communicating with Another Computer or Remote Device
Using System/36 Communications
Communications and Systems Management Guide
Distributed Data Management Guide
Advanced Peer-to-Peer Networking (APPN) Guide
Remote Operation/Support Facility Guide
MSRJE Guide
3270 Device Emulation Guide
SSP-ICF Guide and Examples
Interactive Communications Feature: Base Subsystems Reference
Interactive Communications Feature: Finance Subsystem Reference
Interactive Communications Feature: Upline Subsystems Reference
Interactive Communications Feature: Programming for Subsystems and Intra Subsystem Reference
Using the Asynchronous Communications Support (message manuals)
Determining the Cause of a Problem
(message manuals)
System Problem Determination (online problem determination)
Planning for and Adding New Programming Support
Planning for New Devices and Programming Available at Release 5 (new program product manuals)
Changing Your System Configuration
Using Your IBM PC with the System/36
5250 Emulation Program User's Guide
Enhanced 5250 Emulation Program User's Guide
PC Support/36 Technical Reference
PC Support/36 User's Guide
PC Support/36 Organizer
Managing Your System/36 Office
System/36 in the Office
Planning Your System/36 Office
Setting Up Your System/36 Office
Administering Your System/36 Office
Planning for System/36 Office in a Network
Administering Personal Services/36 in the Office
Doing Office Tasks
Getting Started with Interactive Data Definition Utility
Getting Started with Query/36
Getting Started with Personal Services/36
Getting Started with DisplayWrite/36
Business Graphics Utilities/36 User's Guide
Practicing with DisplayWrite/36
Practicing with Personal Services/36
Practicing with Query/36
Practicing with Interactive Data Definition Utility
Using DisplayWrite/36
Online Information:
Query/36
Personal Services/36
DisplayWrite/36
Interactive Data Definition Utility
S9015015-9


When You Are:
You Can Find Information In:
Planning to Install Your Computer
What to Do before Your Computer Arrives
Converting from System/34 to System/36
System/34 to System/36 Migration Aid
Planning for New Devices and Programming Available at Release 5
Getting Your Computer Ready to Use
Setting Up Your Computer
Installing Your New Features
Performing the First System Configuration
System Security Guide
Updating to a New Release
Operating Your Computer
Learning About Your Computer
Operating Your System
Development Support Utility Guide
Source Entry Utility Guide
Data File Utility Guide
Work Station Utility Guide
Changing Your System Configuration
Using and Programming the 1255 Magnetic Character Reader
Using Your Display Station
Programming Your Computer
(language manuals)
RPG II, BASIC, COBOL
(message manuals)
FORTRAN IV, Assembler
Concepts and Programmer's Guide
System Reference
Getting Started with Interactive Data Definition Utility
Development Support Utility Guide
Source Entry Utility Guide
Creating Displays: Screen Design Aid and System Support Program
Data File Utility Guide
Sort Guide
Functions Reference
Overlay Linkage Editor Guide
System Measurement Facility Guide
Character Generator Utility Guide
Ideographic Sort Guide

S9015001-9

IBM System/36

# Using the Asynchronous Communications Support

Program Numbers: 5727-SS1
5727-SS6

File Number
S36-38

Order Number
SC21-9143-2

**Third Edition (June 1987)**

This major revision makes obsolete SC21-9143-1. See "About This Manual" for a summary of major changes to this edition. Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change or addition.

This edition applies to Release 5, Modification Level 1, of IBM System/36 System Support Program Product (Program 5727-SS1 for the 5360 and 5362 System Units and Program 5727-SS6 for the 5364 System Unit), and to all subsequent releases and modifications until otherwise indicated. Changes are periodically made to the information herein; any such changes will be reported in subsequent revisions or Technical Newsletters.

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

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

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

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

# Contents

# About This Manual

This manual contains information for using the IBM System/36 asynchronous communications support. It is intended primarily for application and system programmers. This manual is also intended for the System/36 user who needs information about how to operate the interactive terminal facility (ITF).

This manual covers the following topics:

- *Configuration:* The procedures needed to describe the asynchronous communications subsystem to the system.
- *The asynchronous communications subsystem:* The commands needed to enable and disable the subsystem, the operation of the subsystem, and the return codes sent by the subsystem.
- *The interactive terminal facility (ITF):* A description of how to start and run ITF.

Information not covered in this manual can be found in the manuals listed in the section "If You Need More Information."

*Note: Throughout this manual, the term* ***remote system*** *refers to the system or device with which System/36 is communicating.*

# What you should know . . .

Before you use this manual, you should know or have the following information for writing application programs using the asynchronous communications support:

- You should be familiar with System/36 programming terminology, particularly work station programming, and you should be able to program in whatever language you intend to use.
- You should know the concepts of data communications as described in the manual *Data Communications Concepts*, GC21-5169.
- You should be familiar with interactive communications concepts as described in Chapters 1 through 5 of the System/36 manual *Interactive Communications Feature: Guide and Examples*, SC21-7911. The *SSP-ICF Guide and Examples* manual introduces SSP-ICF concepts.
- You should use the workbook *Planning for Data Communications*, SA21-9441 for the 5360 System Unit, SA21-9482 for the 5362 System Unit, or SA21-9844 for the 5364 System Unit.

# If you need more information . . .

The following System/36 manuals contain additional information you may need when you use the System/36 asynchronous communications support:

- *SSP-ICF Programming for Communications Subsystems and Intra Communications Subsystem Reference*, SC21-9533 contains general information about SSP-ICF and detailed information about the Intra subsystem. It describes the operations, the OCL statements, and all the return codes for the Intra subsystem. It also contains examples of programming in Assembler, COBOL, BASIC, and RPG II.

- *Using System/36 Communications*, SC21-9082 contains information about:

  - Configuring communications, including the X.25 programming support

  - File transfer subroutines and file transfer subroutine messages

- *Changing Your System Configuration*, SC21-9052 contains instructions for installing communications support.

- *System Security Guide*, SC21-9042 describes how to implement various levels of security on System/36.

- *System Problem Determination*, SC21-7919 for the 5360 System Unit, SC21-9063 for the 5362 System Unit, or SC21-9375 for the 5364 System Unit provides procedures to help you find the cause of communications problems.

- *System Messages*, SC21-7938 describes the system messages that are displayed when you run programs that use the interactive communications features.

- *System Reference*, SC21-9020 describes the OCL statements, system utilities, and system procedures you need when you use System/36.

- *Functions Reference Manual*, SA21-9436 describes the machine instructions, status bytes, and other information needed to understand system programs from the hardware viewpoint.

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

- *Programming with Assembler*, SC21-7908

- *Programming with BASIC*, SC21-9003

- *Programming with COBOL*, SC21-9007

- *Programming with RPG II*, SC21-9006

# How this manual has changed . . .

The following information has been added to this manual since the previous edition:

- X.25 line support on the 5364 System Unit, Chapter 1.
- Changes to the "Programming Considerations," Chapter 3.
- Changes to the DATAL parameter, Chapter 3.
- Carriage return character, Chapter 4.
- Change to the connect command, Appendix A.
- Miscellaneous technical changes have also been made.

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

# Chapter 1. Introduction to the Asynchronous Communications Support

The IBM System/36 asynchronous communications support, part of the base Communications feature, lets System/36 or a host system use the asynchronous communications support to communicate with a remote station, either directly or through a packet switched data network (PSDN).

The asynchronous communications support includes the following parts:

- The asynchronous communications subsystem
- The file transfer subroutines (also used by other subsystems)
- The interactive terminal facility (ITF)
- Support for one to three X.25 lines

The asynchronous communications subsystem provides program-to-program communications between systems using the asynchronous and/or enhanced X.25 data link protocols. It also provides an internal X.25 packet assembler/disassembler (PAD) function, which allows a terminal connected to System/36 to communicate with a host system through a PSDN.

The file transfer subroutines, called from your application program, let you send and receive System/36 data files and library members. See the manual *Using System/36 Communications* for more information.

ITF allows System/36 to connect to applications such as the TELEMAIL service of the GTE Telenet data network [1]. Using ITF, you can send or receive not only simple memos, but also System/36 library members, data files, and DisplayWrite/36 (DW/36) documents. See Chapter 4, "Using the Interactive Terminal Facility," for more information.

The asynchronous communications support includes support for up to three X.25 lines on the 5360 and 5362 System Units, and one X.25 line on a 5364 System Unit.

---

1 TELEMAIL and Telenet are registered servicemarks of the GTE Telenet Communications Corporation

# The Asynchronous Communications Subsystem

The asynchronous communications subsystem allows System/36 to communicate, using an asynchronous communications line, with another asynchronous location or with a PAD, which gives System/36 access to an X.25 PSDN. Using enhanced X.25 support on an X.25 line, the asynchronous communications subsystem can communicate directly through an X.25 network, or it can emulate (imitate) a PAD using CCITT recommendations X.3, X.28, and X.29. The asynchronous communications subsystem, when emulating a PAD, makes a terminal that is locally attached to System/36 appear to be connected to an X.25 PAD over an asynchronous data link. This function allows System/36 terminals to communicate with any X.25 host system that communicates with asynchronous terminals. See Appendix A, "PAD Emulation."

In addition, the asynchronous communications subsystem provides both interactive and batch communications interfaces between application programs. The programs can be written in System/36 Assembler, BASIC, COBOL, or RPG II.

When the asynchronous communications subsystem is in use:

- System/36 programs can initiate procedures on a remote system, and the remote system can initiate procedures on the local System/36. Security options for both systems are supported.

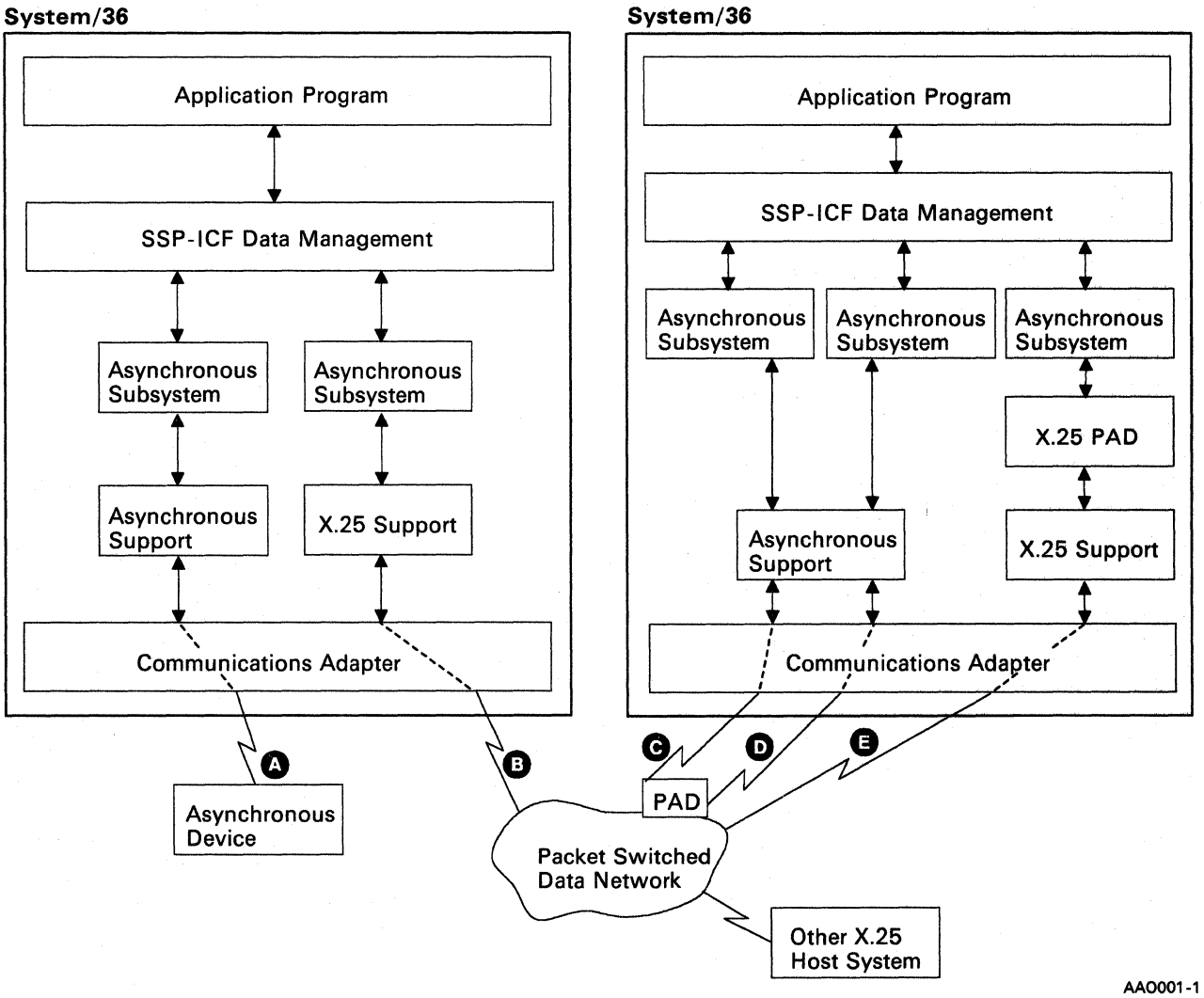
- System/36 can send and receive memos and electronic mail, including data files, library members, and DW/36 documents, using the interactive terminal facility (ITF) along with network or host system services. See Chapter 4, "Using the Interactive Terminal Facility."

- System/36 can send and receive data files and library members using the file transfer subroutines and program-to-program communications. See the manual *Using System/36 Communications* for more information.

- A switched or nonswitched point-to-point communications line can be used. For a subsystem using asynchronous communications, the physical line can be switched or nonswitched. For a subsystem using X.25 support to connect directly to the PSDN, the physical line is nonswitched, but the connection through the network to another system may be a permanent virtual circuit (PVC) or a switched virtual circuit (SVC). If you use the file transfer subroutines, you must have a SVC connection.

- Rotary dial, a function of the System/36 PAD support, can be used. See Appendix B, "Rotary Dial."

The following example shows sample configurations of the asynchronous communications support network using the asynchronous communications subsystem.



Line Ⓐ is a nonswitched communications line from System/36 to an asynchronous device. The asynchronous communications subsystem uses asynchronous communications support for this line. This communications link requires a System/36 application program to communicate with or provide services to the device; such an application program is not part of the asynchronous communications support.

Line Ⓑ is a nonswitched line to the PSDN. This asynchronous communications subsystem communicates through the PSDN directly, using X.25 support.

Lines Ⓒ and Ⓓ are switched asynchronous lines. These asynchronous communications subsystems use the asynchronous communications support; they make switched connections to a PAD, which adapts their data for transmission over the PSDN.

Line Ⓔ is a nonswitched line to the PSDN. The asynchronous communications subsystem connected to line Ⓔ emulates a PAD, using CCITT recommendations X.3, X.28, and X.29. This *internal PAD* interfaces with the X.25 support. The internal PAD lets a locally attached work station communicate through the PSDN, appearing to be an asynchronous terminal.

## Communications Lines Supported

System/36 can have as many as eight communications lines. Each asynchronous communications subsystem requires at least one communications line to communicate with a remote system; an asynchronous communications subsystem can support up to eight asynchronous lines at once. However, the maximum number of lines available is controlled by the communications adapter and the features installed on your system.

Your System/36, the 5360 and 5362 System Units, can have one of several communications adapters:

- The eight-line communications adapter (ELCA)
- The multiline communications adapter (MLCA)
- The single-line communications adapter (SLCA)

The following chart shows, by system unit and adapter, the number of communications lines available, and the number of asynchronous communications lines that can be used.

| **System Unit** | **5360 System Unit** | | | | **5362 System Unit** | |
|---|---|---|---|---|---|---|
| **Adapter** | **SLCA 2500** | **SLCA 2550** | **MLCA 4500** | **ELCA 4550** | **SLCA 2910** | **MLCA 2915** |
| Lines Available | 1 line | 1 line | 1 to 4 lines | 1 to 8 lines | 1 to 2 lines | 1 to 4 lines |
| Asynchronous Communications | N/A | 1 line | N/A | 8 lines | 2 lines | 4 lines |

Refer to the *Functions Reference Manual* for more detailed information on the communications adapters.

### 5360 System Unit

If a System/36 with the 5360 System Unit has the ELCA feature installed, up to three X.25 lines can be configured with the following conditions:

- If one X.25 line is configured, it can be any of the eight lines on the ELCA. The other communications lines are available for use by other protocols (such as asynchronous, BSC, or SDLC).
- If two X.25 lines are configured, they can be any two lines except for line 8, which is reserved. Therefore, only five lines are available for use by other protocols.
- If three X.25 lines are configured, they can be any three lines except for lines 7 and 8, which are reserved. This leaves three lines to be used by other protocols.

### 5362 System Unit

If a System/36 with the 5362 System Unit has the MLCA feature installed, asynchronous communications support allows up to three X.25 lines to be configured. These can be any of the four communications lines on the MLCA. No lines are reserved; the remaining lines are available for use by other protocols.

### 5364 System Unit

A System/36 with the 5364 System Unit emulates an MLCA. However, only two communications lines are available with the following restrictions:

- When a BSC or SDLC adapter is on the system, line 1 must be used for communications.
- When two Asynchronous adapters are on the system unit, both lines 1 and 2 can be used for asynchronous communications.
- When asynchronous communications is active, BSC, SDLC, or local area network (LAN) cannot be used.

If the 5364 System Unit has the IBM Realtime Interface Co-Processor installed, up to three communications lines are available with the following restrictions:

- Line 1 can only be used for asynchronous communications if an Asynchronous adapter is installed.
- Line 2 is not supported.
- The IBM Realtime Interface Co-Processor supports up to two SDLC lines or one X.25 line.
  - For SDLC, lines 3 and 4 can be used for SDLC communications.
  - For X.25, line 3 can be used for X.25 communications. If line 3 is configured as an X.25 line, line 4 is reserved and unavailable for use.

