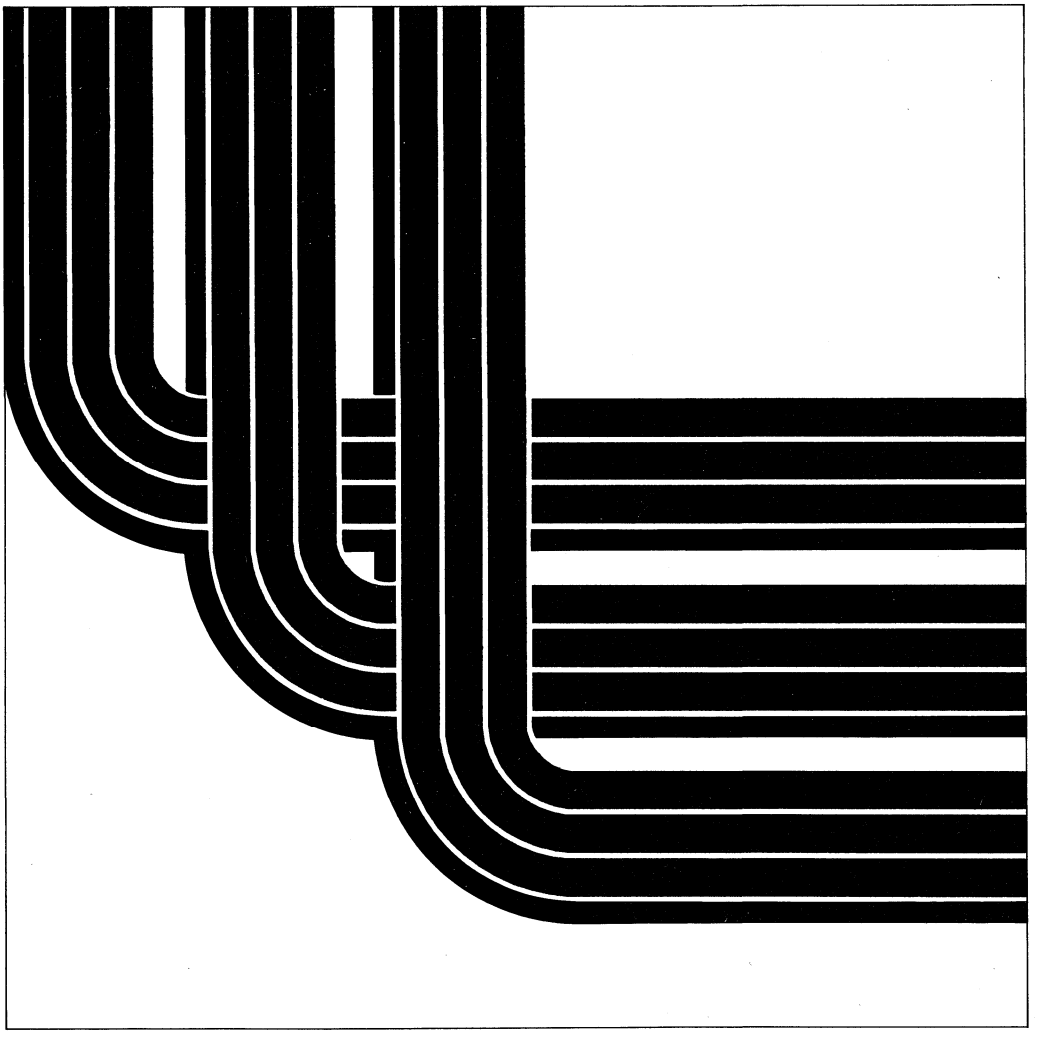
Application System/400™

SC41-9592-00

Communications: Asynchronous Communications Programmer's Guide

Version 2



Application Development

Take Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page vii.

First Edition (May 1991)

This edition applies to the licensed program IBM Operating System/400 (Program 5738-SS1), Version 2 Release 1 Modification 0, and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the proper edition for the level of the product.

Order publications through your IBM representative or the IBM branch serving your locality. Publications are not stocked at the address given below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, you may address your comments to:

Attn Department 245
IBM Corporation
3605 Highway 52 N
Rochester, MN 55901-7899

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you or restricting your use of it.

Contents

Figures

Notices

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 or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577.

The following terms, denoted by an asterisk (*) in this publication, are trademarks of the IBM Corporation in the United States and/or other countries:

Application System/400	AS/400	C/400
COBOL/400	FORTRAN/400	IBM
OfficeVision	Operating System/400	OS/400
RPG/400	400	

The following term, denoted by a double asterisk (**) in this publication, is a trademark of the company listed as follows:

Telemail	GTE Telenet Communications

This publication could contain technical inaccuracies or typographical errors.

This guide may refer to products that are announced but are not yet available.

Information that has changed since Version 1 Release 3 Modification 0 is indicated by a vertical bar (|) to the left of the change.

This publication contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

This guide contains small programs that are furnished by IBM as simple examples to provide an illustration. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. All programs contained herein are provided to you "AS IS". THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE EXPRESSLY DISCLAIMED.

Programming Interface

This programmer's guide is intended to help the customer write communications programs using asynchronous communications. It contains programming information needed to use the AS/400 asynchronous communications support. The *Asynchronous Communications Programmer's Guide* contains no programming interfaces for customers.

About This Guide

This guide supplies the programming information you need to use the AS/400 asynchronous communications support. This guide and the *Communications: Intersystem Communications Function Programmer's Guide*, SC41-9590 are intended to be used together. You should be familiar with the concepts explained in the *ICF Programmer's Guide* and apply those concepts to the detailed information presented here for asynchronous communications.

This guide does not discuss the use or configuration of ASCII work stations attached to work station controllers. For information about attaching ASCII work stations to the AS/400 system, see the *ASCII Work Station Reference and Example*, SA41-9922.

You may need to refer to other IBM manuals for more specific information about a particular topic. The *Publications Guide*, GC41-9678, provides information on all the manuals in the AS/400 library.

For a list of related publications, see the "Bibliography."

Who Should Use This Guide

This guide is intended for programmers who write communications programs using asynchronous communications. It may be used by AS/400 programmers and programmers using other systems and devices that communicate with the AS/400 system using asynchronous communications.

This guide also contains information for the AS/400 user who needs information about how to use the interactive terminal facility (ITF).

Before you use this guide, you should be familiar with the following information:

- AS/400 programming and communications terminology.
- Terminology of the remote system or devices.
- General communications concepts. AS/400 communications concepts are covered in the *System Concepts*, GC41-9802. In addition, specific communications topics are discussed in the online index search. For more information on basic communications, you can also refer to the Discover/ IBM AS/400 course in the communications module. The Discover/ IBM AS/400 course can be ordered separately.
- Communications configuration information for asynchronous support as described in the *Communications: Operating System/400* Communications Configuration Reference*, SC41-0001.
- Intersystem communications function (ICF) support described in the *Communications: Intersystem Communications Function Programmer's Guide*, SC41-9590.
- If you are using asynchronous communications over X.25 lines with integrated packet assembler/disassembler (PAD), you should be familiar with CCITT recommendations X.3, X.28, and X.29. For more information about X.25 line capabilities, see the *Communications: X.25 Network Guide*, SC41-0005.

Chapter 1. Introduction to Asynchronous Communications Support

IBM* Operating System/400* (OS/400*) **asynchronous communications** support allows an AS/400* application program to exchange data with a remote system or device using either an asynchronous (start-stop) or X.25 line. AS/400 application programs can be written in COBOL/400*, RPG/400*, C/400*, or FORTRAN/400* languages. Asynchronous communications support includes file transfer support (also used with other communications types) and interactive terminal facility (ITF).

Asynchronous communications support provides program-to-program and program-to-device communications between systems that use asynchronous (start-stop) or X.25 lines. For X.25 lines, it also supplies an integrated **packet assembler/disassembler (PAD)**[1] that follows CCITT recommendations X.3, X.28, and X.29.

File transfer support (FTS), called from your application program, is a function of the operating system that moves file members from one system to another by using asynchronous, APPC, or BSCEL communications support. See the *ICF Programmer's Guide* for more information about file transfer support.

Interactive terminal facility (ITF) allows AS/400 work stations to connect to applications such as the Telemail** service of the Telenet data network. Using ITF, you can send and receive data, memos, and AS/400 file members. You can also send text from OfficeVision/400* documents. See Chapter 8 for more information about ITF.

AS/400 programs can start programs on a remote system, and the remote system can start programs on the local system. Security options for both systems are supported.

Note: It is the responsibility of the application program to provide error detection, recovery, and data acknowledgement. Data may be lost or received out of sequence if the application program does not provide these checks. When an asynchronous (start-stop) line is used, the physical line can be switched or nonswitched. When an X.25 line is used to connect directly to a packet-switching data network (PSDN), the physical line is nonswitched, but the connection through the network to another system can be a permanent virtual circuit (PVC) or a switched virtual circuit (SVC). A **permanent virtual circuit (PVC)** is a virtual circuit that has a logical channel permanently assigned to it at each data terminal equipment (DTE). A call establishment protocol is not required. The permanent virtual circuit establishes the identity of the called party within the network services contract. A **switched virtual circuit (SVC)** is a virtual circuit that is requested by a virtual call. It is released when the virtual circuit is cleared.

The number of communication lines available for asynchronous communications is dependent on the size of your system and the type of communications adapters attached.

Figure 1-1 on page 1-2 shows an overview of the OS/400 asynchronous communications support.

[1] A functional unit that enables data terminal equipment (DTE) not equipped for packet switching to use a packet-switched network. The **data terminal equipment (DTE)** is that part of a data link that sends data, receives data, and provides the data communications control function according to protocols.

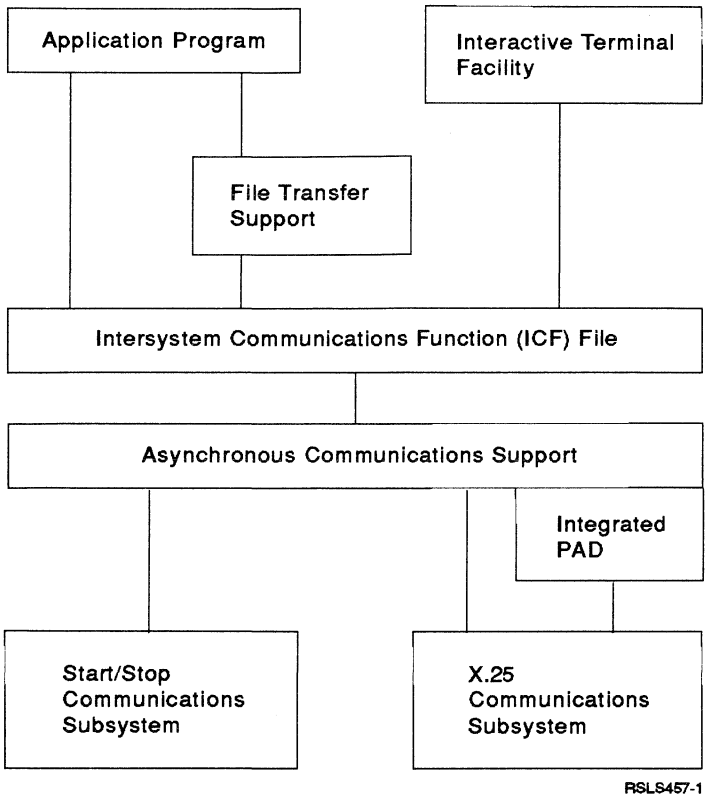


Figure 1-1. OS/400 Asynchronous Communications Support

Chapter 2. Asynchronous Communications Support

This chapter describes the configurations and communications environments that are possible using asynchronous communications support. Asynchronous communications support allows you to send data to and receive data from a remote program or device attached by either an asynchronous (start-stop) or an X.25 line. Your application program must provide the data stream required by the remote device. Asynchronous communications support packages your data stream in either a start-stop format or within X.25 data packets.

You must provide an application program on the AS/400 system to communicate with the remote device. The ICF operations your program uses to communicate with the remote device are the same as those used to communicate with another AS/400 system. The **intersystem communications function (ICF)** is a function of the operating system that allows a program to communicate interactively with another program or system. See Chapter 6 for a description of the ICF operations. For more information about configuring for asynchronous communications, see the *OS/400* Communications Configuration Reference*.

Asynchronous Communications on Start-Stop Lines

Asynchronous support allows an AS/400 system to use an asynchronous (start-stop) line to communicate with another start-stop device. Possible devices include: plotters, printers, terminals, modems, X.25 network-supplied packet assembler/disassemblers (PADs), another AS/400 system, a System/36, or an IBM personal computer. The remote device can be attached by a switched or nonswitched asynchronous line.

The following are some of the parameters you need to specify on the asynchronous communications line description. Use the Create Line Description (Asynchronous) (CRTLINASC) command to create the line description. These parameters must match the characteristics of the remote device.

BITSCHAR — Data bits per character: Specify 7 or 8 bits.

CNN — Connection type: Specify switched (*SWTPP) or nonswitched (*NONSWTPP) point-to-point to describe the physical start-stop communications line being used.

ECHO — Echo support: Specify *NONE, *ALL, or *CNTL.

EORTBL — End-of-record table: Specify up to 8 individual characters. Up to 4 trailing characters can also be specified.

FLOWCNTL — Flow control: Specify whether or not flow control characters will be used to control the flow of your data stream. (The hexadecimal values of the XON and XOFF characters can be specified using the XONCHAR and XOFFCHAR parameters.)

IDLTMR — Idle timer: Specify from 0 to 254 in 0.5 second intervals.

LINESPEED — Line speed: Specify the line speed used, in the range of 50 to 19,200 bits per second.

MAXBUFFER — Maximum buffer size: Specify from 128 to 4096 characters.

PARITY — Type of parity: Specify *EVEN, *ODD, or *NONE.

STOPBITS — Number of stop bits: Specify 1 or 2 bits.

Nonswitched Line Support

You should use a nonswitched asynchronous line description and an asynchronous controller description configuration when:

- The AS/400 system is attached to a nonswitched modem.
- The AS/400 system and the asynchronous device are connected by a modem eliminator or null modem.
- The AS/400 system and the remote device are connected by limited distance modems.

- The attached modem is a command-capable modem. This modem is configured to hold the Data Set Ready (DSR) signal active when the Data Terminal Ready (DTR) signal is active or when the modem is powered on.

Switched Line Support

You should use a switched asynchronous line description and an asynchronous controller description when the AS/400 system is attached to a switched modem. An **asynchronous controller description** represents a remote system or device when using asynchronous transmission methods on an asynchronous communications line or when using non-SNA protocols on an X.25 communications line to communicate with the system. The following types of switched modems can be attached to an AS/400 system:

- Manual dial/answer modems

  The connection to the remote system is made by manually dialing or answering the modem.

- V.25 bis modems (single line, capable of serial automatic dialing)

  When a V.25 bis modem is attached and configured, the AS/400 system issues a dial command to the modem at the time your program acquires the asynchronous device. The number used in the dial command is configured in the asynchronous controller description attached to the device you are acquiring. No other action is required by your application program to start the dial operation to this type of modem.

- Intelligent or command-capable modems

  Asynchronous support allows the attachment of command-capable modems and provides a path through which your application program can send commands to prepare the modem. Your application program may code the modem command sequence in a high-level language (HLL) and send the data to the modem using a write operation. All modem commands and responses appear as application data to the asynchronous support, and are handled as such.

  These types of modems are typically capable, by external switches or keypad configuration, of treating the Data Set Ready (DSR) signal in one of three ways:

  1. Holding DSR signals active at all times when the modem is powered on.
  2. Making DSR signals active when Data Terminal Ready (DTR) is active.

     When the modem is configured as in cases 1 and 2, nonswitched asynchronous line and controller descriptions should be used.
  3. Making DSR signals active only after a successful connection with a remote modem and during the communications with that modem.

     Asynchronous support allows data to be exchanged between the application program and the modem without the DSR signal being active. The modem initialization and dial commands can be issued by the program on a write operation.

     To use this support, specify the following parameters on the Create Line Description (Asynchronous) (CRTLINASC) command:

```
CRTLINASC ... INLCNN(*SWTPP) SWTCNN(*DIAL)
              AUTOANS(*NO) AUTODIAL(*YES)
              DIALCMD(*OTHER)
```

     You must specify the following parameters on the Create Controller Description (Asynchronous) (CRTCTLASC) command:

```
CRTCTLASC ... SWITCHED(*YES) INLCNN(*DIAL)
              CNNNBR(connection-number)
```

Asynchronous (Non-SNA) Communications on X.25 Lines

Asynchronous communications support allows the AS/400 system to use X.25 lines to communicate with another packet-mode host. It also allows the AS/400 system, acting as a packet-mode host, to communicate with start-stop devices that are connected to a packet-switching data network through a PAD.

The physical X.25 communications line can be switched or nonswitched. A nonswitched connection through the network to another system can be a permanent virtual circuit (PVC) or a switched virtual circuit (SVC). A switched connection through the network to another system must be a SVC. For additional information on

X.25 switched or nonswitched lines, see the *OS/400* Communications Configuration Reference*.

The terms permanent virtual circuit (PVC), incoming switched virtual circuit (SVC-IN), and outgoing switched virtual circuit (SVC-OUT) are used in the remainder of this chapter to refer to the various connection capabilities of the asynchronous controller descriptions that are attached to X.25 line descriptions.

The Create Controller Description (Asynchronous) (CRTCTLASC) command is used to specify PVCs and SVCs. PVCs are configured as SWITCHED(*NO) in the controller description; SVCs are configured as SWITCHED(*YES). The Initial Connection (INLCNN) parameter is used to specify SVC-IN (INLCNN(*ANS)) or SVC-OUT (INLCNN(*DIAL)).

Connecting Systems without a Network (DCE-to-DTE)

Asynchronous communications support using X.25 lines can be configured by connecting your AS/400 system to data terminal equipment (DTE) through a modem eliminator (or equivalent) instead of attaching through an X.25 packet-switching data network.

When this method is used, the AS/400 system acts as data circuit terminating equipment (DCE) to the remote DTE. **Data circuit terminating equipment (DCE)** is the equipment installed at the customer location that provides all the functions required to establish, maintain, and end a connection, and the signal conversion and coding between the data terminal equipment and the line. The remote system (DTE) can, for example, be a System/36 or another AS/400 system. The remote DTE acts as though it is attached to an X.25 network, but no packet-switching data network (PSDN) is involved in the connection.

Connections to a Packet-Switching Data Network

Asynchronous communications may be run on an X.25 PSDN. This is done by creating asynchronous controller and device descriptions and either an X.25 or an asynchronous line description. Figure 2-1 on page 2-4 shows the configurations supported.

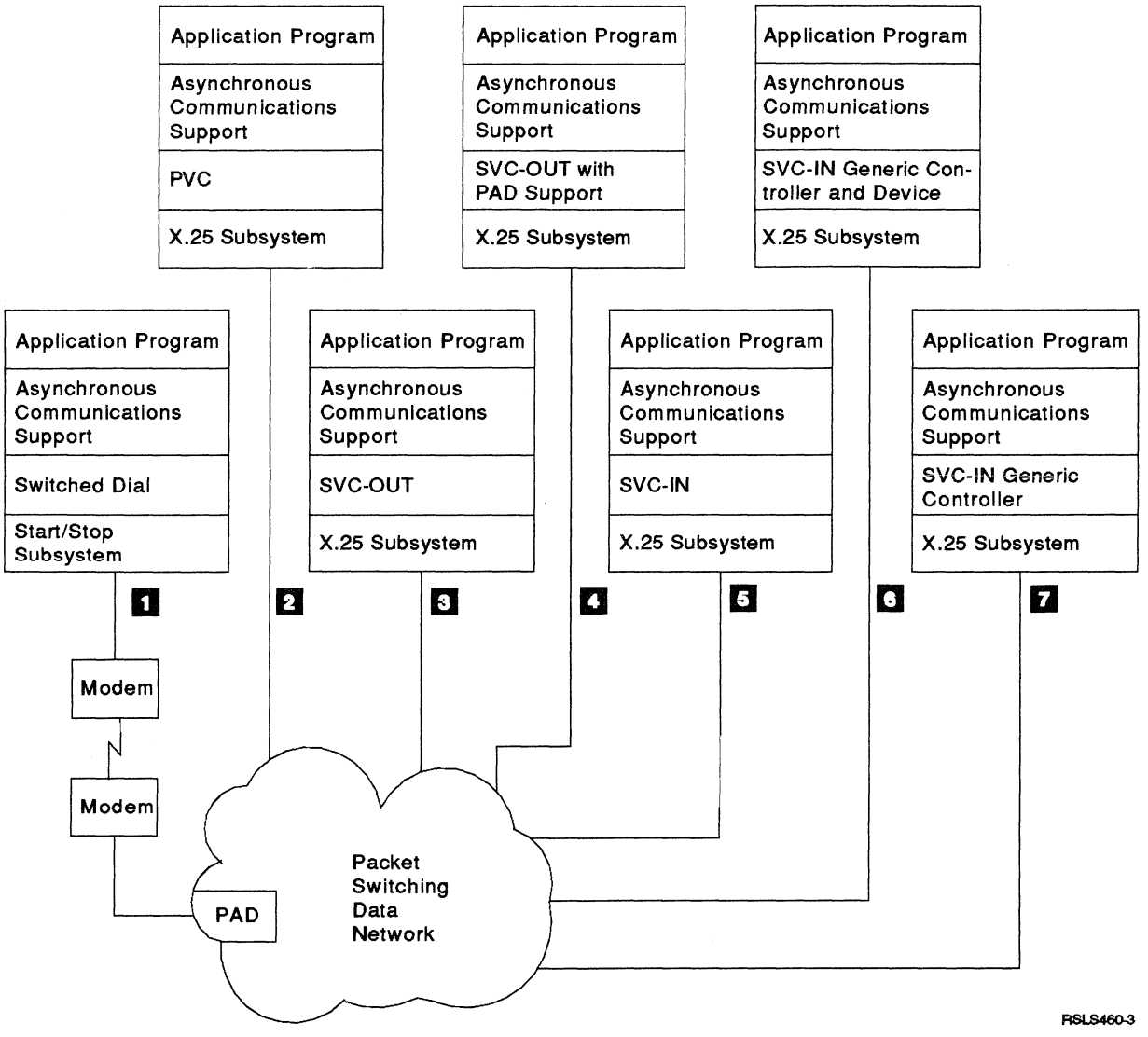


Figure 2-1. X.25 Packet-Switching Data Network

Connection 1 uses asynchronous support on an asynchronous (start-stop) line through a switched connection to the PSDN. Connections 2 through 7 use asynchronous support on separate X.25 lines through physically switched or nonswitched connections to the PSDN.

1 Switched dial connection to a network PAD

This is a switched dial connection using asynchronous communications support on an asynchronous (start-stop) line. A call is made from the AS/400 system to a network PAD. Your application program can then communicate with the PAD to establish a virtual circuit with a packet-mode host that is attached to the network.

The **packet-mode host** (any non-SNA, X.25 host system) could be another AS/400 system configured to accept calls from any network address. See the discussion of generic controllers and devices under 6 and 7.

2 Permanent virtual circuit (PVC) connection

This type of connection is used when the network supports permanently established circuits. No connection is allowed to any network address other than the one specified in your network subscription. This is similar to using a nonswitched connection on an asynchronous (start-stop) line.

3 Outgoing switched virtual circuit (SVC-OUT) to a specific network address

This type of connection is used when you know the network address of the system

that will accept your call. This connection only initiates calls and establishes a virtual circuit with the network address specified in the asynchronous controller description.

Use this type of connection only when you wish to initiate calls to a specific network address. If that address is not valid, busy, or otherwise unable to accept the call, the call is rejected.

The first byte of the call user data (protocol field) in the X.25 call packet used to establish a virtual circuit contains hex C0 to distinguish asynchronous communications from Systems Network Architecture (SNA) protocols.

4 Outgoing switched virtual circuit (SVC-OUT) with integrated PAD support

This type of connection is used when you wish to establish a virtual circuit with a packet-mode host that accepts calls from a PAD.

The first byte of the call user data (protocol field) in the X.25 call packet used to establish a virtual circuit contains hex 01 to inform the host system that the call is from a PAD.

The asynchronous communications integrated PAD provides the following support:

- CCITT recommendations X.3, X.28, and X.29
  - X.3 defines the PAD parameters that the PAD uses to control data and service signals to and from the application program. These parameters can be set by the application program or the packet-mode host.
  - X.28 defines the control procedures used to establish a virtual connection to a packet-mode host, the PAD commands the application program can send to the PAD, and the PAD service signals the program can receive from the PAD.
  - X.29 defines the PAD messages sent between the packet-mode host and the PAD.
- Rotary dial support

  The PAD attempts to establish a virtual circuit with an address contained in a list of network addresses. The PAD network address list is created using the Create Configuration List (CRTCFGL) command.

5 Incoming switched virtual circuit (SVC-IN) from a specific network address

This type of connection is used when you know the network address of the system that initiates the call. This connection only accepts calls and establishes a virtual circuit with the address specified in the asynchronous controller description.

Use this type of connection only when you want to accept calls from a specific network address. Calls received from a network address other than the one specified are rejected.

6 Incoming switched virtual circuit (SVC-IN) from any network address (generic controller and device)

This type of connection requires that you configure a generic controller and device description. This is done by specifying CNNNBR(*ANY) and INLCNN(*ANS) in the controller description and specifying RMTLOCNAME(*NONE) in the attached device description.

This type of connection allows you to accept a call request from any network address. The asynchronous support decides if the incoming call should be accepted based on the following:

- The remote or calling system must have the following configured on the controller description:
  - Remote verify (RMTVFY(*YES))
  - Local location name (LCLLOCNAME)
  - Local identifier (LCLID)

  Connection examples **1**, **3**, and **4** can be used to establish a circuit with a generic controller.
- The local location name and identifier from the above step must be entered in the asynchronous remote location list of the system receiving the call. Asynchronous remote location lists can be created and changed using the Create Configuration List (CRTCFGL) and Change Configuration List (CHGCFGL) commands. See the *OS/400* Commu-*

nications Configuration Reference for more information about using these commands.

- The calling system's local location name cannot be configured as the remote location name in any device description used by the system receiving the call.

Once the call is accepted, the remote verification parameter (configured on the remote or calling system as the local location name in the controller description and on the local system in the remote location list) becomes the remote location name (RMTLOCNAME) of the attached asynchronous device description. When this occurs, the asynchronous device can be acquired by a local program or it can receive program start requests.

If the remote verification parameter is not defined in the asynchronous remote location list, the call is not accepted.

Remote devices can also connect to an AS/400 system on an X.25 network through generic controllers and device descriptions. When an incoming call is received by a generic controller, an ID prompt is sent to the calling device requesting its location name and location identifier. This ID prompt consists of the following ASCII data stream:

<syn>ID<syn>

where <syn> = hex 16

When the device receives this prompt, it must respond by sending its location name and location identifier in the following format:

@<location name><location identifier><CR>

where:

<location name> and <location identifier> represent 8 alphanumeric ASCII characters, left-justified and padded with blanks.

The carriage return character (<CR>) is hex 0D.

@ is the keyboard *at* sign (hex 40).

If the location name and location identifier received by the generic controller are in the system's remote location list, the call is accepted. Otherwise, the call is rejected and the connection is dropped.

When the call is accepted, the asynchronous communications support responds with the following ASCII data stream:

<ETX><CR><LF>CONNECT<CR><LF>

where:

<ETX> = hex 03
<CR> = hex 0D
<LF> = hex 0A

7 Incoming switched virtual circuit (SVC-IN) from any network address (generic controller only)

This type of connection requires that you configure a generic controller. This is done by specifying CNNNBR(*ANY) and INLCNN(*ANS) in the controller description.

As discussed under item **6**, this type of connection allows you to accept a call from any network address. However, because you have configured a remote location name in the device description, your program may attempt to acquire the device before an incoming call is received. The acquire operation will not complete until an incoming call is received.

Remote verification is not done when using this type of connection; therefore, you should specify no remote verification (RMTVFY(*NO)) on the controller description for the remote or calling system.

Chapter 3. Using the Integrated Packet Assembler/Disassembler Support

Packet assembler/disassembler (PAD) is normally used to allow the attachment of start-stop devices to a packet-switching data network (PSDN). This is done by converting the start-stop data stream into X.25 data packets. Integrated PAD support provides the same support for user-written programs, file transfer, and ITF as a network PAD provides for start-stop devices. This support includes:

- Establishing sessions between your program and a packet-mode host
- Processing PAD messages received from the packet-mode host
- Processing PAD commands received from your application program and responding with PAD service signals
- Handling functions that depend on PAD parameter settings
- Routing data between your application program and a packet-mode host

You should consider using integrated PAD support when:

- You have an X.25 line connected to a packet-switching data network (PSDN)
- You want to communicate with a packet-mode host using your application program or ITF
- The packet-mode host communicates with start-stop devices that are connected to the network through a PAD
- The packet-mode host only accepts call requests from a PAD

Using the PAD

You can configure an asynchronous controller to emulate a PAD using the PADEML parameter on the Create Controller Description (Asynchronous) (CRTCTLASC) command. The integrated PAD support follows CCITT recommendations X.3, X.28, and X.29. **CCITT** is the abbreviation for the International Telegraph and Telephone Consultative Committee. These recommendations are as follows:

- X.3 defines the PAD parameters that the PAD uses to control the session
- X.28 defines the PAD commands and service signals exchanged between the PAD and an AS/400 application program
- X.29 defines the PAD messages that are exchanged between a packet-mode host and the PAD

Figure 3-1 shows the relationship between the PAD and the CCITT recommendations.

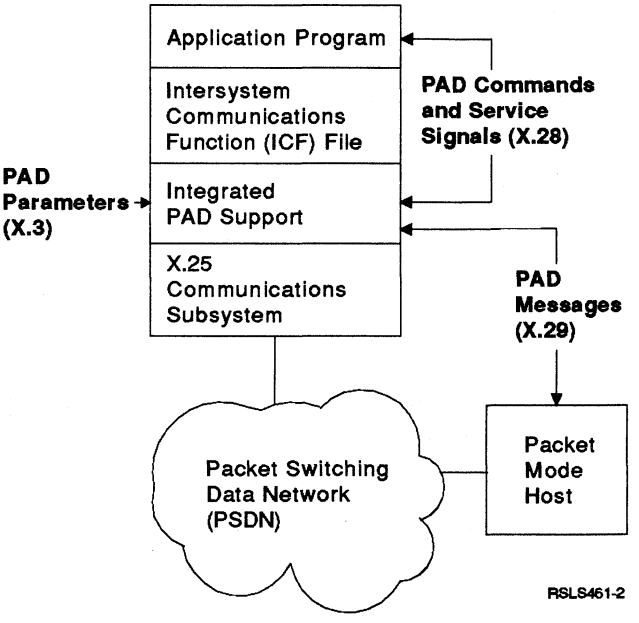


Figure 3-1. CCITT Recommendations

To use the PAD support, your application program must acquire a session with an asynchronous device that is attached to an asynchronous/X.25 controller. The controller must be configured for PAD emulation by specifying PADEML(*YES). Your application program can acquire a session with the PAD even though a connection to a remote system has not been established.

The values specified for the PAD parameters determine how the PAD operates on the data sent and received by your program. You can change the way the PAD operates by changing the values of the PAD parameters. This is done using the appropriate PAD commands. The packet-mode host can also change the values of the PAD parameters by issuing the appropriate PAD messages.

PAD commands are sent as data on write operations that are issued by your application program. Any resulting PAD service signals are returned to your program as data on the next read operation. PAD commands can only be issued when the PAD is in command mode.

A connection with a remote system can be made by issuing the PAD CONNECT command or by using the Rotary Dial function. See "Rotary Dial" on page 3-9 for more information.

Once a connection to a remote system is made, the PAD enters data transfer mode. While in data transfer mode, your program can send data to and receive data from the remote system. You can enter command mode again if PAD parameter 1 is set to 1.

PAD Parameters

The following PAD parameters are used to control the session. The packet-mode host can set and read these parameters by sending a SET, SET and READ, or READ PAD message to the PAD. An application program can change or read these parameters by issuing a SET, SET?, or PAR? PAD command.

Parameters marked *Not supported* in the following table are those that the PAD ignores because they are not used by the programs supported on the AS/400 system. Any attempt to read or change these parameters causes an error to be reported, as follows:

- If a SET, READ, or SET and READ PAD message is received by the PAD from a packet-mode host for a parameter marked *Not supported*, the asynchronous support sends a Parameter Indication PAD message indicating the parameter in error.
- If a SET, SET?, or PAR? PAD command is received by the PAD for a parameter marked *Not supported*, the asynchronous support indicates that the parameter reference is in error by returning the parameter value INV in the PAD service signal.

Figure 3-2 (Page 1 of 2). PAD Parameter Chart

Parameter	Description	Values and Meanings
1	Escape to command mode	0: No escape possible 1: Escape possible Default: 1
2	Echo	0: PAD does not echo 1: PAD will echo characters Default: 1
3	Data forwarding characters **Note:** These values represent the ASCII characters defined in Figure C-2 on page C-2.	0: None 2: Carriage return 6: ENQ, ACK, BEL, ESC, CR 18: ETX, EOT, CR 126: All characters in columns 1 and 2 of Figure C-2 on page C-2, plus DEL Default: 2
4	Idle timer	Not supported
5	PAD suspension of input	Not supported
6	Suppression of service signals	0: Suppress signals 1: Deliver signals Default: 1
7	Break options	0: Do nothing 1: Send interrupt 2: Reset 8: Escape to command mode 21: Discard pending data at the PAD, send interrupt, and send indication of break PAD message Default: 0
8	Discard output	0: Deliver output 1: Discard output Default: 0
9	Carriage return padding	Not supported
10	Line folding	0: None 1-255: Number of characters per line before line folding Default: 0
11	Terminal speed	Not supported
12	Flow control of PAD	0: Not possible 1: Possible Default: 0
13	Line feed insertion after carriage return	0: None 1: To terminal 4: In echoed data to terminal 5: Combination of 1 and 4 6: Combination of 4 and from terminal 7: Combination of 1, 4 and from terminal Default: 0
14	Padding after line feed	Not supported
15	Editing	Not supported
16	Character delete	Not supported
17	Line delete	Not supported
18	Line display	Not supported

Figure 3-2 (Page 2 of 2). PAD Parameter Chart

Parameter	Description	Values and Meanings
19	Editing PAD service signals	Not supported
20	Echo mask	Not supported
21	Parity treatment	Not supported
22	Page wait	Not supported

The PAD parameters are defined as follows:

1 Escape to command mode

This function allows the PAD to change from data transfer mode to command mode using the escape sequence, <CR>@<CR>. Although the PAD is in command mode, your program is still connected to the remote system.

Note: Throughout this section <CR> refers to the carriage return character (hex 0D).

2 Echo

This function allows all characters you send to the PAD to be transmitted back to you at the same time the character is processed.

3 Data forwarding characters

This function allows you to define a set of characters that control how data is sent by the PAD to the packet-mode host. All data up to and including the defined character are sent together. Data from the last forwarding character to the end of the data stream is also sent together. This function is only supported in data transfer mode.

6 Suppression of service signals

This function allows you to determine whether or not you want to receive PAD service signals.

7 Break options

This function allows you to determine how the PAD operates when your application program issues a fail function.

8 Discard output

This function allows the PAD to discard any data received from the packet-mode host.

10 Line folding

This function allows you to set the maximum number of characters per line. The PAD automatically inserts the format effectors (layout characters).

12 Flow control of PAD

This function allows you to control the flow of data between your application program and the PAD. Sending XON or XOFF characters indicates to the PAD whether or not your program is ready to receive data. The XON and XOFF characters are defined as (DC1) and (DC3) in Figure C-2 on page C-2.

13 Line feed insertion after carriage return

This function allows the PAD to automatically insert a line feed after a carriage return. This function is only supported in data transfer mode.

The following parameters are not supported:

4 - Idle timer
5 - PAD suspension of input
9 - Carriage return padding
11 - Terminal speed
14 - Padding after line feed
15 - Editing
16 - Character delete
17 - Line delete
18 - Line display
19 - Editing PAD service signals
20 - Echo mask
21 - Parity treatment
22 - Page wait

PAD Commands

PAD commands are used to manage a virtual circuit and to change the way the PAD operates. The PAD has two modes of operation: command mode and data transfer mode. While in command mode, your application program can send commands to the PAD and receive PAD service signals in response. Your program can enter command mode from data transfer mode by entering the escape sequence:

<CR> @ <CR>

Note: Throughout this section <CR> refers to the carriage return character (hex 0D).

When your program initially acquires a session with the PAD, the PAD is in command mode. Once you have established a connection with a packet-mode host, the PAD enters data transfer mode. If an end-of-session function is issued without disconnecting, and if SWTDSC(*NO) is specified on the controller description, the PAD

resumes the last mode of operation and is still connected when your application program issues the next acquire operation.

The following list describes the PAD commands available.

- **CONNECT**

  This command is used by your application program to request establishment of a virtual call. It allows you to connect to a specified network address. Network addresses have a minimum length of 5 digits and a maximum of 17. The following CONNECT commands are allowed:

  1. Connect with no address specified

     Format: CONNECT

     If your application program issues a CONNECT command without specifying a network address, a connection is attempted using the network address specified in the connection number (CNNNBR) field of the controller description. If this is not the first connection attempt, the address with which you last attempted a connection is used.

  2. Connect with address specified

     Format: CONNECT <*address*>

     A connection is attempted to the specified address. The fully qualified network address is constructed by adding the address you specified to the data network identification code (DNIC) configured in the CNNNBR field of the controller description. The data network identification code (DNIC) is assumed to be the first 4 digits of the CNNNBR field and is used to identify the network. Once a connection has been attempted, the address you entered is the default for any subsequent connection attempts for which no address is specified.

  3. Connect specifying a fully qualified network address

     Format: CONNECT 0DNIC <*address*>

     You can specify a fully qualified network address by preceding the address with a 0 (zero). This allows you to connect to a network address whose data network identification code (DNIC) is different from that specified as the first 4 digits of the CNNNBR field in the controller description. A connection is attempted to the fully qualified address. The data network identification code (DNIC) and address become the default for any subsequent connection attempts for which no address is specified.

  Note: The valid abbreviations for this command are C and CONN.

- **RESET**

  This command resets the session to the pre-connect status. This includes setting the PAD parameters to their default values and clearing the virtual call.

- **STATUS**

  Response to this command indicates whether a virtual circuit is connected or available.

  Note: The valid abbreviation for this command is STAT.

- **DISCONNECT**

  This command is used to request that the virtual call be cleared. Communications with the connected network address is discontinued. Another CONNECT command can be issued after this command to establish a virtual circuit with either the same or a different network address.

  Note: The valid abbreviations for this command are D and DISC.

- **CONTINUE**

  This command is used to return to data transfer mode when your application program is in command mode as a result of entering the PAD escape sequence.

  Note: The valid abbreviation for this command is CONT.

- **SET**

  This command is used to change the values of the PAD parameters. If no list is provided, all PAD parameters are reset to the default values. The following SET commands can be used:

  1. Set all PAD parameters to the default values

     Format: SET

2. Set the values of specified PAD parameters

   Format: SET <*list*>

   The parameter list following the SET command contains pairs of PAD parameters and values. Each pair of parameters is separated by a comma. The list has the following format:

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

- **SET?**

  This command is used to change the values of the PAD parameters and to read the current values after they are changed. If no list is specified, all PAD parameters are reset to the default values and all values are read. The following SET? commands can be used:

  1. Set and read all PAD parameters

     Format: SET?

  2. Set and read specified PAD parameter values

     Format: SET? <*list*>

     The parameter list following the SET? command contains pairs of PAD parameters and values. Each pair of parameters is separated by a comma. The list has the following format:

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

- **PAR?**

  This command is used to read the current values of the PAD parameters. If no parameters are specified, all values are read.

  1. Read all PAD parameters

     Format: PAR?

  2. Read specified PAD parameters

     Format: PAR? <*list*>

     The parameter list following the PAR? command contains the numbers of the PAD parameters you wish to read. Each number is separated by a comma. The list has the following format:

     ```
     number,number,...,number<CR>
     ```

- **PAD escape sequence**

  The PAD escape sequence is used to enter command mode from data transfer mode. The sequence is:

  ```
  <CR>@<CR>
  ```

- **PAD prompt**

  When in command mode, the PAD prompts for the next command by returning the PAD prompt service signal as data on the next read operation issued by your program. This prompt has the following format:

  ```
  * ENTER PAD COMMAND:
  ```

Examples of SET, SET?, and PAR? Commands

The following are examples of the SET, SET?, and PAR? commands. The examples show the PAD service signals returned in response to both successful and unsuccessful commands.

- **SET** - parameter 1 is changed to value 0 and parameter 7 is changed to value 4.

  ```
  PAD Command  - SET 1:0,7:4<CR>
  PAD Response - none unless an error occurs
  ```

- **SET?** - parameter 2 is changed to value 0 and parameter 7 is changed to value 21. Read the values of these parameters after they are changed.

  ```
  PAD Command  - SET? 2:0,7:21<CR>
  PAD Response - PAR 2:0,7:21<CR>
  ```

- **PAR?** - read the value of parameters 1 and 7.

  ```
  PAD Command  - PAR? 1,7<CR>
  PAD Response - PAR 1:0,7:21<CR>
  ```

- **SET** - attempt to change the value of a parameter that is not valid.

  ```
  PAD Command  - SET 23:0<CR>
  PAD Response - PAR 23:INV<CR>
  ```

- **SET** - attempt to change a parameter to a value that is not valid.

  ```
  PAD Command  - SET 7:3<CR>
  PAD Response - PAR 7:INV<CR>
  ```

PAD Service Signals

The following chart shows the PAD service signals issued to the application program in response to PAD commands.

Figure 3-3. PAD Service Signals — Response to PAD Commands

Message ID	PAD Service Signal	Description
CPX6B76	CONNECTED	Response to STATUS command when connected
CPX6B77	DISCONNECTED	Response to DISCONNECT command when session is disconnected
CPX6B78	AVAILABLE	Response to STATUS command when not connected
CPX6B79	<*address*> CONNECTED	Response to CONNECT command when connection complete to specified address
CPX6B7B	ERR	PAD did not understand last command
CPX6B7C	INVALID ADDRESS	Address supplied with CONNECT command is not a valid address
CPX6B7D	ALREADY CONNECTED	Response to CONNECT command when already connected to remote system
CPX6B7E	NOT CONNECTED	Response to DISCONNECT command when not connected to a remote system
CPX6B7F	* ENTER PAD COMMAND:	PAD acknowledgment and prompt

The following chart shows additional PAD service signals issued in response to your unsuccessful call attempts.

Figure 3-4. PAD Service Signals — Response to Unsuccessful Call Attempts

Message ID	PAD Service Signal
CPX6B64	CLR REJECTING
CPX6B65	CLR CALL CLEARED
CPX6B66	CLR NUMBER BUSY
CPX6B67	CLR NOT REACHABLE
CPX6B68	CLR NOT RESPONDING
CPX6B69	CLR REFUSING COLLECT CONNECTION
CPX6B6A	CLR NOT OPERATING
CPX6B6B	CLR STILL PENDING
CPX6B6C	CLR NOT AVAILABLE
CPX6B6D	CLR ILLEGAL ADDRESS
CPX6B6E	CLR ILLEGAL SOURCE ADDRESS
CPX6B6F	CLR NETWORK CONGESTION
CPX6B70	CLR INVALID FACILITY REQUEST
CPX6B71	CLR LOCAL PROCEDURE ERROR
CPX6B72	CLR REMOTE PROCEDURE ERROR
CPX6B73	CLR INVALID LOGICAL CHANNEL TYPE

Figure 3-4. PAD Service Signals — Response to Unsuccessful Call Attempts

Message ID	PAD Service Signal
CPX6B74	CLR CALL USER DATA ERROR
CPX6B75	CLR NO LOGICAL CHANNEL AVAILABLE

PAD Messages

PAD messages allow the packet-mode host to change the PAD parameters as well as request that the PAD clear the virtual circuit.

Two types of PAD messages are supported: requests from the packet-mode host and responses from the PAD. Each PAD message is defined by a message code. The following PAD messages and the associated message codes are supported by the PAD:

- **Set** (hex 02)

  Sent by the packet-mode host to change the PAD parameters.

- **Set and Read** (hex 06)

  Sent by the packet-mode host to change the PAD parameters. The PAD responds by sending a Parameter Indication message.

- **Read** (hex 04)

  Sent by the packet-mode host to find out what the PAD parameters are set to. The PAD responds by sending a Parameter Indication message.

- **Invitation to Clear** (hex 01)

  Sent by the packet-mode host to request the PAD to end the connection. All previously received data is discarded and the PAD enters command mode after dropping the connection.

- **Parameter Indication** (hex 00)

  Sent by the PAD in response to the packet-mode host SET AND READ or READ message. This message tells the packet-mode host what values the PAD parameters are set to.

- **Indication of Break** (hex 03)

  Sent by the PAD in response to the FAIL function. See Appendix D for more information on break and interrupt handling.

- **Error** (hex 05)

  Sent by the PAD to indicate to the packet-mode host that a PAD message that is not valid was received.

Requests from the Packet-Mode Host

Your application program may send PAD messages to a remote PAD using the write function-management-header operation. The PAD message may be a Set, Read, Set and Read, Invitation to Clear, or any other messages supported by the remote PAD; the message code indicates which PAD message is to be sent. Your program supplies the message code and message parameters (if required for that message) as data on the write function-management-header operation. See "Write Operation" on page 6-4 for more information.

Note: The data must be in the exact format required by the remote PAD. The integrated PAD requires that all data received in PAD messages be in hexadecimal. The following discussions assume that the remote PAD to which you are sending PAD messages is the integrated PAD.

Reading and Setting PAD Parameters: The current values of the PAD parameters can be changed or read by sending a Set, Read, or Set and Read message. These messages are sent by your program using the write function-management-header operation. The data to be sent by this operation must be in the following format:

```
hex <message code><parameter><value>
    <parameter><value>...
```

where:

<message code> indicates which PAD message is being sent. <parameter> specifies the PAD parameter that you want to set or read, followed by the <value> that you want it set to. PAD parameters and values supported by the PAD are listed earlier in this chapter. When you send a Read message, you should enter hex 00 for each parameter value, because you are not setting values.

If you do not enter any parameters and values, one of the following occurs:

- For a Set message, all parameters are reset to their default values.
- For a Read message, the values of all parameters are returned to the program by a Parameter Indication.
- For a Set and Read message, all parameters are reset to their default values and the values are returned to the program by a Parameter Indication.

A Parameter Indication or an Error message may be received from the remote PAD in response to the above operations. These messages are returned to your application program as data on the next read operation. A return code of 0004 indicates that the data is a PAD message.

Clearing the Virtual Circuit: Your packet-mode host program can request that the remote PAD end the connection by sending an Invitation to Clear message. This message causes the PAD to clear the virtual circuit and has no message parameters.

Responses from the PAD

The PAD responds to requests from the packet-mode host by sending either a Parameter Indication or an Error message. The PAD may also send an Indication of Break message in response to a fail function issued by your application program.

Parameter Indication Message: The PAD responds to a valid Read or Set and Read message by sending a Parameter Indication to the packet-mode host. This PAD message contains the parameter numbers and the current values (after any changes) of the PAD parameters to which the received PAD message referred. The message has the following format:

```
hex 00<parameter><value><parameter><value>...
```

The PAD does not return a Parameter Indication message in response to a valid Set message.

If the PAD receives a Set, Read, or Set and Read message that contains a reference to a PAD parameter that is not supported, it responds by sending a Parameter Indication message. The parameter field within the Parameter Indication message indicates the parameter that is in error by setting the most significant bit to 1. The corresponding value field is then set to hex 00.

Note: Any remaining valid references to PAD parameters are processed.

Possible reasons for a reference that is not valid to a PAD parameter are:

- The parameter is not supported by the PAD.
- The parameter value is not supported by the PAD.

Indication of Break Message: The Indication of Break message is sent by the PAD when your application program issues a fail function. The break message that is sent depends on the value of PAD parameter 7. See Appendix D, for more information about how the PAD responds to the fail function based on the value of PAD parameter 7.

Error Message: The PAD sends an error message when a PAD message that is not valid is received from a packet-mode host. Possible reasons for a PAD message that is not valid being received are as follows:

- The received PAD message contained less than 8 bits. The reason code is hex 00.
- An unrecognized message code was received in the PAD message. The reason code is hex 01.
- The received PAD message did not contain an integral number of bytes. The reason code is hex 03.

The format of the error message is:

hex 05<reason code>

Note: For reason codes hex 01 and hex 03, the error message also contains the message code of the PAD message that was received from the packet-mode host. The message code follows the reason code in the error message.

Examples of PAD Messages

The following are examples of PAD messages. These examples show the messages that can be sent from a packet-mode host. They also show the responses sent by the PAD to both successful and unsuccessful messages.

Note: The first byte of each PAD message indicates the message being sent.

- Set message
  - The packet-mode host sends a Set message to set PAD parameter 7 to 1 and parameter 10 to 80.

    PAD message - hex 0207010A50

    PAD response - None, unless an error occurs
  - The packet-mode host attempts to set PAD parameter 7 to 13.

    PAD message - hex 02070D

    PAD response - hex 008700
- Read message
  - The packet-mode host issues a message to read the values of PAD parameters 2 and 10 that have the values 1 and 80.

    PAD message - hex 0402000A00

    PAD response - hex 0002010A50
  - The packet-mode host attempts to read the PAD parameter 23, which is not valid.

    PAD message - hex 041700

    PAD response - hex 009700
- Set and Read message
  - The packet-mode host issues a Set and Read message to change the value of PAD parameter 2 to 0 and parameter 7 to 1. Read the values of these parameters after the Set message.

    PAD message - hex 0602000701

    PAD response - hex 0002000701
  - The packet-mode host issues a Set and Read message to change the value of PAD parameter 2 to 1, parameter 7 to 25, and parameter 8 to 0. The value 25 is not valid for parameter 7.

    PAD message - hex 06020107190800

    PAD response - hex 00020187000800
- Error message
  - The packet-mode host issues an unsupported PAD message.

    PAD message - hex 070101

    PAD response - hex 050107
- Invitation to Clear message
  - The packet-mode host requests that the remote PAD clear the virtual circuit.

PAD message - hex 01

PAD response - Clear the virtual circuit

Rotary Dial

Rotary dial is a function of the PAD support that allows you to enter the name of a PAD network address list to use in connecting to a remote system. It provides a function similar to that of the CONNECT command, and is only valid when the PAD is in command mode and not already connected. You create the PAD network address list by running the CRTCFGL command. You should not use a PAD command as the name of a PAD network address list. See the *OS/400* Communications Configuration Reference* for more information about the Create Configuration List (CRTCFGL) command.

The rotary dial function is started by entering the name of the PAD network address list as data either on a write operation issued by your application program or on the input line of ITF. PAD support begins calling the first address in the list for the specified number of times or until a connection is made. If no successful connection is made, the next address in the list is called. This continues until a successful connection is made or until all of the addresses have been called. In either case, a PAD service signal is issued indicating the result of the call attempt. See Figure 3-3 on page 3-5 and Figure 3-4 on page 3-6 for more information about PAD service signals.

Chapter 4. Configuring Asynchronous Communications Support

This chapter lists the configuration commands that allow you to configure the communications environments described in Chapter 2. You can use either the system-supplied menus or the control language (CL) commands to configure asynchronous communications. For more information about communications configuration, see the *OS/400* Communications Configuration Reference*.

Asynchronous Configuration Commands

An asynchronous configuration consists of an asynchronous line, controller, and device description. If you are using X.25, you need to configure an X.25 line with an asynchronous controller and device description. The name of each configuration description must be unique for each configuration object type. This unique name is used when the configuration is varied on or varied off using the Vary Configuration (VRYCFG) command. More than one line description can be varied on and in use at the same time; however, each line must be attached to a different communications port.

The following commands are used to create or change line descriptions:

- CRTLINASC: Create Line Description (Asynchronous)
- CHGLINASC: Change Line Description (Asynchronous)
- CRTLINX25: Create Line Description (X.25)
- CHGLINX25: Change Line Description (X.25)

The following commands are used to create or change controller descriptions:

- CRTCTLASC: Create Controller Description (Asynchronous)
- CHGCTLASC: Change Controller Description (Asynchronous)

The following commands are used to create or change device descriptions:

- CRTDEVASC: Create Device Description (Asynchronous)
- CHGDEVASC: Change Device Description (Asynchronous)

The following commands are used to create or change asynchronous configuration lists. A configuration list can be either a remote location list or a PAD network address list.

- CRTCFGL: Create Configuration List
- CHGCFGL: Change Configuration List

Asynchronous communications configuration lists may consist of either a remote location list or a PAD network address list:

- The remote location list is used by asynchronous support when you have configured generic controllers and devices. See item **6** under the topic "Connections to a Packet-Switching Data Network" on page 2-3 for more information about using generic controllers and devices.
- The PAD network address list is used by the asynchronous support as part of the rotary dial function. See "Rotary Dial" on page 3-9 for more information about this function.

Chapter 5. Running Asynchronous Communications Support

Once configuration for asynchronous communications is complete, the Vary Configuration (VRYCFG) command is used to activate and deactivate the line, controller, and device descriptions used by asynchronous communications support.

The VRYCFG command prepares the local AS/400 system to communicate with the remote system. The remote system must also be prepared to communicate with the local system.

Use the VRYCFG command and specify STATUS(*ON) to vary on the configuration descriptions. Use the VRYCFG command with STATUS(*OFF) to vary off the configured line, controller, and device descriptions.

It is not necessary to activate configuration list support for asynchronous communications. The remote locations or network addresses used in the configuration list are known to the asynchronous communications support at the time the list is created or changed.

| See the *Communications Management Guide* for
| more information about using the VRYCFG command.

Chapter 6. Writing Asynchronous Communications Application Programs

This chapter describes how an application program uses the intersystem communications function (ICF) file and the asynchronous communications support. The program can be coded using C/400, COBOL/400, FORTRAN/400, or RPG/400 languages. These four languages support an interface that allows the program to do the following functions:

- Start a session by opening an ICF file and acquiring a program device.
- Send and receive information by writing or reading to an ICF file.
- End a session by releasing the program device and closing the ICF file.

The chapter also includes a description of the read and write operations that specify a record format containing specific communications functions. Record formats can be defined using data description specifications (DDS), or you may use system-supplied formats.

After an operation completes, a return code (and a high-level language file status) is returned to your application. The return code indicates whether the operation completed successfully or unsuccessfully. Along with the return code, exception messages may also be issued. Refer to Appendix B for more information about return codes and to the appropriate language reference manuals for more information about the high-level language file status.

Intersystem Communications Function Files

An ICF file must be created before your application can use the asynchronous communications support. The **ICF file** is used to describe how data is presented to the program with which your program is communicating, and how data is received from that program. If you are using DDS keywords, use the Create Intersystem Communications Function File (CRTICFF) command to create an ICF file. If you are using the system-supplied formats (such as $$SEND), you do not need to create an ICF file. The ICF file QICDMF, which is in the library QSYS, is supplied by IBM for communications.

The ICF file is a system object of type *FILE with a specific user interface. This interface is made up of a set of commands and operations. The commands allow you to manage the attributes of the file, and the operations allow a program to use the file. Commands allow you to create, delete, change, and display the file description.

The following commands are used to manage the ICF file, and are described in detail in the *ICF Programmer's Guide*.

CRTICFF
: Create ICF File. This command allows you to create an ICF file. Once you have created this file, asynchronous communications support uses the attributes for each session.

CHGICFF
: Change ICF File. This command allows you to make a permanent change to the file attributes of the ICF file.

OVRICFF
: Override ICF File. This command allows you to make a temporary change to the file attributes of the ICF file at run time. These changes are only in effect for the duration of the job and do not affect other users of the file.

DLTF
: Delete File. This command allows you to delete a file from the system.

DSPFD
: Display File Description. This command displays the file description of any file on the system. The information can be printed or displayed.

DSPFFD
: Display File Field Description. This command displays the description of the fields in any file on the system. This information may be printed or displayed.

ADDICFDEVE
: Add ICF Device Entry. This command allows you to add a permanent program

device entry to the ICF file and have it associated with a program device name. Only one program device name can be used for each remote location name in a session. Once you have added a program device entry, the attributes are used for every session.

CHGICFDEVE
Change ICF Device Entry. This command allows you to permanently change the device entry previously added with the ADDICFDEVE command.

RMVICFDEVE
Remove ICF Device Entry. This command allows you to permanently remove the device entries previously added with the ADDICFDEVE command or changed with the CHGICFDEVE command.

OVRICFDEVE
Override ICF Device Entry. This command can be used for two functions:

- To temporarily add the program device entry and the location to the ICF file. You must use an OVRICFDEVE command if you do not use an ADDICFDEVE command to add a program device entry to the ICF file to be used for a session.
- To override (replace) a program device entry with the specified location name and attributes for an ICF file. When the session ends, the attributes revert to the parameters set by the ADDICFDEVE command.

Specifying the Program Device Entry Parameters

The following describes the parameters for the ADDICFDEVE, CHGICFDEVE, and OVRICFDEVE commands and lists the valid values for each parameter for asynchronous communications. For a complete description of all the parameters for these commands, refer to the *ICF Programmer's Guide*.

FILE
Specifies the name and library of the ICF file to which you are adding or changing the program device entry. The FILE parameter is not available on the OVRICFDEVE command.

***LIBL**: Asynchronous communications support uses the library list to locate the ICF file. This is the default.

***CURLIB**: Asynchronous communications support uses the current library for the job to locate the ICF file. If no current library entry exists in the library list, asynchronous communications uses QGPL.

filename: A 1- to 10-character value that specifies the name of the ICF file.

library-name: A 1- to 10-character value that specifies the library where the ICF file is located.

PGMDEV
Specifies the program device name that is defined in the ICF file and specified in the application. The total number of devices that can be acquired to an ICF file is determined by the MAXPGMDEV parameter on the CRTICFF or CHGICFF command.

pgm-device-name: A 1- to 10-character value for the program device name being defined. This name is used on device-specific input and output operations to identify the program device and the attributes.

RMTLOCNAME
Specifies the remote location name with which your program communicates. A remote location name must be specified on the ADDICFDEVE command or an OVRICFDEVE command. If a remote location name is not specified, an 82AA return code is issued when the program device is acquired.

***REQUESTER**: The name used to refer to the communications device through which the program was started. The session that is assigned when the program device is acquired is the same session that receives the program start request. If the program is not started as a result of a program start request, the acquire operation for the program device fails. The target program always uses *REQUESTER as the remote location name in the ICF file to connect to the session that the source program uses to send the program start request.

remote-location-name: A 1- to 8-character name for the remote location name that

should be associated with the program device.

FMTSLT
Specifies the type of record format selection used for input operations for all devices.

***PGM**: The program determines what record formats are selected. If an input (read) operation with a record format name is specified, that format is always selected. If an input operation without a record format is specified, the default format (the first record format in the file) is always selected. This also means that if any record identification (RECID) keywords are specified in the data description specifications (DDS) for the file, they are not taken into consideration when the record is selected. This is the default.

***RECID**: The RECID keywords specified in DDS for the file are used to specify record selection. If no RECID keywords are specified in the file, an error message is sent and an acquire operation for the program device will fail.

***RMTFMT:** Remote format names are not supported by asynchronous communications.

CMNTYPE
Identifies the communications type for which you define a program device entry. You should specify the value *ASYNC or *ALL for this parameter.

***ASYNC**: The prompt for all asynchronous communications-supported attributes.

Note: When you specify *REQUESTER for the remote location name (RMTLOCNAME), you are only prompted for the attributes of the format select parameter (FMTSLT) and the secure from override parameter (SECURE).

Communications Operations

This section provides a description of the operations you can code into a program that uses asynchronous communications support to communicate with another program.

Starting a Session

A communications **session** is a logical connection between two systems through which a local program can communicate with a program at a remote location. A communications session is established with an acquire operation and is ended with a release operation or an end-of-session function.

Open/Acquire Operation

Your program must open an ICF file and acquire a program device before it can direct any read or write operations to the program device. Only program devices defined to the file by the ADDICFDEVE or OVRICFDEVE command can be acquired.

A session can be established explicitly, using an acquire operation, or implicitly, using an open operation. The acquire operation is performed automatically as part of the open operation if you specify the ACQPGMDEV parameter on the ICF file.

You can start the session in one of the following ways:

- For a source program, the session between your program and the remote location with which your program is communicating is started by an open or acquire operation. The program device name on the acquire operation identifies the session and must match the program device name specified in an associated ADDICFDEVE or OVRICFDEVE command.
- For a target program, a source program on the remote system sends a program start request to the AS/400 system to start your program. This also starts the session. Before your program can send or receive data, it must first make a logical connection to the source program. This logical connection is made when your program uses the open or acquire operation. The program device name on the acquire operation identifies the session. This name must match the program device name specified in an associated ADDICFDEVE or OVRICFDEVE command. You must specify a requesting device for the remote location (RMTLOCNAME(*REQUESTER)) on the

ADDICFDEVE or OVRICFDEVE command when your program is started by a program start request.

See "Evoke Function" for the format of the program start request built by asynchronous communications.

Starting a Transaction

A **transaction** is a logical connection between two programs. Use the evoke function to start a transaction between your program and a target program on the remote system.

Evoke Function

Your program uses the evoke function to start a program on the remote system. Control is then returned to your program immediately without confirmation that the target program has started successfully. It is the responsibility of your program to confirm that the target program has started.

For example, after the evoke function has been issued, your program can issue an invite function to request data from the target program. Your program should then use the timer function to set the maximum amount of time your program waits to receive data. Your program can then issue a read-from-invited-program-devices operation until it receives a timer ended return code (0310) or until a confirmation is received from the target program.

If your program sends program initialization parameters on the evoke function, each parameter that is sent should be equal in length to the corresponding parameter specified in the target program. If it is longer than the parameter length in the target program, the parameter is truncated. If it is shorter than the parameter length in the target program, unpredictable results may occur.

For information on how to code the evoke function, refer to the *ICF Programmer's Guide* and the *DDS Reference*.

Syntax of Program Start Requests

When your program issues an evoke function, the asynchronous support builds a program start request that is sent to the remote system. The format of the program start request as received by the remote system is:

```
<'*EXEC' or '*EXEX'><b><PROGRAM NAME><b>
  <PROGRAM DATA><CR>
  <USER ID><CR><LIBRARY NAME>
  <CR><PASSWORD><CR><EOT>
```

where:

```
<*EXEC>          = Normal evoke (Hex 2A45584543)
<*EXEX>          = Evoke with detach
                   (Hex 2A45584558)
<b>              = Blank (Hex 20)
<CR>             = Carriage return (Hex 0D)
<EOT>            = End of transaction (Hex 04)
<PROGRAM NAME>   = Name of program to be started
<PROGRAM DATA>   = Any program initialization
                   parameters sent by your program
<USER ID>        = User identifier
<LIBRARY NAME>   = Name of library where program
                   resides
<PASSWORD>       = Password used by your program
```

Notes:

1. The receiving system always expects the program start request to be in ASCII.
2. If an *, E, C, or X are configured as end-of-record characters in the end-of-record table (EORTBL) of an asynchronous line description, asynchronous communications support may not recognize the program start request.

Sending Data

You can send data during a transaction using the write operation. The following section describes the write operations and functions that are supported for asynchronous communications.

Write Operation

Your program uses the write operation to send data to the remote location. The maximum amount of data your program can send with each write operation is 4096 characters. If an asynchronous line description is used, the asynchronous support does not attempt to maintain

the data as logical records. Therefore, the remote system application program must reassemble the data into logical records.

If an X.25 line description is used, the data sent by your program is maintained as a logical record. This is done by turning on the more-data bit in each data packet sent by the asynchronous support.

If PAD emulation is configured, the settings of the PAD parameters govern how data is sent. See Chapter 3 for more information.

If your program had previously issued an invite function, the write operation causes an implicit cancel invite if no data is available. If data is available, the write operation will receive a 0412 return code. This code indicates that before a write operation can be issued, your program must issue a read operation to receive the data.

Function-Management-Header Function

Your program uses the function-management-header function to affect data translation, to change certain characteristics of data on an asynchronous communications line, or to send PAD messages. All data associated with the function-management-header function, with the exception of the send PAD message function-management-header function, will be used only by the local system.

When your program changes any of the following values, the change remains in effect until the line is varied off or another write function-management-header is issued.

Note: The line description is not changed; therefore, it is not possible to determine what is changed by displaying the line description.

- **Setting translation mode:** When an asynchronous communications session is acquired, the default value for the translation mode is XLATE-Y, which means data is translated. User data is translated from EBCDIC to ASCII on write operations and from ASCII to EBCDIC on read operations. If you do not want user data in a program to be translated, you must turn translation off before you issue any write or read operations. See Appendix C for the code conversion tables that are used to translate your program data.

  XLATE-Y: Data is translated.
  XLATE-N: Data is not translated.

- **Setting parity:** When an asynchronous session is first acquired, the default value for the parity setting is the value configured in the line description. This value remains in effect until you issue another write function-management-header operation or deactivate the line. You can use the write function-management-header operation to change the parity setting in your session as follows:

  PARITY-N: Data is sent with no parity.
  PARITY-O: Data is sent with odd parity.
  PARITY-E: Data is sent with even parity.

- **Changing flow control:** Defaults to line description configuration value. XON/XOFF values are configured and cannot be changed.

  FLOW-Y: Turn on flow control; the hardware stops sending when an XOFF character is received and begins again when an XON character is received.
  FLOW-N: Turn off flow control; the hardware does not recognize XOFF and XON characters received as flow control characters.

  Note: If the function-management-header function is used to turn on flow control when no flow control characters have been specified in the line description, the system assumes hex 11 for XON and hex 13 for XOFF.

- **Changing the ECHO:** Defaults to line description configuration value. Echo is performed by the communications adapter.

  ECHO-N: Turn echo off; do not echo any characters.
  ECHO-A: Echo all characters received.
  ECHO-C: Controlled echo; echo all characters except end-of-record (EOR) characters.

Combinations can also be entered on one write function-management-header operation. However, if done, each operation must be separated by a comma, with no embedded blanks, as in the following example:

```
XLATE-Y,PARITY-N
```

The output length for this example is 16.

Receiving Data

Your program uses the read operation to receive data from a remote location, data echoed by the integrated PAD, or data from a PAD message or service signal. The following section describes the read operations that are supported for asynchronous communications.

Read Operation

Your program uses the read operation to obtain data from either the remote program with which your program is communicating, or an emulated PAD, or a PAD message. The read operation also causes your program to wait for the data if it is not available immediately. Your program then receives control when the data is available.

Note: The read operation obtains data from a specific program while the read-from-invited-program-devices operation allows the data to come from any previously invited device.

In asynchronous communications, the read operation can be issued by itself.

The asynchronous communications support attempts to maintain the data in logical records whenever possible. The following guidelines are used to provide your program with data during a read operation.

- A logical record will not exceed 4096 bytes and is determined in one of the following ways:
  - For asynchronous line descriptions, a logical record is defined as one of the following:
    - Data ended by an end-of-record (EOR) character, including any additional trailing characters received. The EOR character and trailing characters are specified by the EORTBL parameter on the asynchronous line description.
    - All data received prior to an inter-character idle time out. The inter-character idle time out is the length of time elapsed since the last byte of data was received. It is specified by the IDLTMR parameter on the asynchronous line description.
    - All data received in the communications adapter data buffer until the buffer becomes full. The size of the buffer is specified by the MAXBUFFER parameter on the asynchronous line description.
  - For X.25 line descriptions not using PAD support, each data packet received is treated as a logical record unless the more-data bit is on in the data packet. Packets containing the more-data bit are combined and treated as one logical record.
  - For X.25 line descriptions using PAD support configured to echo data to the terminal (your program), all data echoed by the PAD is considered to be one logical record. The echoed data is received by your program prior to any data received from the X.25 line.
- A default record, at least as large as the buffer size configured on the line description, should be specified.
- If the record received contains a parity error or stop bit (frame) error, your program receives a 0016 return code.
- If the record was received and data was lost (overrun situation), your program receives a 0042 return code.
- The asynchronous support does not exceed the input buffer length specified by your application program. If the amount of data available is greater than the amount requested by the read operation in your program, you must issue another read operation to get the remaining data.
- The asynchronous support does not cross a logical record boundary in satisfying a read operation. The actual length of the data supplied to your application program is available in the I/O feedback area. Your application program should always check this length before processing the data received.
- If a fail indication is received, your application program receives a 0302 return code and receives no data on the current read operation. Refer to Appendix D for more information on break and interrupt handling.
- If data is available at the time the read operation is issued, that data is returned to your program immediately. If data is not available, your program waits for the data and

control is returned to your program only when the data becomes available.