*Note: SDLC and X.25 are mutually exclusive.*

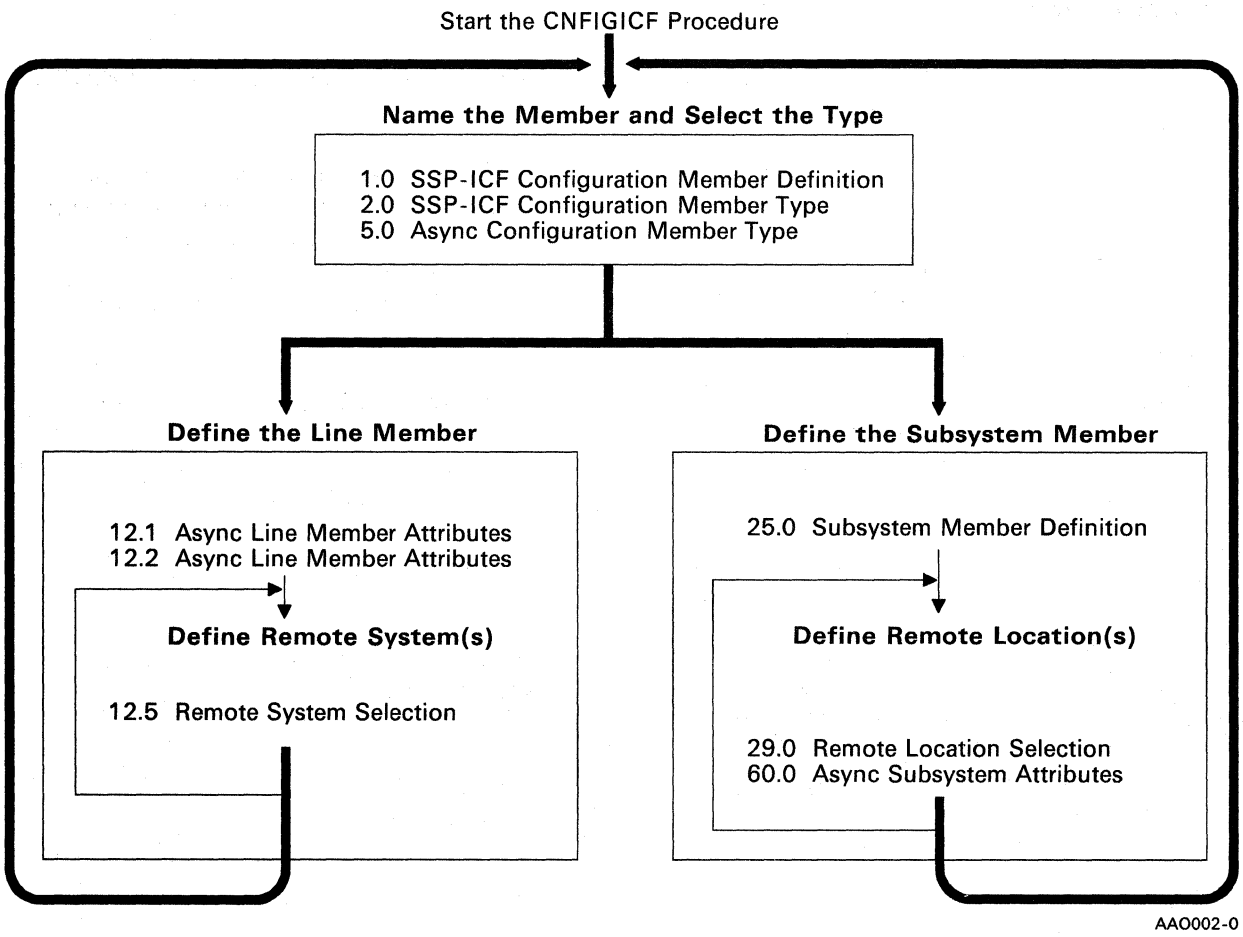
# Chapter 2. Configuring the Asynchronous Communications Subsystem

Before you can use the asynchronous communications support, it must be installed on your system. In addition, a complete asynchronous communications subsystem configuration must be defined. For an asynchronous communications subsystem that uses X.25 support, you must also describe the packet switched data network (PSDN) to System/36 (using the CNFIGX25 procedure).

This section describes the displays and all the parameters (shown in prompt form) needed to define and create an asynchronous communications subsystem configuration, using the CNFIGICF procedure. (A general description of the process of configuration is contained in the manual *Using System/36 Communications.*)

The following diagram shows the sequence in which the CNFIGICF displays are presented, and shows what displays you use to create a communications line member and what displays you use to create a subsystem member.

*Note:* *You **must** define a line member for the communications line support before you can define an asynchronous communications subsystem member. If the line member is later modified, its new attributes are reflected in your subsystem the next time it is enabled. Some changes to the line member, however, also require that you modify each subsystem member using that line member before the subsystem is enabled again. To ensure compatibility between the line and the subsystem, you should always edit (modify) all the subsystem members affected by the changed line member. Refer to the manual **Using System/36 Communications** for information on defining an asynchronous communications line member.*


Start the CNFIGICF Procedure
Name the Member and Select the Type
1.0 SSP-ICF Configuration Member Definition
2.0 SSP-ICF Configuration Member Type
5.0 Async Configuration Member Type
Define the Line Member
12.1 Async Line Member Attributes
12.2 Async Line Member Attributes
Define Remote System(s)
12.5 Remote System Selection
Define the Subsystem Member
25.0 Subsystem Member Definition
Define Remote Location(s)
29.0 Remote Location Selection
60.0 Async Subsystem Attributes
AAO002-0

## Explanation of Displays

On the following displays for the asynchronous communications CNFIGICF procedure:

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

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

- For this set of example displays only, the values to the right of the prompts are shown with:

  - Default values, supplied by the system. If the system provides a default value, that value is shown here. (You can enter a different value if you wish.)

  - Sample values, as typical examples. If fewer characters are shown than the field allows, the remaining positions in the field are underscored. Note that once a value has been entered in a field, it becomes the default value for any related fields on the succeeding displays.

## Subsystem Member Definition

### Display 1.0 SSP-ICF Configuration Member Definition

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

```
 1.0            SSP-ICF CONFIGURATION MEMBER DEFINITION                  W1

1. Configuration member name  . . . . . . . . . . . . . . . . . . . . _______

2. Library name . . . . . . . . . . . . . . . . . . . . . . . . . . . _______

3. Select one of the following:
      1. Create new member
      2. Edit existing member
      3. Create new member from existing member
      4. Remove a member
      5. Review a member
   Option . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1-5 _

4. Existing member name . . . . . . . . . . . . . . . . . . .      _______

5. Existing member library name . . . . . . . . . . . . . . .      _______

Cmd7-End      Cmd19-Cancel
```

1. **Configuration member name:** Enter the name that identifies this configuration of the subsystem. This subsystem member name is used to store the subsystem configuration member in a library, and it is also used in the ENABLE and DISABLE procedures to start and stop the subsystem.

2. **Library name:** Enter the name of the library in which the subsystem configuration member is to be stored. The default is the library that you are currently using.

   *Note: The line member and subsystem member must be in the same library. When X.25 support is needed, the X.25 configuration member also must be in the same library.*

3. **Select one of the following:** Specify which of the five options you want. For example, if you are *creating* a new asynchronous communications subsystem member, select option 1.

4. **Existing member name:** This prompt is displayed only if you selected option 3 for prompt 3. Enter the name of the existing subsystem configuration member that is to be used to create the new member. (The existing member is not changed.)

5. **Existing member library name:** This prompt is displayed only if you selected option 3 for prompt 3. Enter the name of the library that contains the existing member. The default is the library name specified for prompt 2.

### Display 2.0 SSP-ICF Configuration Member Type

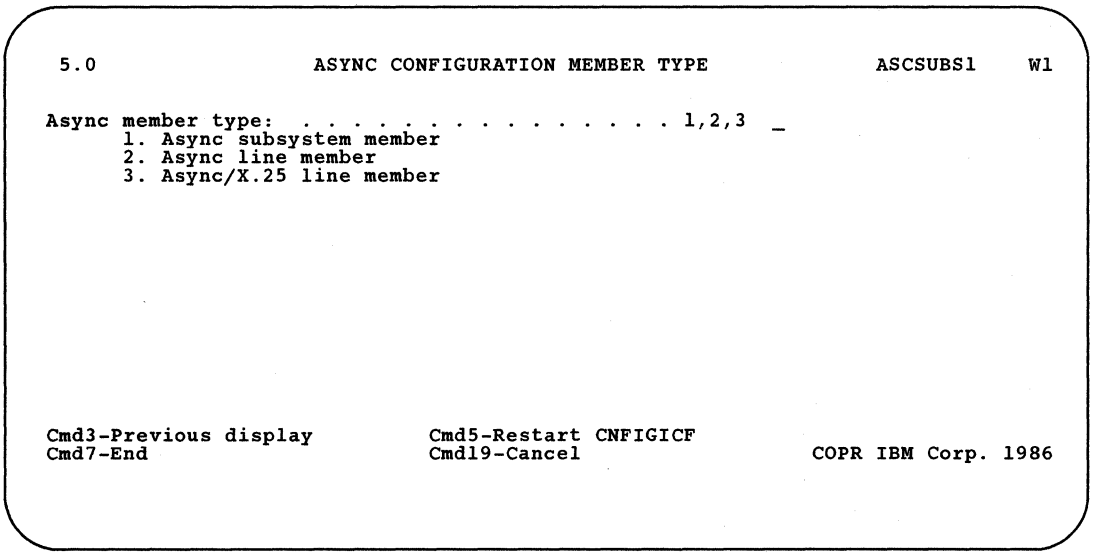
On display 2.0, specify the type of subsystem member you want to define or redefine.

```
 2.0                SSP-ICF CONFIGURATION MEMBER TYPE            ASCSUBS1    W1

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

   Option:  4









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

- **Select one of the following options:** Enter a 4 when you are defining an asynchronous communications configuration member.

### Display 5.0 Asynchronous Configuration Member Type

On display 5.0, specify the type of asynchronous communications configuration member you want to define or redefine.



```
 5.0                ASYNC CONFIGURATION MEMBER TYPE              ASCSUBS1    W1

Async member type:  . . . . . . . . . . . . . . . . 1,2,3  _
      1. Async subsystem member
      2. Async line member
      3. Async/X.25 line member









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

- **Asynchronous member type:** Enter a 1 when you are defining the subsystem member for your asynchronous communications subsystem.

### Display 25.0 Subsystem Member Definition

On display 25.0, enter information that defines the subsystem member.

```
25.0                SUBSYSTEM MEMBER DEFINITION                ASCSUBS1    W1

1. Line member name . . . . . . . . . . . . . . . . . . . . . . . _____
2. Location name . . . . . . . . . . . . . . . . . . . . . . . . _____
3. Local ID . . . . . . . . . . . . . . . . . . . . . . . . . . . _____




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

1. **Line member name:** Enter the name of the *line* member that will be used with this *subsystem* configuration.

   *Note: The line member must be in the same library as the subsystem member. You must define a line member for the communications line support before you can define an asynchronous subsystem member. Refer to the manual* ***Using System/36 Communications*** *for more information.*

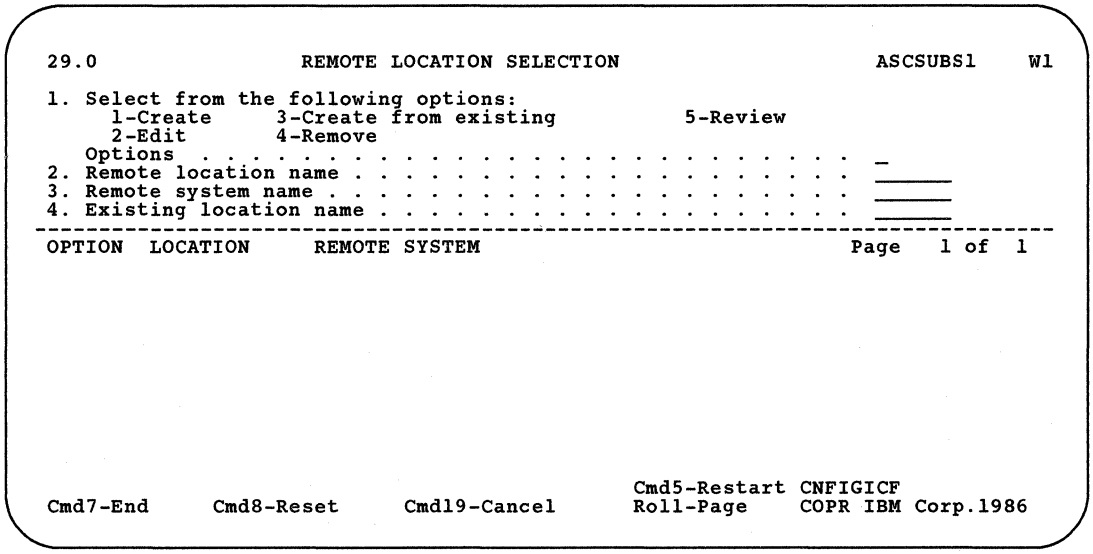
2. **Location name:** Specifies the name that identifies *your* System/36 to any remote locations that your subsystem calls. Enter a name of up to 8 characters. If you do not enter a location name, the subsystem configuration member name is also used as the location name. This name must be the same as the name specified by the remote system using the DEFINLOC procedure.

3. **Local ID:** Specifies the identifier by which this subsystem is known to the remote system. This must be the same as the ID that the remote system specified for this location name using the DEFINLOC procedure. Enter up to 8 characters.

   For more information on the DEFINLOC procedure, see "Using the DEFINLOC Procedure" later in this chapter.

### Display 29.0 Remote Location Selection

On display 29.0, enter information that defines the remote location.




On this display, select the remote locations with which this subsystem communicates. You can define a remote location for each remote system in the corresponding line member.

All previously defined remote locations and all remote systems from the corresponding line member are listed. You may edit, remove, or review any of these remote locations by entering the correct option number in the column to the left of the remote location with which you want to work.

If no locations have been defined for any of the remote systems, the only option shown on display 29.0 is *1-Create*, and prompt 4 is not shown.

1. **Select from the following options:** Specify which of the available options you want to use:

   **1-Create** defines a new remote location.

   **2-Edit** changes an existing remote location.

   **3-Create from existing** defines a new remote location using an existing remote location as a base.

   **4-Remove** deletes a remote location. It will no longer exist in this member.

   **5-Review** displays an existing remote location. The configuration member cannot be changed.

2. **Remote location name:** Enter the location name of the remote system which your system will call. Each remote system identified in this subsystem member must have a unique remote location name. The remote location name specified here must *not* be the same as the location name specified (on display 25.0) at the remote system.

   When a SESSION statement is used by any program in the local System/36 to start a session at the remote location, the remote location name specified here must also be specified in the LOCATION parameter of that statement.

3. **Remote system name:** This is the name you used to identify the remote system during line member definition.

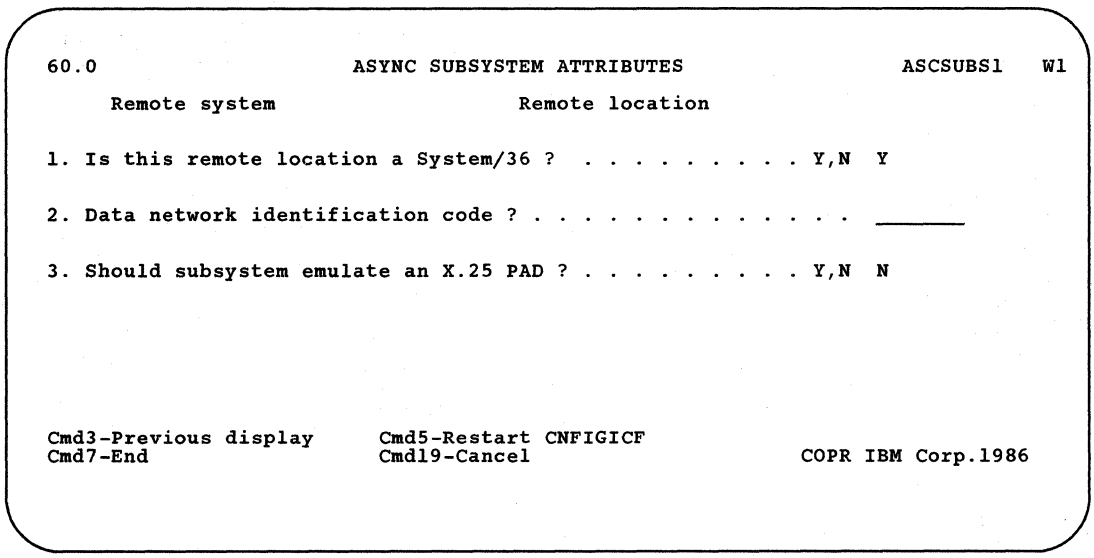
4. **Existing location name:** Specify the name of the existing remote location that is to be used as a base when creating a new remote location.

When you press the Enter key, display 60.0 appears. You further define each remote location on display 60.0.

When you have finished defining remote locations, press the Cmd7 key. Display 1.0 reappears.

### Display 60.0 Asynchronous Subsystem Attributes

On this display, enter the attributes for the asynchronous communications subsystem.




1. **Is this remote location a System/36?** Specify a Y for Yes or an N for No. The default is Y.

   *Note: Prompts 2 and 3 are not displayed unless you are defining an asynchronous/X.25 subsystem member.*

2. **Data network identification code?** Specify the 4-digit data network identification code of the network to which your location is connected.

3. **Should subsystem emulate an X.25 PAD?** Specify a Y for Yes or an N for No. If you specify N, this location will be connected to the remote system as an X.25 host. The default is N.

   *Note: This prompt is not displayed for all data network identification codes.*

   For more information about PAD emulation, see Appendix A.

*Note: Display 29.0 appears again (after you enter the prompts on display 60.0) to allow you to configure additional remote locations.*

## Modifying a Subsystem Configuration

To change one or more of the attributes defined in either a line or subsystem member of an existing asynchronous communications subsystem configuration, you can use the CNFIGICF procedure to change (edit) the member. (For the changed attributes to take effect, any subsystem using the member being changed must be disabled and enabled again.) On display 1.0 of the procedure, specify the name of the member to be changed and specify option 2 (edit existing member) for prompt 3. On display 5.0, specify which type of member it is (line or subsystem), and then, on the following displays, change the values of only the attributes that need to be changed. After the CNFIGICF procedure is completed, the updated member definition is used each time any subsystem associated with the changed member is enabled.

*Note: The ENABLE procedure (if the SHOW parameter is specified in the ENABLE command) can be used to display (not change) all the values specified in the subsystem configuration member of the subsystem that is being enabled.*

## Using the DEFINLOC Procedure

You use the DEFINLOC procedure to set up a list of the names and location IDs of remote locations that you allow to call your subsystem (an asynchronous communications subsystem using X.25 support only).

When your subsystem receives a connect request, the system checks the location ID of the calling system. If the location ID and location name are listed in DEFINLOC, the call is accepted; otherwise, it is rejected. If the call is accepted, the remote system is connected through an available logical channel and assigned to a generic remote system.

When you use the DEFINLOC procedure, you enter information on a set of displays. From the DEFINLOC main menu, you can choose to look at a list of locations already entered, add new locations, change (update) existing locations, or delete locations.

To start, enter the DEFINLOC procedure command, which has no parameters. The following menu appears:

```
  1.0                  REMOTE LOCATION DEFINITION

Select one of the following options:
      1. Display remote locations
      2. Add remote locations
      3. Update remote locations
      4. Delete remote locations

  Option:




Cmd7-End
                                                  COPR IBM Corp. 1986
```

Select an option and press the Enter key. For more information about how to use the DEFINLOC procedure, press the Help key from display 1.0. You can also press the Help key for information about each DEFINLOC display. If you need more information, refer to the manual *Using System/36 Communications*.

# Chapter 3. Using the Asynchronous Communications Subsystem

## Enabling and Disabling an Asynchronous Communications Subsystem

The ENABLE and DISABLE procedures are used to start and end an asynchronous communications subsystem.

### Enabling a Subsystem

The ENABLE procedure is used to start (enable) an asynchronous communications subsystem on System/36. You must specify the name of the subsystem on the ENABLE procedure command, along with the line number of the communications line to be used by the subsystem.

The ENABLE procedure associates the asynchronous communications subsystem with a particular subsystem configuration and with a communications line. The result of the ENABLE procedure is an active subsystem that has the attributes specified during configuration.

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

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

- Ensures compatibility between the subsystem configuration and the communications hardware
- Determines whether the requested communications line is available
- Loads the subsystem support for the asynchronous communications subsystem if it is not already active
- Loads the subsystem configuration that contains the attributes of the subsystem that is being enabled
- Assigns storage for required data areas and buffers

The ENABLE procedure only *prepares* the local end of the line to communicate with the remote location; the remote location must also be prepared for communications. When both ends are prepared, communications can begin.

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

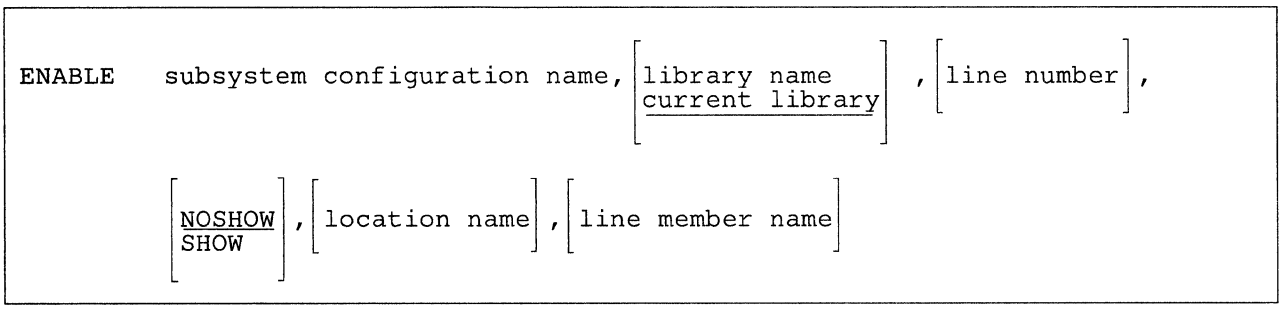
If the line type set in the subsystem configuration record does not correspond to the type of line (identified by line number in the ENABLE command) to be used by the subsystem, a message is issued and the ENABLE procedure is terminated. You can use the SETCOMM or ALTERCOM procedure to change the line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

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

An exception to the rule of unique location names is generic location names. Generic location names are numbers assigned by the system during CNFIGICF that allow incoming calls to be completed (see CNFIGICF display 12.2 in the manual *Using System/36 Communications*). The ENABLE procedure does not check to see if these numbers are unique.

## ENABLE Procedure Command

The syntax of the ENABLE procedure command is:




AAO005-1

**Subsystem configuration name:** Specifies the *subsystem* member name of the subsystem configuration to be enabled. This is the name that was specified when the CNFIGICF procedure was used to configure the subsystem. This parameter is required for an asynchronous communications subsystem.

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

**Line number:** Specifies the number of the communications line for which this subsystem is to be enabled. This parameter is required.

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

**Location name:** Specifies the name of the remote location to be enabled.

**Line member name:** Specifies the name of the line member to be enabled. This prompt is not valid when using asynchronous communications support.

## Disabling a Subsystem

To disable an asynchronous communications subsystem, the DISABLE procedure command must be run. When a disable operation is requested for a subsystem, the following functions are performed:

- If no sessions are active for the subsystem being disabled, the subsystem is disabled, and the main storage being used is freed. Also, if no other asynchronous communications subsystem is active, the asynchronous communications subsystem support is terminated.

- If sessions are active for the subsystem, a message is issued to the operator who issued the DISABLE command. The operator can respond with one of the following options:

  0 Hold (pend) the disable request. New sessions cannot be started for this subsystem; and when all sessions have been completed, a normal disable occurs (see note).

  1 Retry the disable request. Check again for any active sessions for this subsystem.

  2 Cancel active sessions and disable the subsystem or location. Active sessions for this subsystem are immediately terminated, and the DISABLE procedure is performed.

  3 Ignore the disable request. The DISABLE procedure is canceled and must be run again when the subsystem is to be disabled.

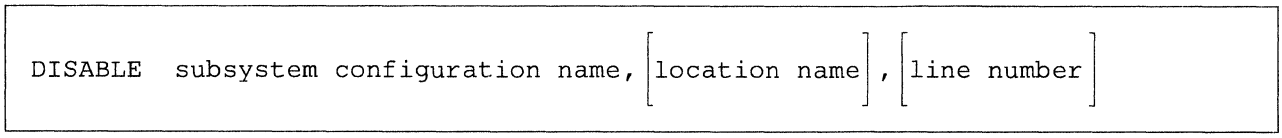
- If a disable request is pending (waiting to be performed) or is in progress, a message is issued to the operator. You cannot disable the subsystem immediately; you must either wait for the session to terminate or have the system operator cancel the session.

*Note: When a disable request is pending, each program performing a successful input operation to the location(s) affected by the DISABLE procedure receives a major return code indicating that a disable operation is pending (02xx).*

When a remote location is disabled, main storage for that location is freed. The rest of the subsystem remains active.

### DISABLE Procedure Command

The syntax of the DISABLE procedure command is:




AA0006-1

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

**Location name:** Specifies the name of the remote location to be disabled. The subsystem remains enabled as long as there are other active locations. If the location name is not specified, all remote locations will be disabled.

**Line number:** Specifies the number of the line to be disabled. This prompt is not valid when using asynchronous communications support.

## Starting Communications Sessions

System/36 communications sessions using an asynchronous communications subsystem can be started in one of two ways:

- Your program can issue an *acquire operation* to start (acquire) the session. The acquire operation identifies the session to be started and must match the session identifier specified in an associated SESSION statement.

- A program on a remote system can also issue an acquire operation to start a session. Then it can issue an *evoke operation*, which causes a *procedure start request* to be sent to the local System/36. The procedure start request initiates a procedure that starts your program, which can then communicate with the remote program.

*Note: Only the local or remote program that issues the acquire operation can issue evoke operations in that session. The program that is evoked* ***cannot*** *issue any evoke operations in that session.*

The following sections describe the SESSION statement and procedure start requests for an asynchronous communications subsystem.

## SESSION Statement

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

- It identifies, using the SYMID parameter, the session to be acquired later in the program.
- It identifies, using the LOCATION parameter, the remote location with which the program is to communicate. Before the SESSION statement is processed, the asynchronous communications subsystem that also specifies the name of the remote location must have already been enabled. (The location name was specified in the subsystem's configuration member.)

The SESSION statement, then, identifies the session and the remote location with which your program is to communicate; it also indirectly identifies the subsystem having the necessary attributes for the session.

*Note: A BASIC program does not require a SESSION statement if an OPEN statement is used that specifies the remote location name in the LOC parameter.*

The syntax of the SESSION statement for the asynchronous communications subsystem is:

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

**LOCATION parameter:** Specifies the remote location name to be associated with this session. The remote location name, specified on display 30.0 during subsystem configuration, refers to the remote location with which your program is to communicate. This parameter has no default.

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

## Procedure Start Requests

When a remote program on another System/36 uses an asynchronous communications subsystem to start a procedure on System/36, the remote program issues an evoke operation, which is sent as a *procedure start request* to the asynchronous communications subsystem on System/36. The subsystem starts the specified System/36 procedure, which then starts a program that communicates with the remote program. Up to 120 bytes of data, including the name of the procedure to be started, can be specified by the user to be sent by the procedure start request.

When the asynchronous communications subsystem receives a procedure start request from the remote system, it uses the information included with the procedure start request statement to start the specified System/36 procedure. (Any procedure parameters needed by the procedure are included in the data.) If the procedure was coded to accept data (PDATA-YES was specified on the COPY control statement for $MAINT, or a Y was entered in response to the Program Data In The Include Statements prompt on the end of job menu for the SEU procedure), the subsystem passes any data to the evoked program on its first input operation. An evoked System/36 program must perform an input operation as its first communications operation.

The format for the procedure start request for an asynchronous communications subsystem is as follows:

```
*EXEC or *EXEX,blank,Procedure Name,
  blank,Procedure Parameters or Program Data,
   <CR>,User ID,<CR>,Library,<CR>,
  Password,<CR>,<EOT>

<CR>  = Carriage return (hex 0D)
<EOT> = End of transmission (hex 37)
```

For information on how to write programs that are to be started by procedure start requests, see "Writing a Program That Is Started by the Remote System" in the *SSP-ICF Guide and Examples* manual.

# Communications Operations for the Asynchronous Communications Subsystem

This section describes all the input and output operations that can be coded in a program that is to communicate, using the asynchronous communications subsystem, with another System/36. For complete details of how these operations work and how to use them, see the *SSP-ICF Guide and Examples* manual.

## Programming Considerations for the Asynchronous Communications Subsystem

Before you start coding the input and output operations, you should note the following programming considerations for an asynchronous communications subsystem:

- Maximum data length of 160 bytes for a read operation
- Maximum data length of 4096 bytes for a write operation
- User application program dependencies:
  - User application is responsible for error detection, recovery, and data acknowledgment.
  - Only parity errors detected with notification are passed to the user application.

    *Note: On the 5360 and 5362 System Units, the parity bit is passed, with the data, to the user application. On the 5364 System Unit, the parity bit is stripped from the data by the communications adapter. If a parity error occurs, the user application is notified.*
  - Variable length data is received and passed to the user application. The user application must examine the data to determine the amount received and form the data into logical records. The define-the-file (DTF) control block contains the length of the data received. Assembler and BASIC programs have access to the DTF. COBOL and RPG II programs may use the asynchronous subroutine, SUBRA1, to obtain the data length.

## SUBRA1 Subroutine

The SUBRA1 subroutine allows COBOL and RPG II programs to retrieve the data length from the DTF for the last get operation.

### Calling the SUBRA1 Subroutine from a COBOL Program

The format for the call to the SUBRA1 subroutine from a COBOL program is as follows:

```
CALL 'SUBRA1' USING WSNAME,SYMID,DATAL,RCODE
```

The parameters to be passed to the subroutine are described under the topic "SUBRA1 Subroutine Parameters" later in this chapter.

### Calling the SUBRA1 Subroutine from a RPG II Program

To call the SUBRA1 subroutine from a RPG II program, make the following entries on the calculation specification:

**RPG CALCULATION SPECIFICATIONS**

IBM International Business Machines Corporation

GX21-9093-3 UM/050• Printed in U.S.A.

Program | Keying Instruction | Graphic | Card Electro Number
Programmer | Date | Key

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

C

| Line | Form Type | Control Level (L0-L9, LR, SR, AN/OR) | Indicators: And Not 9 | And Not 10 | And Not 11 | And Not 12 | And Not 13 | And Not 14 | And Not 15 | And Not 16 | And Not 17 | Factor 1 (18-27) | Operation (28-32) | Factor 2 (33-42) | Result Field: Name (43-48) | Result Field: Length (49-51) | Result Field: Decimal Positions (52) | Half Adjust (H) (53) | Resulting Indicators: Arithmetic: Plus (54-55) | Resulting Indicators: Arithmetic: Minus (56) | Resulting Indicators: Arithmetic: Zero (57-58) | Resulting Indicators: Compare: 1>2 (54-55) | Resulting Indicators: Compare: 1<2 (56) | Resulting Indicators: Compare: 1=2 (57-58) | Resulting Indicators: Lookup(Factor 2)ls: High (54-55) | Resulting Indicators: Lookup(Factor 2)ls: Low (56) | Resulting Indicators: Lookup(Factor 2)ls: Equal (57-58) | Comments (60-74) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 01 | C | | | | | | | | | | | | EXIT | SUBRA1 | | | | | | | | | | | | | | |
| 02 | C | | | | | | | | | | | | RLABL | | SYMID | | 2 | | | | | | | | | | | |
| 03 | C | | | | | | | | | | | | RLABL | | DATAL | | 4 | | | | | | | | | | | |
| 04 | C | | | | | | | | | | | | RLABL | | RCODE | | 2 | | | | | | | | | | | |
| 05 | C | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 06 | C | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 07 | C | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 08 | C | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 09 | C | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10 | C | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11 | C | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 12 | C | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 13 | C | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14 | C | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 15 | C | | | | | | | | | | | | | | | | | | | | | | | | | | | |

AA0004-1

The parameters to be passed to the subroutine are described under the topic "SUBRA1 Subroutine Parameters" later in this chapter.

### SUBRA1 Subroutine Parameters

*WSNAME* This character field contains the name of the file assigned to the work station. This field is required only for COBOL programs.

*SYMID* This 2-character field is the session identifier.

*DATAL* This 4-character name contains the length, in decimal format, of the actual data received from the last get operation.

*RCODE* This 2-character field contains the return code. The subroutine returns this value to the application program to indicate the result of the request. Valid values are as follows:

*40*: Normal completion

*41*: Invalid DTF address or DTF address not found

*42*: Not an asynchronous subsystem session

## Asynchronous Communications Subsystem Operations and Codes

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

| Asynchronous Communications Subsystem Operations | Language Operation Codes | | | |
|---|---|---|---|---|
| | **Assembler** | **BASIC** | **COBOL** | **RPG II** |
| Accept input | ACI | WAITIO and READ[1] | READ[2] | READ[3] |
| Acquire | ACQ | OPEN | ACQUIRE | ACQ |
| Cancel invite | CNI | $$CNLINV | $$CNLINV | $$CNLINV |
| End of session | EOS | $$EOS | $$EOS | $$EOS |
| Evoke | EVK | $$EVOKNI | $$EVOKNI | $$EVOKNI |
| Evoke end of transaction | EVE | $$EVOKET | $$EVOKET | $$EVOKET |
| Evoke then get | EVG | — | — | — |
| Evoke then invite | EVI | $$EVOK | $$EVOK | $$EVOK |
| Fail | FAIL | $$FAIL | $$FAIL | $$FAIL |
| Get | GET | READ | READ[4] | NEXT and READ[5] |
| Get attributes | GTA | ATTRIBUTE$ | ACCEPT | — |
| Invite | INV | $$SEND | $$SEND | $$SEND |
| Put | PUT | $$SENDNI | $$SENDNI | $$SENDNI |
| Put then get | PTG | — | — | — |
| Put then invite | PTI | $$SEND | $$SEND | $$SEND |
| Put FMH | PFM | $$SENDNF | $$SENDNF | $$SENDNF |
| Release | REL | CLOSE | DROP | REL |
| Set timer | STM | $$TIMER | $$TIMER | $$TIMER |

[1]In BASIC, an accept input operation is performed only if the WAITIO operation is followed by a READ operation.

[2]In COBOL, an accept input operation is performed only if the TERMINAL option of the READ statement is not specified or is specified with blanks.

[3]In RPG II, an accept input operation is performed only if the READ operation is not preceded by a NEXT operation.

[4]In COBOL, a get operation is performed only if the TERMINAL option of the READ statement is specified with nonblanks.

[5]In RPG II, a get operation is performed only if a NEXT operation is executed before the READ operation.

## Accept Input Operation

Your program can use the **accept input** operation to perform the following functions:

- Obtain data from any program or any display station that has responded to an invite operation that was previously issued in your program. If data becomes available to your program from more than one program or display station before the accept input operation is issued, your program receives the data that was *first* made available, whether it was from another program or from a display station.

- Wait for a new requester.

  - If your program was evoked, it should issue an accept input operation as its first operation to determine the identifier of the new requester. Your program is notified of the new requester by the resulting 01xx return code at the end of the accept input operation. (See "Return Codes" later in this chapter for information about return code 01xx.)

  - If your program is an MRT NEP program and no previous invite operation is in effect, it should issue an accept input operation so it can wait for a new requester.

Except for the first accept input operation in evoked programs or in MRT NEP programs, all accept input operations in all programs should be issued to receive data only after an invite operation is issued.

| Operation | Assembler | BASIC | COBOL | RPG II |
|---|---|---|---|---|
| Accept input | ACI | WAITIO and READ[1] | READ[2] | READ[3] |

[1]In BASIC, an accept input operation is performed only if the WAITIO operation is followed by a READ operation.

[2]In COBOL, an accept input operation is performed only if the TERMINAL option of the READ statement is not specified or is specified with blanks.

[3]In RPG II, an accept input operation is performed only if the READ operation is not preceded by a NEXT operation.

## Acquire Operation

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

The session started by the acquire operation is initialized with the parameters specified in the SESSION statement.

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

| Operation | Assembler | BASIC | COBOL | RPG II |
|---|---|---|---|---|
| Acquire | ACQ | OPEN | ACQUIRE | ACQ |

### Acquire Operation Examples

***Assembler***

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

This $WSIO macro is used to acquire the session identified as *2S* in the TERMID parameter. The DTF to be used for sending or receiving data is identified as ICDTF2. (For a complete description of the $WSIO macro's communications parameters, see "$WSIO Macro" in the *SSP-ICF Programming for Communications Subsystems and Intra Communications Subsystem Reference* manual.) The SYMID parameter of the SESSION statement must also be 2S.

***BASIC (Normal Acquire)***

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

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

***BASIC (Special Acquire)***

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

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

***COBOL***