Invite Function

Your program uses the invite function to request input data from another program (through the associated session), but it receives control without waiting for the input. To obtain the data, your program must then issue either a read-from-invited-program-devices operation or read operation later in this transaction.

If your program issues a read operation following the invite function, the read operation satisfies the invite function. If you then want to issue a read-from-invited-program-devices operation, you must first issue another invite function because the read operation satisfied the previous invite function.

Read-From-Invited-Program-Devices Operation

Your program can use the read-from-invited-program-devices operation to obtain data from any device that has responded to an invite function that was previously issued in your program. If data becomes available to your program from more than one device before the read-from-invited-program-devices operation is issued, your program receives the data that was first made available.

A read-from-invited-program-devices operation should be issued to receive data only after an invite function is issued and/or a timer function is issued.

Data received on a read-from-invited-program-devices operation follows the same guidelines as those described for the read operation.

Waiting for a Display File, an ICF File, and a Data Queue

Use data queues when a program must wait for a display file, an ICF file, and a data queue, in any combination, at the same time. The following commands are used with the specified DTAQ parameter:

- Create Display File (CRTDSPF)
- Change Display File (CHGDSPF)
- Override Display File (OVRDSPF)
- Create ICF File (CRTICFF)
- Change ICF File (CHGICFF)
- Override ICF File (OVRICFF)

Use these commands to indicate a data queue that will have entries placed in it when one of the following occurs:

- An enabled command key or Enter key is pressed from an invited display device.
- Data becomes available when the session is invited for an ICF device.
- A user-defined entry is made to a data queue by a job running on the system.

For more information, see the *CL Programmer's Guide* and the *ICF Programmer's Guide*.

Notifying the Remote Program of Problems

Your program uses the functions described in this section to indicate that an error has occurred during the transaction with the target program.

Fail Function

Your program uses the fail function to indicate that it has detected an abnormal condition while it was sending or receiving data. Refer to Appendix D for additional information.

When a program that is sending data issues a fail function, either the data just sent was in error or some other condition occurred. However, the last record before the fail function was issued is still sent to the target program.

A program that is receiving data issues a fail function to indicate that the data it received was in error. The program issuing the fail function may then do an output operation so it can indicate why it sent the fail function. However, no data can be sent with a fail function. The record sent by the write operation should identify what the error is and where the other program should start again.

In either case, the program that issued the fail function 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.

Note: When a fail is sent or received, all data that was to be returned to the application program by asynchronous communications support is discarded.

Using Additional Functions and Operations

Additional functions available include the cancel-invite and the timer functions. Also available is the get-attributes operation.

Cancel-Invite Function

Your program uses the cancel-invite function to cancel a valid invite for which no input has yet been received. If data is in the input buffer, the function fails and the return code 0412 is received by the program. Your program must then issue a read operation to receive the data.

Timer Function

Your program uses the timer function to set the maximum amount of time your program waits to receive data when issuing a read-from-invited-program-devices operation.

When your program issues a read-from-invited-program-devices operation and receives data before the timer ends, a 0000 return code is received. However, if no data is received and the timer ends, a 0310 return code is sent to your program.

Get-Attributes Operation

Your program uses the get-attributes operation to determine the status of the session. It can be issued at any time during a session. The operation gets the current status information about the session to which your program is communicating.

Ending Transactions

The detach function is used to end a transaction.

Detach Function

The detach function is used to end a transaction between your program and the program with which it is communicating. The detach function is valid only when used with the evoke function. Any other use will cause an 831E return code to be sent to your program.

Ending a Session

The release operation or the end-of-session function is used to end a session.

Release Operation

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

- If the session was started by a source program, the release operation ends the session immediately. The operation frees the resources that were used during the session. If the release operation is not successful, the end-of-session function can be issued to end the session. The same or another session can then be started.
- If the session was started by a target program, the connection to the source program is ended, but the session still exists. Your program must issue an end-of-session function or go to end of job to end the session.

End-of-Session Function

Your program uses the end-of-session function to end a session. Unlike the release operation, the end-of-session function always ends the session (if it still exists), and gives a normal completion return code (0000). If the session does not exist, the end-of-session operation gives your program an 830B return code.

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

If your program does not issue an end-of-session function, the session exists until your program ends. To prevent your program from ending abnormally because of a communications error, you may want to code the end-of-session function 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 function could be used to end the session rather than trying the failing operation again in that session or specifying some special recovery action for each error.

If you have specified switched disconnect (SWTDSC(*YES)) on a controller description for a switched connection, the physical connection to the remote system is disconnected during end-of-session processing.

Using Response Indicators

Response indicators are defined to your program in the ICF file and are set on each input operation. However, these indicators are optional and major and minor return codes can also be used to indicate the status of input operations.

Receive-Fail Indicator

Use the receive-fail response indicator to determine if a fail function has been received. When a fail function is received, all data received by the asynchronous support and not given to the application program on a read operation is discarded.

Receipt of a fail request is also indicated by the return code 0302.

Using I/O Feedback Areas

Your program may have access to the I/O feedback area. If it does, you should be aware of certain fields when writing applications using asynchronous communications:

Actual received data length
This field contains the length of the data received on an input operation.

Major return code
This field contains the major return code indicating the status of input and output operations.

Minor return code
This field contains the minor return code indicating the status of input and output operations.

For more information about the I/O feedback area, see the *ICF Programmer's Guide*.

Using Return Codes

After each operation, an ICF return code is returned to your program. Your program should check this return code to determine:

- The status of the operation just completed
- The operation that should be issued next

For example, a major return code of 00 indicates that data was received. Along with this major code you can receive from asynchronous communications, for example, one of these minor codes:

- 16: Indicates the data received contains a parity error and/or a stop bit error. Your program should notify the remote program to send data again.
- 42: Indicates that some data was lost, perhaps due to an overrun situation. Your program should notify the remote program to send the data again or ensure that the maximum buffer length configured on the line description is sufficiently large.

Another example would be a major code of 83. In this case, an error was detected that may be recoverable. Different minor codes can be returned, just as for the 00 major. For example, if your program receives an E0 minor return code, your program tried to run an operation using a record format that was not defined for the file. You can check the name of the record format in your program to be sure it is correct, and then check to see whether the record format is defined in the file definition.

It is recommended that your program check the ICF return codes at the completion of every operation to ensure that the operation completed

successfully or that the appropriate recovery action has been taken.

Refer to Appendix B for a description of the return codes that can be returned to your application when it is using asynchronous communications.

Chapter 7. Asynchronous Communications Considerations

This chapter contains application and performance considerations for asynchronous communications programming.

Application Considerations

The following considerations need to be taken into account when writing your applications.

Evoke Confirmation

When your source program issues an evoke function, asynchronous communications does not determine whether or not the evoke was successful. Your program must ensure the evoke was successful. This can be done by issuing a write operation from the target program.

This check can be made using the following steps in your program:

- Issue an invite function after the evoke has been sent.
- Issue the timer function.

  Your program should set the timer value to the maximum length of time you expect to wait before receiving data from the remote system.
- Issue a read-from-invited-program-devices operation until:
  - Your program receives confirmation from the target program that the evoke was received
  - The specified timer value has expired (return code 0310).

The asynchronous communications support does not perform sequence checking, but issues return codes informing you of parity errors, stop bit errors, and data loss. However, because only minimal data integrity checking is done by the asynchronous communications support, it is possible for data to be lost without your program ever being notified. All record sequence error recovery and retransmission of data must be done by your program.

See Appendix B for more information about return codes and their meanings.

Function-Management-Header Function

If your program performs a write function management header (FMH) operation to turn flow control on, and the XON and XOFF characters have not been specified in the line description (XONCHAR and XOFFCHAR parameters), the asynchronous communications support uses the system default values for these characters. The default values are hex 11 for XON and hex 13 for XOFF.

Logical Records

The asynchronous communications support attempts to maintain the data in logical records whenever possible. The following guidelines are used to provide your program with data during a read operation.

A logical record will not exceed 4096 bytes and is determined in one of the following ways:

- For asynchronous line descriptions, a **logical record** is defined as one of the following:
  - Data ended by an end of record (EOR) character, including any additional trailing characters received. The EOR character and trailing characters are specified by the EORTBL parameter on the asynchronous line description.
  - All data received prior to an inter-character idle time out. The inter-character idle time out is the length of time elapsed since the last byte of data was received and is specified by the IDLTMR parameter on the asynchronous line description.
  - All data received in the communications adapter data buffer until the buffer becomes full. The size of the buffer is specified by the MAXBUFFER parameter on the asynchronous line description.
- For X.25 line descriptions not using PAD support, each data packet received is treated as a logical record unless the more-data bit is on in the data packet. Packets containing

the more-data bit are combined and treated as one logical record.

- For X.25 line descriptions using PAD support configured to echo data to the terminal (your program), all data echoed by the PAD is considered to be one logical record. The echoed data is received by your program prior to any data received from the X.25 line.

Prestarting Jobs for Program Start Requests

A program start request is a request made by your program to start a program on the remote system. When a source program issues an evoke function, this signals a program start request to the asynchronous communications support.

If the remote system is an AS/400 system, you can minimize the time required to carry out a program start request by using the prestart job entry to start a job on the remote system before your program sends a program start request. To use prestart jobs, you need to define both communications and prestart job entries in the same subsystem description, and make certain programming changes to the prestart job program with which your program communicates. For information about how to use prestart jobs, refer to the *ICF Programmer's Guide*.

Performance Considerations

The AS/400 system provides support for many devices, application programs, and services using asynchronous communications support. Asynchronous communications is not compatible with Systems Network Architecture (SNA). The performance of this support depends on the application program or service with which it is used and the speed of the line or network used. See the *Communications Management Guide* for general information about communications performance considerations.

Buffer Size

AS/400 asynchronous support uses buffers ranging in size from 128 to 4096 bytes. The maximum buffer size is determined by the value specified for the MAXBUFFER parameter in the line description. This value should be selected based on:

- The amount of time the input/output processor waits before passing the data up (using the *Idle timer* (IDLTMR) prompt).
- The amount of data received within a specified time period
- Whether or not end-of-record processing is being used
- The size of output requests

If EOR processing is used for all received records, MAXBUFFER should be configured to be the largest of the input and output requests, including the EOR character and any trailing characters.

If EOR processing is not used, MAXBUFFER should be configured to be the largest of your input and output requests.

Note: If you use file transfer support (FTS), the MAXBUFFER value must be at least 896.

Data Buffering Using XOFF Characters

When data arrives faster than a user application receives and processes it, the AS/400 system buffers the data until the application can accept it. 12KB of data (where 1KB = 1024 bytes) are buffered before sending an XOFF character to the remote system. The AS/400 system continues to send an XOFF character in response to each logical record received until the amount of data received by the application program reduces the amount of buffered data to less than 4KB. When the buffered data is below 4KB, the AS/400 system sends an XON character.

The AS/400 system will buffer up to 24KB before dropping the connection with the remote system.

Asynchronous Overhead

It may be possible to reduce the amount of overhead needed to send each character on the line by changing the definition of a character in the line description. For example, if you have configured 8 bits even parity and 2 stop bits, the total number of bits sent on the line would be 12. Note that a start bit and at least 1 stop bit are always sent. If instead you configured 8 data bits, 1 stop bit, and no parity, the total number of bits sent on the line would be 10. This would reduce the overhead by 17 percent. The remote system must accept the character format sent by the system.

Chapter 8. Using the Interactive Terminal Facility

The interactive terminal facility (ITF) is included as a part of the OS/400 asynchronous communications support. The **interactive terminal facility (ITF)** allows the AS/400 user to send and receive data through applications such as electronic message services for asynchronous terminals. Through ITF, you can use these applications to send messages such as interoffice memos. In addition, ITF lets you send and receive file members and send OfficeVision/400 documents.

Starting ITF

Before you can start ITF, you must start the asynchronous communications devices. After you have started asynchronous communications, type the following:

```
STRITF nnnnnnnn
```

where *nnnnnnnn* is the name of the remote location with which you want to communicate. This name is the same as the remote location name specified during configuration. For example, if you are using ITF to communicate with TELEMAIL and you specified MAIL as the remote location name for TELEMAIL, type the following:

```
STRITF MAIL
```

Notes:

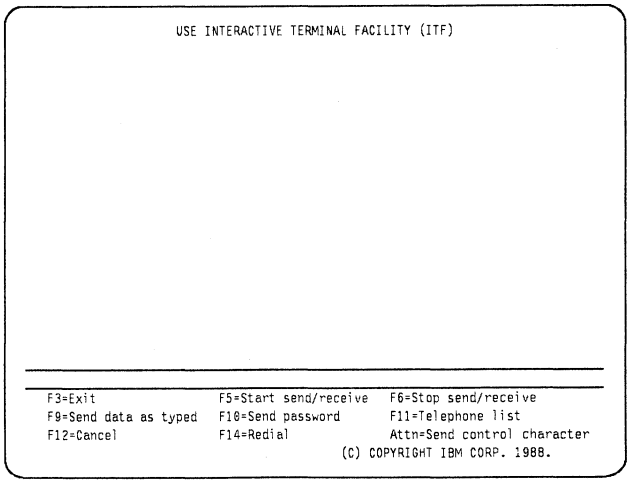
1. ITF can send file members with record lengths up to 2048 characters. However, some applications can only receive data with record lengths up to 132 characters. If the receiving application cannot accept the record length of the file member being sent, unpredictable results may occur.
2. ITF is intended primarily for applications with record lengths up to 132 characters. For longer record lengths, file transfer support can be used. See the *ICF Programmer's Guide* for more information about file transfer support.
3. To improve performance, configure the end-of-record table to match the end-of-record table of the remote application. This configuration reduces time-out conditions.

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

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

- Make a manual connection to the network by dialing the number on the telephone.
- If you are using a command-capable modem, type the modem dial command on the Use Interactive Terminal Facility (ITF) display and press the Enter key. ITF sends this command to the modem, which then calls the number (the modem must support this function).
- For an asynchronous line, press F11 from the Use Interactive Terminal Facility (ITF) display to work with the ITF telephone list. (F11 and F14 are not available for asynchronous/X.25 lines or for lines using packet assembler/disassembler [PAD] support.)

After you type the STRITF command, the Use Interactive Terminal Facility (ITF) display is shown.



The Use Interactive Terminal Facility (ITF) display is the first display for ITF. From this display, you can type commands and answer prompts to start the message service. Function 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. Therefore, when you are typing a message to be sent, you must press the Enter key or F9 at the end of each line. If the data entry line is full, ITF automatically sends the data for you.

When you press the Enter key or F9, ITF sends the data. If you press the Enter key to send the data, a carriage return (CR) is added to the data; if you press F9 to send the data, the CR is not added. The data then disappears from the data entry line of your display. If the remote system echoes the data, it is written again on your display.

Echo can be set on or off at a packet-switching data network (PSDN) PAD. However, you should not set echo off at the PAD when you are using ITF. Echo must be set on at the PAD for data that is sent to appear again on the display after you press the Enter key; it must also be set on for ITF to send file members and OfficeVision/400 documents.

Note: All data sent either from the display or from a data file, file member, or OfficeVision/400 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. A control character sent from the Send Control Character display is not translated.

Incoming data is displayed as it is received or echoed. The data is automatically rolled to the upper portion of the display (lines 2 through 17) if a **format effector** [1] is encountered or if 160 bytes of data have been received. Otherwise, the data is displayed on the data entry line. The old data that is rolled off the top is held in a buffer area. This data can be scanned using the roll keys. Up to six full displays of data, or approximately 96 lines of data, are held in the buffer area. After the buffer is full, old data is overlaid.

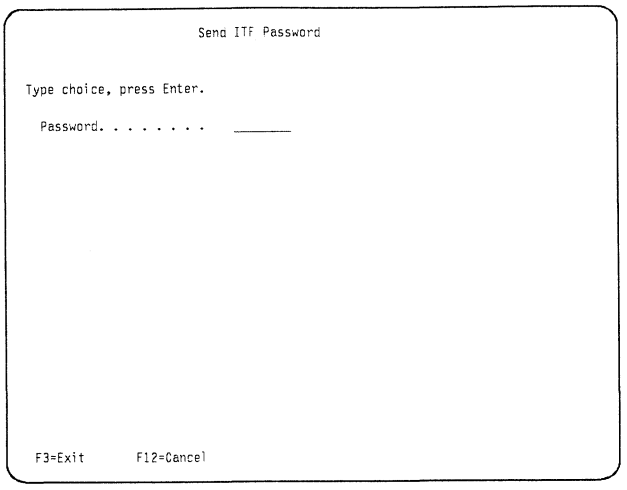
Selecting ITF Functions

The Use Interactive Terminal Facility (ITF) display is the first display for ITF. From this display, you can type commands and answer prompts to start the message service. You can also type a message to be sent. Function keys let you select other ITF functions.

The function keys have the following functions under ITF:

- *F1=Help:* Pressing F1 on any of the ITF displays shows help for using that display.
- *F3=Exit:* If you press F3 from the Use Interactive Terminal Facility (ITF) display, ITF ends. Pressing F3 from any other ITF display returns you to the Use Interactive Terminal Facility (ITF) display; in this case, ITF ignores any data that you may have typed on the display. You cannot use this function key if a data send or receive operation is in progress.
- *F5=Start send/receive:* To send or receive a file member or a document, press F5. The Start Send/Receive display appears; on this display, you can tell ITF to send data from a file or document or to receive data into a file. You cannot use this function key if a data send or receive operation is in progress.
- *F6=Stop send/receive:* To stop the sending or receiving of a file or document, press F6. ITF stops sending or receiving and returns control to you. For better performance, use the Attention key for the stop send/receive function.
- *F9=Send data as typed:* 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 F9 instead of the Enter key after typing data on the data entry line. If you are currently sending a file member or document, pressing F9 or the Enter key has no effect.
- *F10=Send password:* If the network asks for your password, press F10. ITF then shows the Send ITF Password display:

[1] A format effector is a control character used to position printed, displayed, or recorded data.



Type your password to the message application and press the Enter key. To keep your password secure, the characters are not displayed as you type them.

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

You cannot use this function key if a data send or receive operation is in progress.

- *F11=ITF telephone list:* To make a switched connection to the remote end, press F11. The Work with ITF Telephone List display is shown. If you are using PAD support or an asynchronous/X.25 line, F11 is not shown on the Use Interactive Terminal Facility (ITF) display. You cannot use this function key if a data send or receive operation is in progress.
- *F12=Cancel:* Pressing F12 cancels the operation associated with the display. You are returned to the display shown before the display on which F12 is pressed.
- *F14=Redial:* If you want to redial the last telephone number called from the Work with ITF Telephone List display, press F14. The system automatically calls the same number. F14 can only be used to redial numbers that were dialed from the Work with ITF Telephone List display. If you are using PAD support or an asynchronous/X.25 line, F14 is not shown on the Use Interactive Terminal Facility (ITF) display. You cannot use this function key if a data send or receive operation is in progress.
- *Attn=Send control character:* If you press the Attn (Attention) key, the Send Control Character display is shown. From this display, you can select options to stop the send/receive process or to send either a break or a control character.

Sending or Receiving a File Member or OfficeVision/400 Document

With ITF, you can send a file member or OfficeVision/400 document or you can place received data into a file member. ITF adds a carriage return (CR) to the end of each record sent from a file member or OfficeVision/400 document.

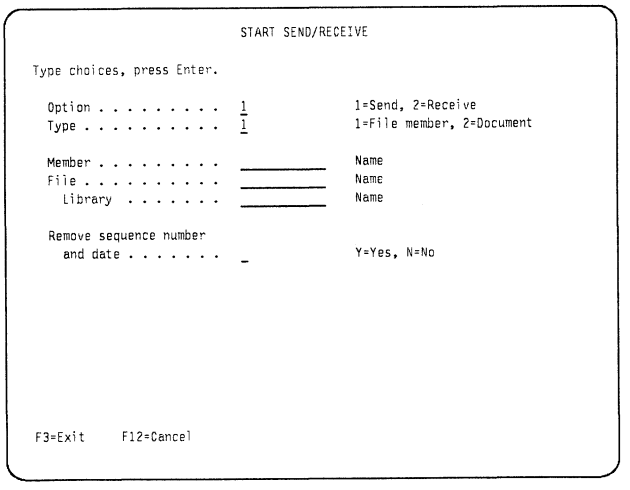
Notes:

1. ITF does not verify data integrity; unpredictable results may occur if you send a file member that contains non-text data (such as hexadecimal characters).
2. ITF can send files containing up to 32 767 records. Unpredictable results will occur if you try to use ITF to send a file containing more than 32 767 records. This limitation does not apply to receiving files.

To select a file member or document, press F5 when the Use Interactive Terminal Facility (ITF) display is shown. The Start Send/Receive display is shown.

Sending or Receiving a File Member:

To send a file member, type 1 (Send) in the *Option* field and 1 (File member) in the *Type* field, and press the Enter key. The Start Send/Receive display is shown again with additional fields.



Type the name of the member to be sent, the name of the file that contains this member, and the name of the library that contains the file. Type either Y (Yes) or N (No) in the *Remove sequence number and date* field. If the member record size is less than 13 bytes, the *Remove sequence number and date* field is not valid. If you type Y, the first 12 bytes of each record you send are deleted. Press the Enter key.

The Use Interactive Terminal Facility (ITF) display is shown again and ITF immediately starts sending the member. As each record in the member is sent, either the PAD or the remote device must echo it so that it is shown on your display. (ITF cannot send file members if the PAD or remote device does not echo.) If you press F6 (Stop send/receive) at this time, you stop sending the data in the file member. ITF always sends the last complete record before it stops sending. When the last record is sent, ITF displays a message:

```
Last record sent
```

If you type 2 (Receive) in the *Option* field and the Type is 1 (File member), the Start Send/Receive display is shown again with additional fields.

```
                        START SEND/RECEIVE

Type choices, press Enter.

  Option . . . . . .  2          1=Send, 2=Receive
  Type . . . . . .    1          1=File member, 2=Document

  Member . . . . . .  _____      Name
  File . . . . . .    _____      Name
    Library . . . . . .  _____   Name

  Replace . . . . . .  _         1=Replace, 2=Append
  Convert to source . . .  _     Y=Yes, N=No

F3=Exit     F12=Cancel
```

If the file you specify already exists, ITF asks if you want to replace the existing file with the new data. If you type 1 (Replace), ITF writes the received data into the existing file member, writing over the previous contents. If you type 2 (Append), ITF adds the received data to the end of the existing file member. Type either Y (Yes) or N (No) in the *Convert to source* field. If you type Y, the data is received, starting at the 13th byte of each record. Bytes 1 through 12 of each record are used for sequence number and date. If the existing member has a record size less than 13 bytes, the *Convert to source* field is not valid. If you type N, the data is received, starting at the first byte of each record.

After you have made your selections, ITF returns to the Use Interactive Terminal Facility (ITF) display while receiving data into the file. If you press F6 (Stop Send/Receive) at this time, you stop receiving data in the file member. However, partial records are not written into the file member. Only complete records are written into the file member.

If you type 2 (Receive) in the *Option* field and the Type is 1 (file member) but the member that you specify does not already exist in the library, ITF prompts you for more information to create a new file member.

If you type 1 (Source) in the *Receive type* field, the following display is shown.

```
                        START SEND/RECEIVE

Type choices, press Enter.

  Option . . . . . .  2          1=Send, 2=Receive
  Type . . . . . .    1          1=File member, 2=Document

  Member . . . . . .  _____      Name
  File . . . . . .    _____      Name
    Library . . . . . .  _____   Name

  Receive type . . . . . .  1    1=Source, 2=Search for header
  Record size . . . . . .  ___   1-2048
  Convert to source . . .  _     Y=Yes, N=No

F3=Exit     F12=Cancel
```

Record size information is handled in two ways. If you are creating a new file and file member, type the record size. If the file you specified already exists but the member does not, the *Record size* field is ignored, even though it is required. The record size of the member is determined by the record size specified in the file attributes.

After the record size is specified, type either Y (Yes) or N (No) in the *Convert to source* field. If you type Y, the data is received, starting at the 13th byte. If you type Y in the *Convert to source* field, you must specify a record size of at least 13 bytes. Bytes 1 through 12 are used for

sequence number and date. If you type N, the data is received starting at the first byte of each record. When you press the Enter key, ITF then returns to the Use Interactive Terminal Facility (ITF) display. While the receive operation is in progress, ITF displays the following message:

```
Receive
```

ITF builds headers for documents and files of documents. These headers contain the number of records and the record length. Maximum record length is 120 characters.

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

If you are receiving a file member that was not sent with a header, specify 1 (Source) in the *Receive type* field and type the record size. All data received is written to the file member.

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

Sending or Receiving OfficeVision/400 Documents: To send an OfficeVision/400 document, type 1 in the *Option* field, type 2 (Document) in the *Type* field, and press the Enter key. The following display is shown:

```
                        START SEND/RECEIVE

Type choices, press Enter.

  Option . . . . . .  1          1=Send, 2=Receive
  Type . . . . . .    2          1=File member, 2=Document

  Document . . . . .  _________  Name, *ALL
    Folder . . . . .             Name

F3=Exit     F12=Cancel
```

Either enter the name of the document or type *ALL for all of the documents in a folder. Then type the name of the folder that contains the documents. After you press the Enter key, the Use Interactive Terminal Facility display is shown again.

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

```
Document(s) sent
```

ITF only sends the first 120 characters of each line in the document. Any characters beyond the first 120 are truncated and the following message is shown:

```
Document(s) sent.  End of data dropped.
```

If *ALL was specified for the document name, all of the documents in the folder are 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 handled as separate records.
2. ITF attempts to maintain line integrity such as blank lines.
3. A blank line is sent between pages of a document and between documents.

You can use the Print Document (PRTDOC) command to resolve the documents to the file that you select. See the online information for more information on the PRTDOC command. After you resolve the documents to a file member, you can use the ITF Start Send/Receive display to send the documents as a file. After the file member is sent, messages are displayed that indicate the number of documents sent and if those documents were truncated.

ITF cannot directly receive OfficeVision/400 documents. Therefore, the information should be received as a new file member with a record length of 120. To receive these documents, use the Start Send/Receive display and specify the Receive type as 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 file member. If

the number of records specified in the header is exceeded, the additional records are not written into the new file member. These records are displayed on the terminal as data.

Work with ITF Telephone List

The Work with ITF Telephone List function allows you to maintain a list of telephone numbers.

Note: This function is not available if the device you are using is attached to an X.25 line. You can call the numbers on the list, or you can change, add, or delete numbers from the telephone list. If you press F11 (ITF telephone list) from the Use Interactive Terminal Facility (ITF) display, the Work with ITF Telephone List display is shown:

```
                    Work with ITF Telephone List

Library . . . . . :

Type options, press Enter.
  1=Call number    2=Change    4=Delete

Option  Description      Prefix   Telephone number

  _     ________         ___      ________________________
  _     ________         ___      ________________________
  _     ________         ___      ________________________
  _     ________         ___      ________________________
  _     ________         ___      ________________________
  _     ________         ___      ________________________
  _     ________         ___      ________________________
  _     ________         ___      ________________________
  _     ________         ___      ________________________
  _     ________         ___      ________________________
  _     ________         ___      ________________________
  _     ________         ___      ________________________
  _     ________         ___      ________________________
  _     ________         ___      ________________________
                                                      More...
F3=Exit     F6=Add entries     F12=Cancel
```

This display lists the telephone numbers that you can call from ITF. Position the cursor in the function field of the telephone number you want to work with. Type the option; then press the Enter key. Using the options, you can call a number, change a number, or delete a number from the telephone list.

- *Option:*
  - To call a number, type 1 in the *Option* field of the number you want to call.
  - To change a number, type 2 in the *Option* field of the number you want to change, and make the changes.
  - To delete a number, type 4 in the *Option* field of the number you want to delete.
- *Description:* This field is optional. You can use this field to type the name of the location with which you wish to communicate.
- *Prefix:* If you are communicating through a command-capable modem, 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 and you can leave it blank.
- *Telephone number:* Type the telephone number of the remote location to which you want to communicate. If you are communicating through a packet-switching data network (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.

You can select options in more than one option field before pressing the Enter key. However, ITF processes all delete options first, followed by any changes, and then calls the first number selected. When the system calls a number from the Work with ITF Telephone List display, it does not process other call options selected after the call request.

If more than one call request is made (option 1), only the first call requested is made.

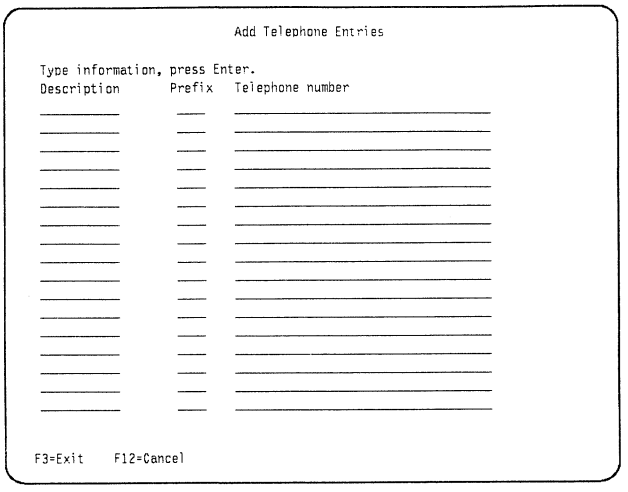
When you press F3, the Use Interactive Terminal Facility (ITF) display is shown again. Any data that you have typed on the Work with ITF Telephone List display is ignored.

After the system makes a connection with the network, the Use Interactive Terminal Facility (ITF) display is shown again. You must now sign on the message application. Refer to the operator's manual for the application sign-on and sign-off commands and for the send and receive message commands.

The telephone list is in the file member #ITFPHONE. The file name is #ITFPHONE and the library is the current one. The *Library* field on the Work with ITF Telephone List display shows the current library name.

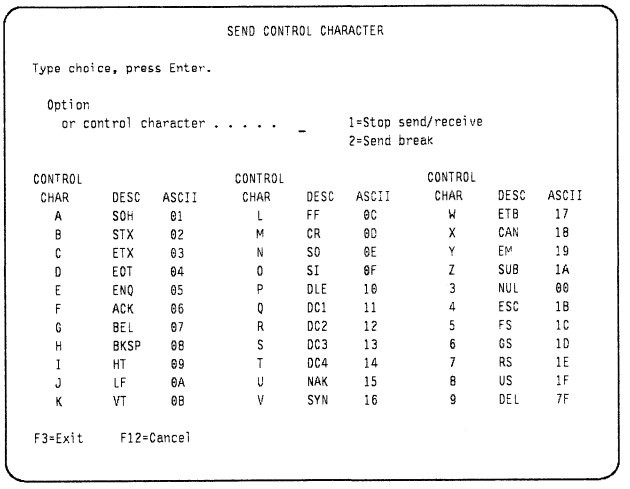
If you press F6 (Add entries), the Add Telephone Entries display is shown. New entries are not processed into the telephone list until you press the Enter key. All entries must be unique. You

can leave the description and prefix fields blank. You can have up to 175 telephone entries.



Using the Attn Key to Send a Control Character

When you press the Attn (Attention) key, the Send Control Character display is shown.



Each letter or number is associated with a function (option 1 or 2) or with a control character. Type your choice and press the Enter key. ITF does the requested action and then returns to the Use Interactive Terminal Facility (ITF) display. Only one function or one control character at a time can be sent from this display.

If you select option 1 (Stop send/receive), all send or receive processing is stopped. If you select option 2 (Send break), a break signal is sent to the host system application. The control characters sent from this display are not translated. No carriage return (CR) is added to the control character.

Appendix A. Language Operations, DDS Keywords, and System-Supplied Formats

This appendix contains charts describing:

- All valid communications operations supported by the intersystem communications function (ICF)
- Valid operations for each programming language that supports ICF
- Data description specifications (DDS) processing keywords
- System-supplied formats

ICF Operations and Supported Language Operations

Figure A-1 describes the language operations supported by ICF.

Figure A-1. ICF Operations

ICF Operations	Description
Open	Opens the ICF file.
Acquire	Establishes a session between the application and the remote location.
Get attributes	Used to determine the status of the session.
Read	Obtains data from a specific session.
Read-from-invited-program-devices	Obtains data from any session that has responded to an invite function.
Write	Passes data records from the issuing program to the other program in the transaction.
Write/Read	Allows a write operation followed by a read operation. Valid for RPG/400 only.
Release	Attempts to end a session.
Close	Closes the ICF file.

Figure A-2 shows all the valid operations for each programming language that supports ICF (C/400, COBOL/400, FORTRAN/400, and RPG/400 programming languages).

Figure A-2 (Page 1 of 2). Valid Operations for Programming Languages

ICF Operation	RPG/400 Operation Code	COBOL/400 Procedure Statement	C/400 Function	FORTRAN/400 Statement
Open	OPEN	OPEN	fopen, _Ropen	OPEN
Acquire	ACQ	ACQUIRE	QXXACQUIRE, _Racquire	Not supported[2]
Get attributes	POST	ACCEPT	QXXDEVATR, _Rdevatr	Not supported
Read	READ	READ	fread, _Rreadn	READ
Read-from-invited-program-devices	READ[1]	READ[1]	QXXREADINVDEV followed by an fread, _Rreadindv	Not supported
Write	WRITE	WRITE	fwrite, _Rwrite	WRITE

Figure A-2 (Page 2 of 2). Valid Operations for Programming Languages

ICF Operation	RPG/400 Operation Code	COBOL/400 Procedure Statement	C/400 Function	FORTRAN/400 Statement
Write/Read	EXFMT	Not supported	_Rwriterd	Not supported
Release	REL	DROP	QXXRELEASE, _Rrelease	Not supported
Close	CLOSE	CLOSE	fclose, _Rclose	CLOSE

1 A read operation can be directed either to a specific program device or to any invited program device. The support provided by the compiler you are using determines whether to issue an ICF read or read-from-invited-program-devices operation, based on the format of the read operation. For example, if a read is issued with a specific format or terminal specified, the read operation is interpreted as an ICF read operation. Refer to the appropriate language reference manual for more information.

2 To acquire a program device using FORTRAN/400, you must specify the program device on the ACQPGMDEV parameter on the CRTICFF, CHGICFF, or OVRICFF commands. The program device will then be implicitly acquired when the ICF file is opened.

DDS Keywords

The following table lists the DDS keywords that are valid for asynchronous communications.

Figure A-3. DDS Keywords

DDS Keyword	Description
CNLINVITE	Cancels any invite function for which no input has been received.
DETACH	Ends the transaction with the target system. **Note:** DETACH is only valid when used with the EVOKE keyword.
EOS	Ends a communications session.
EVOKE	Starts a program on the remote system.
FAIL	Notifies the remote program that an error has occurred.
FMH	Informs the remote program that function-management-header data is being sent.
INVITE	Schedules an invite function.
RCVFAIL	Indicates that the remote program has sent a fail.
RECID	Used to allow the data content to identify the record format to use to receive the data. **Note:** Refer to the *ICF Programmer's Guide* for more information about the RECID keyword.
SECURITY	Includes security information needed to start a program on the target system. Valid only with an EVOKE keyword.
TIMER	Allows you to specify an interval of time to wait before a read-from-invited-program-devices operation receives a timer-expired return code.
VARLEN	Allows you to specify, at run time, the length of the data to be sent across the communications line. **Note:** Refer to the *ICF Programmer's Guide* for more information about the VARLEN keyword.

System-Supplied Formats

The following table lists all the keyword functions performed by the system-supplied formats that are valid for asynchronous communications. Refer to *ICF Programmer's Guide* for more information about system-supplied formats.

Figure A-4. System-Supplied Formats

System-Supplied Formats	Description
$$CNLINV	Cancel invite
$$EOS	End of session
$$EVOK	Evoke with invite
$$EVOKET	Evoke with detach
$$EVOKNI	Evoke
$$FAIL	Fail
$$SEND	Write then invite or invite
$$SENDNF	Write function-management-header
$$SENDNI	Write
$$TIMER	Timer

Appendix B. Return Codes, Messages, and Sense Codes

Return Codes

This section describes all the return codes that are valid for asynchronous communications. These return codes are set in the I/O feedback area of the ICF file; they report the results of each I/O operation issued by your application program. Your program should check the return code and act accordingly. Refer to your high-level language manual for more information on how to access these return codes.

Each return code is a four-digit hexadecimal value. The first two digits contain the *major code*, and the last two digits contain the *minor code*.

With some return codes, a message is also sent to the job log or the system operator message queue (QSYSOPR). You can refer to the message for additional information.

Notes:

1. In the return code descriptions, *your program* refers to the local AS/400 application program that issues the operation and receives a return code from ICF communications. The *remote program* refers to the application program on the remote system with which your program is communicating through ICF.
2. Several references to input and output operations are made in the descriptions. These operations can include DDS keywords and system-supplied formats, which are listed in Appendix A.

Major Code 00

Major Code 00 — Operation completed successfully.

Description: The operation issued by your program completed successfully. Your program may have sent or received some data, or may have received a message from the remote system.

Action: Examine the minor return code and continue with the next operation.

Code	Description/Action
0000	**Description:** For input operations issued by your program, 0000 indicates that your program received some data on a successful input operation. Your program can continue to receive data, or it can send data to the remote program. For output operations issued by your program, 0000 indicates that the last output operation completed successfully and that your program can continue to send data.

Action: For the actions which can be taken after 0000 is received, refer to the following table:

Figure B-1. Actions for Return Code 0000

Type of Session	Last Operation Issued	Actions Your Program Can Take
Started by a source program	Acquire or open	Issue an evoke or timer function, or a get-attributes operation.
	Evoke with detach or write with detach	Issue another evoke function, issue a release operation, continue local processing, or end.
	Any other output operation	Issue another output operation (except evoke), or issue an input operation.
	End-of-Session	Continue local processing or end.
Started by a remote program start request[1]	Acquire or open	Issue an input or output operation.
	Write with detach	Continue local processing or end. This session has ended.
	Any other output operation	Issue another output operation (except evoke), or issue an input operation.
	End-of-Session	Continue local processing or end.

[1] A target program (started by a program start request) cannot issue an evoke function in this session; it can issue an evoke function only in a different session that it has first acquired.

0004 **Description:** On a successful input operation, your program received a PAD message from the remote PAD. The message may be a parameter indication, an error indication, or an invitation to clear. See "PAD Messages" on page 3-6 for more information about PAD messages.

Action: Process the PAD message.

0016 **Description:** On a successful input operation, your program received some data containing a parity error and/or a stop bit error (framing).

Action: Notify the remote program to send the data again.

Messages:

CPD6B91 (Diagnostic)

0042 **Description:** Your program received some data on a successful input operation. However, some data was lost, possibly due to an overrun situation.

Action: Notify the remote program to send the data again, and ensure that the maximum buffer length configured on the line description is large enough to contain any expected data.

Messages:

CPD6B92 (Diagnostic)

Major Code 02

Major Code 02 — Input operation completed successfully, but your job is being ended (controlled).

Description: The input operation issued by your program completed successfully. Your program may have received some data or a message from the remote system. However, your job is being ended (controlled).

Action: Your program should complete its processing and end as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

Code	Description/Action
0200	**Description:** On a successful input operation, your program received some data. Also, your job is being ended (controlled). **Action:** Your program can continue to receive data, or it can send data to the remote program. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.
0204	**Description:** On a successful input operation, your program received a PAD message from the remote PAD. The message may be a parameter indication, an error indication, or an invitation to clear. See "PAD Messages" on page 3-6 for more information about PAD messages. Also, your job is being ended (controlled). **Action:** Your program can process the PAD message. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.
0216	**Description:** On a successful input operation, your program received some data containing a parity error and/or a stop bit error (framing). Also, your job is being ended (controlled). **Action:** Your program can notify the remote program to send the data again. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job. **Messages:** CPD6B91 (Diagnostic)
0242	**Description:** Your program received some data on a successful input operation. However, some data was lost, possibly due to an overrun situation. Also, your job is being ended (controlled). **Action:** Your program can notify the remote program to send the data again, and ensure that the maximum buffer length configured on the line description is large enough to contain any expected data. However, the recommended action is to complete all processing and

end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

Messages:

CPD6B92 (Diagnostic)

Major Code 03

Major Code 03 — Input operation completed successfully, but no data received.

Description: The input operation issued by your program completed successfully, but no data was received.

Action: Examine the minor return code and continue with the next operation.

Code	Description/Action
0300	**Description:** On a successful input operation, your program received no data to process. The session is still active. **Action:** Issue an input or output operation.
0302	**Description:** On a successful input operation, your program received a fail indication without any data. Either the remote program has sent a fail function, or the system has detected a break condition. All data received by the asynchronous support that is not given to your program on an input operation is discarded. **Action:** Issue an input operation to receive the reason for the fail from the remote program. **Messages:** CPD6B92 (Diagnostic)
0309	**Description:** On a read-from-invited-program-devices operation, your program did not receive any data. Also, your job is being ended (controlled). **Action:** Your program can continue processing. However, the recommended action is to complete all processing and end your program as soon as possible. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job. **Messages:** CPF4741 (Notify)
0310	**Description:** On a read-from-invited-program-devices operation, the time interval specified by a timer function in your program or by the WAITRCD value specified for the ICF file expired. **Action:** Issue the intended operation after the specified time interval has ended. For example, if you were using the time interval to control the length of time to wait for data, you can issue another read-from-invited-program-devices operation to receive the data.

Note: Since no specific program device name is associated with the completion of this operation, the program device name in the common I/O feedback area is set to *N. Therefore, your program should not make any checks based on the program device name after receiving the 0310 return code.

Messages:

CPF4742 (Status)
CPF4743 (Status)

Major Code 04

Major Code 04 — Output exception occurred.

Description: An output exception occurred because your program attempted to send data when it should be receiving data. The data from your output operation was not sent. You can attempt to send the data later.

Action: Issue an input operation to receive the data.

Code	Description/Action
0412	**Description:** An output exception occurred because your program attempted to send data when it should be receiving data available from the remote program or from the PAD. The data from your output operation was not sent to the remote system. Your program can attempt to send the data later. **Action:** Issue an input operation to receive the data. **Note:** If your program issues another output operation before an input operation, your program receives a return code of 831C. **Messages:** CPF4750 (Notify) CPF5076 (Notify)

Major Codes 08 and 11

Major Codes 08 and 11 — Miscellaneous program errors occurred.

Description: The operation just attempted by your program was not successful. The operation may have failed because it was issued at the wrong time.

Action: Refer to the minor code description for the appropriate recovery action.

Code	Description/Action
0800	**Description:** The acquire operation just attempted by your program was not successful. Your program tried to acquire a program device that was already acquired and is still active. **Action:** If the session associated with the original acquire operation is the one needed, your program can begin communicating in that session since it is already available. If you want a different session, issue another acquire operation for the new session by specifying a different program device name in the PGMDEV parameter of the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command that precedes the program. **Messages:** CPD4077 (Diagnostic) CPF5041 (Status) CPF50A0 (Status)
1100	**Description:** The read-from-invited-program-devices operation just attempted by your program was not successful because your program tried this operation when no program devices were invited and no timer function was in effect. **Action:** Issue an invite function (or a combined operation that includes an invite) followed by a read-from-invited-program-devices operation. **Messages:** CPF4740 (Notify)

Major Code 34

Major Code 34 — Input exception occurred.

Description: The input operation attempted by your program was not successful. The data received was too long for your program's input buffer or was not compatible with the record format specified on the input operation.

Action: Refer to the minor code description for the appropriate recovery action.

Code	Description/Action
3441	**Description:** A valid record format name was specified with format selection type *RECID. However, although the data received matched one of the record formats in the ICF file, it did not match the format specified on the read operation. **Action:** Correct your program to issue a read operation that does not specify a record format name, or specify the correct record format name to process the data based on the format selection option for the file. **Messages:** CPF5058 (Notify)

3451 **Description:** Your program specified a file record size that was not large enough for the indicators to be included with the data sent by the remote program (for a file defined with a nonseparate indicator area). Your program did not receive any data. For a file using a nonseparate indicator area, the actual record length field in the device-dependent I/O feedback area contains the number of indicators specified by the record format.

Action: End the session; close the file; correct the file record size; then open the file again.

Messages:

CPF4768 (Notify)

Major Code 80

Major Code 80 — Permanent system or file error (irrecoverable).

Description: An irrecoverable file or system error has occurred. The underlying communications support may have ended and your session has ended. If the underlying communications support ended, it must be established again before communications can resume. Recovery from this error is unlikely until the problem causing the error is detected and corrected.

Action: You can perform the following general actions for all 80xx return codes. Specific actions are given in each minor code description.

- Close the file, open the file again, then establish the session. If the operation is still not successful, your program should end the session.
- Continue local processing.
- End.

Note: If the session is started again, it starts from the beginning, not at the point where the session error occurred.

Code **Description/Action**

8081 **Description:** The operation attempted by your program was not successful because a system error condition was detected.

Action: Your communications configurations may need to be varied off and then on again. Your program can do one of the following:

- Continue local processing.
- Close the ICF file, open the file again, and establish the session again.
- End.

Messages:

CPF4170 (Escape)
CPF4510 (Escape)
CPF5257 (Escape)
CPF5447 (Escape)

8082 **Description:** The operation attempted by your program was not successful because the device supporting communications between your program and the remote location is not usable. For example, this may have occurred because communications were stopped for the device by a Hold Communications Device (HLDCMNDEV) command, or because a cancel reply was issued in response to an error recovery message for the device. Your program should not issue any operations to the device.

Action: Communications with the remote program cannot resume until the device has been reset to a varied on state. If the device has been held, use the Release Communications Device (RLSCMNDEV) command to reset the device. If the device is in an error state, vary the device off and then on again. Your program can attempt to establish the session again, continue local processing, or end.

Messages:

CPF4744 (Escape)
CPF5269 (Escape)

80B3 **Description:** The open operation issued by your program was not successful because the ICF file is in use by another process.

Action: Wait for the file to become available, then issue another open operation. Otherwise, your program may continue processing, or it can end.

Consider increasing the WAITFILE parameter with the Change ICF File (CHGICFF) or Override ICF File (OVRICFF) command to allow more time for the file resources to become available.

Messages:

CPF4128 (Escape)

80EB **Description:** The open operation attempted by your program was not successful due to one of the following:

- Your program used an option of update or delete to open the file, but that option is not supported by the program device.
- Your program requested both blocked data and user buffers on an open option, but these formats cannot be selected together.
- Your program tried to open a source file, but the file was not created as a source file.
- There is a mismatch on the INDARA keyword between your program and the ICF file as to whether or not a separate indicator area should be used.
- The file was originally opened as a shared file; however, no program devices were ever acquired for the file before your program attempted the current open operation.

Action: After performing one of the following actions, your program can try the open operation again:

- If the update and delete options are not supported for the program device, use an option of input, or output, or both.
- If your program tried selecting user buffers and blocked data together, it should try selecting one or the other, but not both.
- If your program tried to open a non-source file as a source file, either change the file name or change the library name.

- If there was a mismatch on the INDARA keyword, either correct the file or correct your program so that the two match.
- If no program devices were previously acquired for a shared file, acquire one or more program devices for the file.

Messages:

CPF4133 (Escape)
CPF4156 (Escape)
CPF4238 (Escape)
CPF4250 (Escape)
CPF4345 (Escape)
CPF5522 (Escape)
CPF5549 (Escape)

80ED **Description:** The open operation attempted by your program was not successful because there is a record format level mismatch between your program and the ICF file.

Action: Close the file. Compile your program again to match the file level of the ICF file, or change or override the file to LVLCHK(*NO); then open the file again.

Messages:

CPF4131 (Escape)

80EF **Description:** Your program attempted an open operation on a file or library for which the user is not authorized.

Action: Close the file. Either change the file or library name on the open operation, or obtain authority for the file or library from your security officer. Then issue the open operation again.

Messages:

CPF4104 (Escape)

80F8 **Description:** The open operation attempted by your program was not successful because one of the following occurred:

- The file is already open.
- The file is marked in error on a previous return code.

Action:

- If the file is already open, close the file and end your program. Remove the duplicate open operation from your program, then issue the open operation again.
- If the file is marked in error, your program can check the job log to see what errors occurred previously, then take the appropriate recovery action for those errors.

Messages:

CPF4132 (Escape)
CPF5129 (Escape)

Major Code 81

Major Code 81 — Permanent session error (irrecoverable).

Description: An irrecoverable session error occurred during an I/O operation. Your session cannot continue and has ended. Before communications can resume, the session must be established again by using an acquire operation or another program start request. Recovery from this error is unlikely until the problem causing the error is detected and corrected. Operations directed to other sessions associated with the file should work.

Action: You can perform the following general actions for all 81xx return codes. Specific actions are given in each minor return code description.

If your program initiated the session, you can:

- Correct the problem and establish the session again. If the operation is still not successful, your program should end the session.
- Continue processing without the session.
- End.

If your session was initiated by a program start request from the remote program, you can:

- Continue processing without the session.
- End.

Several of the minor codes indicate that an error condition must be corrected by changing a value in the communications configuration or in the file.

- To change a parameter value in the communications configuration, vary the configuration off, make the change to the configuration description, then vary the configuration on.
- To change a parameter value in the file, use the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command.

  Note: When a parameter can be specified both in the ADDICFDEVE or OVRICFDEVE command and in the configuration, the value in the ADDICFDEVE or OVRICFDEVE command overrides the value specified in the configuration (for your program only). Therefore, in some cases, you may choose to make a change with the ADDICFDEVE or OVRICFDEVE command rather than in the configuration.

Several other minor codes indicate a line or remote system error and may require an operator to correct the error.

Note: If the session is started again, it starts from the beginning, not at the point where the session error occurred.

Code	Description/Action
8140	**Description:** A cancel reply was received from your program or from the operator in response to a notify message, or was the result of a system default, causing the session to be ended. The session is no longer active. **Action:** If your program started the session, issue an acquire operation to start the session again. If your program was started by a program start request, it can continue local processing or end. **Messages:** CPF5104 (Escape)
8191	**Description:** A permanent line or controller error occurred on an input or output operation, and the system operator attempted recovery in response to the error message. You can learn what type of line error occurred by checking the system operator's message queue. The session has ended. Data may have been lost. **Action:** If your program started the session, issue an acquire operation to start the session again. If your program was started by a program start request from the remote program, it can continue local processing or end. **Messages:** CPF4146 (Escape) CPF4155 (Escape) CPF5128 (Escape) CPF5138 (Escape) CPF6B82 (Escape) CPF6B83 (Escape)
81E9	**Description:** An input operation was issued and the format selection option for the ICF file was *RECID, but the data received did not match any record formats in the file. There was no format in the file defined without a RECID keyword, so there was no default record format to use. The session has ended. **Action:** Verify that the data sent by the remote program was correct. If the data was not correct, have the operator on the remote system change the remote program to send the correct data. If the data was correct, add a RECID keyword definition to the file that matches the data, or define a record format in the file without a RECID keyword so that a default record format can be used on input operations. If your program started the session, use another acquire operation to start the session again. If a program start request started your program, continue local processing or end. **Messages:** CPF5291 (Escape)

Major Code 82

Major Code 82 — Open or acquire operation failed.

Description: Your attempt to establish a session was not successful. The error may be recoverable or permanent, and recovery from it is unlikely until the problem causing the error is detected and corrected.

Action: You can perform the following general actions for all 82xx return codes. Specific actions are given in each minor code description.

If your program was attempting to start the session, you can:

- Correct the problem and attempt to establish the session again. The next operation could be successful only if the error occurred because of some temporary condition such as the communications line being in use at the time. If the operation is still not successful, your program should end.
- Continue processing without the session.
- End.

If your session was initiated by a program start request from the remote program, you can:

- Correct the problem and attempt to connect to the requesting program device again. If the operation is still not successful, your program should end.
- Continue processing without the session.
- End.

Several of the minor codes indicate that an error condition must be corrected by changing a value in the communications configuration or in the file.

- To change a parameter value in the communications configuration, vary the configuration off, make the change to the configuration description, then vary the configuration on.
- To change a parameter value in the file, use the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command.

  Note: When a parameter can be specified both in the ADDICFDEVE or OVRICFDEVE command and in the configuration, the value in the ADDICFDEVE or OVRICFDEVE command overrides the value specified in the configuration (for your program only). Therefore, in some cases, you may choose to make a change with the ADDICFDEVE or OVRICFDEVE command rather than in the configuration.

If no changes are needed in your file or in the configuration (and depending on what the return code description says):

- If the attempted operation was an acquire, issue the acquire operation again.
- If the attempted operation was an open, close the file and issue the open operation again.

Code	Description/Action
8209	**Description:** The open or acquire operation issued by your program was not successful because a prestart job is being canceled. One of the following may have occurred: • An End Job (ENDJOB), End Prestart Job (ENDPJ), End Subsystem (ENDSBS), End System (ENDSYS), or Power Down System (PWRDWNSYS) command was being issued. • The maximum number of prestart jobs (MAXJOBS parameter) was reduced by the Change Prestart Job Entry (CHGPJE) command. • The value for the maximum number of program start requests allowed (specified in the MAXUSE parameter on the ADDPJE or CHGPJE command) was exceeded. • Too many unused prestart jobs exist. • The prestart job had an initialization error. **Action:** Complete all processing and end your program as soon as possible. Correct the system error before starting this job again. **Messages:** CPF4292 (Escape) CPF5313 (Escape)
8233	**Description:** A program device name that was not valid was detected. Either an ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command was not run, or the program device name in your program does not match the program device name specified in the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command for the session being acquired. The session was not started. **Action:** If the error was in your program, change your program to specify the correct program device name. If an incorrect identifier was specified in the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command, specify the correct value in the PGMDEV parameter. **Messages:** CPF4288 (Escape) CPF5068 (Escape)
8281	**Description:** On an unsuccessful open or acquire operation, a system error condition was detected. For example, the file may previously have been in error, or the file could not be opened due to a system error. **Action:** Your communications configurations may need to be varied off and then on again. Your program can do one of the following: • Continue local processing. • Close the ICF file, open the file again, and acquire the program device again. However, if this results in another 8281 return code, your program should close the file and end. • Close the file and end. **Messages:** CPF4168 (Escape) CPF4182 (Escape) CPF4304 (Escape) CPF4369 (Escape)

CPF4370 (Escape)
CPF4375 (Escape)
CPF5257 (Escape)
CPF5274 (Escape)
CPF5317 (Escape)
CPF5318 (Escape)
CPF5355 (Escape)

8282 **Description:** The open or acquire operation attempted by your program was not successful because the device supporting communications between your program and the remote location is not usable. For example, this may have occurred because communications were stopped for the device by a Hold Communications Device (HLDCMNDEV) command, or because a cancel reply was issued in response to an error recovery message for the device. Your program should not issue any operations to the device. The session was not started.

Action: Communications with the remote program cannot resume until the device has been reset to a varied on state. If the device has been held, use the Release Communications Device (RLSCMNDEV) command to reset the device. If the device is in an error state, vary the device off, then on again. Your program can attempt to acquire the program device again, continue local processing, or end.

Messages:

CPF4298 (Escape)
CPF4354 (Escape)
CPF5269 (Escape)
CPF5548 (Escape)

8285 **Description:** On an open or acquire operation, the attempt by your program to call a remote location automatically using a switched connection was not successful. The number specified on the controller description was dialed, but the connection was not established. Possible causes are that the line was busy, that there was no answer, or that the number dialed was disconnected. The session was not started.

Action: Verify that the number you are dialing is correct and that the remote system is ready for the call. Also verify that the line description you are using is varied on and is included in the switched line list on the controller description. Your program can issue the open or acquire operation again, continue local processing, or end.

Messages:

CPF5260 (Escape)

82A8 **Description:** The acquire operation attempted by your program was not successful because the maximum number of program devices allowed for the ICF file has been reached. The session was not started.

Action: Your program can recover by releasing a different program device and issuing the acquire operation again. If more program devices are needed, close the file and increase the MAXPGMDEV value for the ICF file.

Messages:

CPF4745 (Diagnostic)
CPF5041 (Status)

82A9 **Description:** The acquire operation issued by your program to a *REQUESTER device was not successful due to one of the following causes:

- Your program has already acquired the *REQUESTER device.
- The job was started by a program start request with the *REQUESTER device detached.
- The *REQUESTER device was released because an end-of-session was requested.
- The job does not have a *REQUESTER device; that is, the job was not started by a program start request.
- A permanent error occurred on the session.

Action:

- If the *REQUESTER device is already acquired and your program expects to communicate with the *REQUESTER device, use the program device that acquired the *REQUESTER.
- If the *REQUESTER device is not available and your program expects to communicate with the *REQUESTER device, the remote program must send a program start request without a detach function.
- If your program released its *REQUESTER device, correct the error that caused your program to release its *REQUESTER device before trying to acquire it.
- If this job does not have a *REQUESTER device, correct the error that caused your program to attempt to acquire a *REQUESTER device.
- If a permanent error caused the acquire operation to fail, verify that your program correctly handles the permanent error return codes (80xx, 81xx) it received on previously issued input and output operations. Because your program was started by a program start request, your program cannot attempt error recovery after receiving a permanent error return code. It is the responsibility of the remote program to initiate error recovery.

Messages:

CPF4366 (Escape)
CPF5380 (Escape)
CPF5381 (Escape)

82AA **Description:** The open or acquire operation attempted by your program was not successful because the remote location name specified on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command does not match any remote location configured on the system. The session was not started.

Action: Your program can continue local processing, or close the file and end. Verify that the name of the remote location is specified correctly in the RMTLOCNAME parameter on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command.

Messages:

CPF4103 (Escape)
CPF4363 (Escape)
CPF4364 (Escape)
CPF4747 (Escape)
CPF5378 (Escape)
CPF5379 (Escape)

82AB **Description:** The open or acquire operation attempted by your program was not successful because the device description for the remote location was not varied on. The session was not started.

Action: Your program can wait until the communications configuration is varied on and then issue the acquire operation again, it can try the acquire operation again using a different device description, continue local processing, or end.

Messages:

82B3 **Description:** The open or acquire operation attempted by your program was not successful because your program is trying to use a device description that is already in use by another job. The session was not started.

Action: Wait for the device description to become available, then issue the acquire operation again. You can use the Work with Configuration Status (WRKCFGSTS) command to determine which job is using the device description. Consider increasing the WAITFILE parameter of the CHGICFF or OVRICFF command to allow more time for the device to become available. Otherwise, your program can continue local processing or end.

Messages:

CPF4106 (Escape)
CPF5507 (Escape)

82EA **Description:** The open or acquire operation attempted by your program was not successful. A format selection of *RECID was specified on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command, but cannot be used with the ICF file because the RECID DDS keyword is not used on any of the record formats in the file. The session was not started.

Action: Close the ICF file. Change the record format selection (FMTSLT) parameter to select formats by some means other than *RECID, or use a file that has a RECID DDS keyword specified for at least one record format. Open the file again.

Messages:

CPF4348 (Escape)
CPF5521 (Escape)

82EE **Description:** Your program attempted an open or acquire operation to a device that is not supported. Your program tried to acquire a device that is not a valid ICF communications type, or it is trying to acquire the requesting program device in a program that was not started by a program start request. The session was not started.

Action: Your program can continue local processing or end. Verify that the name of the remote location is specified correctly in the RMTLOCNAME parameter on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command. If your program was attempting to acquire a non-ICF device, use the appropriate interface for that communications type. If your program was attempting to acquire a requesting program device, verify that your program is running in the correct environment.

Messages:

CPF4105 (Escape)
CPF4223 (Escape)
CPF4251 (Escape)
CPF4760 (Escape)
CPF5038 (Escape)
CPF5550 (Escape)

82EF **Description:** Your program attempted an acquire operation, or an open operation that implicitly acquires a session, to a device that the user is not authorized to, or that is in service mode. The session was not started.

Action: If the operation was an acquire, correct the problem and issue the acquire again. If the operation was an open, close the file, correct the problem, then issue the open operation again. To correct an authority error, obtain authority for the device from your security officer or device owner. If the device is in service mode, wait until machine service function (MSF) is no longer using the device before issuing the operation again.

Messages:

CPF4104 (Escape)
CPF4186 (Escape)
CPF5278 (Escape)
CPF5279 (Escape)

82F0 **Description:** The open or acquire operation attempted by your program to a requesting program device was not successful because there is an error in the ICF file.

Action: End your program, correct the error, then have the remote program send the program start request again.

Messages:

CPF4324 (Escape)
CPF5540 (Escape)

82F5 **Description:** The open or acquire operation was not successful because your program tried to use a format selection option of *RMTFMT in the FMTSLT parameter on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command. The session was not started.

Action: Change the value in the FMTSLT parameter on the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command, then issue the open or acquire operation again.

Messages:

CPF4347 (Escape)

Major Code 83

Major Code 83 — Session error occurred (the error is recoverable).

Description: A session error occurred, but the session may still be active. Recovery within your program might be possible.

Action: You can perform the following general actions for all 83xx return codes. Specific actions are given in each minor code description.

- Correct the problem and continue processing with the session. If the error occurred because of a resource failure on the remote system or because the remote system was not active at the time, a second attempt may be successful. If the operation is still not successful, your program should end the session.
- Issue an end-of-session function and continue processing without the session.
- End.

Several of the minor codes indicate that an error condition must be corrected by changing a value in the communications configuration or in the file.

- To change a parameter value in the communications configuration, vary the configuration off, make the change to the configuration description, then vary the configuration on.
- To change a parameter value in the file, use the ADDICFDEVE, CHGICFDEVE, or OVRICFDEVE command.

  Note: When a parameter can be specified both in the ADDICFDEVE or OVRICFDEVE command and in the configuration, the value in the ADDICFDEVE or OVRICFDEVE command overrides the value specified in the configuration (for your program only). Therefore, in some cases, you may choose to make a change with the ADDICFDEVE or OVRICFDEVE command rather than in the configuration.

If no changes are needed in your file or in the configuration, and depending on what the return code description says, you should notify the remote location that a change is required at that location to correct the error received.

Code	Description/Action
830B	**Description:** Your program attempted an operation that was not valid because the session was not yet acquired or has ended. The session may have ended because of a release operation, an end-of-session function, or a permanent error. Your program may have incorrectly handled a previous error. **Action:** Verify that your program does not attempt any operations without an active session. Also verify that your program correctly handles the permanent error or session-not-acquired return codes (80xx, 81xx, 82xx) it received on previously issued input and output operations. To recover from an incorrectly handled error condition, your program may or may not be able to issue another acquire operation, depending on the return code.

Messages:

CPD4079 (Diagnostic)
CPF4739 (Status)
CPF5067 (Escape)
CPF5068 (Escape)
CPF5070 (Escape)

831C **Description:** Your program's previous output operation received a return code of 0412, indicating that your program must receive information sent by the remote program or the PAD; however, your program did not handle the return code correctly. The current output operation was not successful because your program should have issued an input operation to receive the information already sent by the remote program.

Action: Issue an input operation to receive the previous information.

Messages:

CPF4934 (Notify)
CPF5076 (Notify)

831E **Description:** The operation attempted by your program was not valid, or a combination of operations that was not valid was specified. The session is still active. The error may have been caused by one of the following:

- Your program issued an operation that is not recognizable or not supported by asynchronous communications.
- Your program requested a combination of operations or keywords that was not valid, such as a combined write-then-read operation with the invite function specified.
- Your program issued an input operation, or an output operation with the invite function, for a file that was opened for output only.
- Your program issued an output operation for a file that was opened for input only.
- Your program issued a close operation with a temporary close option.

Action: Your program can try a different operation, issue a release operation or end-of-session function, or end. Correct the error in your program before trying to communicate with the remote program.

If the file was opened for input only, do not issue any output operations; or, if the file was opened for output only, do not issue any input operations, and do not use the invite or allow-write function on an output operation. If such an operation is needed, then release the session, close the ICF file, and open the file again for input and output.

Messages:

CPF4564 (Escape)
CPF4764 (Notify)
CPF4766 (Notify)
CPF4790 (Notify)
CPF5132 (Escape)
CPF5149 (Escape)

831F **Description:** Your program specified data or a length for the operation that was not valid; however, the session is still active. One of the following caused the error indication:

- On an output operation, your program tried to send a data record that was longer than the MAXRCDLEN value specified for the ICF file.
- The program used a read or write operation that specified a data length greater than the record format in the ICF file.
- If this was a timer function, the format of the timer interval was not HHMMSS.
- If a system-defined format was used to specify the operation, or if the variable-length-data-record (VARLEN) function was used, then the length of the user buffer was not valid.

Action: If you want your program to recover, try the operation again with a smaller data length. If you do not need your program to recover immediately, do one of the following:

- Change the record format length in the ICF file, or change the record length in your program and compile your program again.
- For an input operation, specify a data length equal to or less than the record format length, or do not specify a length at all.
- If the timer function was used, verify that the format of the timer interval is HHMMSS.
- For an output operation that used the variable-length-data-record (VARLEN) function, verify that the length specified is less than the record length specified for the ICF file when it was opened.

Messages:

CPF4762 (Notify)
CPF4765 (Notify)
CPF4767 (Notify)

8329 **Description:** An evoke function that was not valid was detected in this session. Your program was started by a program start request and, therefore, cannot issue any evoke functions in this session.

Action: To recover, your program can try a different operation or function. To issue an evoke function in a different session, first issue an acquire operation (using a different program device name), then try the evoke function. Otherwise, your program can issue an end-of-session function, continue local processing, or end. If a coding error caused your program to attempt an evoke that was not valid, correct your program.

Messages:

CPF5099 (Notify)

832C **Description:** A release operation following an invite function was detected. Because your program issued the invite function, it cannot issue a release operation to end the invited session.

Action: Issue an input operation to satisfy the invite function, or issue a cancel-invite function to cancel the invite function; then try the release operation again. Otherwise, issue an end-of-session function to end the session. If a coding error caused your program

to attempt a release operation that was not valid, correct your program.

Messages:

CPF4769 (Notify)

832D **Description:** Following an invite function, your program issued an additional invite function. This operation failed because the original invite function must first be satisfied by an input operation.

Action: Issue an input operation to receive the data that was invited. Otherwise, issue an end-of-session function to end the session. If a coding error caused your program to attempt a request-to-write indication or an additional invite function, correct your program.

Messages:

CPF4924 (Notify)

83E0 **Description:** Your program attempted an operation using a record format that was not defined for the ICF file.

Action: Verify that the name of the record format in your program is correct, then check to see whether the record format is defined in the file definition.

Messages:

CPF5054 (Notify)

83E8 **Description:** Your program attempted to issue a cancel-invite function to a session that was not invited. One of the following may have occurred:

- The invite function was implicitly canceled earlier in your program by a valid output operation.
- The invite function was satisfied earlier in your program by a valid input operation.
- Your program had already canceled the invite function, then tried to cancel it again.
- Your program never invited the session.

The session is still active.

Action: Your program can issue an input or output operation, issue an end-of-session function, continue local processing, or end. However, you should correct the error that caused your program to attempt the cancel-invite to a session that was not invited.

Messages:

CPF4763 (Notify)

83F8 **Description:** Your program attempted to issue an operation to a program device that is marked in error due to a previous I/O or acquire operation. Your program may have handled the error incorrectly.

Action: Release the program device, correct the previous error, then acquire the program device again.

Messages:

CPF5293 (Escape)

Failed Program Start Requests

Message CPF1269 is sent to the system operator message queue when the local system rejects an incoming program start request. You can use the message information to determine why the program start request was rejected.

The CPF1269 message contains two reason codes. One of the reason codes can be zero, which can be ignored. If only one nonzero reason code is received, that reason code represents the reason the program start request was rejected. If the System/36 environment is installed on your AS/400 system, there can be two nonzero reason codes. These two reason codes occur when OS/400 cannot determine whether the program start request was to start a job in the System/36 environment or in OS/400. One reason code explains why the program start request was rejected in the System/36 environment and the other explains why the program start request was rejected in OS/400. Whenever you receive two reason codes, you should determine which environment the job was to run in and correct the problem for that environment.