```
ACQUIRE COMM-SESSION FOR COMMUNICATIONS-FILE.
```

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

***RPG II***

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

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

## Cancel Invite Operation

Your program uses the **cancel invite** operation to cancel any valid invite operation for which no input has yet been received from any invited session. (The cancel invite operation is the only valid cancel operation for the asynchronous subsystem.)

The cancel invite operation is valid only when it is issued after any valid invite operation. Normally, no data is in the subsystem's input buffer when the cancel invite operation is issued. If the data is in the input buffer, the operation fails and the return code 0412 is received by the program. Your program must issue an input operation to receive the data.

| **Operation** | **Assembler** | **BASIC** | **COBOL** | **RPG II** |
|---|---|---|---|---|
| Cancel invite | CNI | $$CNLINV | $$CNLINV | $$CNLINV |

## End of Session Operation

Your program uses the **end of session** operation to terminate a session. Unlike the release operation, the end of session operation always terminates the session (if it still exists), and it *always* gives a normal completion return code (0000). For example, your program could issue the end of session operation after an error has occurred on one of its previous operations; it may be an error from which your program cannot easily recover.

### Ending a Session Started by an Evoke Operation from Another Program

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

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

| Operation | Assembler | BASIC | COBOL | RPG II |
|---|---|---|---|---|
| End of session | EOS | $$EOS | $$EOS | $$EOS |

## Evoke Operations

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

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

The following types of evoke operations can be used in an asynchronous communications session to start another procedure on a remote system.

- **Evoke:** Evokes the specified procedure, sends data to the subsystem (if specified by the user), and then waits until that procedure has been started before control is returned to your program.

- **Evoke end of transaction:** Evokes the specified procedure, sends any data specified by the user to one of the programs started by that procedure, and then ends the transaction without allowing the program to communicate in return. Control is returned to your program immediately, without confirmation that the remote program has or has not started successfully.

- **Evoke then get (assembler only):** Evokes the specified procedure, sends any data specified by the user, and then waits for input to be received from one of the programs started by the procedure.

- **Evoke then invite:** Evokes the specified procedure, sends any data specified by the user, and invites one of the programs started by that procedure to send data; your program regains control without having it wait for the invited data to be received. Control is returned to your program after the remote system acknowledges that the remote program has or has not started successfully. An accept input or a get operation must be issued later in this transaction to receive the data in your program's input buffer.

| Operation | Assembler | BASIC | COBOL | RPG II |
|---|---|---|---|---|
| Evoke | EVK | $$EVOKNI | $$EVOKNI | $$EVOKNI |
| Evoke end of transaction | EVE | $$EVOKET | $$EVOKET | $$EVOKET |
| Evoke then get | EVG | — | — | — |
| Evoke then invite | EVI | $$EVOK | $$EVOK | $$EVOK |

The evoke parameter list associated with each evoke operation contains the name of the procedure to be started, the name of the library in which the procedure is located, and the password and user identifier associated with that procedure. (The password and user identifier are needed only if security is being used on System/36.) The evoke operation can optionally include either parameters to be sent to the evoked procedure or data to be passed to one of the programs started by the procedure.

The total length of the procedure name and data (or procedure parameters) specified in the program to be sent to the subsystem cannot exceed 120 bytes. (This does not include the other three evoke list parameters, each of which can be 8 bytes long.)

### Assembler Evoke Operation (Macroinstructions)

***$WSIO Macro:*** To perform an evoke operation in assembler, use the $WSIO macro. You specify the evoke **operation code** (EVK) in the OPC parameter of the macro (for example, OPC-EVK). You must use another macro, $EVOK, to specify the evoke **parameters** needed to perform the evoke operation specified on the $WSIO macro. (For a complete description of the communications parameters for the $WSIO macro, see "$WSIO Macro" in the *SSP-ICF Programming for Communications Subsystems and Intra Communications Subsystem Reference* manual.)

User data or procedure parameters (in either positional or keyword form) to be passed to the other program or procedure are specified in the RCAD and OUTLEN parameters on the $WSIO macro. The INLEN parameter is ignored.

***Example of $WSIO Macro:***

```
EVOK    $WSIO  DTF-ICDTF1,,RCAD-IOBUFF,
               OPC-EVK,PL@-EVKLST,OUTLEN-112
```

This $WSIO macro (in your program) evokes a procedure on the remote System/36, starts a transaction in the acquired session, and then waits until that procedure has been started before control is returned to your program. The parameters to be used in the operation are those identified by the label EVKLST (shown in the following $EVOK example). There are 112 bytes of *output* data or procedure parameters in your program buffer named IOBUFF that are to be sent to the other program or procedure. Then, when *input* is received from the program, the data is placed in your program's buffer (IOBUFF), which is 256 bytes long.

***$EVOK Macro:*** The $EVOK macro builds a parameter list to be associated with an evoke operation. The label on this macro should be the label specified on the PL@ parameter of the $WSIO macro performing the evoke operation. (For a complete description of the $EVOK macro and its parameters, see "$EVOK Macro" in the *SSP-ICF Programming for Communications Subsystems and Intra Communications Subsystem Reference* manual.)

***Example of $EVOK Macro:***

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

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

### BASIC Evoke Operation Parameters

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

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

***BASIC Example (Evoke Operation):***

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

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

## COBOL Evoke Operation Parameters

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

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

### *COBOL Example (Evoke Operation)*

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

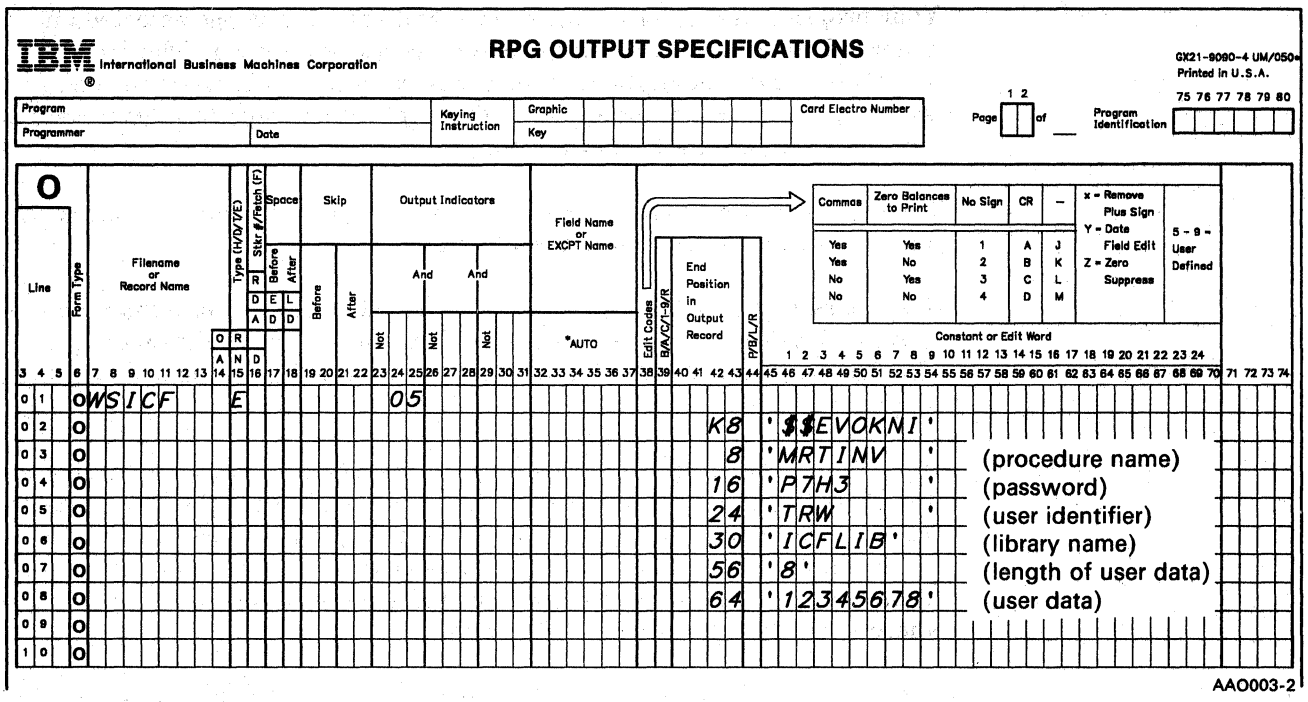
The WRITE statement at line 95 issues the $$EVOKNI (evoke) operation to evoke a procedure (ICFREM) in the session identified by ICF-SESSION. Lines 57 through 63 give the values of the parameters used in the evoke operation performed by the WRITE statement.

### RPG II Evoke Operation Parameters

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

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

*RPG II Example (Evoke Operation)*



RPG OUTPUT SPECIFICATIONS

IBM International Business Machines Corporation

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

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

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

| Line | Form Type | Filename or Record Name | Type (H/D/T/E) O A / R N D | Str #/Fetch (F) | Space After | Space Before | Skip Before | Skip After | Output Indicators And Not | Field Name or EXCPT Name | Edit Codes | End Position in Output Record | P/B/L/R | Constant or Edit Word |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 01 | O | WSICF | E | | | | | | 05 | | | | | |
| 02 | O | | | | | | | | | | | K8 | | '$$EVOKNI ' |
| 03 | O | | | | | | | | | | | 8 | | 'MRTINV ' (procedure name) |
| 04 | O | | | | | | | | | | | 16 | | 'P7H3 ' (password) |
| 05 | O | | | | | | | | | | | 24 | | 'TRW ' (user identifier) |
| 06 | O | | | | | | | | | | | 30 | | 'ICFLIB' (library name) |
| 07 | O | | | | | | | | | | | 56 | | '8' (length of user data) |
| 08 | O | | | | | | | | | | | 64 | | '12345678' (user data) |
| 09 | O | | | | | | | | | | | | | |
| 10 | O | | | | | | | | | | | | | |

Comma | Zero Balances to Print | No Sign | CR | – | x = Remove Plus Sign, Y = Date Field Edit, Z = Zero Suppress | 5 – 9 = User Defined

| Yes | Yes | 1 | A | J |
|---|---|---|---|---|
| Yes | No | 2 | B | K |
| No | Yes | 3 | C | L |
| No | No | 4 | D | M |

AAO003-2

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

## Fail Operation

Your program uses the fail operation to indicate that it has detected an abnormal condition while it was sending or receiving data. The fail operation causes the asynchronous communications subsystem to send a break signal to the remote system. The application programs determine the error recovery after a fail operation.

The fail operation causes a return code to be sent to the other program, indicating that the fail operation was issued.

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

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

In either case, the program that issued the fail operation should send, and the program that receives the fail return code (0302) should receive. Otherwise, the program that was sending cannot determine which record failed or with which record it should begin sending again.

If both programs issue a fail operation at the same time, the program that was receiving will be successful and should send. The program that was sending will receive return code 0302, indicating that its next operation *must* be an input operation.

| Operation | Assembler | BASIC | COBOL | RPG II |
|---|---|---|---|---|
| Fail | FAIL | $$FAIL | $$FAIL | $$FAIL |

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

## Get Operation

Your program uses the get operation to obtain data from either a specific program or a specific display station. In an asynchronous communications session, the get operation causes the subsystem to get data from the program with which your program is communicating (and which has already been evoked). The get operation also causes your program to wait for the data if it is not available immediately. Your program receives control when the data is available.

*Note: The get operation obtains data from a specific program or display station, and the accept input operation allows the data to come from any previously invited program or display station.*

In the asynchronous communications subsystem, the get operation can be issued by itself. Only in assembler language can the get operation be issued in combination with another operation such as evoke and followed with get or put then get.

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

| Operation | Assembler | BASIC | COBOL | RPG II |
|---|---|---|---|---|
| Get | GET | READ | READ[1] | NEXT and READ[2] |
| Get attributes | GTA | ATTRIBUTE$ | ACCEPT | — |

[1]In COBOL, a get operation is performed only if the TERMINAL option of the READ statement is specified with nonblanks.

[2]In RPG II, a get operation is performed only if a NEXT operation is executed before the READ operation.

*Note: The get attributes operation is not used in RPG II.*

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

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

## Invite Operation

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

| Operation | Assembler | BASIC | COBOL | RPG II |
|---|---|---|---|---|
| Invite | INV | $$SEND[1] | $$SEND[1] | $$SEND[1] |

[1]In BASIC, COBOL, or RPG II, only an invite operation is performed if $$SEND is issued with a record length of 0 bytes. Otherwise, $$SEND performs a *put then invite* operation.

## Put Operation

The put operation passes data records from the issuing program to the other program in this transaction and returns control to your program without waiting for the operation to be completed. Each put operation sends only one record to the subsystem. You can issue put operations only during a transaction. To issue a put operation without sending any data, specify an output record length of zero.

The following types of put operations can be issued in an asynchronous session.

- **Put:** Issues a put operation to the subsystem to send a data record to the remote program, and returns control to your program without waiting for the operation to complete.

- **Put then get** (assembler only): Issues a put operation to send a record to the remote program, and then waits for the remote program to send data to your program. Control is not returned to your program until the data is received. (See "Get Operation" earlier in this chapter.)

- **Put then invite:** Issues a put operation to send a record to the remote program, followed by an invite operation so it can receive data from that program. (See "Invite Operation" earlier in this chapter.) Control is returned to your program without waiting for the remote system to send the data. (An accept input or a get operation must be issued later in this transaction to receive the invited input.)

- **Put FMH:** Used either to set the translation mode or parity setting for your session or to send X.29 PAD messages.

  - **Setting translation mode:** When an asynchronous communications translation mode is XLATE-Y, perform translation. User data is translated from EBCDIC to ASCII on put operations and from ASCII to EBCDIC on get operations.

    You can use the put FMH operation to change the translation mode in your program:

    - If you issue a put FMH operation with the data string XLATE-N in your output buffer, data is not translated.

    - If you issue a put FMH operation with the data string XLATE-Y in your output buffer, data is translated.

    The output record length must be set to 7.

    Therefore, if you do not want user data in a program to be translated, you must issue a put FMH (XLATE-N) operation before you issue any put or get operations in your program.

  - **Setting parity:** When an asynchronous session is first acquired, the specified in display 12.1 of the CNFIGICF procedure (see Chapter 11 in the manual *Using System/36 Communications*). This value remains in effect until you issue another put FMH operation, or disable the line. You can use the put FMH operation to change the parity setting in your session:

    - If you issue a put FMH operation with the data string PARITY-N in your output buffer, data will be sent with no parity.

    - If you issue a put FMH operation with the data string PARITY-O in your output buffer, data will be sent with odd parity.

    - If you issue a put FMH operation with the data string PARITY-E in your output buffer, data will be sent with even parity.

    The output record length must be set to 8.

– **Sending X.29 PAD messages:** For information about using the put FMH operation to send X.29 PAD messages, see Appendix A, "PAD Emulation," later in this manual.

| Operation | Assembler | BASIC | COBOL | RPG II |
|---|---|---|---|---|
| Put | PUT | $$SENDNI | $$SENDNI | $$SENDNI |
| Put then get | PTG | — | — | — |
| Put then invite | PTI | $$SEND | $$SEND | $$SEND |
| Put FMH | PFM | $$SENDNF | $$SENDNF | $$SENDNF |

## Release Operation

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

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

- If the session was started when your program was *evoked* by another program, and your program is:

  - An MRT program, the release operation passes the session to the next step in your procedure. The system then executes any additional OCL statements in the procedure.

  - An SRT program, the release operation is delayed until your program terminates.

| Operation | Assembler | BASIC | COBOL | RPG II |
|---|---|---|---|---|
| Release | REL | CLOSE | DROP | REL |

## Set Timer Operation

Your program can use the set timer operation to set a timer and wait for it to expire before performing some specified function such as an accept input operation. The set timer operation specifies an interval of time (in hours, minutes, and seconds) to wait before your program receives a timer expired return code (0310). Your program continues to execute, and all operations are valid during the time interval. Your program must issue an accept input operation some time after it has issued the set timer operation, so that it can accept the 0310 return code after the timer has expired.

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

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

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

| Operation | Assembler | BASIC | COBOL | RPG II |
|---|---|---|---|---|
| Set timer | STM | $$TIMER | $$TIMER | $$TIMER |

# Return Codes

This section describes all the return codes that are valid for the asynchronous communications subsystem. These are interactive communications return codes that are sent to your program at the end of each subsystem operation to indicate the results of that operation. The appropriate return code is sent by the subsystem to the application program that issued the operation; the program can then check the results and act accordingly.

The return code is a 4-digit value; the first 2 digits contain the major code, and the last 2 digits contain the minor code. Assembler programs receive the return codes in binary form (2 bytes long). BASIC, COBOL, and RPG II programs receive the return codes in EBCDIC hexadecimal form (4 bytes).

*Notes:*

1. *In the return code descriptions, **your program** refers to the local System/36 application program that initiates the operation and receives the return code from the subsystem. The **remote program** refers to the remote system's application program with which your application program is communicating.*

2. *Several references are also made in the descriptions to **input** and **output** operations. The following chart shows all the input and output operations that are valid for the asynchronous communications subsystem. Although all the operations shown are valid for asynchronous communications subsystems, their validity also depends on the logical sequence of communications events occurring between your System/36 and the remote system.*

| **Input Operations to Your Program** | **Output Operations from Your Program** |
|---|---|
| Accept input | Acquire[1] |
| Get | End of session |
| Invite | Evoke |
| | Evoke end of transaction |
| | Fail |
| | Put |
| | Release |
| [1]Normally, the acquire operation should be followed by an evoke operation in order to establish a transaction. | |

**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 was received on a successful input operation. The remote program now wants to receive some data; your program must send it.<br><br>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.<br><br>**Normal Action:** If return code 0000 was received on an input operation, issue an output operation. |

For the actions that can be taken (in this session) after 0000 is returned for an output operation, refer to the following chart:

| **In This Session, If Your Program:** | **And the Last Output Operation Was:** | **Then (In This Session):** |
|---|---|---|
| Started the session (this is an acquired session) | Acquire operation | Issue an evoke operation. |
| | Evoke end of transaction 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) | Any output operation | Your session has ended. Continue local processing, or terminate your program. |
| | | 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.

| Code | Indication/Action |
|---|---|
| 0001 | **Normal Indication:** Your program has received some data on a successful input operation. It must continue to receive input until SSP-ICF returns a code of xx00 (a change direction indication, which allows your program to send data).<br><br>**Normal Action:** Issue another input operation. However, if your program detects something that indicates the remote program is ready to receive data, your program can issue an output operation. |
| 0004 | **Normal Indication:** An X.29 PAD message was received. The message may be a parameter indication or an error message. See Appendix A, "PAD Emulation." |
| 0016 | **Normal Indication:** Your program has received some data on a successful input operation. However, the data received contains a parity error.<br><br>**Normal Action:** Notify remote program to resend the data. |

**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:** A 01xx return code 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 was received, and some data may have been received with the request.<br><br>For *output* operations performed by an evoked SRT program, the operation completed successfully.<br><br>**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. |

---

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 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 |
|---|---|
| 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 was received.<br><br>**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.<br><br>**Normal Action:** Issue another input operation. If your program detects something that indicates the remote program is ready to receive data, your program can issue an output operation. |

**Major Code 03** – Successful operation, but no data received.

**Normal Description:** The input 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:** *No* data was received on a successful input operation. This return code indicates that a get operation issued earlier has completed successfully and there is no data to process.

**Normal Action:** Issue an output operation or continue to issue input operations.

**0302** **Normal Indication:** A fail indication was received with *no* data on a *successful* input operation. The remote program has issued a FAIL or the session has abnormally terminated.

**Normal Action:** Issue end of session.

**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.

**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 Messages* manual for additional information. The error indication is also returned to your program as a return code; the minor code portion indicates the specific cause. (Each return code is described on the following pages.) The subsystem must be enabled again before communications can resume.

**General Recovery Actions:** The following general actions can be taken for each 80xx return code. Other specific actions are given in each return code description.

- Issue, to the system operator or to the display station operator who started the program, a message requesting that the subsystem be enabled again.
- Issue an end of session (EOS or $$EOS) operation for the session that has terminated. Your program can: (1) wait for the subsystem to be enabled by issuing a set timer ($$TIMER) operation or by using the TIMER intrinsic function (in BASIC only); (2) continue local processing; or (3) terminate. Note that if your program is a BASIC or RPG II program, the $$TIMER operation is not valid unless at least one display station or session is attached to your program.
- If the session should be started again after the subsystem is enabled, it must be reacquired by your program or restarted by the 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 the abnormal termination of either this subsystem or its interrupt handler.<br><br>**Recovery Action:** This subsystem has been disabled; it must be enabled again before communications can resume. Your program can continue local processing, wait to reissue the acquire operation, or terminate. |
| 8082 | **Error Indication:** This session is being terminated immediately because the subsystem controlling the session is currently being disabled; the subsystem is not waiting for any of its active sessions to be completed normally.<br><br>**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 until the subsystem has been enabled again and reissue the acquire operation, or it can 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 Messages* manual for additional information. The error indication is also returned to your program as a return code; the minor code portion indicates the specific cause. Before communications can resume, the session must be acquired again or be started by another procedure start request.

**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.

- 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 I/O error has occurred; data may have been lost. The session has been terminated.<br><br>**Recovery Action:** If your program started the session, reissue the acquire operation to restart the session. If your program was evoked, it can wait 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.<br><br>**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 to be evoked again (MRT programs only), continue local processing, or terminate. |

**Major Code 82** – Acquire operation failed.

**Error Description:** An attempt to acquire a session was not successful; the session was not started. An error indication was returned to your program as a return code; the minor portion of the code indicates the specific cause. (Each return code is described on the following pages.) The error indication has also been sent as a message to the display station or to the system console; the operator can refer to the *System Messages* manual for additional information.

**General Recovery Actions:** Determine why the 82xx error code was returned to your program. Read the description of that return code to determine what action is needed.

**Code** **Indication/Action**

**8233** **Error Indication:** On an unsuccessful acquire operation, an invalid session identifier was detected. Either no SESSION statement was specified between the LOAD and RUN statements for this program, or the session identifier in your program does not match the identifier specified on the SESSION statement for the session being acquired. The session was not started.

**Recovery Action:** If the error is in your program, specify the correct session identifier in your program. If an incorrect identifier was specified on the SESSION statement, specify the correct value in the SYMID parameter.

**8281** **Error Indication:** On an unsuccessful acquire operation, an SSP-ICF error condition was detected. The error caused the abnormal termination of either this subsystem or its 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 to reissue the acquire operation, or terminate.

**8282** **Error Indication:** The acquire operation just performed was unsuccessful because the subsystem controlling the session is currently being disabled; no sessions can be acquired in the subsystem.

**Recovery Action:** Communications with the remote program cannot be resumed until the subsystem has been enabled again. Your program can continue local processing, it can wait until the subsystem has been enabled again and reissue the acquire operation, or it can terminate.

**82A8** **Error Indication:** The acquire operation was not successful because the maximum number of active sessions allowed in the system has been reached. No more than 360 sessions can be active in System/36 at one time. The session was not started.

If this acquire operation is associated with a SESSION statement (normal acquire), the maximum of 260 normally acquired sessions are already active at this time. If this acquire operation is *not* associated with a SESSION statement (BASIC special acquire), the maximum of 100 specially acquired and/or evoked sessions are already active at this time.

**Recovery Action:** Your program can wait for another session to end and then reissue the acquire operation. Otherwise, your program can continue local processing or terminate.

**82AA** **Error Indication:** The acquire operation just performed was not successful either because no subsystems are enabled at this time or because, of all the subsystems that are enabled, none contained the name of an *activated* remote location definition that matches the location name you specified on your SESSION statement. (That is, although the remote location was defined for a subsystem during the CNFIGICF procedure, that remote location definition may not have been activated when the subsystem was enabled.)

The subsystem that must be enabled or that must have its remote location activated is the one whose subsystem configuration member contains the same remote location name as that specified by the location parameter in the SESSION statement or on the OPEN statement in BASIC. That location name must also have been specified on display 30.0 during configuration of the subsystem. The session was not started.

**Recovery Action:** Verify that the name of the *remote* location (with which your program is attempting to communicate) was specified correctly on the location parameter of the SESSION statement or on the OPEN statement in BASIC. If the correct name was specified, enable the specified subsystem by entering the ENABLE procedure command. Then reissue the acquire operation. Otherwise, your program can continue local processing, wait to reissue the acquire operation, or terminate.

**82AB** **Error Indication:** The acquire operation just performed was not successful because the specified subsystem is currently being enabled; or, if the subsystem is already enabled for another remote location, communications is currently being activated for the specified remote location. This condition also may have occurred because the subsystem on the *remote* system has not been enabled. The session was not started.

**Recovery Action:** Your program can wait until the subsystem has been enabled, or until communications has been activated with the specified remote location; then it can reissue the acquire operation to start the session. If the *remote* subsystem has not been enabled, notify the remote location.

**82B0** **Error Indication:** The acquire operation just performed was not successful either because the specified subsystem or the specified remote location definition for this session is currently being disabled, or because a disable subsystem request for this subsystem or location is pending. No new sessions can be started.

**Recovery Action:** Your program can wait until the subsystem is enabled again or until communications with this location is activated 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 for one of the sessions in the subsystem 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 Messages* manual for additional information.

**General Recovery Actions:** The following general actions can be taken for each 83xx return code. Other specific actions are given in each return code description.

1. Determine why the 83xx error code was returned to your program. Read the description of that return code to determine what action is needed.

2. If a parameter value must be changed in the 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 (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. If another operation is not successful, retry it only a limited number of times. (The limit for retries should be specified in your program.)

| Code | Indication/Action |
|---|---|
| 830B | **Error Indication:** Your program has attempted to execute a communications input or output operation either before the session was acquired or after it has ended. Your program may have: (1) issued an input or output operation either *before* it issued an acquire operation or *after* it has released the session (by a release or end of session operation); or (2) it may have improperly handled an 81xx (session was terminated) or 82xx (session was not acquired) error return code.<br><br>**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. |
| 831E | **Error Indication:** The operation just issued by your program was invalid. Either the subsystem did not recognize the operation code, or the specified operation is not supported by the subsystem. The session is still active.<br><br>**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. |
| 8327 | **Error Indication:** An invalid input or output operation was issued when no transaction existed; your program may have expected more data when there was none. The remote program has already ended the transaction, or 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.<br><br>**Recovery Action:** If you want your program to recover from this error, issue an evoke operation to start a transaction. Otherwise, issue an end of session operation, then continue local processing or terminate your program. If a coding error in your program caused the error, correct your program. |

| Code | Indication/Action |
|---|---|
| 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.<br><br>**Recovery Action:** If you want your program to recover from this error dynamically, 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.<br><br>**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.<br><br>**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. |

| Code | Indication/Action |
|---|---|
| 8333 | **Error Indication:** On an input or output operation; an invalid session identifier was detected. The session is still active.<br><br>**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. |
| 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.<br><br>**Recovery Action:** Your program can wait until the subsystem is enabled again, and then reissue the acquire operation. Otherwise, your program can continue local processing or terminate. |

# Chapter 4. Using the Interactive Terminal Facility

The interactive terminal facility (ITF) allows the System/36 user to send and receive data through applications such as electronic message services for asynchronous terminals.

Through ITF, you can use such applications to send messages (for example, interoffice memos). In addition, ITF lets you send and receive files, library members, and DisplayWrite/36 (DW/36) documents.

## Starting ITF

Before you can start ITF, you must enable an asynchronous communications subsystem using the ENABLE command. See Chapter 3 in this manual. After you have enabled the subsystem, enter the following:

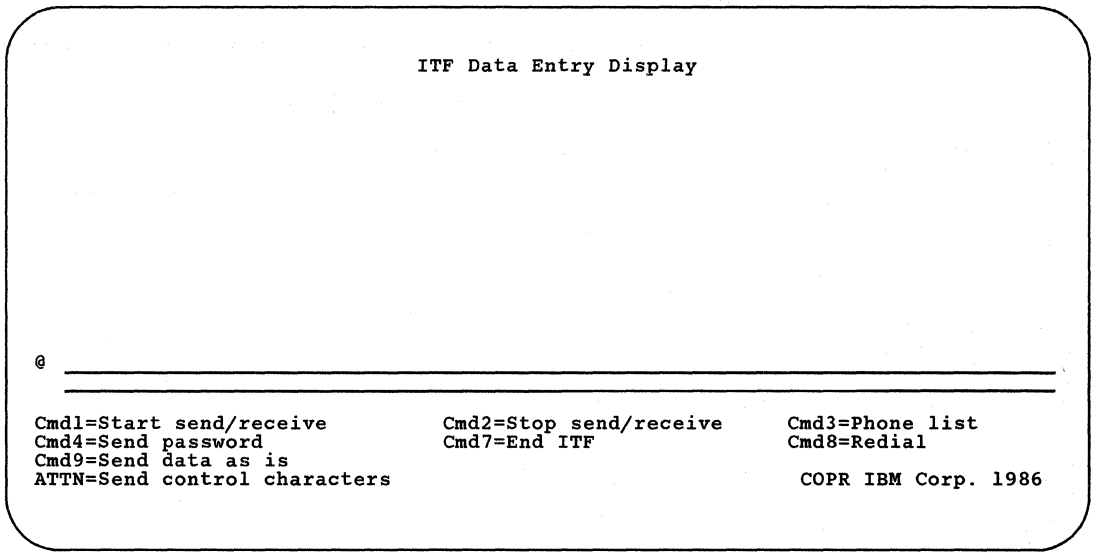
ITF *nnnnnnnn*

where *nnnnnnnn* is the name of the remote location with which you want to communicate. This is the same as the remote location name specified on display 29.0 of the CNFIGICF procedure. See Chapter 2 in this manual. For example, if you are using ITF to communicate with TELEMAIL, and you specified MAIL as the remote location name for TELEMAIL, enter the following:

ITF *MAIL*

*Note: ITF cannot be started on a generic remote location.*

After you enter the ITF command, the following display appears:




If you are communicating through a packet switched data network (PSDN), you must be connected to the network before you can send or receive messages.

For an asynchronous/X.25 configuration, that connection is made for you when you enable the asynchronous subsystem.

For a configuration using an asynchronous line, you can use one of the following ways to make this connection:

- Make a manual connection to the network by dialing the number on the telephone.
- Type in the telephone number to the network on the ITF Data Entry Display and press the Enter key. ITF sends a command to the modem, which then calls the number (the modem must support this function).
- For an asynchronous line, press Cmd3 from the ITF Data Entry Display. (Cmd3 and Cmd8 are not available for asynchronous/X.25 lines or for lines using packet assembler/disassembler (PAD) emulation.)

If you press Cmd3 from the ITF Data Entry Display, the Phone List Display appears:

```
                        ITF Phone List Display

Type options, press Enter.
 Options:  1=Add  2=Delete  3=Call number

    FUNCTION    LOCATION      PREFIX              PHONE NUMBER
1.              Chicago        ATDT       9-1-312-280-9489
2.      3       St. Paul       ATDT       9-1-612-545-8788
3.
4.
5.
6
7
8
9
10.
11.
12.
13.
14.
15.

Cmd7=Data Entry Display                                  COPR IBM Corp. 1986
```

This display lists the telephone numbers that you can call from ITF. Enter a 3 in the function field of the number that you want to call. The system calls that number and returns to the ITF Data Entry Display.

*Notes:*

1. *The prefix field gives the modem information about how to make the switched connection. If your modem does not make the switched connection for you, this field is not necessary; leave it blank.*

2. *If you are communicating through a PSDN, the telephone number that you call is a number for a PAD, which gives you access to the network. It is* ***not*** *the number for the remote location. Once you have signed on the network, it will route your message to the remote location that you specify.*

3. *Only one system user at a time can access the Phone List Display.*

The Phone List Display lets you add, delete, or connect to a remote location.

To add a new remote location to the display, fill in the fields on the display with the correct information. Type a 1 in the function field and press the Enter key.

To delete a number from this display, enter a 2 in the function field of that number.

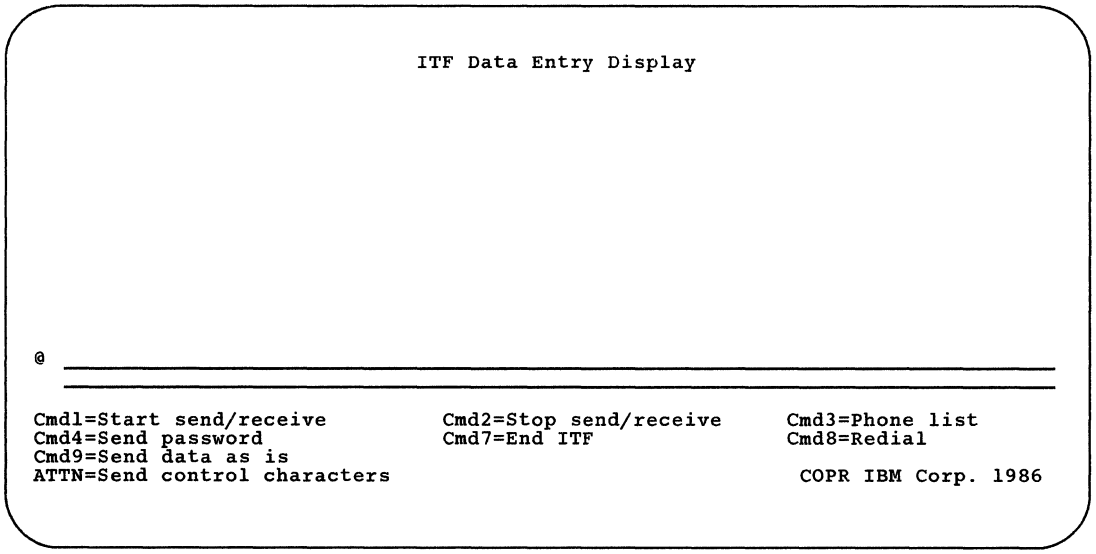
You can type options in more than one function field before pressing the Enter key. However, ITF will perform only those functions before the first call (3); when the system calls a number from the Phone List Display, it stops processing other functions from that display.

If you press Cmd7, the ITF Data Entry Display appears again. Any data that you have typed on the Phone List Display is ignored.

After the system makes a connection with the network, the Data Entry Display appears again. You must now sign on the message application. For the commands that you enter to sign on and off, as well as those used to send and receive messages, refer to the operator's manual for the application.

# Selecting ITF Functions

The ITF Data Entry Display acts as the main menu for ITF. From this display, you can enter commands and answer prompts to start the message service. You can also enter a message to be sent. Command keys let you perform other ITF functions.




When you are using ITF, the display station functions as an asynchronous terminal. Data is sent and received one record at a time. Thus, when you are entering a message to be sent, you must press the Enter key or Cmd9 at the end of each line. The display does not automatically roll up at the end of a line (unless you type a character on the last position of the second line, which automatically enters the line).

When you press the Enter key or Cmd9, ITF sends the data. The data then disappears from the data entry line of your display screen. If the remote system echoes the data back to your display, it is written again on another line above the data entry line.

Echo can be set on or off at a PSDN PAD. You should not set echo off at the PAD when you are using ITF. Echo must be set on at the PAD in order for sent data to appear again on the display after you press the Enter key; it must be set on in order for ITF to send files, library members, and DW/36 documents.

*Note: All data sent either from the display or from a data file, library member, or DW/36 document is assumed to be EBCDIC and is translated to ASCII. All data received is assumed to be ASCII and is translated to EBCDIC before being displayed or placed in a file or member. The exception is when sending control characters from the ATTN key screen.*

## ITF Command Keys

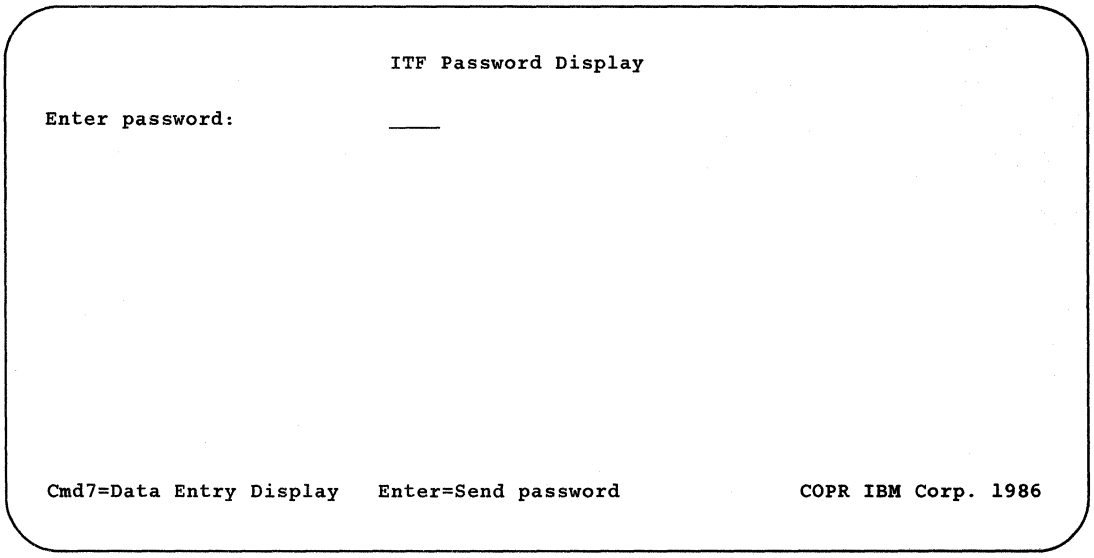
Command keys have the following functions under ITF:

- *Cmd1-Start send/receive:* To select a library member, a file, or a DW/36 document, press Cmd1. The Start Send/Receive Process display appears; on this display, you can tell ITF whether to send data from a member, file, or document or receive data into a member or file. This display is described under "Sending or Receiving a Library Member, Data File, or DisplayWrite/36 Document" later in this chapter.

- *Cmd2-Stop send/receive:* To stop the sending or receiving of a library member, file, or document, press Cmd2. ITF immediately stops sending or receiving and returns control to you.

- *Cmd3-Phone list:* To make a switched connection to the remote end, press Cmd3. The Phone List Display appears. This display is described under "Starting ITF" earlier in this chapter. If you are using PAD emulation or an asynchronous/X.25 line, Cmd3 does not appear on the ITF Data Entry Display.

- *Cmd4-Send password:* If the network asks for your password, press Cmd4. ITF presents the ITF Password Display.




Type in your password to the message application and press the Enter key. So that your password remains secure, the characters will **not** be displayed as you type them.

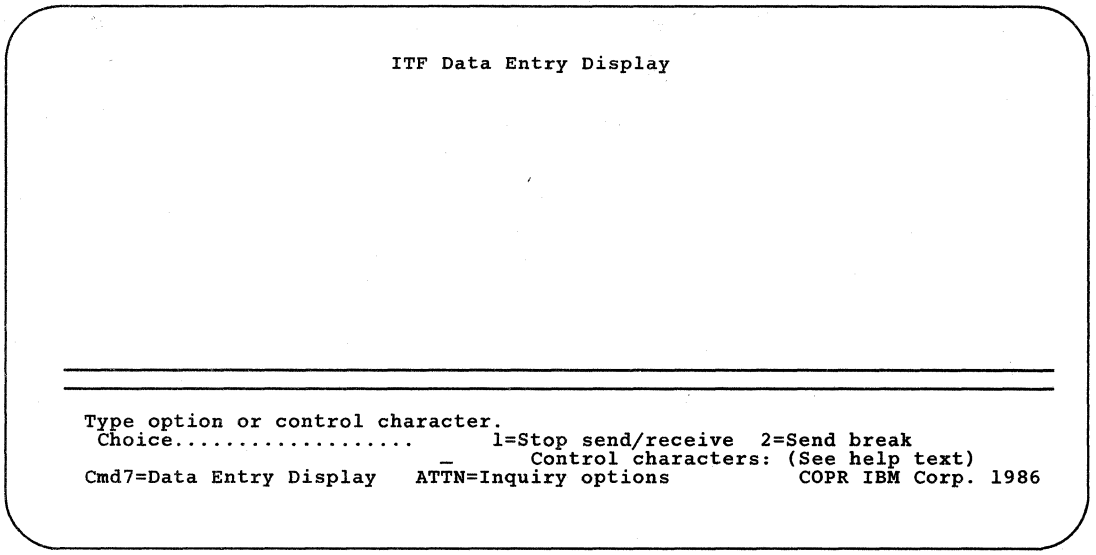
When you press the Enter key, ITF sends your password.

- *Cmd7-End ITF:* If you press Cmd7 from the ITF Data Entry Display, ITF ends. Pressing Cmd7 from any other ITF display returns you to the ITF Data Entry Display; in this case, ITF ignores any data that you may have typed on the display. You cannot end ITF if a data send or receive is in operation.

- *Cmd8-Redial:* If you want to call the last telephone number called from the Phone List Display again, press Cmd8. The system automatically calls the same number. Cmd8 can only be used to redial numbers that were dialed from the Phone List Display. If you are using PAD emulation or an asynchronous/X.25 line, Cmd8 does not appear on the ITF Data Entry Display.

- *Cmd9-Send data as is:* In normal data entry, a carriage return character is added to the end of the data when you press the Enter key. If you want to send data without a carriage return (for example, to send commands to a modem), press Cmd9 instead of the Enter key after typing data on the data entry line.

If you press the Attn key, new lines appear at the bottom of the ITF Data Entry Display:




- *1-Stop send/receive:* Select option 1 to stop sending a member, file, or document or to stop receiving data into a member or file. The remote station is not affected by this option. It may continue to send data, but the data will not be received into a member or file.

- *2-Send break:* Select option 2 to send a break signal to the remote station.

- *Control characters:* The control characters are used to perform program functions, such as a pause in receiving data. Their meanings are determined by the application with which you are communicating. The help text tells you which control characters correspond with the work station keys.

  Type a 1 or a 2 to select an option. To send a control character, type the letter that corresponds to it. You do not need to press the Enter key. Press the Attn key to display the system inquiry options.

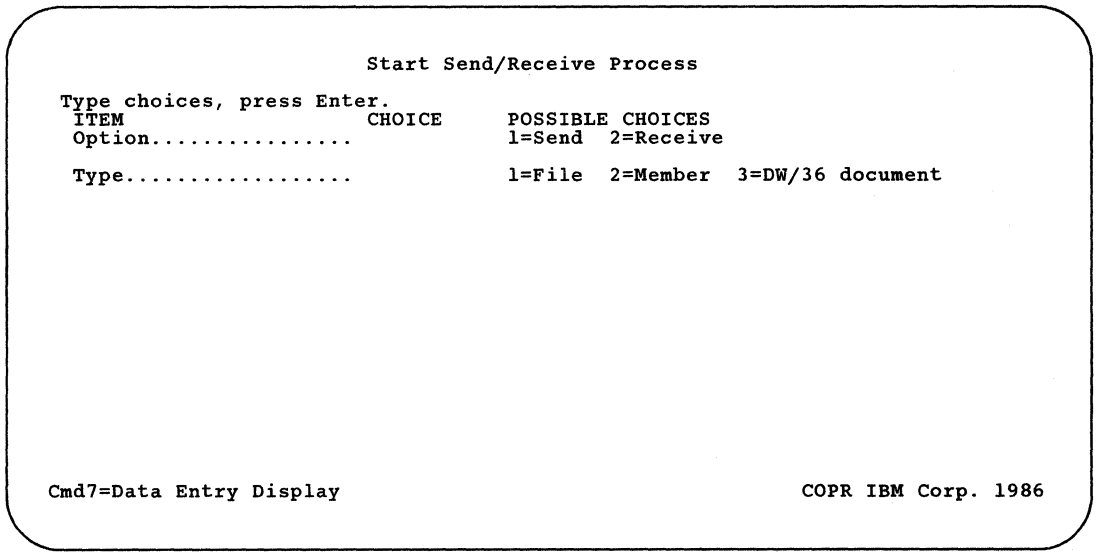
## Sending or Receiving a Library Member, Data File, or DisplayWrite/36 Document

With ITF, you can send a library member, data file, or DW/36 document; or, you can place received data into a library member or data file.

ITF will append a carriage return character (CR) to the end of each record sent from a data file, library member, or DW/36 document.

*Note: There is no verification of data integrity performed by ITF; unpredictable results may occur if you send a data file or library member that contains non-text data (such as hexadecimal characters).*

To select a member, file, or document, press Cmd1 when the ITF Data Entry Display is shown. The Start Send/Receive Process display appears:



### Sending or Receiving a Library Member

To send a library member, enter 1 in the Option field; to receive data into a library member, enter 2 in the Option field. Enter 2 in the Type field, and press the Enter key. The following display appears:

```
                        Start Send/Receive Process

Type choices, press Enter.
 ITEM                   CHOICE      POSSIBLE CHOICES
 Option................ 1           1=Send  2=Receive

 Type.................. 2           1=File  2=Member  3=DW/36 document

 Library name.......... TESTLIBR   Name of library

 Member name........... TESTMMBR   Name of member









Cmd7=Data Entry Display                                  COPR IBM Corp. 1986
```

Enter the name of the library member and the library that contains this member.

If you entered 1 in the Option field, an additional prompt appears:

```
                        Start Send/Receive Process

Type choices, press Enter.
 ITEM                   CHOICE      POSSIBLE CHOICES
 Option................ 1           1=Send  2=Receive

 Type.................. 2           1=File  2=Member  3=DW/36 document

 Library name.......... TESTLIBR   Name of library

 Member name........... TESTMMBR   Name of member

 Member type........... 1           1=Source  2=Procedure




Cmd7=Data Entry Display                                  COPR IBM Corp. 1986
```

Type 1 in the member type field if you are sending a source member, or 2 if you are sending a procedure member. Press the Enter key. ITF immediately starts sending the member. As each record in the member is sent, the PAD echoes it so that it appears on your display. (ITF cannot send library members if the PAD does not echo.) When the last record is sent, ITF displays a message:

```
Last record sent
```

If you entered a 2 in the Option field and if the member that you specify does not already exist in the library, ITF prompts you for more information.

```
                        Start Send/Receive Process

Type choices, press Enter.
 ITEM                   CHOICE      POSSIBLE CHOICES
 Option................ 2           1=Send  2=Receive

 Type.................. 2           1=File  2=Member  3=DW/36 document

 Library name.......... TESTLIBR   Name of library

 Member name........... TESTMMBR   Name of member

 Member type........... 1           1=Source  2=Search for header

 Record size........... 120         Size of record (80,96,120)




Cmd7=Data Entry Display                                  COPR IBM Corp. 1986
```

When ITF sends a library member that is not a normal source member, it builds a header that is sent as the first record of the member. ITF builds headers for procedure members, TMS source members, BGU format members, screen format members, or documents and files of documents. This header contains the library member type, the member subtype, the number of records in the member, and the length of the records.

If you are receiving a library member that was sent with a header, specify a 2 in the member type field. ITF then uses the header record to create a new library member. Only data received after the header record is written to the new member.

If you are receiving a normal source member, specify a 1 in the member type field. All data received is written to the library member.

Enter 1 or 2 in the member type field, and enter the record size in characters.

*Note: If you are receiving data into a library member, users at other work stations cannot send data from or receive data into that library until your operation is completed.*

If the member that you specify already exists in the library, ITF asks if you want to replace the existing member with the data received.

```
                        Start Send/Receive Process

Type choices, press Enter.
 ITEM                   CHOICE      POSSIBLE CHOICES
 Option................ 2           1=Send  2=Receive

 Type.................. 2           1=File  2=Member  3=DW/36 document

 Library name.......... TESTLIBR   Name of library

 Member name........... TESTMMBR   Name of member

 Replace............... 1           1=Yes  2=No









Cmd7=Data Entry Display                                  COPR IBM Corp. 1986
```

If you enter 1 (Yes), ITF writes the received data into the existing library member, writing over the existing data. If you enter 2 (No), you receive an error message, and you must try again with another member.

*Note: ITF cannot append data to an existing library member.*

### Sending or Receiving a Data File

To send a data file, type 1 in the Option field; to receive data into a file, type 2 in the Option field. Type 1 in the Type field, and press the Enter key. The following display appears:

```
                        Start Send/Receive Process

Type choices, press Enter.
 ITEM                   CHOICE     POSSIBLE CHOICES
 Option................ 1          1=Send  2=Receive

 Type.................. 1          1=File  2=Member  3=DW/36 document

 File name............. TESTFILE   Name of file

 File date............. 091085     Date of file (optional)









Cmd7=Data Entry Display                                  COPR IBM Corp. 1986
```

Enter the name of the data file. You can also enter the file date, but this field is optional. If you do not enter a file date and more than one file with this name exists, ITF uses the file with the most current date.

If you entered a 1 in the Option field, ITF immediately starts sending. Each record of the file appears on the display as the receiving application echoes it back. (ITF cannot send files if the receiving application does not echo.) When the last record is sent, ITF displays a message:

```
Last record sent
```

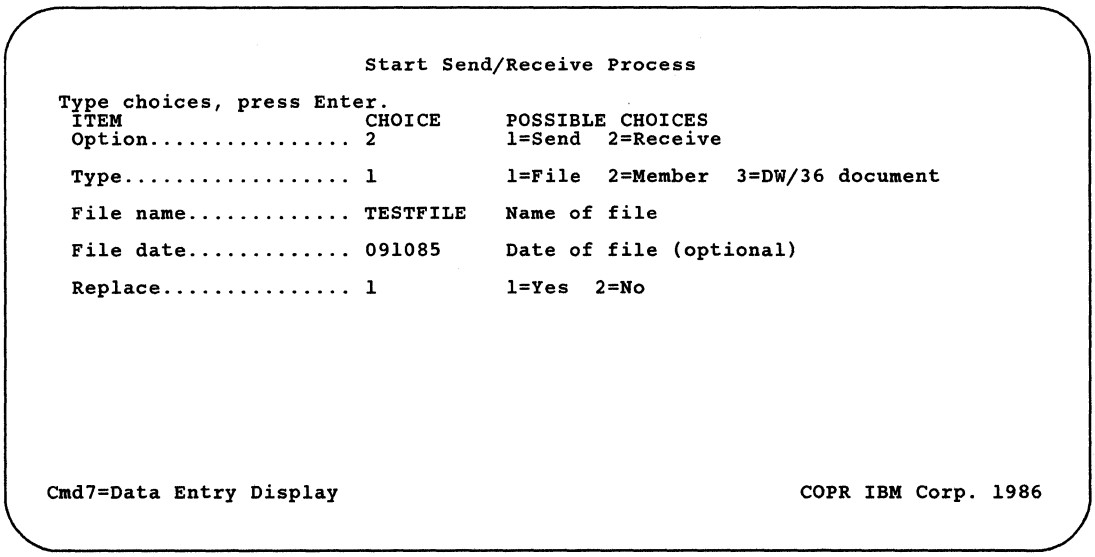
*Note: ITF can send files with record lengths up to 2048 characters. However, some applications can receive only shorter records. For example, some electronic mail services can receive only files with record lengths of 132 characters or less. If the application receiving the file cannot accept files with the record length that you are sending, unpredictable results may occur.*

If you entered a 2 in the Option field, ITF prompts you for more information. If the file that you specify is a new file, the following prompt appears:

```
                        Start Send/Receive Process

Type choices, press Enter.
 ITEM                   CHOICE      POSSIBLE CHOICES
 Option................ 2           1=Send  2=Receive

 Type.................. 1           1=File  2=Member  3=DW/36 document

 File name............. TESTFILE    Name of file

 File date............. 091085      Date of file (optional)


 Record size........... 080         Size of record




Cmd7=Data Entry Display                                  COPR IBM Corp. 1986
```

Enter the record size. Valid values are from 1 to 999.

If the file that you specify already exists, ITF asks if you want to replace the existing file with the new data.




If you enter 1 (Yes), ITF writes the message into the existing file, writing over the existing data. If you enter 2 (No), the message is appended to the end of the existing data.

*Note: If you specified an existing file with record lengths greater than 2048, you receive an error message.*

## Sending DW/36 Documents

To send a DW/36 document, enter 1 in the Option field. Enter 3 in the Type field and press the Enter key. The following display appears:

```
                        Start Send/Receive Process

Type choices, press Enter.
 ITEM                   CHOICE     POSSIBLE CHOICES
 Option................ 1          1=Send  2=Receive

 Type.................. 3          1=File  2=Member  3=DW/36 document

 Document name......... TESTDOCT   Name of document

 Folder name........... TESTFLDR   Name of folder









Cmd7=Data Entry Display                                  COPR IBM Corp. 1986
```

Enter the name of the document, or enter ALL for all of the documents in a folder. Enter the name of the folder that contains the document(s). After you press the Enter key, the ITF Data Entry Display reappears.

ITF immediately starts sending the document(s). Each record appears on the display as the PAD echoes it back. (ITF cannot send if the PAD does not echo.) When the last record is sent, ITF displays a message:

```
nnnn document(s) sent
```

ITF will only send the first 120 characters of each line in the document. Any characters beyond the first 120 are truncated and the following message appears:

```
nnnn document(s) sent. Data truncated.
```

If ALL was specified for the document name, all of the documents in the folder will be sent.

*Notes:*

1. *ITF only sends the base line, superscript, and subscript text from a document. No document control characters are sent. Superscripts and subscripts are treated as separate records.*

2. *ITF attempts to maintain line integrity such as blank lines.*

3. *Blank lines are sent between pages of a document and between documents.*

### Sending a File of DW/36 Documents

You will have to use the TEXTDOC PRTFILE command to resolve the documents to the file that you select. See the DW/36 online information for more information on TEXTDOC PRTFILE. After you have resolved the documents to a file, you can use the ITF Start Send/Receive Process display to send the file of documents as a normal file. After the file is sent, the messages that indicate the number of documents and whether the documents were truncated are displayed.

### Receiving DW/36 Documents

ITF cannot directly receive DW/36 documents. The information should be received as a new library member with a record length of 120. To receive these documents, use the Start Send/Receive Process display and specify the Member type as a 2 (Search for header).

When ITF sends documents or a file of documents, a header is sent as the first record. ITF searches for the header and uses the information in the header to create the library member.

# Appendix A. PAD Emulation

You can configure the asynchronous communications subsystem to emulate an X.25 packet assembler/disassembler (PAD). The System/36 PAD supports terminals running the interactive terminal facility or application programs, and allows them to communicate with any X.25 host system that communicates with asynchronous terminals.

The System/36 PAD follows CCITT recommendations X.3, X.28, and X.29. These recommendations are as follows:

- X.3 defines the parameters that the PAD uses in controlling the session.
- X.28 defines the interface between the PAD and a terminal.
- X.29 defines how PAD messages are exchanged between an X.25 host system and the PAD.

## Recommendation X.3

The following X.3 parameters are used to control the session. The host system can set and/or read these parameters by sending a SET, SET and READ, or READ X.29 PAD message to the System/36 PAD.

Parameters marked *not meaningful* in this table are those that System/36 ignores because they are not used by the terminals supported by System/36.

| Parameter | Description | Values and Meanings |
|---|---|---|
| 1 | Escape to command mode | 0: No escape possible<br>1: Escape possible |
| 2 | Echo | 0: PAD does not echo<br>1: PAD will echo characters |
| 3 | Data forwarding characters | 0: None<br>1: Alphamerics<br>2: Carriage return<br>4: Escape<br>8: Editing<br>16: Terminators<br>32: Form effectors<br>64: Other control characters<br>128: Other characters |
| 4 | Idle timer | 0: No time-out<br>1-255: 1 to 255 multiples of 0.05 seconds |
| 5 | PAD suspension of input from terminal (System/36 PAD never suspends input) | 0: No (PAD cannot suspend input)<br>1: Yes (PAD can suspend input) |
| 6 | Suppression of service signals | 0: Suppress signals<br>1: Deliver signals |
| 7 | Break options | 0: Do nothing<br>1: Send interrupt<br>2: Reset<br>4: Send break indication<br>8: Escape to command mode<br>16: Discard output |
| 8 | Discard output | 0: Deliver output<br>1: Discard output |
| 9 | Carriage return padding (not meaningful) | 0: None<br>1-7: Number of character delays |
| 10 | Line folding | 0: None<br>1-255: Number of characters per line before line folding |
| 11 | Terminal speed (not meaningful) | 1-15 |
| 12 | Flow control of PAD | 0: Not possible<br>1: Possible |
| 13 | Line feed insertion after carriage return | 0: None<br>1: To terminal<br>2: From terminal<br>4: In echo |
| 14 | Line feed padding (not meaningful) | 0-7 |
| 15 | Editing | 0: Do not use editing<br>1: Use editing |
| 16 | Character delete (not meaningful) | 0-127 |
| 17 | Line delete (not meaningful) | 0-127 |
| 18 | Line display (not meaningful) | 0-127 |

# Recommendation X.28

Recommendation X.28 describes the interface between the PAD and a terminal. There are two modes of operation: command mode and data transfer mode. While in command mode, the terminal sends commands to the PAD and receives PAD messages as responses to the commands. The commands available to a user are given below.

| Command | Abbreviation | Description |
|---|---|---|
| Connect <address><br><br><0DNIC address> | C | Request connection to specified network address<br><br>***This connect command allows you to connect to a different network by specifying an alternate data network identification code (DNIC) other than the one you specified on display 60.0.*** |
| Reset | RESE | Reset to default X.3 parameter values and reset session to preconnect status |
| Status | STAT | Return network address of terminal location |
| Disconnect | D | Discontinue communications |
| Continue | CONT | Return to data transfer mode from command mode |
| SET <list> | | Set or change X.3 parameter values to those specified in the list |
| PAR? <list> | | Request the current values of the X.3 parameters listed to be displayed |
| SET? <list> | | Request the specified X.3 parameters to be set and displayed |
| <CR>@<CR> (escape sequence) | | Escape from data transfer mode to command mode |

The parameter lists following the SET, SET?, and PAR? commands have the following format:

```
parameter number:value,...,number:value <CR>
```

Where <CR> = carriage return ('0D'X).

*Note: If you are accessing the PAD using ITF, the <CR> is appended automatically by ITF.*

In the following example, parameter 1 is set to 0 and parameter 7 is set to 4:

```
SET 1:0,7:4
```

The following are the messages from the PAD to the terminal in response to the terminal commands.

| Message | Description |
|---|---|
| @ | Acknowledgment and prompt |
| ? | PAD did not understand last command |
| <address> connected | Connection complete to specified address |
| Disconnected | Session is disconnected |
| <address> available | Response to status command when not connected to the specified address |
| Not connected | Response to disconnect command when not connected to a remote system |
| Invalid address | Address supplied with connect command is not a valid address |

The following service signals from the PAD include an X.25 cause code (CC) and diagnostic code (DC). Descriptions of these codes can be found in the manual, *Multiline Communications Attachment Maintenance Information*, SY31-9018.

- < Address > busy CC DC
- < Address > invalid facility request CC DC
- < Address > not reachable CC DC
- < Address > not operating CC DC
- < Address > illegal source address CC DC
- < Address > not responding CC DC
- < Address > remote procedure error CC DC
- < Address > local procedure error CC DC
- < Address > refusing collect connection CC DC
- < Address > still pending CC DC
- < Address > not available CC DC
- < Address > illegal address CC DC
- < Address > network congestion CC DC
- < Address > invalid logical channel type CC DC
- < Address > call user data error CC DC
- < Address > no logical channel available CC DC
- < Address > call cleared CC DC
- < Address > rejecting CC DC

# Recommendation X.29

Recommendation X.29 specifies how PAD messages are sent and received between an X.25 host system and an X.25 PAD. The following X.29 PAD messages are supported by the System/36 PAD.

- SET: Sent by the host system to set PAD X.3 parameters.
- SET and READ: Sent by the host system to set PAD X.3 parameters. The PAD responds by sending a parameter indication.
- READ: Sent by the host system to find out what the PAD X.3 parameters are set to. The PAD responds by sending a parameter indication.
- PARAMETER INDICATION: Sent by the PAD in response to the host system's Set and Read or Read command; it tells the host system what values the PAD parameters are set to.
- INDICATION OF BREAK: Sent by the PAD when the terminal user sends a break signal.
- INVITATION TO CLEAR: Sent by the host system to request the PAD to terminate the connection to the terminal after transmission to the terminal of all previously received data.
- ERROR: Used by the System/36 PAD to indicate to a host system that an invalid PAD message was received.

## Using the Asynchronous Communications Subsystem to Send and Receive X.29 PAD Messages

You can use an SSP-ICF operation code and return code to send and receive X.29 PAD messages within your application programs. The put FMH operation lets you send the SET, SET and READ, and READ PAD messages to an asynchronous terminal; return code 0004 indicates that a PARAMETER INDICATION or ERROR message has been received from the terminal.

The INDICATION OF BREAK PAD message is sent by the fail operation.

You cannot send an INVITATION TO CLEAR PAD message within an application program.

### Put FMH Operation

For an asynchronous communications subsystem using PAD emulation, the put FMH operation sends X.29 PAD messages. The format of the put FMH operation is as follows:

| Operation | Assembler | COBOL | RPG II |
|---|---|---|---|
| Put FMH | PFM | $$SENDNF | $$SENDNF |

The put FMH operation sends SET, SET and READ, and READ messages to the PAD. These messages are used to set parameter values at the PAD or to find out what those values are set to.

The data sent by this operation must be in the following format:

```
code parameter value parameter value parameter value ...
```

*Code* is a 2-digit number that indicates which X.29 message is being sent:

- 02 = SET
- 04 = READ
- 06 = SET and READ

*Parameter* specifies the X.3 parameter that you want to set or to read, followed by the *value* to which you want it set. X.3 parameters and their valid values are listed earlier in this appendix. When you send a READ message, you can enter a blank for each parameter value, because you are not setting values.

If you do not enter any parameters and values:

- For a SET message, all parameters are set to their default values.
- For a SET and READ message, all parameters are set to their default values and the values are returned to the program.
- For a READ message, the values of all parameters are returned to the program.

### Return Code 0004

Return code 0004 indicates that an X.29 PAD message was received. The message may be a PARAMETER INDICATION or an ERROR message.

# Appendix B. Rotary Dial

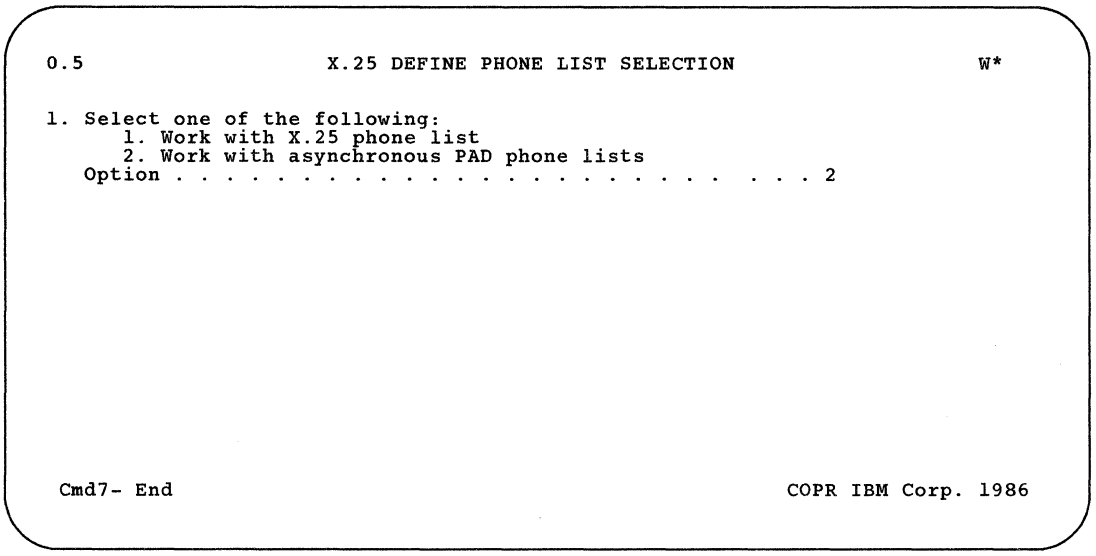
Rotary dial is a function of the System/36 PAD support that automatically calls X.25 network addresses specified in a sequentially ordered phone list. It continues calling for the specified number of retries, or until a successful connection is made. If no successful connection is made, a message is returned to the terminal.

To use this function you must enable an X.25 subsystem with PAD emulation configured. You must also create an asynchronous PAD phone list by executing the DEFINX25 procedure. Rotary dial can be implemented within a user application program or by entering the phone list name via ITF.

*Note: The phone list must reside in the same library as the subsystem configuration. System/36 will only search for the phone list name in the library that is enabled.*

## Creating an Asynchronous PAD Phone List

1. To create an asynchronous PAD phone list, enter the DEFINX25 procedure command. On display 0.5, select option 2 to work with an asynchronous PAD phone list.



2. On display 1.0, select option 1 to create an asynchronous PAD phone list.

```
1.0                    X.25 DEFINE PHONE LIST MENU                      W*
                    Define asynchronous PAD phone list

1. Select one of the following:
      1. Create an asynchronous PAD phone list
      2. Change an asynchronous PAD phone list
      3. Remove an asynchronous PAD phone list
      4. Print an asynchronous PAD phone list
   Option . . . . . . . . . . . . . . . . . . . . . . . . . . 1

 Cmd7-End    Cmd19-Cancel                                 COPR IBM Corp. 1986
```

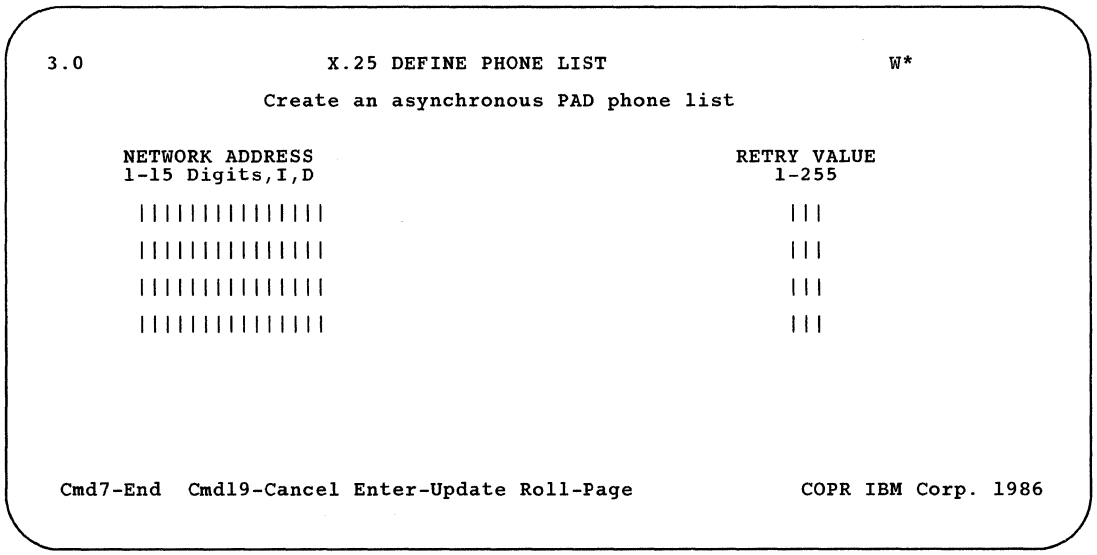
3. On display 2.0, enter information to define the phone list to be created.

```
2.0                  X.25 PHONE LIST MEMBER SELECTION                   W*
                    Create an asynchronous PAD phone list

1. Phone list name . . . . . . . . . . . . . . . . . . . . .
2. Phone list library name . . . . . . . . . . . . . . . . .

 Cmd7-End    Cmd19-Cancel                                 COPR IBM Corp. 1986
```

**1. Phone list name:** Specify the name of the phone list member.

**2. Phone list library name:** Specify the name of the library that contains the phone list.

4. On display 3.0, specify the list of network addresses and the number of times you want each address to be called.




- **Network address:** Specify the address of the network to be called. You can enter up to 15 digits for each address. To add a new entry, enter an **I** in the leftmost position of this field. To delete the current entry, enter a **D** in the leftmost position of this field.

- **Retry value:** Specify the number of times the address should be called.

At this point you have completed defining the phone list.

# Appendix C. Establishing a Communications Link

To use the communications support on your system, you need to establish a communications link between your system and another location.

This appendix contains the form that you should fill in so your users know the steps to use to establish a communications link with the communications support that is on your system.

See the *System Messages* manual for an explanation of messages received.

The steps you can take to determine the cause of a problem for communications lines are described in the manual *System Problem Determination.*

# Using Asynchronous Communications Support

## To Establish the Communications Link

At a display station, perform the following steps:

_____

_____

_____

_____

_____

## To End the Communications Link

At a display station, perform the following steps:

_____

_____

_____

_____

_____

# Glossary

**#LIBRARY.** The library, provided with the system, that contains the System Support Program Product. See *system library.*

**abnormal termination.** A system failure or operator action that causes a job to end unsuccessfully.

**access.** To go to or reach; to get at.

**access method.** The way that records in files are referred to by the system. The reference can be consecutive (records are referred to one after another in the order in which they appear in the file), or it can be random (the individual records can be referred to in any order).

**acquire.** To assign a display station or session to a program.

**acquired session.** A session that has been started by a System/36 program using an acquire operation, or in BASIC, using an OPEN statement.

**adapter.** See *communications adapter.*

**address pool.** In data communications, a collection of multipoint addresses. Each address can be associated with an individual SSP-ICF session.

**addressing.** (1) In data communications, the way that the sending or control station selects the station to which it is sending data. (2) A means of identifying storage locations.

**advanced program-to-program communications (APPC).** Communications support that allows System/36 to communicate with other systems having the same support. APPC is the way that System/36 puts the IBM SNA LU-6.2 protocol into effect.

**Advanced Peer-to-Peer Networking (APPN).** A communications feature that routes data in a network between two or more APPC systems that are not directly attached. See also *node* and *network node.*

**alert.** An error message sent to the system services control point (SSCP) at a host system. On System/36, the problem management portion of the Communications and Systems Management feature is used to generate and send alerts.

**allocate.** To assign a resource, such as a disk file or a diskette file, to perform a specific task.

**alphabetic character.** Any one of the letters A through Z (uppercase and lowercase). Some program products extend the alphabet to include the special characters #, $, and @.

**alphameric.** Consisting of letters, numbers, and often other symbols, such as punctuation marks and mathematical symbols.

**alphanumeric.** See *alphameric.*

**American National Standard Code for Information Interchange (ASCII).** The code developed by ANSI for information interchange among data processing systems, data communications systems, and associated equipment. The ASCII character set consists of 7-bit control characters and symbolic characters.

**American National Standards Institute (ANSI).** An organization sponsored by the Computer and Business Equipment Manufacturers Association for establishing voluntary industry standards.

**ANSI.** See *American National Standards Institute (ANSI).*

**APAR.** See *authorized program analysis report (APAR).*

**APPC.** See *advanced program-to-program communications (APPC).*

**APPN.** See *Advanced Peer-to-Peer Networking (APPN).*

**application program.** A program used to perform an application or part of an application.

**ASCII.** See *American National Standard Code for Information Interchange (ASCII).*

**assembler.** A program that converts assembler language statements to machine instructions.

**assembler instruction statement.** A statement that controls what the assembler does, rather than what the user program does.

**assembler language.** A symbolic programming language in which the set of instructions includes the instructions of the machine and whose data structures correspond directly to the storage and registers of the machine.

**asynchronous transmission.** In data communications, a method of transmission in which the bits included in a character or block of characters occur during a specific time interval. However, the start of each character or block of characters can occur at any time during this interval. Contrast with *synchronous transmission.*

**attribute.** A characteristic. For example, an attribute for a displayed field could be blinking.

**authorized program analysis report (APAR).** A request for correction of a defect in a current release of an IBM-supplied program.

**autoanswer.** In data communications, the ability of a station to receive a call over a switched line without operator action. Contrast with *manual answer.*

**autocall.** In data communications, the ability of a station to place a call over a switched line without operator action. Contrast with *manual call.*

**autocall unit.** A common carrier device that allows System/36 to automatically call a remote location.

**automatic reconnect.** An option specified during system configuration that allows a remote work station controller to be reconnected automatically on a switched or nonswitched line.

**BASIC (beginner's all-purpose symbolic instruction code).** A programming language designed for interactive systems and originally developed at Dartmouth College to encourage people to use computers for simple problem-solving operations.

**batch.** Pertaining to activity involving little or no operator action. Contrast with *interactive.*

**batch BSC.** The System Support Program Product support that provides data communications with BSC computers and devices via the RPG T specification or the assembler $DTFB macroinstruction.

**batch processing.** A processing method in which a program or programs process records with little or no operator action. Contrast with *interactive processing.*

**binary.** (1) Pertaining to a system of numbers to the base two; the binary digits are 0 and 1. (2) Involving a choice of two conditions, such as on-off or yes-no.

**bind command.** An SNA command used to define the protocols for a session. Contrast with *unbind command.*

**bit.** Either of the binary digits 0 or 1. See also *byte.*

**bps.** Bits per second.

**buffer.** (1) A temporary storage unit, especially one that accepts information at one rate and delivers it at another rate. (2) An area of storage, temporarily reserved for performing input or output, into which data is read or from which data is written.

**byte.** The amount of storage required to represent one character; a byte is 8 bits.

**cable thru.** A standard function or special feature that allows multiple work stations to be attached to a particular line.

**call.** (1) To activate a program or procedure at its entry point. Compare with *load.* (2) In data communications, the action necessary in making a connection between two stations on a switched line.

**cancel.** To end a task before it is completed.

**carriage return character (CR).** A format effector that causes the print or display position to move to the first position on the same line.

**carrier.** A continuous frequency that can be modulated with a second (information-carrying) signal.

**CCITT.** Consultative Committee on International Telegraphy and Telephone.

**character.** A letter, digit, or other symbol.

**character key.** A keyboard key that allows the user to enter the character shown on the key. Compare with *command key* and *function key*.

**COBOL (common business-oriented language).** A high-level programming language, similar to English, that is used primarily for commercial data processing.

**code.** (1) Instructions for the computer. (2) To write instructions for the computer. Same as *program*. (3) A representation of a condition, such as an error code.

**command.** A request to the system to perform an operation or a procedure.

**command key.** A keyboard key that is used to request specific programmed actions. Compare with *character key* and *function key*.

**communications.** See *data communications.*

**communications adapter.** A hardware feature that enables a computer or device to become a part of a data communications network.

**communications line.** The line over which data communications takes place; for example, a telephone line.

**communications link.** See *data link.*

**communications security.** A System Support Program Product option that allows the identity of a remote location to be verified before that location can run programs on your system.

**communications subsystem.** See *subsystem.*

**condition.** An expression in a program or procedure that can be evaluated to a value of either true or false when the program or procedure is running.

**configuration.** The group of machines, devices, and programs that make up a data processing system. See also *system configuration.*

**constant.** A data item with a value that does not change. Contrast with *variable.*

**control station.** The primary or controlling computer on a multipoint line. The control station controls the sending and receiving of data.

**current library.** The first library searched for any required members. The current library can be specified during sign-on or while running programs and procedures.

**cursor.** A movable symbol on a display, used to indicate to the operator where to type the next character.

**data area.** A storage area used by a program or device to hold information.

**data circuit-terminating equipment (DCE).** The equipment installed at the user's location that provides all the functions required to establish, maintain, and terminate a connection, and the signal conversion and coding between the data terminal equipment (DTE) and the line.

**data communications.** The transmission of data between computers and/or remote devices (usually over a long distance).

**data link.** The equipment and rules (protocols) used for sending and receiving data.

**data link escape (DLE) character.** In BSC, a transmission control character usually used in transparent text mode to indicate that the next character is a transmission control character.

**data management.** See *disk data management.*

**data mode.** In data communications, a time during which BSC is sending or receiving characters on the communications line.

**data stream.** All information (data and control information) transmitted over a data link.

**data terminal equipment (DTE).** The data processing unit that uses communications lines.

**DCE.** See *data circuit-terminating equipment (DCE).*

**debug.** To detect, locate, and remove errors from a program.

**decimal.** (1) Pertaining to a system of numbers to the base ten; decimal digits range from 0 through 9. (2) A proper fraction in which the denominator is a power of 10.

**default.** See *default value.*

**default value.** A value stored in the system that is used when no other value is specified.

**define-the-file.** A control block containing information that is passed between data management routines and users of the data management routines.

**delete.** To remove. For example, to delete a file.

**disable.** In interactive communications, to end a subsystem and free the area of main storage used by that subsystem. Contrast with *enable.*

**disk data management.** The System Support Program Product support that processes a request to read or write data.

**display.** (1) A visual presentation of information on a display screen. (2) To show information on the display screen.

**display station.** A device that includes a keyboard from which an operator can send information to the system and a display screen on which an operator can see the information sent to or the information received from the system.

**DLE.** See *data link escape (DLE) character.*

**DTE.** See *data terminal equipment (DTE).*

**DTF.** See *define-the-file.*

**duplex.** Pertains to communications in which data can be sent and received at the same time. Same as full duplex. Contrast with *half duplex.*

**EBCDIC.** See *extended binary-coded decimal interchange code (EBCDIC).*

**EBCDIC character.** Any one of the symbols included in the 8-bit EBCDIC set.

**eight-line communications adapter/attachment (ELCA).** A feature that allows up to eight communication lines to be connected to a 5360 System Unit.

**ELCA.** See *eight-line communications adapter/attachment (ELCA).*

**enable.** In interactive communications, to load and start a subsystem. Contrast with *disable.*

**enter.** To type in information from a keyboard and press the Enter key in order to send the information to the computer.

**error code.** See *system reference code.*

**evoke.** To start a program or procedure so that it can communicate with your program.

**expression.** A representation of a value. For example, variables and constants appearing alone or in combination with operators.

**extended binary-coded decimal interchange code (EBCDIC).** A set of 256 eight-bit characters.

**feature.** A programming or hardware option, usually available at an extra cost. For example, Communications is a feature of the System Support Program Product.

**field.** One or more characters of related information (such as a name or an amount).

**file.** A set of related records treated as a unit.

**file name.** The name used by a program to identify a file. See also *label.*

**first-level message.** A message that is issued immediately when an error occurs. See also *second-level message.*

**folder.** A named area on disk that contains documents, profiles, mail, or data definitions. Compare with *library.*

**full duplex.** Same as *duplex.*

**function key.** A keyboard key that requests an action but does not display or print a character. The cursor movement and Help keys are examples of function keys. Compare with *command key* and *character key.*

**function management header.** In SNA, a special record or part of a record that contains control information for the data that follows.

**generic remote location.** A remote location name that is reserved for calls from any remote location defined by the DEFINLOC procedure to an asynchronous subsystem.

**half duplex.** Pertains to communications in which data can be sent in only one direction at a time. Contrast with *duplex.*

**Help key.** A function key that, when pressed, displays online information or some part of the system help support.

**help text.** The part of the system help support that offers additional information about displays and messages.

**hex.** See *hexadecimal.*

**hexadecimal.** Pertaining to a system of numbers to the base sixteen; hexadecimal digits range from 0 (zero) through 9 (nine) and A (ten) through F (fifteen).

**host system.** The primary or controlling computer in a communications network. See also *control station.*

**I/O.** See *input/output (I/O).*

**identifier.** (1) A sequence of bits or characters that identifies a program, device, or system to another program, device, or system. (2) In COBOL, a data name that is unique or is made unique by the correct combination of qualifiers, subscripts, or indexes.

**informational message.** A message that provides information to the operator, but does not require a response.

**initial program load (IPL).** The process of loading the system programs and preparing the system to run jobs.

**input.** Data to be processed.

**input/output (I/O).** Pertaining to either input or output, or both.

**interactive.** Pertaining to activity involving requests and replies as, for example, between an operator and a program or between two programs. Contrast with *batch.*

**interactive communications feature (SSP-ICF).** A feature of the System Support Program Product that allows a program to interactively communicate with another program or system.

**interactive processing.** A processing method in which each operator action causes a response from the program or the system. Contrast with *batch processing.*

**interactive terminal facility (ITF).** An asynchronous communications feature that allows a System/36 to communicate with applications that can send and receive data such as electronic mail, memos, library members, and data files.

**invite.** To ask for input data from either a display station or an SSP-ICF session.

**IPL.** See *initial program load (IPL).*

**job.** (1) A unit of work to be done by a system. (2) One or more related procedures or programs grouped into a procedure.

**job step.** A unit of work represented by a single program or a procedure that contains a single program. A job consists of one or more job steps.

**key.** One or more characters used to identify the record and establish the record's order within an indexed file.

**label.** (1) The name in the disk or diskette volume table of contents or on a tape that identifies a file. See also *file name.* (2) The name that identifies a statement.

**LAN.** See *local area network (LAN).*

**library.** (1) A named area on disk that can contain programs and related information (not files). A library consists of different sections, called library members. Compare with *folder.* (2) The set of publications for a system.

**library member.** A named collection of records or statements in a library. The types of library members are *load member, procedure member, source member,* and *subroutine member.*

**library member subtype.** A specific classification of a library member type. For example, a source member can be identified as a COBOL source member or a DFU source member.

**licensed program.** An IBM-written program that performs functions related to processing user data.

**link level.** A part of Recommendation X.25 that defines the link protocol used to get data into and out of the network across the full-duplex link connecting the subscriber's machine to the network node. LAP and LAPB are the link access protocols recommended by the CCITT.

**link protocol.** See *link level.*

**load.** (1) To move data or programs into storage. (2) To place a diskette into a diskette drive or a diskette magazine into a diskette magazine drive. (3) To insert paper into a printer. (4) To mount a tape or insert a tape cartridge into a tape drive.

**load member.** A library member that contains information in machine language, a form that the system can use directly. Contrast with *source member.*

**load module.** A program in a form that can be loaded into main storage and run. The load module is the output of the overlay linkage editor.

**local.** Pertaining to a device, file, or system that is accessed directly from your system, without the use of a communications line. Contrast with *remote.*

**local area network (LAN).** The physical connection among devices located on the same premises for information transfer.

**log.** (1) To record; for example, to log all messages on the system printer. (2) See *mail log.*

**logical channel.** In a packet switching data network, a path over which data packets flow between the sending data terminal equipment and the network, and between the network and the receiving data terminal equipment.

**mail log.** A record of all the mail sent or received by a user.

**manual answer.** In data communications, a line type requiring operator actions to receive a call over a switched line. Contrast with *autoanswer.*

**manual call.** In data communications, a line type requiring operator actions to place a call over a switched line. Contrast with *autocall.*

**menu.** A displayed list of items from which an operator can make a selection.

**message.** (1) Information sent to one or more users or display stations from a program or another user. A message can be either displayed or printed. (2) An indication of the condition of the system sent by the system. (3) For IMS/IRSS, a unit of data sent over the communications line.

**message identification.** A field in the display or printout of a message that directs the user to the description of the message in a message guide or a reference manual. This field consists of up to four alphabetic characters, followed by a dash, followed by the message identification code.

**message identification code (MIC).** A four-digit number that identifies a record in a message member. This number can be part of the message identification.

**MIC.** See *message identification code (MIC).*

**MLCA.** See *multiline communications adapter/attachment (MLCA).*

**modem.** See *modulator-demodulator (modem).*

**modified data tag.** A bit in each input field that, when set, causes that field to be transferred to the host system.

**modulation.** Changing the frequency or size of one signal by using the frequency or size of another signal.

**modulator-demodulator (modem).** A device that converts data from the computer to a signal that can be transmitted on a communications line, and converts the signal received to data for the computer.

**multiline communications adapter/attachment (MLCA).** A feature that allows up to four communication lines to be connected to System/36 with a 5360 or 5362 System Unit.

**multipoint.** In data communications, pertains to a network that allows two or more stations to communicate with a single system on one line.

**network.** A collection of data processing products connected by communications lines for information exchange between stations.

**network node.** A node which is capable of performing the intermediate routing functions, directory services, and route selection services in an APPC network.

**node.** (1) An addressable location in a communications network that provides host processing services. (2) A point where packets are received, sorted, and forwarded to another node (or DTE) according to a routing method the network has defined.

**node identification.** A string of characters that identifies a node to the system.

**nonswitched line.** A connection between computers or devices that does not have to be established by dialing. Contrast with *switched line.*

**numeric.** Pertaining to any of the digits 0 through 9.

**offline.** Neither controlled directly by, nor communicating with, the computer, or both. Contrast with *online.*

**online.** Being controlled directly by, or directly communicating with, the computer, or both. Contrast with *offline.*

**operation.** A defined action, such as adding or comparing, performed on one or more data items.

**operation code.** (1) A code used to represent the operations of a computer. (2) In SSP-ICF, a code used by a System/36 application program to request SSP-ICF data management and/or the subsystem to perform an action. For example, the operation $$SEND asks that data be sent.

**operation control language (OCL).** A language used to identify a job and its processing requirements to the System Support Program Product.

**optional network facilities.** Facilities a packet switching data network user may request when establishing a virtual circuit. See also *reverse charging*, *closed user group*, and *throughput class negotiation.*

**output.** The result of processing data.

**packet.** A data transmission information unit. It has a header on the front that indicates the destination of the packet. Commonly used data field lengths in packets are 128 or 256 bytes.

**packet assembly/disassembly (PAD).** A functional unit that enables data terminal equipment (DTEs) not equipped for packet switching to access a packet-switched network.

**packet level.** A part of Recommendation X.25 that defines the protocol for establishing logical connections between two DTEs and for transferring data on these connections.

**packet switching.** The act of transferring and routing packets from source to destination based on information contained in their headers.

**packet switching data network (PSDN).** A communications network that uses packet switching as a means of transmitting data.

**packet window.** A specified number of packets that can be sent by the DTE before it receives an acknowledgement.

**PAD.** See *packet assembly/disassembly (PAD).*

**parameter.** A value supplied to a procedure or program that either is used as input or controls the actions of the procedure or program.

**password.** A string of characters that, when entered along with a user ID, allows an operator to sign on to the system.

**password security.** A System Support Program Product option that helps prevent the unauthorized use of a display station, by checking the password entered by each operator at sign-on.

**permanent virtual circuit (PVC).** A virtual circuit that has a logical channel permanently assigned to it at each DTE. The usual call establishment protocol is therefore not required.

**phone list.** A list of telephone numbers to be called using a communications program and the autocall or X.25 feature.

**physical connection.** See *physical level (X.25).*

**physical level (X.25).** A standard that defines the electrical, physical, functional, and procedural methods used to control the physical link running between the DTE and the DCE.

**point-to-point line.** A communications line that connects a single remote station to a computer.

**procedure.** A set of related operation control language statements (and, possibly, utility control statements and procedure control expressions) that cause a specific program or set of programs to be performed.

**procedure command.** A command that runs a procedure.

**procedure member.** A library member that contains the statements (such as operation control language statements) necessary to perform a program or set of programs.

**program.** (1) A sequence of instructions for a computer. See *source program* and *load module.* (2) To write a sequence of instructions for a computer. Same as *code.*

**program product.** A licensed program for which a fee is charged.

**prompt.** A displayed request for information or operator action.

**protocol.** A set of rules governing the communication and transfer of data between two or more devices in a communications system.

**PSDN.** See *packet switching data network (PSDN).*

**receive time-out.** In data communications, the result of no data being received in a given period of time.

**Realtime Interface Co-Processor.** A feature that allows up to three communications lines to be connected to a System/36 with a 5364 System Unit.

**Recommendation X.25.** A document, CCITT Recommendation X.25, that outlines standards for the connection of processing equipment to a packet switching data network.

**remote.** Pertaining to a device, file, or system that is accessed by your system through a communications line. Contrast with *local.*

**remotely started session.** A session started by an incoming procedure start request from the remote system. Contrast with *acquired session.*

**return code.** In data communications, a value generated by the system or subsystem that is returned to a program to indicate the results of an operation issued by that program.

**reverse charging.** A packet switching data network optional facility. It enables the DTE to request that the cost of a communications session it initiates be charged to the DTE that is called. See also *optional network facilities.*

**rotary dial.** (1) In a switched system, the conventional dialing method that creates a series of pulses to identify the called station. (2) A function of the System/36 PAD support that automatically calls X.25 network addresses specified in a sequentially ordered phone list.

**routine.** A set of statements in a program that causes the system to perform an operation or a series of related operations.

**RPG.** A programming language specifically designed for writing application programs that meet common business data processing requirements.

**run.** To cause a program, utility, or other machine function to be performed.

**RWS.** Remote work station.

**second-level message.** A message that supplies additional information about an error condition when the Help key is pressed for a first-level message. See also *first-level message.*

**session.** (1) The logical connection by which a System/36 program or device can communicate with a program or device at a remote location. (2) The length of time that starts when an operator signs on the system and ends when the operator signs off the system.

**sign off.** To end a session at a display station.

**sign on.** (Verb) To begin a session at a display station.

**sign-on.** (Noun) The action an operator uses at a display station in order to begin working at the display station.

**single line communications adapter/attachment (SLCA).** In data communications, a feature that allows a single communications line to be connected to System/36.

**SLCA.** See *single line communications adapter/attachment (SLCA).*

**source member.** A library member that contains information in the form in which it was entered, such as RPG specifications. Contrast with *load member.*

**source program.** A set of instructions that are written in a programming language and that must be translated to machine language before the program can be run.

**SSCP.** See *system services control point (SSCP).*

**SSP.** See *System Support Program Product (SSP).*

**SSP-ICF.** See *interactive communications feature (SSP-ICF).*

**statement.** An instruction in a program or procedure. (COBOL) A syntactically valid combination of words and symbols, beginning with a verb, that is written in the Procedure Division.

**station.** A computer or device that can send or receive data.

**status.** A condition. For example, the status of a printer, a job, or a communications line.

**subroutine.** A group of instructions that can be called by another program or subroutine.

**subroutine member.** A library member that contains information that must be combined with one or more members before being run by the system.

**subsystem.** The part of communications that handles the requirements of the remote system, isolating most system-dependent considerations from the application program.

**subtype.** See *library member subtype.*

**switched line.** In data communications, a connection between computers or devices that is established by dialing. Contrast with *nonswitched line.*

**switched virtual circuit (SVC).** A virtual circuit that is requested from the network through a virtual call. It is released when the virtual circuit is cleared.

**synchronous.** Occurring in a regular or predictable sequence.

**synchronous transmission.** In data communications, a method of transmission in which the sending and receiving of characters is controlled by timing signals. Contrast with *asynchronous transmission.*

**system.** The computer and its associated devices and programs.

**system configuration.** A process that specifies the machines, devices, and programs that form a particular data processing system.

**system library.** The library, provided with the system, that contains the System Support Program Product and is named #LIBRARY.

**system program.** An IBM-supplied program that is installed on the system. The System Support Program Product (SSP) is an example.

**system reference code.** A four-character code that contains information for a service representative. This code either is provided as part of a message or is displayed on the control panel.

**system services control point (SSCP).** A focal point within an SNA network for managing the configuration, coordinating network operator and problem determination requests, and providing directory support and other session services for network users.

**System Support Program Product (SSP).** A group of licensed programs that manage the running of other programs and the operation of associated devices, such as the display station and printer. The SSP also contains utility programs that perform common tasks, such as copying information from diskette to disk.

**system unit.** The part of the system that contains the processing unit, the control panel, the disk drive and the disk, and either a diskette drive or a diskette magazine drive.

**temporary-text-delay (TTD) character.** A BSC transmission control character that indicates to the receiving station that there is a temporary delay in the transmission of data.

**terminal.** In data communications, a device, usually equipped with a keyboard and a display device, capable of sending and receiving information over a communications line.

**transparent text mode.** A mode that allows BSC to send and receive messages containing any of the 256 character combinations in hexadecimal, including transmission control characters.

**unbind command.** An SNA command used to reset the protocols for a session. Contrast with *bind command.*

**unique.** The only one.

**user ID.** See *user identification (user ID).*

**user identification (user ID).** A string of characters that identifies a user to the system.

**valid.** (1) Allowed. (2) True, in conforming to an appropriate standard or authority.

**variable.** A name used to represent a data item whose value can change while the program is running. Contrast with *constant.*

**virtual circuit.** A logical connection established between two DTEs. It can be permanent, that is, defined when you subscribe to your network port, or it can be dynamically established when creating a switched virtual circuit.

**volume table of contents (VTOC).** An area on a disk or diskette that describes the location, size, and other characteristics of each file, library, and folder on the disk or diskette.

**VTOC.** See *volume table of contents (VTOC).*

**work station.** A device that lets people transmit information to or receive information from a computer; for example, a display station or printer.

**X.21.** In data communications, a specification of the CCITT that defines the connection of data terminal equipment to an X.21 (public data) network.

**X.21 feature.** The feature that allows System/36 to be connected to an X.21 network.

**X.21 short hold mode.** An option specified during system configuration that allows a circuit switched line to be disconnected when the line is not active.

**X.25.** In data communications, a specification of the CCITT that defines the interface to an X.25 (packet switching) network.

**X.75.** A standard that defines ways of interconnecting two X.25 networks.

# Index

## Special Characters



## A

## B

## C



## D

## E

## F



## G



## H



## I

## J



## K



## L



## M

## N



## O



## P

## Numerics

## READER'S COMMENT FORM

**Please use this form only to identify publication errors or to request changes in publications.** Direct any requests for additional publications, technical questions about IBM systems, changes in IBM programming support, and so on, to your IBM representative or to your IBM-approved remarketer. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

☐ If your comment does not need a reply (for example, pointing out a typing error), check this box and do not include your name and address below. If your comment is applicable, we will include it in the next revision of the manual.

☐ If you would like a reply, check this box. Be sure to print your name and address below.

Page number(s): Comment(s):

**Please contact your IBM representative or your IBM-approved remarketer to request additional publications.**

Name _____

Company or Organization _____

Address _____

_____

City State Zip Code

Phone No. _____

Area Code

No postage necessary if mailed in the U.S.A.

Fold and tape. **Please do not staple.**

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

# BUSINESS REPLY MAIL

FIRST CLASS / PERMIT NO. 40 / ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

**International Business Machines Corporation**
Information Development
Department 245
Rochester, Minnesota, U.S.A. 55901

Fold and tape. **Please do not staple.**

IBM

IBM®

# Using the Asynchronous Communications Support

International Business Machines Corporation

File Number
S36-38

Order Number
SC21-9143-2

Printed in U.S.A.