Figure B-2 describes reason codes for failed program start requests.

Figure B-2 (Page 1 of 3). Reason Codes for Rejected Program Start Requests

Reason Code	Reason Description
401	Program start request received to a device that is not allocated to an active subsystem.
402	Requested device is currently being held by a Hold Communications Device (HLDCMNDEV) command.
403	User profile is not accessible.
404	Job description is not accessible.
405	Output queue is not accessible.
406	Maximum number of jobs defined by subsystem description are already active.
407	Maximum number of jobs defined by communications entry are already active.
408	Maximum number of jobs defined by routing entry are already active.
409	Library on library list is exclusively in use by another job.
410	Group profile cannot be accessed.
411	Insufficient storage in machine pool to start job.
412	System values not accessible.
501	Job description was not found.
502	Output queue was not found.
503	Class was not found.
504	Library on initial library list was not found.
505	Job description or job description library is damaged.
506	Library on library list is destroyed.
507	Duplicate libraries were found on library list.
508	Storage-pool defined size is zero.

Figure B-2 (Page 2 of 3). Reason Codes for Rejected Program Start Requests

Reason Code	Reason Description
602	Transaction program-name value is reserved but not supported.
604	Matching routing entry was not found.
605	Program was not found.
704	Password is not valid.
705	User is not authorized to device.
706	User is not authorized to subsystem description.
707	User is not authorized to job description.
708	User is not authorized to output queue.
709	User is not authorized to program.
710	User is not authorized to class.
711	User is not authorized to library on library list.
712	User is not authorized to group profile.
713	User ID is not valid.
714	Default user profile is not valid.
715	Neither password nor user ID was provided, and no default user profile was specified in the communications entry.
718	No user ID.
722	A user ID was received but no password was sent.
723	No password was associated with the user ID.
725	User ID does not follow naming convention.
726	User profile has been disabled.
801	Program initialization parameters are present but not allowed.
802	Program initialization parameter exceeds 2000 bytes.
803	Subsystem is ending.
804	Prestart job is inactive or is ending.
805	WAIT(NO) was specified on the prestart job entry and no prestart job was available.
806	The maximum number of prestart jobs that can be active on a prestart job entry was exceeded.
807	Prestart job ended when a program start request was being received.
901	Program initialization parameters are not valid.
902	Number of parameters for program not valid.
903	Program initialization parameters required but not present.
1001	System logic error. Function check or unexpected return code encountered.
1002	System logic error. Function check or unexpected return code encountered while receiving program initialization parameters.

Figure B-2 (Page 3 of 3). Reason Codes for Rejected Program Start Requests

Reason Code	Reason Description
1501	Character in procedure name not valid.
1502	Procedure not found.
1503	System/36 environment library not found.
1504	Library QSSP not found.
1505	File QS36PRC not found in library QSSP.
1506	Procedure or library name is greater than 8 characters.
1507	Current library not found.
1508	Not authorized to current library.
1509	Not authorized to QS36PRC in current library.
1510	Not authorized to procedure in current library.
1511	Not authorized to System/36 environment library.
1512	Not authorized to file QS36PRC in System/36 environment library.
1513	Not authorized to procedure in System/36 environment library.
1514	Not authorized in library QSSP.
1515	Not authorized to file QS36PRC in QSSP.
1516	Not authorized to procedure in QS36PRC in QSSP.
1517	Unexpected return code from System/36 environment support.
1518	Problem phase program not found in QSSP.
1519	Not authorized to problem phase program in QSSP.
1520	Maximum number of target programs started (100 per System/36 environment).

Appendix C. Code Conversion Tables

The following tables show how asynchronous communications support translates your program data. Tables included show:

- The EBCDIC character set
- The ASCII (IA-5) character set
- EBCDIC-to-ASCII translation table
- ASCII-to-EBCDIC translation table

You can also create your own translation table using the Create Table (CRTTBL) command. See the online information for a description of this command. A system-supplied program, QDCXLATE, can be used to translate individual fields, using any specified translation table. See the *CL Programmer's Guide* for more information about using QDCXLATE.

If you do choose to create your own translation table, you must turn translation off by issuing a function-management-header function. Your application program is then responsible for translating all user data. See "Write Operation" on page 6-4 for more information about the function-management-header function.

EBCDIC Character Set

		Main Storage Bit Positions 0,1,2,3															
Main Storage Bit Positions 4,5,6,7		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	NUL	DLE			SP	&	-						{	}	\	0
0001	1	SOH	DC1					/		a	j	~		A	J		1
0010	2	STX	DC2		SYN					b	k	s		B	K	S	2
0011	3	ETX	DC3							c	l	t		C	L	T	3
0100	4									d	m	u		D	M	U	4
0101	5	HT	NL	LF						e	n	v		E	N	V	5
0110	6		BS	ETB						f	o	w		F	O	W	6
0111	7	DEL		ESC	EOT					g	p	x		G	P	X	7
1000	8		CAN							h	q	y		H	Q	Y	8
1001	9		EM						'	i	r	z		I	R	Z	9
1010	A					[	]	¦	:								
1011	B	VT				.	$	,	#								
1100	C	FF	FS		DC4	<	*	%	@								
1101	D	CR	GS	ENQ	NAK	(	)	_	'								
1110	E	SO	RS	ACK		+	;	>	=								
1111	F	SI	US	BEL	SUB	!	^	?	"								

RSLS452-5

Figure C-1. EBCDIC Character Set

International Alphabet (IA-5) ASCII Character Set

The following table shows the ASCII characters as defined by the international alphabet (IA-5) used by the integrated PAD support to determine data forwarding characters.

		Main Storage Bit Positions 0,1,2,3															
Main Storage Bit Positions 4,5,6,7		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	NUL	DLE	SP	0	[2]	P	[2]	p								
0001	1	SOH	DC1	!	1	A	Q	a	q								
0010	2	STX	DC2	"	2	B	R	b	r								
0011	3	ETX	DC3	# [1] / £	3	C	S	c	s								
0100	4	EOT	DC4	¤ [1] / $	4	D	T	d	t								
0101	5	ENQ	NAK	%	5	E	U	e	u								
0110	6	ACK	SYN	&	6	F	V	f	v								
0111	7	BEL	ETB	'	7	G	W	g	w								
1000	8	BS	CAN	(	8	H	X	h	x								
1001	9	HT	EM	)	9	I	Y	i	y								
1010	A	LF	SUB	*	:	J	Z	j	z								
1011	B	VT	ESC	+	;	K	[2]	k	[2]								
1100	C	FF	IS4	,	<	L	[2]	l	[2]								
1101	D	CR	IS3	-	=	M	[2]	m	[2]								
1110	E	SO	IS2	.	>	N	[2]	n	[2]								
1111	F	SI	IS1	/	?	O	_	o	DEL								

[1] Multiple characters defined.
[2] No specific characters defined.

RSLS462-0

Figure C-2. International Alphabet (IA-5) ASCII Character Set

EBCDIC-to-ASCII Translation Table

The following table shows the hexadecimal values used when translating characters from EBCDIC to ASCII.

For example, EBCDIC uses hex 82 to represent the letter b; the ASCII equivalent for the letter b, as shown in the table, is hex 62. Likewise, EBCDIC uses hex 2E to represent the ACK character; the ASCII equivalent is hex 06.

Blank squares in the table (for example, hex 8A) are translated to hex FF.

		Main Storage Bit Positions 0,1,2,3															
Main Storage Bit Positions 4,5,6,7		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	00	10			20	26	2D						7B	7D	5C	30
0001	1	01	11					2F		61	6A	7E		41	4A		31
0010	2	02	12		16					62	6B	73		42	4B	53	32
0011	3	03	13							63	6C	74		43	4C	54	33
0100	4									64	6D	75		44	4D	55	34
0101	5	09	0A	0A						65	6E	76		45	4E	56	35
0110	6		08	17						66	6F	77		46	4F	57	36
0111	7	7F		1B	04					67	70	78		47	50	58	37
1000	8		18							68	71	79		48	51	59	38
1001	9		19						60	69	72	7A		49	52	5A	39
1010	A					5B	5D	7C	3A								
1011	B	0B				2E	24	2C	23								
1100	C	0C	1C		14	3C	2A	25	40								
1101	D	0D	1D	05	15	28	29	5F	27								
1110	E	0E	1E	06		2B	3B	3E	3D								
1111	F	0F	1F	07	1A	21	5E	3F	22								

RSLS463-1

Figure C-3. EBCDIC-to-ASCII Translation Table

ASCII-to-EBCDIC Translation Table

The following table shows the hexadecimal values used when translating characters from ASCII (using the IA-5 alphabet) to EBCDIC.

For example, ASCII uses hex 4E to represent the letter N; the EBCDIC equivalent for the letter N, as shown in the table, is hex D5.

		Main Storage Bit Positions 0,1,2,3															
Main Storage Bit Positions 4,5,6,7		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	00	10	40	F0	7C	D7	79	97	00	10	40	F0	7C	D7	79	97
0001	1	01	11	4F	F1	C1	D8	81	98	01	11	4F	F1	C1	D8	81	98
0010	2	02	12	7F	F2	C2	D9	82	99	02	12	7F	F2	C2	D9	82	99
0011	3	03	13	7B	F3	C3	E2	83	A2	03	13	7B	F3	C3	E2	83	A2
0100	4	37	3C	5B	F4	C4	E3	84	A3	37	3C	5B	F4	C4	E3	84	A3
0101	5	2D	3D	6C	F5	C5	E4	85	A4	2D	3D	6C	F5	C5	E4	85	A4
0110	6	2E	32	50	F6	C6	E5	86	A5	2E	32	50	F6	C6	E5	86	A5
0111	7	2F	26	7D	F7	C7	E6	87	A6	2F	26	7D	F7	C7	E6	87	A6
1000	8	16	18	4D	F8	C8	E7	88	A7	16	18	4D	F8	C8	E7	88	A7
1001	9	05	19	5D	F9	C9	E8	89	A8	05	19	5D	F9	C9	E8	89	A8
1010	A	15	3F	5C	7A	D1	E9	91	A9	15	3F	5C	7A	D1	E9	91	A9
1011	B	0B	27	4E	5E	D2	4A	92	C0	0B	27	4E	5E	D2	4A	92	C0
1100	C	0C	1C	6B	4C	D3	E0	93	6A	0C	1C	6B	4C	D3	E0	93	6A
1101	D	0D	1D	60	7E	D4	5A	94	D0	0D	1D	60	7E	D4	5A	94	D0
1110	E	0E	1E	4B	6E	D5	5F	95	A1	0E	1E	4B	6E	D5	5F	95	A1
1111	F	0F	1F	61	6F	D6	6D	96	07	0F	1F	61	6F	D6	6D	96	07

RSLS464-1

Figure C-4. ASCII-to-EBCDIC Translation Table

Appendix D. Break and Interrupt Handling

This appendix describes what actions and responses asynchronous communications support takes when your program issues a fail function or when a break signal, interrupt packet, or `Indication of Break` message is received from the remote device. The actions taken by asynchronous communications support depend on whether:

- The line being used is an asynchronous (start-stop) or X.25 line
- Your application program is a packet-mode host application
- You configured PAD emulation
- The remote device is attached to a PAD

Note: `Indication of Break` messages are qualified X.25 data packets.

Fail Function

The following lists the actions taken by asynchronous support when your application program issues a fail function.

- If the application program uses an asynchronous (start-stop) line:

  The I/O adapter creates a break signal of at least 300 milliseconds space-time duration.

- If the application program uses an X.25 line with PAD emulation configured, and the value of the PAD parameter 7 is:

  0 No action is taken.

  1 An X.25 interrupt packet is sent with user data of hex 01.

  2 Clear the virtual circuit. The PAD parameters are reset to the default values.

  8 Escape to command mode.

  21 An X.25 interrupt packet is sent with user data of hex 00. PAD parameter 8 is set to a value of 1. An `Indication of Break` message is sent that also informs the packet-mode host that PAD parameter 8 has been set to a value of 1.

- If the application program uses an X.25 line as a packet-mode host and the remote device is a:

  Packet-mode host
  An X.25 interrupt packet is sent with user data of hex 01.

  PAD
  An `Indication of Break` message is sent.

Receive Break or Interrupt Actions

The following list shows the actions and responses that occur when asynchronous communications support receives a break signal, Indication of Break message, or interrupt packet from the remote device:

- If the application program uses an asynchronous (start-stop) line, and a break signal is detected by the I/O adapter, your application program will receive a 0302 return code.

  Note: A framing (stop bit) error on a null character is treated as a break signal and a 0302 return code is sent to the receiving program.

- If the application program uses an X.25 line with PAD emulation configured, and asynchronous communications support receives the following:

  Indication of Break message
  Your application program receives a 0302 return code.

  Indication of Break message with data of hex 0801
  Your application program receives a 0302 return code. PAD parameter 8 is set to a value of 0.

  X.25 interrupt packet
  No action is taken.

- If the application program uses an X.25 line as a packet-mode host and asynchronous communications support receives the following:

  Indication of Break message
  Your application program receives a 0302 return code. If the remote device is a PAD, a Set message is sent to set parameter 8 to a value of 0.

  X.25 interrupt packet with data of hex 00
  No action is taken.

  X.25 interrupt packet with data of hex 01
  Your application program receives a 0302 return code.

Appendix E. Asynchronous Communications Configuration Examples

This appendix contains asynchronous communications configuration examples for:

- A nonswitched line (for use in connecting directly to an asynchronous device, such as a printer or plotter).
- A switched line where the AS/400 system uses a command-capable modem, such as the IBM 5842, to connect to a remote device.
- Asynchronous communications on an X.25 packet-switching data network (PSDN).

Each example uses the command prompt displays shown by typing the name of the command on the command line, then pressing F4 (Prompt).

Nonswitched Asynchronous Communications Example

The following nonswitched configuration example is used to communicate with a directly attached asynchronous communications device or a device connected through a modem eliminator. The line has the following characteristics:

- Line speed of 9600 bits per second
- Even parity
- 1 stop bit
- 7 data bits (ASCII)
- Duplex (*FULL)
- Device buffer size of 128 bytes
- Device provides pacing by using flow control with the default XON/XOFF characters
- Device ends each record sent to the AS/400 system with an ASCII carriage return followed by a line feed

Because the device is not an AS/400 device, the file transfer acknowledgment timer and file transfer retry do not apply to the controller description, nor do the remote verification parameters of verify, local location name, and local identifier.

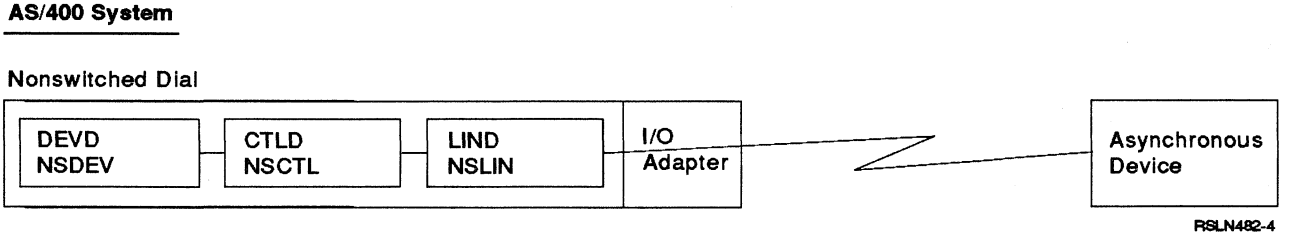


Figure E-1. Nonswitched Asynchronous Communications Example

Nonswitched Asynchronous Line Description:

```
                    CREATE LINE DESC (ASYNC) (CRTLINASC)

Type choices, press Enter.

Line description . . . . . > NSLIN        Name
Resource name  . . . . . > LIN031       Name
Online at IPL  . . . . . *YES           *YES, *NO
Physical interface . . . . . *RS232V24      *RS232V24
Connection type  . . . . . *NONSWTPP        *NONSWTPP, *SWTPP
Switched network backup  . . . . *NO            *NO, *YES
Data bits per character  . . . . > 7            8, 7
Type of parity . . . . . > *EVEN          *NONE, *ODD, *EVEN
Stop bits  . . . . . 1             1, 2
Duplex . . . . . *FULL          *FULL, *HALF
Echo support . . . . . *NONE          *NONE, *ALL, *CNTL
Line speed . . . . . > 9600          50, 75, 110, 150, 300, 600...
Modem type supported . . . . . *NORMAL        *NORMAL, *V54, *IBMWRAP
Maximum buffer size  . . . . . > 128           128-4096
Flow control . . . . . > *YES           *NO, *YES
XON character  . . . . . 11            01-FF
                                                                  MORE...
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display       F24=More keys
```

```
                    CREATE LINE DESC (ASYNC) (CRTLINASC)

Type choices, press Enter.

XOFF character . . . . . 13            01-FF
End-of-Record table:
  End-of-Record character  . . > 0A            00-FF
  Trailing characters  . . . . . 0             0-4

  End-of-Record character  . . > 0D            00-FF
  Trailing characters  . . . . . 0             0-4
            + for more values
Text 'description' . . . . . > 'Asynchronous line to fast device'

                                                                  BOTTOM
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display       F24=More keys
```

Figure E-2. Prompt Displays for Nonswitched Asynchronous Line Description

Nonswitched Asynchronous Controller Description:

```
                    CREATE CTL DESC (ASYNC) (CRTCTLASC)

Type choices, press Enter.

Controller description . . . . . > NSCTL         Name
Link type  . . . . . > *ASYNC        *ASYNC, *X25
Online at IPL  . . . . . *YES           *YES, *NO
Switched connection  . . . . . *NO            *NO, *YES
Switched network backup  . . . . *NO            *NO, *YES
Attached nonswitched line  . . . > NSLIN         Name
Text 'description' . . . . . > 'Asynchronous controller to fast device'

                                                                  BOTTOM
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display       F24=More keys
```

Figure E-3. Prompt Display for Nonswitched Asynchronous Controller Description

Nonswitched Asynchronous Device Description:

```
                    CREATE DEVICE DESC (ASYNC) (CRTDEVASC)

Type choices, press Enter.

Device description . . . . . > NSDEV         Name
Remote location name . . . . . > FASTDEV       Name, *NONE
Online at IPL  . . . . . *YES           *YES, *NO
Attached controller  . . . . . > NSCTL         Name
Text 'description' . . . . . > 'Fast asynchronous device'

                                                                  BOTTOM
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display       F24=More keys
```

Figure E-4. Prompt Display for Nonswitched Asynchronous Device Description

Switched Asynchronous Communications Configuration Example

The following switched configuration is used to communicate with another AS/400 system that is capable of receiving a call. The application program uses the file transfer subroutines, and the line has the following characteristics:

- Line speed of 2400 bps
- No parity
- 1 stop bit
- 8 data bits (EBCDIC)
- Duplex (*FULL)
- Device buffer size of 896 bytes (required for file transfer)
- No flow control
- No echo

Asynchronous communications allows you to connect to the modem and to send dial commands without the data set ready (DSR) signal being active. This is called a **deferred connection** and is configured using the AUTODIAL(*YES) and DIALCMD(*OTHER) parameters on the line description. If you have the modem switches set to hold DSR active, you must provide a nonswitched line and controller configuration to communicate with the modem.

A command-capable modem requires that you send a dial command to call the remote modem. This command must be provided by your program as data on the first write operation.

Because file transfer runs on this line, you need to provide a retry value and an acknowledgement timer value for file transfer. If your lines are noisy or if the network you are using is slow, you could choose to increase both values. The values shown cause file transfer to wait for a response from the remote system for up to 30 seconds before considering the transmission unsuccessful. Each unsuccessful transmission is tried again a maximum of ten times. The value provided for parity, bits per character, flow control, and echo are also required for file transfer.

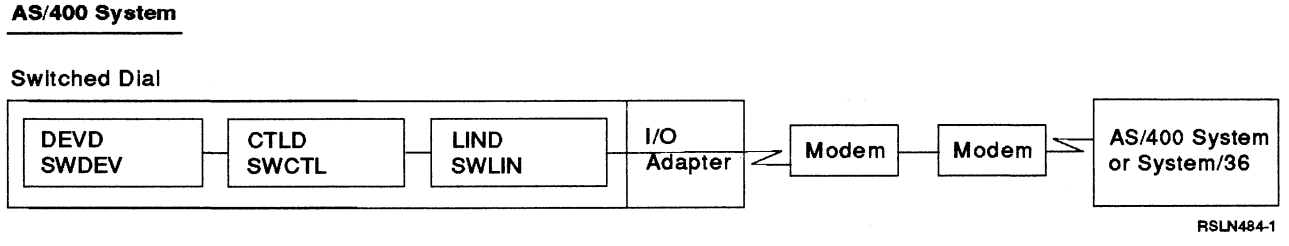


Figure E-5. Switched Asynchronous Communications Example

Switched Asynchronous Line Description:

```
                    CREATE LINE DESC (ASYNC) (CRTLINASC)

Type choices, press Enter.

Line description . . . . . > SWLIN          Name
Resource name  . . . . . > LIN031         Name
Online at IPL  . . . . . > *NO            *YES, *NO
Physical interface . . . . . *RS232V24    *RS232V24
Connection type  . . . . . > *SWTPP       *NONSWTPP, *SWTPP
Vary on wait . . . . . *NOWAIT            *NOWAIT, 15-180 (1 second)
Autocall unit  . . . . . *NO              *NO, *YES
Data bits per character  . . . . 8        8, 7
Type of parity . . . . . *NONE            *NONE, *ODD, *EVEN
Stop bits  . . . . . 1                    1, 2
Duplex . . . . . *FULL                    *FULL, *HALF
Echo support . . . . . *NONE              *NONE, *ALL, *CNTL
Line speed . . . . . > 2400               50, 75, 110, 150, 300, 600...
Modem type supported . . . . . *NORMAL    *NORMAL, *V54, *IBMWRAP
Switched connection type . . . . > *DIAL  *BOTH, *ANS, *DIAL
Autoanswer . . . . . > *NO                *YES, *NO
                                                                  MORE...
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

```
                    CREATE LINE DESC (ASYNC) (CRTLINASC)

Type choices, press Enter.

Autodial . . . . . > *YES          *NO, *YES
Dial command type  . . . . . > *OTHER      *NONE, *V25BIS, *OTHER
Calling number . . . . . *NONE
Inactivity timer . . . . . > *NOMAX        *NOMAX, 150-4200 (0.1 sec)
Maximum buffer size  . . . . . 896         128-4096
Flow control . . . . . *NO                 *NO, *YES
End-of-Record table:
  End-of-Record character  . . . 00        00-FF
  Trailing characters  . . . . . 0         0-4
          + for more values
Data Set Ready drop timer  . . . 6         3-60 (seconds)
Autoanswer type  . . . . . *DTR            *DTR, *CDSTL
Remote answer timer  . . . . . 60          30, 35, 40, 45 (seconds)...
Text 'description' . . . . . > 'Switched asynchronous line for file transfe
r'

                                                                  BOTTOM
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

Figure E-6. Prompt Displays for Switched Asynchronous Line Description

Switched Asynchronous Controller Description:

```
                    CREATE CTL DESC (ASYNC) (CRTCTLASC)

Type choices, press Enter.

Controller description . . . . . > SWCTL        Name
Link type  . . . . . > *ASYNC       *ASYNC, *X25
Online at IPL  . . . . . *YES       *YES, *NO
Switched connection  . . . . . > *YES       *NO, *YES
Switched line list . . . . . > SWLIN        Name
          + for more values
Initial connection . . . . . *DIAL          *DIAL, *ANS
Connection number  . . . . . > '555-1234'
Text 'description' . . . . . > 'Switched controller for file transfer'


                                                                  BOTTOM
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

Figure E-7. Prompt Display for Switched Asynchronous Controller Description

Switched Asynchronous Device Description:

```
                    CREATE DEVICE DESC (ASYNC) (CRTDEVASC)

Type choices, press Enter.

Device description . . . . . > SWDEV        Name
Remote location  . . . . . > ASYNDIAL     Name, *NONE
Online at IPL  . . . . . *YES             *YES, *NO
Attached controller  . . . . . > SWCTL    Name
Text 'description' . . . . . > 'Asynchronous device for file transfer'


                                                                  BOTTOM
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

Figure E-8. Prompt Display for Switched Asynchronous Device Description

Asynchronous/X.25 Network Examples

Asynchronous communications can be run on a switched or nonswitched line to an X.25 packet-switching data network (PSDN). This is done by creating an X.25 line description and asynchronous controller and device descriptions. The following examples show how to configure these descriptions for running asynchronous communications over an X.25 PSDN using :

- A permanent virtual circuit (PVC).
- An incoming call on a switched virtual circuit (SVC), *SVCIN.
- An incoming call on an SVC (*SVCIN) for generic asynchronous communications controllers and devices. Generic controllers are created to accept incoming calls from any network address whose local location name and local ID are defined in your asynchronous remote location configuration list.
- An incoming call on an SVC (*SVCIN) for generic asynchronous controllers. Generic controllers are created to accept incoming calls from any network address.
- An outgoing call on an SVC (*SVCOUT).
- An outgoing call on an SVC (*SVCOUT) for packet assembler/disassembler (PAD) emulation.

Permanent Virtual Circuit (*PVC)

This X.25 line description example supports one PVC and up to four SVC controllers. Asynchronous communications allows only one active session (device) for each controller. The controller and device descriptions attached to this line could be a combination of any of the communications types supported by X.25.

The following controller and device description examples show only asynchronous communications possibilities.

An exchange identifier (EXCHID) is required in the X.25 line description although it is never used in establishing or confirming an asynchronous communications connection. The EXCHID is used only by the SNA controller descriptions attached to the X.25 line.

X.25 Line Description:

```
                    CREATE LINE DESC (X.25) (CRTLINX25)

Type choices, press Enter.

Line description . . . . . > X25LINE      Name
Resource name  . . . . . > LIN03      Name
Logical channel entries:
  Logical channel identifier . . > 001        001-FFF
  Logical channel type . . . . . > *PVC       *PVC, *SVCIN, *SVCBOTH...
  PVC controller . . . . . _____      Name
            + for more values _
Local network address  . . . . . > 40100030
Connection initiation  . . . . . > *LOCAL     *LOCAL, *REMOTE, *WAIT
Online at IPL  . . . . . *YES       *YES, *NO
Physical interface . . . . . *X21BISV24   *X21BISV24, *X21BISV35...
Connection type  . . . . . *NONSWTPP  *NONSWTPP, *SWTPP
Vary on wait . . . . . *NOWAIT    *NOWAIT, 15-180 (1 second)
Line speed . . . . . 9600       600, 1200, 2400, 4800...
Exchange identifier  . . . . . *SYSGEN    05600000-056FFFFF, *SYSGEN
Extended network addressing  . . *NO        *YES, *NO
                                                              MORE...
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

Figure E-9 (Part 1 of 3). Prompt Displays for X.25 Line Description

```
                    SPECIFY MORE VALUES FOR PARAMETER LGLCHLE

Type choices, press Enter.

Logical channel entries:
  Logical channel identifier . . > 001          001-FFF
  Logical channel type . . . . . > *PVC         *PVC, *SVCIN, *SVCBOTH...
  PVC controller . . . . . .       _________    Name

  Logical channel identifier . . > 002          001-FFF
  Logical channel type . . . . . > *SVCBOTH     *PVC, *SVCIN, *SVCBOTH...
  PVC controller . . . . . .       _________    Name

  Logical channel identifier . . > 003          001-FFF
  Logical channel type . . . . . > *SVCBOTH     *PVC, *SVCIN, *SVCBOTH...
  PVC controller . . . . . .       _________    Name

  Logical channel identifier . . > 004          001-FFF
  Logical channel type . . . . . > *SVCBOTH     *PVC, *SVCIN, *SVCBOTH...
  PVC controller . . . . . .       _________    Name
                                                                    MORE...
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

```
                    SPECIFY MORE VALUES FOR PARAMETER LGLCHLE

Type choices, press Enter.

  Logical channel identifier . . > 005          001-FFF
  Logical channel type . . . . . > *SVCBOTH     *PVC, *SVCIN, *SVCBOTH...
  PVC controller . . . . . .       _________    Name

  Logical channel identifier . .   _____        001-FFF
  Logical channel type . . . . .   _____        *PVC, *SVCIN, *SVCBOTH...
  PVC controller . . . . . .       _________    Name

  Logical channel identifier . .   _____        001-FFF
  Logical channel type . . . . .   _____        *PVC, *SVCIN, *SVCBOTH...
  PVC controller . . . . . .       _________    Name

  Logical channel identifier . .   _____        001-FFF
  Logical channel type . . . . .   _____        *PVC, *SVCIN, *SVCBOTH...
  PVC controller . . . . . .       _________    Name
                                                                    MORE...
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

```
                    CREATE LINE DESC (X.25) (CRTLINX25)

Type choices, press Enter.

  Logical channel identifier . . > 005          001-FFF
  Logical channel type . . . . . > *SVCBOTH     *PVC, *SVCIN, *SVCBOTH...
  PVC controller . . . . . .       _________    Name
            + for more values _
Local network address  . . . . . > 40100036
Connection initiation  . . . . . > *LOCAL       *LOCAL, *REMOTE, *WAIT
Online at IPL  . . . . . .         *YES         *YES, *NO
Physical interface . . . . . .     *X21BISV24   *X21BISV24, *X21BISV35...
Connection type  . . . . . .       *NONSWTPP    *NONSWTPP, *SWTPP
Vary on wait . . . . . .           *NOWAIT      *NOWAIT, 15-180 (1 second)
Line speed . . . . . .             9600         600, 1200, 2400, 4800...
Exchange identifier  . . . . . .   *SYSGEN      05600000-056FFFFF, *SYSGEN
Extended network addressing  . .   *NO          *YES, *NO


                                                                    MORE...
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

Figure E-9 (Part 2 of 3). Prompt Displays for X.25 Line Description

```
                    CREATE LINE DESC (X.25) (CRTLINX25)

Type choices, press Enter.

Default packet size:
  Transmit value . . . . . .  128          64, 128, 256, 512, 1024
  Receive value  . . . . . .  *TRANSMIT    *TRANSMIT, 64, 128, 256...
Maximum packet size:
  Transmit value . . . . . .  *DFTPKTSIZE  *DFTPKTSIZE, 64, 128, 256...
  Receive value  . . . . . .  *DFTPKTSIZE  *DFTPKTSIZE, *TRANSMIT, 64...
Modulus  . . . . . .          8            8, 128
Default window size:
  Transmit value . . . . . .  2            1-15
  Receive value  . . . . . .  *TRANSMIT    1-15, *TRANSMIT
Insert net address in packets  *YES        *YES, *NO
Text 'description' . . . . . > 'X.25 line at address 40100030'
____


                                                                    BOTTOM
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

Figure E-9 (Part 3 of 3). Prompt Displays for X.25 Line Description

Asynchronous Controller Description to PVC:

```
                    CREATE CTL DESC (ASYNC) (CRTCTLASC)

Type choices, press Enter.

Controller description . . . . . > PVCCTL     Name
Link type  . . . . . .            > *X25      *ASYNC, *X25
Online at IPL  . . . . . .          *YES      *YES, *NO
Switched connection  . . . . . .    *NO       *NO, *YES
Attached nonswitched line  . . . > X25LINE    Name
X.25 logical channel ID  . . . . > 001        001-FFF
Text 'description' . . . . . . . > 'X.25 asynchronous controller to PVC'
____


                                                                    BOTTOM
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

Figure E-10. Prompt Display for Asynchronous PVC Controller

Asynchronous Device Description to PVC:

```
                CREATE DEVICE DESC (ASYNC) (CRTDEVASC)

Type choices, press Enter.

Device description . . . . . > PVCDEV      Name
Remote location name . . . . . > PVCLOC     Name, *NONE
Online at IPL . . . . . .      *YES        *YES, *NO
Attached controller . . . . . > PVCCTL     Name
Text 'description' . . . . . > 'X.25 asynchronous device to PVC'
___

                                                                  BOTTOM
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

Figure E-11. Prompt Display for Asynchronous PVC Device

Incoming Call on a Switched Virtual Circuit (*SVCIN)

This example should be used if you know the network address of the system that starts the call. This configuration is connected only to the address specified for the connection number (CNNNBR parameter).

After creating the controller description, the name must be added to the switched controller list (SWTCTLLST) of the appropriate X.25 line description. To add the name to the switched controller list, type the following command:

```
CHGLINX25 LIND(X25LINE) SWTCTLLST(SVCINCTL1)
```

Asynchronous Controller Description: *SVCIN from a Specific Address:

```
                    CREATE CTL DESC (ASYNC) (CRTCTLASC)

Type choices, press Enter.

Controller description . . . . . > SVCINCTL1     Name
Link type  . . . . . . . . . . . > *X25          *ASYNC, *X25
Online at IPL  . . . . . . . . . > *NO           *YES, *NO
Switched connection  . . . . . . > *YES          *NO, *YES
Switched line list . . . . . . . > X25LINE       Name
            + for more values
Initial connection . . . . . . . > *ANS          *DIAL, *ANS
Connection number  . . . . . . . > 40100055
Text 'description' . . . . . . . > 'X.25 async controller *SVCIN from 40100055'


                                                                      BOTTOM
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

*Figure E-12. Prompt Display for Asynchronous Controller: *SVCIN from Address 40100055*

Asynchronous Device Description *SVCIN from a Specific Address:

```
                    CREATE DEVICE DESC (ASYNC) (CRTDEVASC)

Type choices, press Enter.

Device description . . . . . > SVCINDEV1     Name
Remote location  . . . . . . > SVC1          Name, *NONE
Online at IPL  . . . . . . . > *NO           *YES, *NO
Attached controller  . . . . > SVCINCTL1     Name
Text 'description' . . . . . > 'X.25 async device *SVCIN'


                                                                      BOTTOM
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

*Figure E-13. Prompt Display for Asynchronous Device: *SVCIN from Address 40100055*

Incoming Call on a Switched Virtual Circuit (*SVCIN) for Generic Controllers and Devices

This example describes an incoming call on a switched virtual circuit (*SVCIN) for generic controllers and devices. These descriptions accept a call from any system on the network that satisfies the following conditions:

- The local system controller description must specify INLCNN(*ANS) and CNNNBR(*ANY); no remote location name (RMTLOCNAME(*NONE)) is specified in the device description.
- The remote or calling system must have configured RMTVFY(*YES) and have provided a local location name and local identifier (LCLLOCNAME and LCLID parameters) in the controller description.
- The local system must enter the local location name and local identifier specified by the remote system in the asynchronous communications remote location list (using CRTCFGL TYPE(*ASYNCLOCE)). The local location name cannot currently be specified as a remote location name on an existing asynchronous device.

Figure E-14 on page E-9 shows a configuration of this type. The remote system controller description has specified RMTVFY(*YES) and provided a local location name and local identifier. When the remote system calls the local system, the local system will check the asynchronous remote location list for the LCLLOCNAME and LCLID sent by the remote system. If these entries are included, the call is accepted and the local location name specified by the remote controller is used as the remote location name (RMTLOCNAME parameter) for the generic device description.

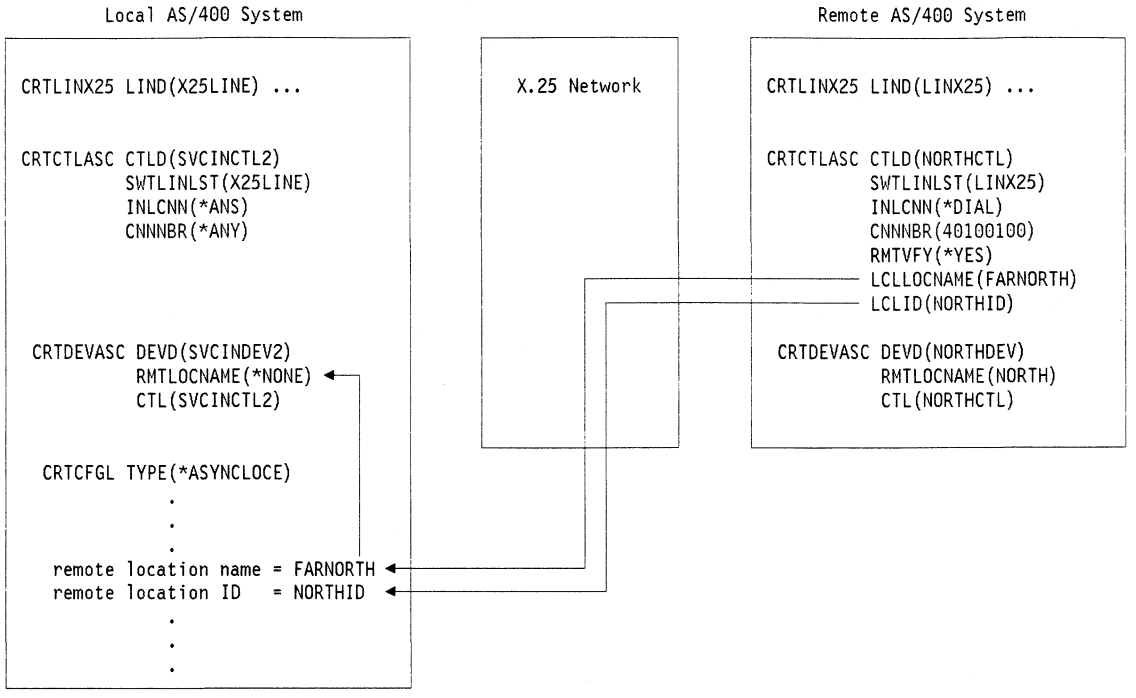


Figure E-14. Using a Generic Device Description with Asynchronous Remote Location Entries

"Outgoing Call on a Switched Virtual Circuit (*SVCOUT)" on page E-10 shows an example of an SVC configured like that of the remote system in Figure E-14.

After creating the controller description, the name must be added to the switched controller list (SWTCTLLST) of the appropriate X.25 line description. To add the name to the switched controller list, type the following command:

```
CHGLINX25 LIND(X25LINE) SWTCTLLST(SVCINCTL2)
```

Asynchronous Controller Description: *SVCIN from Any Address:

```
                    CREATE CTL DESC (ASYNC) (CRTCTLASC)

Type choices, press Enter.

Controller description . . . . . > SVCINCTL2      Name
Link type . . . . . > *X25        *ASYNC, *X25
Online at IPL . . . . . > *NO     *YES, *NO
Switched connection . . . . . > *YES  *NO, *YES
Switched line list . . . . . X25LINE  Name
          + for more values
Initial connection . . . . . > *ANS  *DIAL, *ANS
Connection number . . . . . > *ANY
Text 'description' . . . . . > 'X.25 async controller from any network addr
ess'

                                                                      BOTTOM
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

*Figure E-15. Prompt Display for Asynchronous Controller: *SVCIN from Any Network Address. Display shows a generic controller description (CNNNBR(*ANY)) used with generic device description (RMTLOCNAME(*NONE)) shown in Figure E-16.*

Asynchronous Device Description: *SVCIN from Any Address:

```
                CREATE DEVICE DESC (ASYNC) (CRTDEVASC)

Type choices, press Enter.

Device description . . . . . > SVCINDEV2     Name
Remote location  . . . . . . > *NONE         Name, *NONE
Online at IPL  . . . . . . . > *NO           *YES, *NO
Attached controller  . . . . > SVCINCTL2     Name
Text 'description' . . . . . > 'X.25 async device *SVCIN from any network a
ddress'

                                                                      BOTTOM
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

*Figure E-16. Prompt Display for Asynchronous Device: *SVCIN from Any Network Address. Display shows a generic device description (RMTLOCNAME(*NONE)) used with generic controller description (CNNNBR(*ANY)) shown in Figure E-15.*

Incoming Call on a Switched Virtual Circuit (*SVCIN) for Generic Controllers

This example describes an incoming call on a switched virtual circuit (*SVCIN) for generic controllers. These descriptions accept a call from any system on the network.

After creating the controller description, the name must be added to the switched controller list (SWTCTLLST) of the appropriate X.25 line description. To add the name to the switched controller list, type the following command:

```
CHGLINX25 LIND(X25LINE) SWTCTLLST(SVCINCTL3)
```

Asynchronous Controller Description: *SVCIN from Any Address:

```
                CREATE CTL DESC (ASYNC) (CRTCTLASC)

Type choices, press Enter.

Controller description . . . . . > SVCINCTL3     Name
Link type  . . . . . . . . . . . > *X25          *ASYNC, *X25
Online at IPL  . . . . . . . . . > *NO           *YES, *NO
Switched connection  . . . . . . > *YES          *NO, *YES
Switched line list . . . . . . . > X25LINE       Name
              + for more values
Initial connection . . . . . . . > *ANS          *DIAL, *ANS
Connection number  . . . . . . . > *ANY
Text 'description' . . . . . . . > 'X.25 async ctl *SVCIN from any network addr
ess'

                                                                      BOTTOM
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

*Figure E-17. Prompt Display for Asynchronous Controller: *SVCIN from Any Network Address. Display shows a generic controller description (CNNNBR(*ANY)) used with the device description shown in Figure E-18.*

Asynchronous Device Description: *SVCIN from Any Address:

```
                CREATE DEVICE DESC (ASYNC) (CRTDEVASC)

Type choices, press Enter.

Device description . . . . . > SVCINDEV3     Name
Remote location  . . . . . . > SVC3          Name, *NONE
Online at IPL  . . . . . . . > *NO           *YES, *NO
Attached controller  . . . . > SVCINCTL3     Name
Text 'description' . . . . . > 'X.25 async device *SVCIN from any network a
ddress'

                                                                      BOTTOM
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

*Figure E-18. Prompt Display for Asynchronous Device: *SVCIN from Any Network Address*

Outgoing Call on a Switched Virtual Circuit (*SVCOUT)

This example describes an outgoing switched virtual circuit using remote verification (RMTVFY(*YES)) and specifying a local location name and local identifier to connect to a generic controller and device at the remote system. The

local location name and the local identifier specified for the controller description must also be specified in the asynchronous remote location list (*ASYNCLOCE list when using CRTCFGL) at the remote system.

This configuration is similar to that of the remote system described in Figure E-14 on page E-9.

Asynchronous Controller Description: *SVCOUT to a Specific Address:

```
                    CREATE CTL DESC (ASYNC) (CRTCTLASC)

Type choices, press Enter.

Controller description . . . . . > SVCOUTCTL     Name
Link type  . . . . . . . . . . . > *X25          *ASYNC, *X25
Online at IPL  . . . . . . . . . > *NO           *YES, *NO
Switched connection  . . . . . . > *YES          *NO, *YES
Switched line list . . . . . . . > X25LINE       Name
            + for more values
Initial connection . . . . . . .  *DIAL          *DIAL, *ANS
Connection number  . . . . . . . > 40100100
Text 'description' . . . . . . . > 'X.25 async ctl *SVCOUT to 40100100'

                         Additional Parameters

Attached devices . . . . . .  _________  Name
Predial delay  . . . . . . .  6          0-254 (0.5 seconds)
Redial delay . . . . . . . .  120        0-254 (0.5 seconds)
                                                              MORE...
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

```
                    CREATE CTL DESC (ASYNC) (CRTCTLASC)

Type choices, press Enter.

Dial retry . . . . . . . . . .  2          0-254
Switched disconnect  . . . . .  *NO        *NO, *YES
File transfer ack timer  . . .  16         16-65535 seconds
File transfer retry  . . . . .  7          1-255
Remote verify  . . . . . . . . > *YES      *NO, *YES
Local location . . . . . . . . > FARNORTH  Name
Local identifier . . . . . . . > NORTHID   Name
PAD Emulation  . . . . . . . .  *NO        *NO, *YES
X.25 switched line selection .  *FIRST     *FIRST, *CALC
X.25 default packet size:
  Transmit value . . . . . . .  *LIND      *LIND, 64, 128, 256, 512...
  Receive value  . . . . . . .  *LIND      *LIND, *TRANSMIT, 64, 128...
X.25 default window size:
  Transmit value . . . . . . .  *LIND      1-15, *LIND
  Receive value  . . . . . . .  *LIND      1-15, *LIND, *TRANSMIT
X.25 user group identifier . . .  _        00-99
                                                              MORE...
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

*Figure E-19. Prompt Displays for Asynchronous Controller: *SVCOUT to Address 40100100*

Asynchronous Device Description: *SVCOUT to a Specific Address:

```
                    CREATE DEVICE DESC (ASYNC) (CRTDEVASC)

Type choices, press Enter.

Device description . . . . . > SVCOUTDEV     Name
Remote location  . . . . . . > SVC0          Name, *NONE
Online at IPL  . . . . . . . > *NO           *YES, *NO
Attached controller  . . . . > SVCOUTCTL     Name
Text 'description' . . . . . > 'X.25 async device *SVCOUT'


                                                              BOTTOM
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

*Figure E-20. Prompt Display for Asynchronous Device: *SVCOUT to Address 40100100*

Outgoing Call on a Switched Virtual Circuit (*SVCOUT) for PAD Emulation

This example describes how to configure packet assembler/disassembler (PAD) emulation. PAD emulation support allows an AS/400 application program to communicate with host support that requires the PAD function to be performed.

For a more detailed discussion of the PAD support provided, refer to the *Asynchronous Communications Programmer's Guide*.

Asynchronous Controller Description: *SVCOUT to PAD:

```
                    CREATE CTL DESC (ASYNC) (CRTCTLASC)

Type choices, press Enter.

Controller description . . . . . > SVCPADCTL     Name
Link type  . . . . . . . . . . . > *X25          *ASYNC, *X25
Online at IPL  . . . . . . . . . > *NO           *YES, *NO
Switched connection  . . . . . . > *YES          *NO, *YES
Switched line list . . . . . . . > X25LINE       Name
          + for more values
Initial connection . . . . . .    *DIAL          *DIAL, *ANS
Connection number  . . . . . . > 40100150
Text 'description' . . . . . . > 'X.25 async ctl *SVCOUT to PAD'

                         Additional Parameters

Attached devices . . . . . .      _____          Name
Predial delay  . . . . . . . .    6              0-254 (0.5 seconds)
Redial delay . . . . . . . . .    120            0-254 (0.5 seconds)
                                                                  MORE...
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

*Figure E-21. Prompt Displays for Asynchronous Controller: *SVCOUT to PAD*

```
                    CREATE CTL DESC (ASYNC) (CRTCTLASC)

Type choices, press Enter.

Dial retry . . . . . . . . . .    2              0-254
Switched disconnect  . . . . .    *NO            *NO, *YES
File transfer ack timer  . . .    16             16-65535 seconds
File transfer retry  . . . . .    7              1-255
Remote verify  . . . . . . . .    *NO            *NO, *YES
Local location . . . . . . . .    _____          Name
Local identifier . . . . . . .    _____          Name
PAD Emulation  . . . . . . . . > *YES            *NO, *YES
X.25 switched line selection . .  *FIRST         *FIRST, *CALC
X.25 default packet size:
  Transmit value . . . . . . .    *LIND          *LIND, 64, 128, 256, 512...
  Receive value  . . . . . . .    *LIND          *LIND, *TRANSMIT, 64, 128...
X.25 default window size:
  Transmit value . . . . . . .    *LIND          1-15, *LIND
  Receive value  . . . . . . .    *LIND          1-15, *LIND, *TRANSMIT
X.25 user group identifier . . .  _              00-99
                                                                  MORE...
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Asynchronous Device Description: *SVCOUT to PAD:

```
                    CREATE DEVICE DESC (ASYNC) (CRTDEVASC)

Type choices, press Enter.

Device description . . . . . > SVCPADDEV     Name
Remote location  . . . . . . > PAD           Name, *NONE
Online at IPL  . . . . . . . > *NO           *YES, *NO
Attached controller  . . . . > SVCPADCTL     Name
Text 'description' . . . . . > 'X.25 async device *SVCOUT to PAD'

                                                                  BOTTOM
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

*Figure E-22. Prompt Display for Asynchronous Device: *SVCOUT to PAD*

Appendix F. Program Examples

This appendix contains program examples written in COBOL/400, C/400, and RPG/400 languages. Each language is represented by two sample programs (a source and a target program) that demonstrate passing data using asynchronous communications support. A sample program in FORTRAN/400 is provided in the *ICF Programmer's Guide*.

Not all programming considerations or techniques are illustrated in these examples. You should review the examples before you begin application design and coding.

COBOL/400 Program Examples

The COBOL/400 source program starts a session with a remote location and issues an evoke function, with no invite, to start the target program. The source program sends an item number to the target program and then waits 30 seconds (specified using the DDS TIMER keyword) to receive a response from the target program indicating the evoke function completed successfully. If the source program receives a major return code equal to or greater than 03, the program goes to end of job.

In the following sample programs, the source program sends an item number to the target program requesting item information, then waits 30 seconds. The target program then sends the item information (description) to the source program. The source program sends the value 99999 to the target program, to indicate end-of-transaction. At this point, both programs go to end-of-job.

If the source program does not receive a response from the target program within 30 seconds of sending a request, the source program issues a time-out message and goes to end-of-job.

COBOL/400 Program Descriptions

The following information describes the structure of the sample programs shown in Figure F-3 on page F-5 and Figure F-5 on page F-13. The reference numbers in the figures correspond to those in the descriptions.

COBOL/400 Source Program: The following describes the COBOL/400 inquiry program that runs on the local system.

Program Files: The COBOL/400 inquiry program uses the following files:

ASYNFILS The ICF file used to send records to and receive responses from the target program.

DSPFIL The display device file used to request part number entry at the display station.

DDS Source: The DDS for the ICF file (ASYNFILS) is shown in Figure F-1; the DDS for the display device file (DSPFILE) is shown in Figure F-2.

```
SEQNBR  *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8  Date
   100      A
   200      A                                     INDARA
   300      A
   400      A          R STRTIM
   500      A                                     TIMER(000030)
   600      A
   700      A          R PGMSTR
   800      A                                     EVOKE(&LIB/&PGMID)
   900      A                                     SECURITY(2 'ASYNCPWD' 3 'ASYNCUSR')
  1000      A            PGMID         10A  P
  1100      A            LIB           10A  P
  1200      A* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
  1300      A*                                                               *
  1400      A*  *  Depending on the security level of the Target Source,     *
  1500      A*     a user profile of "ASYNCUSR" having a password of         *
  1600      A*     "ASYNCPWD" may be required on the Target System.          *
  1700      A*                                                               *
  1800      A*  *  User, "ASYNCUSR", must have authority to the device       *
  1900      A*     object (device description) being used on the Target      *
  2000      A*     System.                                                   *
  2100      A*                                                               *
  2200      A* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
  2300      A
  2400      A          R ITEMRQ
  2500      A                                     INVITE
  2600      A            PARTNM         5A
  2700      A
  2800      A          R JOBEND
  2900      A            EOJIND         5A
  3000      A
  3100      A          R INVIT
  3200      A                                     INVITE
  3300      A
  3400      A          R ITEMDS
  3500      A            PARTDS        25A
  3600      A
  3700      A          R ERRDES
  3800      A            ERRORD        40A
  3900      A
  4000      A          R PGMERR
  4100      A                                     INVITE
  4200      A                                     FAIL
                     * * * * *  E N D  O F  S O U R C E  * * * * *
```

Figure F-1. DDS Source for ICF File ASYNFILS, COBOL Source and Target Programs

```
SEQNBR  *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8  Date
   100      A                                     DSPSIZ(24 80 *DS3)
   200      A                                     INDARA
   300      A                                     CF03(99)
   400      A          R PROMPT
   500      A                                  5 10'Part Number: '
   600      A            PARTN          5A  I  5 25
   700      A                                 10 10'Part Description: '
   800      A            PARTD         25A  O 10 30
   900      A            ERRORL        40A  O 12 10DSPATR(HI)
  1000      A                                 23  5'F3 = Exit'
                     * * * * *  E N D  O F  S O U R C E  * * * * *
```

Figure F-2. DDS Source for Display Device File, COBOL Source Program

ICF File Creation and Program Device Entry Definition: The following command is used to create the ICF file. Note that the same ICF file is used for both the source and target programs.

```
CRTICFF    FILE(ASYNLIBCBL/ASYNFILS)
           SRCFILE(ASYNLIBCBL/QDDSSRC)
           SRCMBR(ASYNFIL) MAXPGMDEV(2)
           WAITRCD(30)
```

The following command is used to define the program device entry.

```
ADDICFDEVE FILE(ASYNLIBCBL/ASYNFILS)
           PGMDEV(ICF00)
           RMTLOCNAME(*CHICAGO)
```

The following two commands can also be used.

```
OVRICFDEVE PGMDEV(ICF00) RMTLOCNAME(CHICAGO)

OVRICFF    FILE(ASYNFILS)
           TOFILE(ASYNLIBCBL/ASYNFILS)
```

Display Device File Creation: The following command is used to create the display device file:

```
CRTDSPF    FILE(ASYNLIBCBL/DSPFIL)
           SRCFILE(ASYNLIBCBL/QDDSSRC)
           SRCMBR(DSPFILE)
```

Program Explanation: The following describes the structure of the program example illustrated in Figure F-3 on page F-5.

1 The files used in the program are described in the file control section. ASYNFILS is the ICF file used to send records to and receive records from the target system.

DSPFIL is the name of the display device file used to request an entry from the work station and to display the results of the inquiry.

Note that ICF files are defined to COBOL/400 as work station files.

2 This section of the program redefines the feedback areas for use within the program. See the *ICF Programmer's Guide* for a description of the I/O feedback areas.

3 The ICF file (ASYNFILS) and the display device file (DSPFIL) are opened. The program device named ICF00 is acquired by the program. This program device was previously added to the ICF file (ASYNFILS) by the ADDICFDEVE command.

Once the program device is acquired, routine EVOKE-ROUTINE (**8**) is called to build the evoke request to be sent to the remote system. Because the DDS for the record format only specifies the field identifiers with the record, the program moves the literal value ASYNTINQ to field PGMID, the value ASYNLIBCBL to field LIB, and the value ICF00 to the field PGM-DEV-NME.

The write operation is then issued using record format PGMSTR, which has the evoke function specified in the DDS.

When the program start request is received at the target AS/400 system, ASYNLIBCBL is searched for program ASYNTINQ and that program is then started. The target program for this example is shown in Figure F-5 on page F-13.

4 The program builds the first prompt display to request the entry of a part number and to read the part number.

Routine DISPLAY-PROC (**6**) is called to send the results of the read from the display station to the remote program.

If F3 was pressed while the prompt was displayed, processing goes to the end-of-job routine, END-JOB (**5**).

5 This part of the program does the end-of-job processing. Control passes here whenever the program is going to end normally. The program ends when the operator presses F3 while the part number prompt is displayed.

It first calls DETACH-ROUTINE (**9**) to end the transaction. The files used by the program are then implicitly closed and the program ends.

6 This routine (DISPLAY-PROC) is called from **4** to build the record to send to the target AS/400 system. If F3 is pressed while the prompt is displayed, control passes to **5** .

Routine **7** is called to build the record and send it to the target system. Control then returns here. The results are displayed and input is again requested.

7 This routine (REMOTE-PROC) builds and sends the item request record to the target system. It sends the request using format ITEMRQ. When the operation completes, the routine checks for a successful return code (00 major code, as defined in **2**); if successful, the item description (ITEMDS) is read from the program device ICF00 and then moved to the part description field (PARTD) for the display device (DSPFIL). Control then returns to **6** .

If the target system returned a fail (return code 0302), the error description is read from ICF00 and moved to the display device

error field (ERRORL). Control then returns to 6 .

If any other return code is received, the program goes to end of job (5).

8 This routine (EVOKE-ROUTINE) is called from 3 to build and send the program start request to the remote program. Record format PGMSTR is used to issue the evoke function.

9 This routine (DETACH-ROUTINE) is called from 5 to end the transaction by issuing a write operation using format PGMEND.

10 If an exception occurs, routine ASYNFILS-EXCEPTION is automatically called to check the return code on all operations to ASYNFILS. If the major code is other than a 00 or 03, it ends the program.

```
5738CB1 V2R1M0  910524          AS/400 COBOL Source            ASYNLIBCBL/ASYNSINQ   RCH38321  12/19/90 11:26:45     Page     2
Program . . . . . . . . . . . . . :   ASYNSINQ
  Library . . . . . . . . . . . . :     ASYNLIBCBL
Source file . . . . . . . . . . . :   QCBLSRC
  Library . . . . . . . . . . . . :     ASYNLIBCBL
Source member . . . . . . . . . . :   ASYNSINQ     12/19/90 11:14:31
Generation severity level . . . . :   29
Text 'description' . . . . . . . . :   Source System's asynchronous COBOL program example
Source listing options . . . . . . :   *NONE
Generation options . . . . . . . . :   *NONE
Print file . . . . . . . . . . . . :   QSYSPRT
  Library . . . . . . . . . . . . :     *LIBL
FIPS flagging . . . . . . . . . . :   *NOFIPS *NOSEG *NODEB *NOOBSOLETE
SAA flagging . . . . . . . . . . . :   *NOFLAG
Flagging severity . . . . . . . . :   0
Replace program . . . . . . . . . :   *YES
Target release . . . . . . . . . . :   *CURRENT
User profile . . . . . . . . . . . :   *USER
Authority . . . . . . . . . . . . :   *LIBCRTAUT
Compiler . . . . . . . . . . . . . :   IBM AS/400 COBOL/400

5738CB1 V2R1M0  910524          AS/400 COBOL Source            ASYNLIBCBL/ASYNSINQ   RCH38321  12/19/90 11:26:45     Page     2
 STMT SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S  COPYNAME   CHG DATE

    1  000100 IDENTIFICATION DIVISION.
       000200
    2  000300 PROGRAM-ID.  ASYNSINQ.
       000400
       000500*****************************************************************
       000600* THIS PROGRAM STARTS A PROGRAM CALLED 'ASYNTINQ' A TARGET      *
       000700* SYSTEM.  IT THEN BRINGS UP A DISPLAY WHICH PROMPTS THE USER   *
       000800* FOR A PART NUMBER.  THE PART NUMBER IS PASSED ON PROGRAM      *
       000900* DEVICE 'ICF00' TO 'ASYNTINQ'.  IF THE PART IS FOUND IN THE    *
       001000* DATABASE OF 'ASYNTINQ', THE 'WRITE' IS CONFIRMED, AND THE     *
       001100* DESCRIPTION OF THE PART IS SENT BACK.  IF THE PART IS NOT     *
       001200* FOUND OR THE PART NUMBER IS INVALID, A 'FAIL' IS RETURNED     *
       001300* IN RESPONSE TO THE WRITE ALONG WITH ERROR MESSAGE TEXT.       *
       001400* EITHER THE PART DESCRIPTION (IF THE PART WAS FOUND) OR THE    *
       001500* ERROR MESSAGE TEXT (IF THE PART WAS NOT FOUND) IS DISPLAYED   *
       001600* TO THE USER.  THE USER MAY THEN ENTER A NEW PART NUMBER OR    *
       001700* END THE PROGRAM.  THE PROGRAM IS ENDED BY F3.                 *
       001800*****************************************************************
       001900
    3  002000 ENVIRONMENT DIVISION.
       002100
    4  002200 CONFIGURATION SECTION.
       002300
    5  002400 SOURCE-COMPUTER. IBM-AS400.
    6  002500 OBJECT-COMPUTER. IBM-AS400.
    7  002600 SPECIAL-NAMES.   I-O-FEEDBACK IS IO-FEEDBACK
    8  002700                  OPEN-FEEDBACK IS OPEN-FBA.
       002800
    9  002900 INPUT-OUTPUT SECTION.
       003000
    1
   10  003100 FILE-CONTROL.
       003200
   11  003300     SELECT ASYNFILS ASSIGN TO WORKSTATION-ASYNFILS
   12  003400         ORGANIZATION IS TRANSACTION
   13  003500         CONTROL-AREA IS TR-CTL-AREA
   14  003600         FILE STATUS IS STATUS-IND MAJ-MIN.
   15  003700     SELECT DSPFIL ASSIGN TO WORKSTATION-DSPFIL
   16  003800         ORGANIZATION IS TRANSACTION
   17  003900         CONTROL-AREA IS DISPLAY-FEEDBACK
   18  004000         FILE STATUS IS STATUS-DSP.
       004100
   19  004200 DATA DIVISION.
       004300
   20  004400 FILE SECTION.
       004500
```

Figure F-3 (Part 1 of 6). COBOL/400 Inquiry Example — Source Program

```
      004600*****************************************************************
      004700* FILE DESCRIPTION FOR THE ICF FILE FOR THIS PROGRAM.          *
      004800*****************************************************************
      004900
   21 005000 FD  ASYNFILS
   22 005100     LABEL RECORDS ARE STANDARD.
   23 005200 01  ASYNREC.
   24 005300     COPY DDS-ALL-FORMATS-I-O OF ASYNFILS.
   25 +000001       05  ASYNFILS-RECORD PIC X(40).                          <-ALL-FMTS
      +000002*    I-O FORMAT:STRTIM     FROM FILE ASYNFILS   OF LIBRARY ASYNLIBCBL  <-ALL-FMTS

5738CB1 V2R1M0  910524           AS/400 COBOL Source            ASYNLIBCBL/ASYNSINQ   RCH38321  12/19/90 11:26:45     Page    3
 STMT SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S  COPYNAME   CHG DATE

      +000003*                                                               <-ALL-FMTS
      +000004*      05  STRTIM        REDEFINES ASYNFILS-RECORD.             <-ALL-FMTS
      +000005*  INPUT FORMAT:PGMSTR     FROM FILE ASYNFILS   OF LIBRARY ASYNLIBCBL  <-ALL-FMTS
      +000006*                                                               <-ALL-FMTS
      +000007*      05  PGMSTR-I      REDEFINES ASYNFILS-RECORD.             <-ALL-FMTS
      +000008* OUTPUT FORMAT:PGMSTR     FROM FILE ASYNFILS   OF LIBRARY ASYNLIBCBL  <-ALL-FMTS
      +000009*                                                               <-ALL-FMTS
   26 +000010       05  PGMSTR-O      REDEFINES ASYNFILS-RECORD.             <-ALL-FMTS
   27 +000011           06 PGMID               PIC X(10).                    <-ALL-FMTS
   28 +000012           06 LIB                 PIC X(10).                    <-ALL-FMTS
      +000013*    I-O FORMAT:ITEMRQ     FROM FILE ASYNFILS   OF LIBRARY ASYNLIBCBL  <-ALL-FMTS
      +000014*                                                               <-ALL-FMTS
   29 +000015       05  ITEMRQ        REDEFINES ASYNFILS-RECORD.             <-ALL-FMTS
   30 +000016           06 PARTNM               PIC X(5).                     <-ALL-FMTS
      +000017*    I-O FORMAT:JOBEND     FROM FILE ASYNFILS   OF LIBRARY ASYNLIBCBL  <-ALL-FMTS
      +000018*                                                               <-ALL-FMTS
   31 +000019       05  JOBEND        REDEFINES ASYNFILS-RECORD.             <-ALL-FMTS
   32 +000020           06 EOJIND              PIC X(5).                     <-ALL-FMTS
      +000021*    I-O FORMAT:INVIT      FROM FILE ASYNFILS   OF LIBRARY ASYNLIBCBL  <-ALL-FMTS
      +000022*                                                               <-ALL-FMTS
      +000023*      05  INVIT         REDEFINES ASYNFILS-RECORD.             <-ALL-FMTS
      +000024*    I-O FORMAT:ITEMDS     FROM FILE ASYNFILS   OF LIBRARY ASYNLIBCBL  <-ALL-FMTS
      +000025*                                                               <-ALL-FMTS
   33 +000026       05  ITEMDS        REDEFINES ASYNFILS-RECORD.             <-ALL-FMTS
   34 +000027           06 PARTDS              PIC X(25).                    <-ALL-FMTS
      +000028*    I-O FORMAT:ERRDES     FROM FILE ASYNFILS   OF LIBRARY ASYNLIBCBL  <-ALL-FMTS
      +000029*                                                               <-ALL-FMTS
   35 +000030       05  ERRDES        REDEFINES ASYNFILS-RECORD.             <-ALL-FMTS
   36 +000031           06 ERRORD              PIC X(40).                    <-ALL-FMTS
      +000032*    I-O FORMAT:PGMERR     FROM FILE ASYNFILS   OF LIBRARY ASYNLIBCBL  <-ALL-FMTS
      +000033*                                                               <-ALL-FMTS
      +000034*      05  PGMERR        REDEFINES ASYNFILS-RECORD.             <-ALL-FMTS
      005400
      005500*****************************************************************
      005600* FILE DESCRIPTION FOR THE DISPLAY FILE FOR THIS PROGRAM.      *
      005700*****************************************************************
      005800
   37 005900 FD  DSPFIL
   38 006000     LABEL RECORDS ARE STANDARD.
   39 006100 01  DSPREC.
   40 006200     COPY DDS-ALL-FORMATS-I-O OF DSPFIL.
   41 +000001       05  DSPFIL-RECORD PIC X(65).                            <-ALL-FMTS
      +000002*  INPUT FORMAT:PROMPT     FROM FILE DSPFIL     OF LIBRARY ASYNLIBCBL  <-ALL-FMTS
      +000003*                                                               <-ALL-FMTS
   42 +000004       05  PROMPT-I      REDEFINES DSPFIL-RECORD.               <-ALL-FMTS
   43 +000005           06 PARTN               PIC X(5).                     <-ALL-FMTS
      +000006* OUTPUT FORMAT:PROMPT     FROM FILE DSPFIL     OF LIBRARY ASYNLIBCBL  <-ALL-FMTS
      +000007*                                                               <-ALL-FMTS
   44 +000008       05  PROMPT-O      REDEFINES DSPFIL-RECORD.               <-ALL-FMTS
   45 +000009           06 PARTD               PIC X(25).                    <-ALL-FMTS
   46 +000010           06 ERRORL              PIC X(40).                    <-ALL-FMTS
      006300
   47 006400 WORKING-STORAGE SECTION.
      006500
   48 006600 77 STATUS-IND       PIC XX.
```

Figure F-3 (Part 2 of 6). COBOL/400 Inquiry Example — Source Program

```
5738CB1 V2R1M0  910524          AS/400 COBOL Source            ASYNLIBCBL/ASYNSINQ   RCH38321  12/19/90 11:26:45     Page    4
 STMT SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S  COPYNAME   CHG DATE

   49  006700 77 STATUS-DSP       PIC XX.
   50  006800 77 MAJ-MIN-SAV      PIC X(4).
   51  006900 77 ERR-SW           PIC X       VALUE "0".
   52  007000 77 INDON            PIC 1       VALUE B"1".
   53  007100 77 INDOFF           PIC 1       VALUE B"0".
   54  007200 77 OPEN-COUNT       PIC 9(1)    VALUE 0.
       007300
   55  007400 01 LGTHERR          PIC X(40)
   56  007500                 VALUE "Invalid data received                  ".
   57  007600 01 TIMEERR          PIC X(40)
   58  007700                 VALUE "Timer expired on READ operation        ".
       007800
   59  007900 01 TR-CTL-AREA.
   60  008000    05 FILLER        PIC XX.
   61  008100    05 PGM-DEV-NME   PIC X(10).
   62  008200    05 RCD-FMT-NME   PIC X(10).
       008300
   63  008400 01 DSPF-INDIC-AREA.
   64  008500    05 CMD3          PIC 1       INDIC 99.
   65  008600    88 CMD3-ON    VALUE B"1".
   66  008700    88 CMD3-OFF   VALUE B"0".
       008800
    2
   67  008900 01 IO-FBA.
   68  009000    05 FILLER        PIC X(44).
   69  009100    05 DATA-LEN      PIC 9(2)     USAGE IS COMP-4.
   70  009200    05 FILLER        PIC X(369).
       009300
   71  009400 01 MAJ-MIN.
   72  009500    88 PARITY-ERR    VALUE "0016".
   73  009600    88 DATA-LOST     VALUE "0042".
   74  009700    88 FAIL-RETURNED VALUE "0302".
   75  009800    88 TIME-OUT      VALUE "0310".
   76  009900    05 MAJ           PIC XX.
   77  010000       88 OK-RETURNED   VALUE "00".
   78  010100    05 MIN           PIC XX.
       010200
   79  010300 01 DISPLAY-FEEDBACK.
   80  010400    05 CMD-KEY       PIC XX.
   81  010500    05 FILLER        PIC X(10).
   82  010600    05 RCD-FMT       PIC X(10).
       010700
   83  010800 PROCEDURE DIVISION.
       010900
       011000 DECLARATIVES.
       011100
       011200 ERR-SECTION SECTION.
       011300
       011400*****************************************************************
       011500* ICF FILE ERROR HANDLER.                                       *
       011600*****************************************************************
       011700
       011800        USE AFTER STANDARD ERROR PROCEDURE ON ASYNFILS.
       011900
    10
       012000 ASYNFILS-EXCEPTION.
       012100

5738CB1 V2R1M0  910524          AS/400 COBOL Source            ASYNLIBCBL/ASYNSINQ   RCH38321  12/19/90 11:26:45     Page    5
 STMT SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S  COPYNAME   CHG DATE

   84  012200     IF MAJ NOT = "00" AND MAJ NOT = "03" THEN
   85  012300         STOP RUN.
       012400
       012500 EXIT-DECLARATIVES.
       012600
       012700 END DECLARATIVES.
       012800
```

Figure F-3 (Part 3 of 6). COBOL/400 Inquiry Example — Source Program

```
      012900*****************************************************************
      013000* START OF PROGRAM.                                             *
      013100*                                                               *
      013200* FILES ARE OPENED.  THE PROGRAM DEVICE IS ACQUIRED.  THE       *
      013300* EVOKE PROCESSING IS DONE.  THE INITIAL DISPLAY IS SHOWN       *
      013400* AND THE INITIAL READ FROM THE DISPLAY IS PERFORMED.           *
      013500* PROCESSING CONTINUES UNTIL A F3 IS RECEIVED, THEN             *
      013600* CLEAN UP ROUTINES ARE PERFORMED.                              *
      013700*****************************************************************
      013800
      013900 START-PROGRAM SECTION.
      014000
      014100 START-PROGRAM-PARAGRAPH.
      014200
    3
   86 014300     OPEN I-O ASYNFILS DSPFIL.
   87 014400     MOVE ZEROS TO DSPF-INDIC-AREA.
   88 014500     IF ERR-SW = "1" THEN
   89 014600         IF OPEN-COUNT = 9 THEN
   90 014700             PERFORM ERROR-RECOVERY
   91 014800             STOP RUN
      014900         ELSE
   92 015000            ADD 1 TO OPEN-COUNT
   93 015100            PERFORM ERROR-RECOVERY
   94 015200            GO TO START-PROGRAM-PARAGRAPH
      015300     ELSE
   95 015400         MOVE 0 TO OPEN-COUNT.
   96 015500     ACQUIRE "ICF00   " FOR ASYNFILS.
   97 015600     MOVE "ICF00   " TO PGM-DEV-NME.
   98 015700     PERFORM EVOKE-ROUTINE THRU EVOKE-EXIT.
    4
   99 015800     MOVE SPACES TO DSPREC.
  100 015900     WRITE DSPREC FORMAT IS "PROMPT"
      016000         INDICATORS ARE DSPF-INDIC-AREA.
  101 016100     READ DSPFIL INDICATORS ARE DSPF-INDIC-AREA.
  102 016200     PERFORM DISPLAY-PROC THRU DISPLAY-EXIT
      016300         UNTIL CMD3-ON.
  103 016400     PERFORM END-JOB.
      016500
      016600*****************************************************************
      016700* PROCESS DISPLAY DATA.                                         *
      016800*                                                               *
      016900* SEND THE RESULTS OF THE PREVIOUS READ TO THE TARGET           *
      017000* PROGRAM.  PUT THE RESULTS RETURNED TO THE DISPLAY.            *
      017100* PERFORM ANOTHER READ FROM THE DISPLAY.                        *
      017200*****************************************************************
      017300
    6
      017400 DISPLAY-PROC.
      017500
  104 017600     PERFORM REMOTE-PROC THRU REMOTE-EXIT.

5738CB1 V2R1M0  910524           AS/400 COBOL Source            ASYNLIBCBL/ASYNSINQ   RCH38321  12/19/90 11:26:45     Page    6
 STMT SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S  COPYNAME   CHG DATE

  105 017700     WRITE DSPREC FORMAT IS "PROMPT".
  106 017800     READ DSPFIL INDICATORS ARE DSPF-INDIC-AREA.
      017900
      018000 DISPLAY-EXIT.
      018100
      018200     EXIT.
      018300
      018400*****************************************************************
      018500* PROCESS TARGET SYSTEM INFORMATION.                            *
      018600*                                                               *
      018700* PREPARE AND SEND PART NUMBER RECORD TO THE TARGET PROGRAM.    *
      018800* PERFORM A READ TO GET DATA FROM THE TARGET PROGRAM.  IF A      *
      018900* FAIL IS RECEIVED, PERFORM ANOTHER READ TO GET THE ERROR       *
      019000* MESSAGE FROM THE TARGET PROGRAM.  IF THERE IS A TIME OUT      *
      019100* WAITING FOR THE READ TO COMPLETE, PUT OUT AN APPROPRIATE      *
      019200* MESSAGE.  ALSO, IF THERE IS A DATA LENGTH ERROR OR PARITY     *
      019300* ERROR, AN ERROR MESSAGE IS OUTPUT.                            *
      019400*****************************************************************
```

Figure F-3 (Part 4 of 6). COBOL/400 Inquiry Example — Source Program

```
      019500
    7
  107  019600 REMOTE-PROC.
       019700
  108  019800     MOVE PARTN OF PROMPT-I TO PARTNM.
  109  019900     WRITE ASYNREC FORMAT IS "ITEMRQ"
       020000         TERMINAL IS PGM-DEV-NME.
  110  020100     WRITE ASYNREC FORMAT IS "STRTIM"
       020200         TERMINAL IS PGM-DEV-NME.
  111  020300     READ ASYNFILS.
  112  020400     IF TIME-OUT THEN
  113  020500         MOVE SPACES TO PARTD
  114  020600         MOVE TIMEERR TO ERRORL
       020700     ELSE
  115  020800         IF FAIL-RETURNED THEN
  116  020900             WRITE ASYNREC FORMAT IS "INVIT"
       021000                 TERMINAL IS PGM-DEV-NME
  117  021100             WRITE ASYNREC FORMAT IS "STRTIM"
       021200                 TERMINAL IS PGM-DEV-NME
  118  021300             READ ASYNFILS
  119  021400             READ ASYNFILS FORMAT IS "ERRDES"
  120  021500             MOVE SPACES TO PARTD
  121  021600             MOVE ERRORD TO ERRORL
       021700         ELSE
  122  021800             READ ASYNFILS FORMAT IS "ITEMDS"
  123  021900             ACCEPT IO-FBA FROM IO-FEEDBACK FOR ASYNFILS
  124  022000             IF OK-RETURNED THEN
  125  022100                 IF DATA-LEN NOT = 25 OR PARITY-ERR OR DATA-LOST
  126  022200                     MOVE SPACES TO PARTD
  127  022300                     MOVE LGTHERR TO ERRORL
       022400                 ELSE
  128  022500                     MOVE PARTDS TO PARTD
  129  022600                     MOVE SPACES TO ERRORL
       022700             ELSE
  130  022800                 PERFORM END-JOB.
       022900
       023000 REMOTE-EXIT.
       023100
```

```
5738CB1 V2R1M0  910524           AS/400 COBOL Source            ASYNLIBCBL/ASYNSINQ   RCH38321  12/19/90 11:26:45     Page    7
 STMT SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S  COPYNAME   CHG DATE

       023200     EXIT.
       023300
       023400*****************************************************************
       023500* START TARGET PROGRAM.                                         *
       023600*                                                               *
       023700* MOVE DATA INTO THE APPROPRIATE FIELDS AND WRITE THE EVOKE     *
       023800* REQUEST RECORD OUT TO THE TARGET SYSTEM.                      *
       023900*****************************************************************
       024000
    8
  131  024100 EVOKE-ROUTINE.
       024200
  132  024300     MOVE "ASYNCTCL  " TO PGMID.
  133  024400     MOVE "ASYNLIBCBL" TO LIB.
  134  024500     WRITE ASYNREC FORMAT IS "PGMSTR"
       024600         TERMINAL IS PGM-DEV-NME.
       024700
       024800 EVOKE-EXIT.
       024900
       025000     EXIT.
       025100
       025200*****************************************************************
       025300* PERFORM ERROR RECOVERY.                                       *
       025400*                                                               *
       025500* SEND A DETACH TO THE TARGET SYSTEM.  CLOSE THE FILES.         *
       025600* RESET THE ERROR SWITCH.                                       *
       025700*****************************************************************
       025800
  135  025900 ERROR-RECOVERY.
       026000
  136  026100     CLOSE ASYNFILS DSPFIL.
  137  026200     MOVE "0" TO ERR-SW.
       026300
       026400 ERROR-RECOVERY-EXIT.
       026500
       026600     EXIT.
       026700
```

Figure F-3 (Part 5 of 6). COBOL/400 Inquiry Example — Source Program

```
        026800*****************************************************************
        026900* DETACH FROM TARGET SYSTEM.                                    *
        027000*                                                               *
        027100* SEND '99999' RECORD TO THE TARGET SYSTEM.                     *
        027200*****************************************************************
        027300
      9
    138 027400 DETACH-ROUTINE.
        027500
    139 027600     MOVE "99999" TO EOJIND.
    140 027700     WRITE ASYNREC FORMAT IS "JOBEND"
        027800         TERMINAL IS PGM-DEV-NME.
        027900
        028000 DETACH-EXIT.
        028100
        028200     EXIT.
        028300
        028400*****************************************************************
        028500* END THE JOB.                                                  *
        028600*                                                               *

5738CB1 V2R1M0  910524           AS/400 COBOL Source            ASYNLIBCBL/ASYNSINQ   RCH38321  12/19/90 11:26:45     Page    8
 STMT SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S  COPYNAME   CHG DATE

        028700* PERFORM DETACH PROCESSING.  STOP PROGRAM.                     *
        028800*****************************************************************
        028900
      5
    141 029000 END-JOB.
        029100
    142 029200     PERFORM DETACH-ROUTINE THRU DETACH-EXIT.
        029300
    143 029400     STOP RUN.

                        *****  E N D   O F   S O U R C E  *****

5738CB1 V2R1M0  910524           AS/400 COBOL Messages          ASYNLIBCBL/ASYNSINQ   RCH38321  12/19/90 11:26:45     Page    9

  STMT
*   24  MSGID: LBL0600  SEVERITY: 10  SEQNBR:  005300
        Message . . . . :   No INPUT fields found for format STRTIM.
*   24  MSGID: LBL0600  SEVERITY: 10  SEQNBR:  005300
        Message . . . . :   No INPUT fields found for format PGMSTR.
*   24  MSGID: LBL0600  SEVERITY: 10  SEQNBR:  005300
        Message . . . . :   No INPUT fields found for format INVIT.
*   24  MSGID: LBL0600  SEVERITY: 10  SEQNBR:  005300
        Message . . . . :   No INPUT fields found for format PGMERR.
*   86  MSGID: LBL0335  SEVERITY: 00  SEQNBR:  012700
        Message . . . . :   Empty paragraph or section precedes 'END
          DECLARATIVES' paragraph or section.

                      *****  E N D   O F   M E S S A G E S  *****

                                    Message Summary

  Total     Info(0-4)    Warning(5-19)    Error(20-29)    Severe(30-39)    Terminal(40-99)
      5             1                4               0               0                  0

 Source records read . . . . . . . . . :   294
 Copy records read . . . . . . . . . . :   44
 Copy members processed  . . . . . . . :   2
 Sequence errors . . . . . . . . . . . :   0
 Highest severity message issued . . :   10

  LBL0901 00  Program ASYNSINQ created in library ASYNLIBCBL.

                     *****  E N D   O F   C O M P I L A T I O N  *****
```

Figure F-3 (Part 6 of 6). COBOL/400 Inquiry Example — Source Program

COBOL/400 Target Program: The following describes the COBOL/400 inquiry program that runs on the remote AS/400 system.

Program Files: The COBOL/400 inquiry target program uses the following files:

ASYNFILT The ICF file used to receive part numbers from and send responses to the source program.

Note: The DDS for the ICF file is the same at the source and target program.

DBFIL The database file that contains the part numbers and associated descriptions. This file is used to validate the part number received from the source program.

DDS Source: The DDS for the database file is shown in Figure F-4.

```
SEQNBR  *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8  Date
   100      A* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   200      A*   This file has UNIQUE key values.
   300      A*   In other words, Duplicate Keys are NOT allowed.
   400      A* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   500      A                                     UNIQUE
   600      A          R DBRCD
   700      A            ITEMNM         5
   800      A            ITEMD         25A
   900      A          K ITEMNM

                       * * * * *  E N D  O F  S O U R C E  * * * * *
```

Figure F-4. DDS Source for Database File, COBOL Target Program

ICF File Creation and Program Device Entry Definition: The following command is used to create the ICF file. Note that the same ICF file is used for both the source and target programs.

```
CRTICFF     FILE(ASYNLIBCBL/ASYNFILT)
            SRCFILE(ASYNLIBCBL/QDDSSRC)
            SRCMBR(ASYNFIL)
            MAXPGMDEV(2) WAITRCD(30)
```

The following command is used to define the program device entry.

```
ADDICFDEVE FILE(ASYNLIBCBL/ASYNFILT)
           PGMDEV(ICF01)
           RMTLOCNAME(*REQUESTER)
```

The following two commands can also be used.

```
OVRICFDEVE PGMDEV(ICF01) RMTLOCNAME(*REQUESTER)

OVRICFF    FILE(ASYNFILT)
           TOFILE(ASYNLIBCBL/ASYNFILT)
```

Database File Creation: The following command is used to create the database file:

```
CRTPF       FILE(ASYNLIBCBL/DBFIL)
            SRCFILE(ASYNLIBCBL/QDDSSRC)
            SRCMBR(DBFILE)
```

In order to use the database file with this example, data must be entered in the file. The program requires the item numbers to be greater than 10000.

Program Explanation: The following describes the structure of the program example illustrated in Figure F-5 on page F-13.

1 The file division section defines the files used in the program.

ASYNFILT is the ICF file used to receive records from and send records to the source program. DBFIL is the database file that contains the valid part numbers and part descriptions.

2 This section defines the input/output feedback area for use within the program.

3 The ICF file (ASYNFILT) and the database file (DBFIL) are opened. The program then establishes a session using program device ICF01 in ICF file ASYNFILT. This is the program device that is associated with a remote location name of *REQUESTER. The name of the program device is then moved to field PGM-DEV-NME to define the device used.

A read operation is issued to the program device to receive an inquiry request from the source program. If the read is successful, control passes to **5**.

4 This routine ends the job. It is called from **9** if an error has occurred; or it is performed after a detach is received from the source program (subroutine **5** has completed).

The files used by the program are implicitly closed. The program then ends.

5 This subroutine is called from **3** to process the request from the source program. If the part number received is less than 10000, routine **8** is called to send the error message to the source program.

If the part number is greater than 10000, the database file is read to find the part numbers and associated description. If the part number is not found, routine **7** is called to build the error response. If the part number is found, routine **6** builds the response and sends the record. Routine **5** is repeated until a detach is received from the source program.

6 This routine is called when the part number is found in the database. It builds the response by moving the part description to the output file and then sends the response to the source program.

7 This routine is called from **5** if the part number is not found in the database file. It builds the error response indicating that the record was not found and calls **8** to send the response. Control then returns to **5**.

8 This routine is called from either **5** or **7** to send an error response to the source program.

9 This routine is the exception handler for ASYNFILT. When an exception is issued against the file, control passes here to check the return code. If any return code other than normal (0000) is returned to the program, the program is ended.

10 This routine closes all opened files, resets the error flag, and ends the program.

```
5738CB1 V2R1M0  910524           IBM AS/400 COBOL/400           ASYNLIBCBL/ASYNTINQ   RCH38321  12/19/90 11:27:07     Page    1
Program . . . . . . . . . . . . . :   ASYNTINQ
  Library . . . . . . . . . . . . :     ASYNLIBCBL
Source file . . . . . . . . . . . :   QCBLSRC
  Library . . . . . . . . . . . . :     ASYNLIBCBL
Source member . . . . . . . . . . :   ASYNTINQ     12/19/90 11:14:51
Generation severity level . . . . :   29
Text 'description' . . . . . . . . :   Target System's asynchronous COBOL program example
Source listing options . . . . . . :   *NONE
Generation options . . . . . . . . :   *NONE
Print file . . . . . . . . . . . . :   QSYSPRT
  Library . . . . . . . . . . . . :     *LIBL
FIPS flagging . . . . . . . . . . :   *NOFIPS *NOSEG *NODEB *NOOBSOLETE
SAA flagging . . . . . . . . . . . :   *NOFLAG
Flagging severity . . . . . . . . :   0
Replace program . . . . . . . . . :   *YES
Target release . . . . . . . . . . :   *CURRENT
User profile . . . . . . . . . . . :   *USER
Authority . . . . . . . . . . . . :   *LIBCRTAUT
Compiler . . . . . . . . . . . . . :   IBM AS/400 COBOL/400

5738CB1 V2R1M0  910524           AS/400 COBOL Source            ASYNLIBCBL/ASYNTINQ   RCH38321  12/19/90 11:27:07     Page    2
 STMT SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S  COPYNAME   CHG DATE

    1  000100 IDENTIFICATION DIVISION.
       000200
    2  000300 PROGRAM-ID.  ASYNTINQ.
       000400
       000500*****************************************************************
       000600* THIS PROGRAM IS STARTED BY THE PROGRAM 'ASYNSINQ' ON A        *
       000700* SOURCE SYSTEM SENDING A PROGRAM START REQUEST.                *
       000800* USING THE PROGRAM DEVICE 'ICF01', IT RECEIVES A PART NUMBER   *
       000900* FROM THE SOURCE SYSTEM.  IF THE PART NUMBER IS INVALID, A     *
       001000* 'FAIL' IS SENT TO THE SOURCE SYSTEM AND AND FOLLOWED WITH     *
       001100* THE TEXT TO BE USED AS AN ERROR MESSAGE ON THE SOURCE SYSTEM.*
       001200* THE SAME IS TRUE IF THE PART NUMBER IS SEARCHED FOR BUT       *
       001300* NOT FOUND.  IF THE PART NUMBER IS FOUND, A PART DESCRIPTION   *
       001400* WHICH WAS RETRIEVED FROM THE DATABASE IS SENT TO THE SOURCE   *
       001500* SYSTEM.  IF THE PART NUMBER IS LESS THAN 10000, AN ERROR      *
       001600* MESSAGE IS SENT TO THE SOURCE SYSTEM.                         *
       001700*                                                               *
       001800* THE PROGRAM IS ENDED WHEN A '99999' IS RECEIVED FROM THE      *
       001900* SOURCE SYSTEM.                                                *
       002000*****************************************************************
       002100
    3  002200 ENVIRONMENT DIVISION.
       002300
    4  002400 CONFIGURATION SECTION.
       002500
    5  002600 SOURCE-COMPUTER. IBM-AS400.
    6  002700 OBJECT-COMPUTER. IBM-AS400.
    7  002800 SPECIAL-NAMES.   I-O-FEEDBACK IS IO-FEEDBACK
    8  002900                  OPEN-FEEDBACK IS OPEN-FBA.
       003000
    9  003100 INPUT-OUTPUT SECTION.
       003200
    1
   10  003300 FILE-CONTROL.
       003400
   11  003500     SELECT ASYNFILT ASSIGN TO WORKSTATION-ASYNFILT
   12  003600         ORGANIZATION IS TRANSACTION
   13  003700         CONTROL-AREA IS TR-CTL-AREA
   14  003800         FILE STATUS IS STATUS-IND MAJ-MIN.
   15  003900     SELECT DBFIL ASSIGN TO DATABASE-DBFIL
   16  004000         ORGANIZATION IS INDEXED
   17  004100         ACCESS IS RANDOM
   18  004200         RECORD KEY IS ITEMNM.
       004300
   19  004400 DATA DIVISION.
       004500
   20  004600 FILE SECTION.
       004700
       004800*****************************************************************
       004900* FILE DESCRIPTION FOR THE ICF FILE USED BY THIS PROGRAM.       *
       005000*****************************************************************
       005100
   21  005200 FD  ASYNFILT
   22  005300     LABEL RECORDS ARE STANDARD.
   23  005400 01  ASYNREC.
   24  005500     COPY DDS-ALL-FORMATS-I-O OF ASYNFILT.
```

Figure F-5 (Part 1 of 6). COBOL/400 Inquiry Example — Target Program

```
5738CB1 V2R1M0  910524           AS/400 COBOL Source            ASYNLIBCBL/ASYNTINQ   RCH38321  12/19/90 11:27:07     Page    3
 STMT SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S  COPYNAME   CHG DATE

  25 +000001       05  ASYNFILT-RECORD PIC X(40).                                    <-ALL-FMTS
     +000002*    I-O FORMAT:STRTIM     FROM FILE ASYNFILT   OF LIBRARY ASYNLIBCBL   <-ALL-FMTS
     +000003*                                                                       <-ALL-FMTS
     +000004*       05  STRTIM        REDEFINES ASYNFILT-RECORD.                    <-ALL-FMTS
     +000005*  INPUT FORMAT:PGMSTR     FROM FILE ASYNFILT   OF LIBRARY ASYNLIBCBL   <-ALL-FMTS
     +000006*                                                                       <-ALL-FMTS
     +000007*       05  PGMSTR-I      REDEFINES ASYNFILT-RECORD.                    <-ALL-FMTS
     +000008* OUTPUT FORMAT:PGMSTR     FROM FILE ASYNFILT   OF LIBRARY ASYNLIBCBL   <-ALL-FMTS
     +000009*                                                                       <-ALL-FMTS
  26 +000010       05  PGMSTR-O      REDEFINES ASYNFILT-RECORD.                    <-ALL-FMTS
  27 +000011           06 PGMID                PIC X(10).                          <-ALL-FMTS
  28 +000012           06 LIB                  PIC X(10).                          <-ALL-FMTS
     +000013*    I-O FORMAT:ITEMRQ     FROM FILE ASYNFILT   OF LIBRARY ASYNLIBCBL   <-ALL-FMTS
     +000014*                                                                       <-ALL-FMTS
  29 +000015       05  ITEMRQ        REDEFINES ASYNFILT-RECORD.                    <-ALL-FMTS
  30 +000016           06 PARTNM                PIC X(5).                          <-ALL-FMTS
     +000017*    I-O FORMAT:JOBEND     FROM FILE ASYNFILT   OF LIBRARY ASYNLIBCBL   <-ALL-FMTS
     +000018*                                                                       <-ALL-FMTS
  31 +000019       05  JOBEND        REDEFINES ASYNFILT-RECORD.                    <-ALL-FMTS
  32 +000020           06 EOJIND                PIC X(5).                          <-ALL-FMTS
     +000021*    I-O FORMAT:INVIT      FROM FILE ASYNFILT   OF LIBRARY ASYNLIBCBL   <-ALL-FMTS
     +000022*                                                                       <-ALL-FMTS
     +000023*       05  INVIT         REDEFINES ASYNFILT-RECORD.                    <-ALL-FMTS
     +000024*    I-O FORMAT:ITEMDS     FROM FILE ASYNFILT   OF LIBRARY ASYNLIBCBL   <-ALL-FMTS
     +000025*                                                                       <-ALL-FMTS
  33 +000026       05  ITEMDS        REDEFINES ASYNFILT-RECORD.                    <-ALL-FMTS
  34 +000027           06 PARTDS                PIC X(25).                         <-ALL-FMTS
     +000028*    I-O FORMAT:ERRDES     FROM FILE ASYNFILT   OF LIBRARY ASYNLIBCBL   <-ALL-FMTS
     +000029*                                                                       <-ALL-FMTS
  35 +000030       05  ERRDES        REDEFINES ASYNFILT-RECORD.                    <-ALL-FMTS
  36 +000031           06 ERRORD                PIC X(40).                         <-ALL-FMTS
     +000032*    I-O FORMAT:PGMERR     FROM FILE ASYNFILT   OF LIBRARY ASYNLIBCBL   <-ALL-FMTS
     +000033*                                                                       <-ALL-FMTS
     +000034*       05  PGMERR        REDEFINES ASYNFILT-RECORD.                    <-ALL-FMTS
      005600
      005700*****************************************************************
      005800* FILE DESCRIPTION FOR THE DATABASE FILE USED BY THIS PROGRAM. *
      005900*****************************************************************
      006000
  37  006100 FD  DBFIL
  38  006200     LABEL RECORDS ARE STANDARD.
  39  006300 01  DBREC.
  40  006400     COPY DDS-ALL-FORMATS OF DBFIL.
  41 +000001       05  DBFIL-RECORD PIC X(30).                                      <-ALL-FMTS
     +000002*    I-O FORMAT:DBRCD      FROM FILE DBFIL      OF LIBRARY ASYNLIBCBL   <-ALL-FMTS
     +000003*                                                                       <-ALL-FMTS
     +000004*           USER SUPPLIED KEY BY RECORD KEY CLAUSE                      <-ALL-FMTS
  42 +000005       05  DBRCD         REDEFINES DBFIL-RECORD.                       <-ALL-FMTS
  43 +000006           06 ITEMNM                PIC X(5).                          <-ALL-FMTS
  44 +000007           06 ITEMD                 PIC X(25).                         <-ALL-FMTS
      006500
  45  006600 WORKING-STORAGE SECTION.
      006700
  46  006800   77 STATUS-IND       PIC XX.
  47  006900   77 ERR-SW           PIC X       VALUE "0".

5738CB1 V2R1M0  910524           AS/400 COBOL Source            ASYNLIBCBL/ASYNTINQ   RCH38321  12/19/90 11:27:07     Page    4
 STMT SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S  COPYNAME   CHG DATE

  48  007000   77 OPEN-COUNT       PIC 9(1)    VALUE 0.
  49  007100   77 ERROR-FND        PIC X       VALUE "0".
      007200
  50  007300   01 TR-CTL-AREA.
  51  007400      05 FILLER        PIC X(2).
  52  007500      05 PGM-DEV-NME   PIC X(10).
  53  007600      05 RCD-FMT-NME   PIC X(10).
      007700
    2
  54  007800   01 IO-FBA.
  55  007900      05 FILLER        PIC X(37).
  56  008000      05 DSP-FMT       PIC X(10).
  57  008100      05 FILLER        PIC X(225).
  58  008200      05 PGM-DEVICE-NAME   PIC X(10).
  59  008300      05 FILLER        PIC X(84).
```

Figure F-5 (Part 2 of 6). COBOL/400 Inquiry Example — Target Program

```
 60  008400      05 DEV-DEP-AREA.
 61  008500         10 FILLER     PIC X(4).
 62  008600         10 DATA-LEN   PIC 9(4).
 63  008700         10 FILLER     PIC X(34).
 64  008800         10 MAJ-MIN-S.
 65  008900            15 MAJ-S   PIC XX.
 66  009000            15 MIN-S   PIC XX.
 67  009100         10 FILLER     PIC X(8).
 68  009200      05 FILLER        PIC XXX.
     009300
 69  009400   01 MAJ-MIN.
 70  009500      05 MAJ           PIC XX.
 71  009600      05 MIN           PIC XX.
     009700
 72  009800   01 NOT-FND-MSG      PIC X(40)
 73  009900            VALUE "The requested part was not found.     ".
     010000
 74  010100   01 INV-PRT-MSG      PIC X(40)
 75  010200            VALUE "The part number must be over 10000.   ".
     010300
 76  010400 PROCEDURE DIVISION.
     010500
     010600 DECLARATIVES.
     010700
     010800 ERR-SECTION SECTION.
     010900
     011000*****************************************************************
     011100* ICF FILE ERROR HANDLER.                                       *
     011200*****************************************************************
     011300
     011400     USE AFTER STANDARD ERROR PROCEDURE ON ASYNFILT.
     011500
 9
     011600 ASYNFILT-EXCEPTION.
     011700
 77  011800     IF MAJ-MIN NOT = "0000"
 78  011900         STOP RUN.
     012000
     012100 EXIT-DECLARATIVES.
     012200
     012300 END DECLARATIVES.
     012400
```

```
5738CB1 V2R1M0  910524           AS/400 COBOL Source            ASYNLIBCBL/ASYNTINQ   RCH38321  12/19/90 11:27:07     Page    5
 STMT SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S  COPYNAME   CHG DATE
```

```
     012500*****************************************************************
     012600* START OF PROGRAM.                                             *
     012700*****************************************************************
     012800
     012900 START-PROGRAM SECTION.
     013000
     013100 START-PROGRAM-PARAGRAPH.
     013200
 3
 79  013300     OPEN I-O ASYNFILT DBFIL.
 80  013400     MOVE "ICF01   " TO PGM-DEV-NME.
 81  013500     IF ERR-SW = "1" THEN
 82  013600         IF OPEN-COUNT IS = 9 THEN
 83  013700             PERFORM ERROR-RECOVERY
 84  013800             STOP RUN
     013900         ELSE
 85  014000            ADD 1 TO OPEN-COUNT
 86  014100            PERFORM ERROR-RECOVERY
 87  014200            GO TO START-PROGRAM-PARAGRAPH
     014300     ELSE
 88  014400         MOVE 0 TO OPEN-COUNT.
 89  014500     ACQUIRE "ICF01   " FOR ASYNFILT.
 90  014600     READ ASYNFILT FORMAT IS "ITEMRQ".
 91  014700     PERFORM READ-REQUEST THRU READ-EXIT
     014800             UNTIL ITEMRQ = "99999".
 92  014900     PERFORM END-JOB.
```

Figure F-5 (Part 3 of 6). COBOL/400 Inquiry Example — Target Program

```
      015000
      015100*****************************************************************
      015200* PROCESS INPUT FROM SOURCE SYSTEM.                             *
      015300*                                                               *
      015400* IF THE PART NUMBER IS LESS THAN 10000, AN ERROR MESSAGE IS    *
      015500* MOVED TO THE ERROR MESSAGE OUTPUT FIELD.  OTHERWISE, THE      *
      015600* DATABASE IS SEARCHED FOR THAT PART.  IF IT IS NOT FOUND,      *
      015700* ERROR PROCESSING IS DONE.  IF IT IS FOUND, NORMAL PROCESSING  *
      015800* CONTINUES.                                                    *
      015900*****************************************************************
      016000
    5
      016100 READ-REQUEST.
      016200
   93 016300     MOVE "0" TO ERROR-FND.
   94 016400     IF PARTNM IS LESS THAN "10000" THEN
   95 016500         MOVE INV-PRT-MSG TO ERRORD
   96 016600         PERFORM ERROR-SEND THRU ERROR-EXIT
      016700     ELSE
   97 016800         MOVE PARTNM TO ITEMNM
   98 016900         READ DBFIL FORMAT IS "DBRCD"
   99 017000           INVALID KEY PERFORM RECORD-NOT-FOUND
      017100                       THRU RECORD-NF-EXIT.
  100 017200     IF ERROR-FND = "0" THEN
  101 017300         PERFORM SEND-RECORD THRU SEND-REC-EXIT.
  102 017400     READ ASYNFILT FORMAT IS "ITEMRQ".
      017500
      017600 READ-EXIT.
      017700
      017800     EXIT.
      017900

5738CB1 V2R1M0  910524           AS/400 COBOL Source            ASYNLIBCBL/ASYNTINQ   RCH38321  12/19/90 11:27:07     Page    6
 STMT SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S  COPYNAME   CHG DATE

      018000*****************************************************************
      018100* RECORD WAS FOUND IN DATABASE FILE, RETURN DATA.               *
      018200*                                                               *
      018300* THE ITEM DESCRIPTION OF THE PART IS MOVED TO THE OUTPUT       *
      018400* FIELD AND SENT TO THE SOURCE PROGRAM.                         *
      018500*****************************************************************
      018600
    6
  103 018700 SEND-RECORD.
      018800
  104 018900     MOVE ITEMD TO PARTDS
  105 019000     WRITE ASYNREC FORMAT IS "ITEMDS"
      019100         TERMINAL IS PGM-DEV-NME.
      019200
      019300 SEND-REC-EXIT.
      019400
      019500     EXIT.
      019600
      019700*****************************************************************
      019800* RECORD NOT FOUND IN DATABASE FILE, INDICATE ERROR.            *
      019900*                                                               *
      020000* MOVE THE ERROR MESSAGE TO THE OUTPUT FIELD AND EXECUTE THE    *
      020100* ERROR SEND PROCESSING.                                        *
      020200*****************************************************************
      020300
    7
  106 020400 RECORD-NOT-FOUND.
      020500
  107 020600     MOVE NOT-FND-MSG TO ERRORD.
  108 020700     PERFORM ERROR-SEND THRU ERROR-EXIT.
      020800
      020900 RECORD-NF-EXIT.
      021000
      021100     EXIT.
      021200
```

Figure F-5 (Part 4 of 6). COBOL/400 Inquiry Example — Target Program

```
        021300*****************************************************************
        021400* SEND ERROR BACK TO SOURCE SYSTEM.                             *
        021500*                                                               *
        021600* TURN ERROR INDICATOR FLAG ON.  SEND A 'FAIL' TO THE SOURCE    *
        021700* PROGRAM.  SEND THE ERROR MESSAGE WHICH WAS PREVIOUSLY PUT     *
        021800* INTO THE OUTPUT FIELD TO THE SOURCE PROGRAM.                  *
        021900*****************************************************************
        022000
      8
    109 022100 ERROR-SEND.
        022200
    110 022300     MOVE "1" TO ERROR-FND.
    111 022400     WRITE ASYNREC FORMAT IS "PGMERR"
        022500         TERMINAL IS PGM-DEV-NME.
    112 022600     WRITE ASYNREC FORMAT IS "ERRDES"
        022700         TERMINAL IS PGM-DEV-NME.
        022800
        022900 ERROR-EXIT.
        023000
        023100     EXIT.
        023200
        023300*****************************************************************
        023400* PERFORM ERROR RECOVERY.                                       *

 5738CB1 V2R1M0  910524           AS/400 COBOL Source            ASYNLIBCBL/ASYNTINQ   RCH38321  12/19/90 11:27:07     Page    7
  STMT SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S  COPYNAME   CHG DATE

        023500*                                                               *
        023600* CLOSE OPENED FILES.  RESET THE ERROR FLAG.                    *
        023700*****************************************************************
        023800
     10
    113 023900 ERROR-RECOVERY.
        024000
    114 024100     CLOSE ASYNFILT DBFIL.
    115 024200     MOVE "0" TO ERR-SW.
        024300
        024400 ERROR-RECOVERY-EXIT.
        024500
        024600     EXIT.
        024700
        024800*****************************************************************
        024900* END THE JOB.                                                  *
        025000*                                                               *
        025100* STOP PROGRAM.                                                 *
        025200*****************************************************************
        025300
      4
    116 025400 END-JOB.
        025500
    117 025600     STOP RUN.

                        *****  E N D   O F   S O U R C E  *****

 5738CB1 V2R1M0  910524           AS/400 COBOL Messages          ASYNLIBCBL/ASYNTINQ   RCH38321  12/19/90 11:27:07     Page    8

   STMT
*    24  MSGID: LBL0600  SEVERITY: 10  SEQNBR:  005500
        Message . . . . :   No INPUT fields found for format STRTIM.
*    24  MSGID: LBL0600  SEVERITY: 10  SEQNBR:  005500
        Message . . . . :   No INPUT fields found for format PGMSTR.
*    24  MSGID: LBL0600  SEVERITY: 10  SEQNBR:  005500
        Message . . . . :   No INPUT fields found for format INVIT.
*    24  MSGID: LBL0600  SEVERITY: 10  SEQNBR:  005500
        Message . . . . :   No INPUT fields found for format PGMERR.
*    79  MSGID: LBL0335  SEVERITY: 00  SEQNBR:  012300
        Message . . . . :   Empty paragraph or section precedes 'END
          DECLARATIVES' paragraph or section.

                        *****  E N D   O F   M E S S A G E S  *****
```

Figure F-5 (Part 5 of 6). COBOL/400 Inquiry Example — Target Program

```
                                Message Summary

  Total     Info(0-4)     Warning(5-19)     Error(20-29)     Severe(30-39)     Terminal(40-99)
    5           1              4                0                0                 0

 Source records read . . . . . :   256
 Copy records read . . . . . :   41
 Copy members processed  . . . . . :   2
 Sequence errors . . . . . :   0
 Highest severity message issued . . :   10

  LBL0901 00  Program ASYNTINQ created in library ASYNLIBCBL.

                    *****  E N D  O F  C O M P I L A T I O N  *****
```

Figure F-5 (Part 6 of 6). COBOL/400 Inquiry Example — Target Program

RPG/400 Program Examples

The RPG/400 source program starts a session with a remote location and issues an evoke function, with no invite, to start the target program. The source program sends item numbers to the target program and then waits 30 seconds (the value specified by the WAITRCD parameter on the CRTICFF command) to receive an acknowledgment from the target program indicating that the evoke function completed successfully. If the source program receives a major return code equal to or greater than 03, the program goes to end of job.

In the following sample programs, the source program sends an item number to the target program requesting item information. The target program then sends the item information (description and quantity) to the source program. The source program sends the value 99999 to the target program, to indicate end-of-transaction. At this point, both programs go to end of job.

RPG/400 Program Descriptions

The following information describes the structure of the example programs in Figure F-7 on page F-21 and Figure F-9 on page F-28. The reference numbers in the figures correspond to those in the descriptions.

RPG/400 Source Program: The following describes the RPG/400 inquiry program that runs on the local system.

Program Files: The RPG/400 source program uses the following files:

CMNFILS An ICF file used to send records to and receive records from the target program.

QPRINT An AS/400 printer file that is used to print records, both sent and received, as well as major and minor ICF return codes.

DDS Source: The DDS used in the ICF file is shown in the following example. QPRINT is a program-described file and does not require DDS.

```
SEQNBR  *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8  Date

   100       A* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
   200       A*                                                                     *
   300       A* ICF communications file used by the Source System to                *
   400       A*   send/receive interactively with the Target System.                *
   500       A*                                                                     *
   600       A* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
   700       A*
   800       A          R ITMREC
   900       A            ITMDSC        25A
  1000       A            ITMQTY         5S
  1100       A*
  1200       A          R EVOKE
  1300       A                                    EVOKE(&LIB/&PGM)
  1400       A                                    SECURITY( 2 &USRPWD +
  1500       A                                              3 &USRID)
  1600       A* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
  1700       A*                                                                     *
  1800       A*  *  The data placed in USERID and USRPWD must correspond to        *
  1900       A*     a user profile and password, respectively, on the Target       *
  2000       A*     System.                                                        *
  2100       A*                                                                     *
  2200       A*  *  The user in USRID must have authority to the device            *
  2300       A*     object (device description) being used on the Target           *
  2400       A*     System, as well as to the program and library indicated        *
  2500       A*     in PGM and LIB, respectively.                                  *
  2600       A*                                                                     *
  2700       A* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
  2800       A            PGM           10A  P
  2900       A            LIB           10A  P
  3000       A            USRID         10A  P
  3100       A            USRPWD        10A  P
  3200       A*
  3300       A          R ITMREQ
  3400       A                                    INVITE
  3500       A            ITMNO          5A

                        * * * * *  E N D  O F  S O U R C E  * * * * *

5738SS1 V2R1M0  910524               Data Description              ASYNLIBRPG/CMNFILS      12/14/90  9:48:47       Page    2

                                      Expanded Source

                                                                                                  Field       Buffer position
SEQNBR  *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8 length        Out     In

   800                 R ITMREQ
   900                   ITMDSC        25A  B                                                    25           1      1
  1000                   ITMQTY         5S 0B                                                    5          26     26
  1200                 R EVOKE                    EVOKE(&LIB/&PGM) +
  1400                                            SECURITY( 2 &USRPWD 3 &USRID)
  2800                   PGM           10A  P                                                   10           1
  2900                   LIB           10A  P                                                   10          11
  3000                   USRID         10A  P                                                   10          21
  3100                   USRPWD        10A  P                                                   10          31
  3300                 R ITMREQ                   INVITE
  3500                   ITMNO          5A  B                                                    5           1      1

              * * * * *  E N D  O F  E X P A N D E D  S O U R C E  * * * * *

5738SS1 V2R1M0  910524               Data Description              ASYNLIBRPG/CMNFILS      12/14/90  9:48:47       Page    3

                                      Message Summary

  Total          Informational          Warning        Error       Severe
                     (0-9)               (10-19)       (20-29)     (30-99)
      0              0                     0             0           0

* CPC7301      00              Message . . . . :   File CMNFILS created in library ASYNLIBRPG.

                      * * * * *  E N D  O F  C O M P I L A T I O N  * * * * *
```

Figure F-6. DDS Source for ICF File CMNFILS, RPG/400 Source Program

ICF File Creation and Program Device Entry Definition: The following command is needed to create the ICF file:

```
CRTICFF     FILE(ASYNLIBRPG/CMNFILS)
            SRCFILE(ASYNLIBRPG/QDDSSRC)
            ACQPGMDEV(CMNFILS)
            MAXPGMDEV(2) WAITRCD(30)

OVRICFDEVE PGMDEV(CMNFILS)
            RMTLOCNAME(CHICAGO)
```

Program Explanation: The following describes the structure of the program example shown in Figure F-7 on page F-21. The ICF file used in this example contains externally described data formats (DDS) defined by the user. The reference letters in the figure correspond to those in the following program description.

1 This section identifies the files used in the program. CMNFILS is the name of the ICF file used to send/receive records to and from the target program.

The files used in the program are opened at the beginning of the RPG cycle and the ICF program device is implicitly acquired because the ACQPGMDEV parameter was specified on the CRTICFF command.

2 IOFB is the name of the file information data structure (INFDS) used with CMNFILS. It contains the following information:

- File status (STS)
- Major and minor return code (MAJMIN, MAJCOD, MINCOD)

3 This section builds the evoke function to send to the target system. Because the DDS for the record format only specifies the field identifiers with the record, this code moves the values for the program name, library name, user-id, and password from the array, TARGET, to fields PGM, LIB, USRID, and USRPWD, respectively.

When the program start request is received at the target system, CMNLIB is searched for ASYNCRCL and that program is then started. ASYNCRCL is a CL program that contains the following:

```
ADDLIBLE    LIB(ASYNLIBRPG)
OVRICFDEVE PGMDEV(CMNFILT)
            RMTLOCNAME(*REQUESTER)
OVRPRTF     FILE(QPRINT)
            OUTQ(ASYNLIBRPG/ASYNCT)
CALL        PGM(ASYNLIBRPG/ASYNCT)
```

4 Item numbers are sequentially retrieved from the array ITM# and sent to the target program.

5 In this section, the value accessed in the array ITM# is sent to the target program. The record format ITMREQ contains the item number. The DDS keyword INVITE allows the target program to respond.

6 A read-from-invited-program-devices operation is performed to receive the data from the target program. This operation continues to wait for data until data is received or until the timer value, specified in the WAITRCD parameter of the CRTICFF command, is exceeded. If the time specified for the WAITRCD parameter is exceeded, an ICF major or minor return code of 0310 is received by the source program, and the program ends. Using a read-from-invited-program-devices operation and the WAITRCD parameter prevents the source program from waiting indefinitely if no data is available.

7 This section sends a flag to the target program to indicate the end of transaction.

8 If an error occurs, the session ends.

9 The program ends by setting the last run indicator (LR) to ON and returning to the program that called the program. The ICF file is closed, and the session ends at the end of the RPG/400 cycle.

```
5738RG1 V2R1M0  910524            IBM AS/400 RPG/400              ASYNLIBRPG/ASYNCS       12/14/90  09:48:48      Page       1

Compiler . . . . . :   IBM AS/400 RPG/400
Command Options:
  Program  . . . . . :   ASYNLIBRPG/ASYNCS
  Source file  . . . . . :   ASYNLIBRPG/QRPGSRC
  Source member  . . . . . :   *PGM
  Source listing options . . . . . :   *SOURCE     *XREF       *GEN        *NODUMP     *NOSECLVL
  Generation options . . . . . :   *NOLIST     *NOXREF     *NOATR      *NODUMP     *NOOPTIMIZE
  Source listing indentation . . . :   *NONE
  SAA flagging . . . . . :   *NOFLAG
  Generation severity level  . . . :   9
  Print file . . . . . :   *LIBL/QSYSPRT
  Replace program  . . . . . :   *YES
  Target release . . . . . :   *CURRENT
  User profile . . . . . :   *USER
  Authority  . . . . . :   *LIBCRTAUT
  Text . . . . . :   *SRCMBRTXT
  Phase trace  . . . . . :   *NO
  Intermediate text dump . . . . . :   *NONE
  Snap dump  . . . . . :   *NONE
  Codelist . . . . . :   *NONE
  Ignore decimal data error  . . . :   *NO
Actual Program Source:
  Member . . . . . :   ASYNCS
  File . . . . . :   QRPGSRC
  Library  . . . . . :   ASYNLIBRPG
  Last Change  . . . . . :   12/14/90  09:48:07
  Description  . . . . . :   Source System's RPG program example (source code).

5738RG1 V2R1M0  910524            IBM AS/400 RPG/400              ASYNLIBRPG/ASYNCS       12/14/90  09:48:48      Page       2
SEQUENCE                                                                                IND   DO    LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE   NUM   UPDATE     LINE   ID

                      S o u r c e   L i s t i n g

     100  H                                                                                                        ASYNCS
     200   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
     300   *   THIS IS AN INTERACTIVE SEND/RECEIVE PROGRAM THAT USES AN ARRAY
     400   *   (ITM#) TO SIMULATE THE RETRIEVING OF AN ITEM NUMBER FROM A DATA
     500   *   FILE AND THEN SENDS THAT ITEM NUMBER TO A TARGET SYSTEM IN ORDER
     600   *   TO RETRIEVE AN ITEM DESCRIPTION AND A QUANTITY FROM THE TARGET
     700   *   SYSTEM'S DATA FILE ON A SUBSEQUENT READ TO THE ICF FILE.
     800   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      1
     900  F*                                                      -               12/14/90
    1000  FCMNFILS CF  E                    WORKSTN
    1100  F                                           KINFDS IOFB
    1200  F                                           KNUM       1
    1300  F                                           KID    ID
           RECORD FORMAT(S):  LIBRARY ASYNLIBRPG FILE CMNFILS.
                    EXTERNAL FORMAT ITMREC RPG NAME ITMREC
                    EXTERNAL FORMAT EVOKE RPG NAME EVOKE
                    EXTERNAL FORMAT ITMREQ RPG NAME ITMREQ

    1400  FQPRINT  O   F     132     OF     PRINTER

    1500  E* ARRAYS
    1600  E                   TARGET  1   4 10              Target Sys. Info.               12/12/90
    1700  E                   FILERR  1   6 66              File Error Msgs.                12/13/90
    1800  E                   ITM#    1  20  5              ITEM NUMBER

A000000   INPUT  FIELDS FOR RECORD ITMREC FILE CMNFILS FORMAT ITMREC.
A000001                                      1  25 ITMDSC
A000002                                     26  300ITMQTY
B000000   INPUT  FIELDS FOR RECORD EVOKE FILE CMNFILS FORMAT EVOKE.
C000000   INPUT  FIELDS FOR RECORD ITMREQ FILE CMNFILS FORMAT ITMREQ.
C000001                                      1   5 ITMNO
      2
    1900  IIOFB       DS
    2000  I* I/O FEEDBACK AREA VALUES
    2100  I                                  *STATUS  STS
    2200  I                                   401 404 MAJMIN
    2300  I                                   401 402 MAJCOD
    2400  I                                   403 404 MINCOD
```

Figure F-7 (Part 1 of 5). RPG/400 Inquiry Example — Source Program

```
5738RG1 V2R1M0  910524            IBM AS/400 RPG/400              ASYNLIBRPG/ASYNCS       12/14/90  09:48:48      Page       3
SEQUENCE                                                                    IND   DO    LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE   NUM   UPDATE     LINE   ID

    2600  C* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    2700  *   EVOKE program 'ASYNCTCL' in library 'ASYNLIBRPG' on the Target              12/11/90
    2800  *   system, using the user-id and password indicated by USRID and              12/11/90
    2900  *   USRPWD, respectively.  The 'acquire' function is performed by              12/11/90
    3000  *   the 'ACQPGMDEV' parameter specified in the ICF file.                       12/11/90
    3100  * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    3200  *  Indicator 98 tells you whether the WRITE command completed                 12/13/90
    3300  *  successfully.  (OFF means the command completed successfully.)             12/13/90
    3400  * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -          12/13/90
    3500  *
          3
    3600  C                   MOVEL'CMNFILS' ID
    3700  C                   MOVELTARGET,3  PGM            Target Program              12/12/90
    3800  C                   MOVELTARGET,4  LIB            Target Library              12/12/90
    3900  C                   MOVELTARGET,1  USRID          Target User-ID              12/12/90
    4000  C                   MOVELTARGET,2  USRPWD         Target Password             12/12/90
    4100  C                   WRITEEVOKE               98   DO THE EVOKE      2
    4200  C   98              TIME           TIME    60    GET TIME SENT               12/13/90
    4300  C   98              MOVELFILERR,1  PART1   66                                12/13/90
    4400  C   98              MOVELFILERR,2  PART2   66                                12/13/90
    4500  C   98      MAJCOD  CABGE'  '      ERROR                                     12/13/90
    4600  C   98      MAJCOD  CABGE'03'      ERROR
    4700  *
    4800  C           REQST   TAG
    4900  *
    5000  * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    5100  *   THE FOLLOWING IS THE ARRAY PROCESSING THAT IS USED TO SIMULATE
    5200  *   READING A DATA BASE OR DISPLAY FILE TO GET THE ITEM NUMBER TO
    5300  *   SEND TO THE TARGET SYSTEM.
    5400  * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    5500  *
          4
    5600  C           X       CABGE20        LAST          LAST ELEMENT?               12/13/90
    5700  C                   ADD  1         X       20
    5800  C                   MOVE ITM#,X    ITMNO         ITEM NUMBER
    5900  *
    6000  * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    6100  *   THIS WRITE WILL SEND THE ITEM NUMBER (ITMNO) TO THE TARGET
    6200  *   SYSTEM AND THEN INVITE IT TO SEND.
    6300  * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    6400  *  Indicator 98 tells you whether the WRITE command completed                 12/13/90
    6500  *  successfully.  (OFF means the command completed successfully.)             12/13/90
    6600  * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -          12/13/90
    6700  *
          5
    6800  C                   WRITEITMREQ              98   SEND W/INVITE     2
    6900  C                   TIME           TIME    60    GET TIME SENT
    7000  C   98              MOVELFILERR,5  PART1   66                                12/13/90
    7100  C   98              MOVEL*BLANKS   PART2   66                                12/13/90
    7200  C   98      MAJCOD  CABGE'  '      ERROR                                     12/13/90
    7300  C   98      MAJCOD  CABGE'03'      ERROR
    7400  C                   EXCPTREQ                     PRINT LOG
    7500  *
    7600  * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    7700  *   THIS READ-FROM-INVITED-DEVICES IS TO RECEIVE THE ITEM                      12/11/90

5738RG1 V2R1M0  910524            IBM AS/400 RPG/400              ASYNLIBRPG/ASYNCS       12/14/90  09:48:48      Page       4
SEQUENCE                                                                    IND   DO    LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE   NUM   UPDATE     LINE   ID

    7800  *   DESCRIPTION AND QUANTITY.  USING A READ-FROM-INVITED-DEVICES (ON            12/11/90
    7900  *   THE TIME VALUE SPECIFIED ON THE WAITRCD PARAMETER)  PREVENTS               12/11/90
    8000  *   WAITING INDEFINITELY ON THE READ STATEMENT.  THE 'READ'
    8100  *   STATEMENT REFERENCES THE FILE NAME 'CMNFILS' TO BE ABLE TO
    8200  *   DO A 'READ-FROM-INVITED-DEVICES' WHICH IS REQUIRED TO GET
    8300  *   THE TIMER TO INTERRUPT THE 'READ'.  SINCE RECORD-IDENTIFYING
    8400  *   CHARACTERS WERE NOT USED FOR THE FORMATS IN THE FILE,
    8500  *   THE 'ITMREC' FORMAT WAS PLACED FIRST IN THE FILE AND WILL                  12/11/90
    8600  *   RECEIVE THE INPUT FROM 'READ CMNFILS'STATEMENT.
    8700  * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    8800  *  Indicator 10 tells you whether you have reached the end of the             12/13/90
    8900  *  file (EOF).  (ON means EOF has been reached.)                             12/13/90
```

Figure F-7 (Part 2 of 5). RPG/400 Inquiry Example — Source Program

```
 9000   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -              12/13/90
 9100   *  Indicator 98 tells you whether the READ command completed                 12/13/90
 9200   *  successfully.  (OFF means the command completed successfully.)            12/13/90
 9300   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -              12/13/90
 9400   *
         6
 9500   C                   READ CMNFILS              9810                2 3
 9600   C                   TIME          TIME            GET TIME RECVD
 9700   C                   MOVEL*BLANKS  PART2  66                                12/13/90
 9800   C   98              MOVELFILERR,3 PART1  66                                12/13/90
 9900   C   98 10           MOVELFILERR,4 PART2  66                                12/13/90
10000   C  N98 10           MOVELFILERR,4 PART1  66                                12/13/90
10100   C      10           GOTO ERROR                                             12/13/90
10200   C   98      MAJMIN  CABGE'0310'   ERROR           CK INCL TIMER
10300   C                   EXCPTRECVD
10400   C                   GOTO REQST

5738RG1 V2R1M0  910524            IBM AS/400 RPG/400              ASYNLIBRPG/ASYNCS       12/14/90  09:48:48      Page       5
SEQUENCE                                                                    IND    DO    LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE    NUM   UPDATE     LINE   ID

10600   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
10700   *   THIS SECTION WILL SEND A DUMMY 'ITMNO' OF '99999' TO THE TARGET
10800   *   PROGRAM TO INDICATE THE END OF THE TRANSACTIONS.
10900   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
11000   *  Indicator 98 tells you whether the WRITE command completed                12/13/90
11100   *  successfully.  (OFF means the command completed successfully.)            12/13/90
11200   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -              12/13/90
11300   *
         7
11400   C           LAST      TAG
11500   C                     MOVE '99999'  ITMNO           ITMNO = 99999
11600   C                     WRITEITMREQ             98    SEND LAST REC     2
11700   C                     TIME          TIME            GET TIME SENT
11800   C   98                MOVELFILERR,6 PART1  66                                12/13/90
11900   C   98                MOVEL*BLANKS  PART2  66                                12/13/90
12000   C   98      MAJCOD    CABGE' '      ERROR           CK FOR ERROR             12/13/90
12100   C   98      MAJCOD    CABGE'03'     ERROR           CK FOR ERROR
12200   C                     EXCPTREQ
12300   C                     GOTO END
12400   *
12500   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
12600   *   ERROR OCCURRED.  PRINT THE MAJOR/MINOR RETURN CODES.
12700   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
12800   *
         8
12900   C           ERROR     TAG
13000   *                                                                          12/13/90
13100   C   10                                                                     12/13/90
13200   COR 98                EXCPTERRDSC                 ERROR DESCRIPTN          12/13/90
13300   *                                                                          12/13/90
13400   C                     EXCPTERR
13500   *
13600   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
13700   *   END-OF-JOB.  SETON 'LR' INDICATOR.
13800   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
13900   *
         9
14000   C           END       TAG
14100   C                     SETON                   LR                1
14200   C                     RETRN
```

Figure F-7 (Part 3 of 5). RPG/400 Inquiry Example — Source Program

```
5738RG1 V2R1M0  910524            IBM AS/400 RPG/400              ASYNLIBRPG/ASYNCS       12/14/90  09:48:48      Page       6
SEQUENCE                                                                                  IND    DO    LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE    NUM   UPDATE     LINE   ID

  14400  O* - - - - - - - - - *
  14500   *  PRINT HEADINGS   *
  14600   * - - - - - - - - - *
  14700  OQPRINT  H  301   1P
  14800  O       OR        OF
  14900  O                                  22 'SOURCE TRANSACTION LOG'
  15000  O                        UDATE Y   35
  15100  O                                 110 'PAGE'                                  12/14/90
  15200  O                        PAGE  J  116                                        12/14/90
  15300  O                                 130 'ASYNCS'
  15400   * - - - - - - - - - - - - - - - *
  15500   *  PRINT REQUEST TRANSACTION  *
  15600   * - - - - - - - - - - - - - - - *
  15700  O        EF 1            REQ
  15800  O                                  18 'ITEM NUMBER SENT -'
  15900  O                        ITMNO     25
  16000  O                                  90 'TIME -'
  16100  O                        TIME      99 '0 :  :  '
  16200  O                                 130 'ASYNCS'
  16300   * - - - - - - - - - - - - - - - - *
  16400   *  PRINT RECEIVED TRANSACTION   *
  16500   * - - - - - - - - - - - - - - - - *
  16600  O        EF 2            RECVD
  16700  O                                  22 'RECEIVED FROM TARGET :'
  16800  O                                  32 'ITMDSC -'
  16900  O                        ITMDSC    58
  17000  O                                  68 'ITMQTY -'
  17100  O                        ITMQTYJ   76
  17200  O                                  90 'TIME -'
  17300  O                        TIME      99 '0 :  :  '
  17400  O                                 130 'ASYNCS'
  17500   * - - - - - - - - - - - - - - - - - *                                      12/13/90
  17600   *  PRINT ERROR DESCRIPTIONS         *                                      12/13/90
  17700   * - - - - - - - - - - - - - - - - - *                                      12/13/90
  17800  O        EF 1            ERRDSC                                           12/13/90
  17900  O                        PART1     66                                     12/13/90
  18000  O                        PART2    132                                     12/13/90
  18100   * - - - - - - - - - - - - - - - - *
  18200   *  PRINT MAJOR/MINOR RETURN CODES  *
  18300   * - - - - - - - - - - - - - - - - *
  18400  O        EF 2            ERR
  18500  O                                  24 'MAJOR/MINOR RETURN CODE:'
  18600  O                        MAJCOD    27
  18700  O                                  28 '/'
  18800  O                        MINCOD    30
  18900  O                                  40 'STATUS -'
  19000  O                        STS       46
  19100  O                                  52 'ID -'
  19200  O                        ID        63
  19300  O                                  80 'JOB ENDED'
  19400  O                                  90 'TIME -'
  19500  O                        TIME      99 '0 :  :  '
  19600  O                                 130 'ASYNCS'

 D000000   OUTPUT FIELDS FOR RECORD EVOKE FILE CMNFILS FORMAT EVOKE.
 D000001                          PGM       10  CHAR   10
 D000002                          LIB       20  CHAR   10
 D000003                          USRID     30  CHAR   10
 D000004                          USRPWD    40  CHAR   10
 E000000   OUTPUT FIELDS FOR RECORD ITMREQ FILE CMNFILS FORMAT ITMREQ.
 E000001                          ITMNO      5  CHAR    5


          *****  E N D   O F   S O U R C E  *****

         A d d i t i o n a l   D i a g n o s t i c   M e s s a g e s
```

Figure F-7 (Part 4 of 5). RPG/400 Inquiry Example — Source Program

```
5738RG1 V2R1M0  910524            IBM AS/400 RPG/400              ASYNLIBRPG/ASYNCS       12/14/90  09:48:48      Page       7
SEQUENCE                                                                                                    LAST
 NUMBER    *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8                 UPDATE

                  C o m p i l e - T i m e   T a b l e s

 Table/Array . . . . . :   TARGET

    19800  ASYNCUSR                     /* Target System:  User-ID */                    12/12/90
    19900  ASYNCPWD                     /* Target System:  Password */                   12/12/90
    20000  ASYNCTCL                     /* Target System:  Program name */               12/12/90
    20100  ASYNLIBRPG                   /* Target System:  Library name */               12/12/90

 Table/Array . . . . . :   FILERR

    20300  Error occurred when issuing the initial WRITE.  The ICF file may h            12/13/90
    20400  ave been created without the proper value for ACQPGMDEV parameter.           12/13/90
    20500  Error occurred when issuing the READ CMNFILT command.  ... ...              12/13/90
    20600  EOF is indicated.  (Has the CMNFILS been ACQUIREd?)                          12/13/90
    20700  Error occurred when issuing the WRITE ITMREQ command.                        12/13/90
    20800  Error occurred when issuing the Last WRITE ITMREQ command.                   12/13/90

 Table/Array . . . . . :   ITM#

    21000  00001
    21100  00002
    21200  00003
    21300  00004
    21400  00005
    21500  00006
    21600  00007
    21700  00008
    21800  00009
    21900  00010
    22000  00011
    22100  00012
    22200  00013
    22300  00014
    22400  00015
    22500  00016
    22600  00017
    22700  00018
    22800  00019
    22900  00020

5738RG1 V2R1M0  910524            IBM AS/400 RPG/400              ASYNLIBRPG/ASYNCS       12/14/90  09:48:48      Page       8
                        F i n a l   S u m m a r y
No errors found in source program.
Program Source Totals:
   Records . . . . . :   229
   Specifications  . . . . . :   101
   Table Records . . . . . :   30
   Comments  . . . . . :   92
PRM has been called.
Program ASYNCS is placed in library ASYNLIBRPG. 00 highest Error-Severity-Code.
        * * * * *  E N D  O F  C O M P I L A T I O N  * * * * *
```

Figure F-7 (Part 5 of 5). RPG/400 Inquiry Example — Source Program

RPG/400 Target Program: The following describes an RPG/400 asynchronous target program.

Program Files: The RPG/400 target program uses the following files:

CMNFILR An ICF file used to send records to and receive records from the source program.

QPRINT An AS/400 printer file that is used to print records, both sent and received, as well as major and minor ICF return codes.

DDS Source: The following example shows the DDS keywords that are used in the ICF file. (QPRINT is a program-described file; no DDS is required.)

```
SEQNBR  *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8  Date
   100       A* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
   200       A*                                                                     *
   300       A* ICF communications file used by the Target System to                *
   400       A*   receive/send interactively with the Source System.                *
   500       A*                                                                     *
   600       A* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
   700       A*
   800       A          R ITMREQ
   900       A            ITMNO          5A
  1000       A*
  1100       A          R ITMREC
  1200       A                                     INVITE
  1300       A            ITMDSC        25A
  1400       A            ITMQTY         5S
  1500       A*
  1600       A          R INVITE
  1700       A                                     INVITE
                        * * * * *  E N D  O F  S O U R C E  * * * * *
5738SS1 V2R1M0  910524                Data Description              ASYNLIBRPG/CMNFILT      12/14/90  9:49:01        Page    2
                                      Expanded Source
                                                                                  Field       Buffer position
SEQNBR  *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8 length        Out     In
   800                R ITMREQ
   900                  ITMNO          5A  B                                        5            1       1
  1100                R ITMREC                    INVITE
  1300                  ITMDSC        25A  B                                      25            1       1
  1400                  ITMQTY         5S 0B                                      5           26      26
  1600                R INVITE                    INVITE
              * * * * *  E N D  O F  E X P A N D E D  S O U R C E  * * * * *
5738SS1 V2R1M0  910524                Data Description              ASYNLIBRPG/CMNFILT      12/14/90  9:49:01        Page    3
                                      Message Summary
   Total          Informational         Warning        Error       Severe
                      (0-9)              (10-19)       (20-29)      (30-99)
       0              0                    0             0            0
* CPC7301      00              Message . . . . :   File CMNFILT created in library ASYNLIBRPG.
                        * * * * *  E N D  O F  C O M P I L A T I O N  * * * * *
```

Figure F-8. DDS Source for ICF File CMNFILT, RPG/400 Target Program

ICF File Creation and Program Device Entry Definition: The command needed to create the ICF file is:

```
CRTICFF    FILE(ASYNLIBRPG/CMNFILT)
           SRCFILE(ASYNLIBRPG/QDDSSRC)
           ACQPGMDEV(CMNFILT)
           MAXPGMDEV(2) WAITRCD(30)
```

The following command is needed to define the program device entry:

```
OVRICFDEVE PGMDEV(CMNFILT)
           RMTLOCNAME(*REQUESTER)
```

Program Explanation

The following describes the structure of the program example shown in Figure F-9 on page F-28. The ICF file used in this example uses externally described data formats (DDS) defined by the user. The reference letters in the figure correspond to those in the following program.

1 This section identifies the files used in the program. CMNFILT is the ICF file used to send/receive records to and from the source program.

The files used in the program are opened at the beginning of the RPG cycle and the ICF program device is implicitly acquired because the ACQPGMDEV parameter was specified on the CRTICFF command.

2 IOFB is the name of the file information data structure (INFDS) used with CMNFILT. It contains the following information:

- File status (STS)
- Major/minor return code (MAJMIN,MAJCOD,MINCOD)

3 A read-from-invited-program-devices operation is performed to receive data from the source program and will continue to wait for data until data is received or the timer value, specified in the WAITRCD parameter of the CRTICFF command, is exceeded. Using a read-from-invited-program-devices operation and the WAITRCD parameter prevents the target program from waiting indefinitely if no data is available.

4 This section checks the ITMNO field for the value 99999 that indicates the end of transaction. When 99999 is received, the session ends; otherwise, the item corresponding to the value sent from the source program is accessed in the arrays ITMDES and ITMQNT, and placed in the fields ITMDSC and ITMQTY.

5 Item information is retrieved and sent to the source program.

6 If an error occurs, the session ends.

7 The program ends by setting the last run indicator (LR) to ON and returning to the program that called the program. The ICF file is closed, and the session ends at the end of the RPG/400 cycle.

```
5738RG1 V2R1M0  910524            IBM AS/400 RPG/400              ASYNLIBRPG/ASYNCT       12/14/90  09:49:02      Page       1
Compiler . . . . . . . . . . . . . :   IBM AS/400 RPG/400
Command Options:
  Program  . . . . . . . . . . . . :   ASYNLIBRPG/ASYNCT
  Source file  . . . . . . . . . . :   ASYNLIBRPG/QRPGSRC
  Source member  . . . . . . . . . :   *PGM
  Source listing options . . . . . :   *SOURCE     *XREF       *GEN        *NODUMP     *NOSECLVL
  Generation options . . . . . . . :   *NOLIST     *NOXREF     *NOATR      *NODUMP     *NOOPTIMIZE
  Source listing indentation . . . :   *NONE
  SAA flagging . . . . . . . . . . :   *NOFLAG
  Generation severity level  . . . :   9
  Print file . . . . . . . . . . . :   *LIBL/QSYSPRT
  Replace program  . . . . . . . . :   *YES
  Target release . . . . . . . . . :   *CURRENT
  User profile . . . . . . . . . . :   *USER
  Authority  . . . . . . . . . . . :   *LIBCRTAUT
  Text . . . . . . . . . . . . . . :   *SRCMBRTXT
  Phase trace  . . . . . . . . . . :   *NO
  Intermediate text dump . . . . . :   *NONE
  Snap dump  . . . . . . . . . . . :   *NONE
  Codelist . . . . . . . . . . . . :   *NONE
  Ignore decimal data error  . . . :   *NO
Actual Program Source:
  Member . . . . . . . . . . . . . :   ASYNCT
  File . . . . . . . . . . . . . . :   QRPGSRC
  Library  . . . . . . . . . . . . :   ASYNLIBRPG
  Last Change  . . . . . . . . . . :   12/14/90  09:48:24
  Description  . . . . . . . . . . :   Target System's RPG program example (source code).
5738RG1 V2R1M0  910524            IBM AS/400 RPG/400              ASYNLIBRPG/ASYNCT       12/14/90  09:49:02      Page       2
SEQUENCE                                                                                  IND   DO    LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE   NUM   UPDATE     LINE   ID
                     S o u r c e   L i s t i n g
     100  H                                                                                                        ASYNCT
     200   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
     300   *   THIS IS THE TARGET PROGRAM THAT WILL PERFORM THE INTERACTIVE
     400   *   RECEIVE/SEND FUNCTION WITH THE SOURCE SYSTEM'S PROGRAM.  IT WILL
     500   *   ALSO USE ARRAYS TO SIMULATE THE RETRIEVAL OF DATA FROM A DATA
     600   *   BASE FILE.  IT USES THE 'ITMNO' SENT FROM THE SOURCE SYSTEM AS
     700   *   THE INDEX FOR THE ARRAYS TO RETRIEVE THE APPROPRIATE ITEM
     800   *   DESCRIPTION AND QUANTITY.
     900   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

    1000  F*
           1
    1100  FCMNFILT CF  E                    WORKSTN
    1200  F                                           KINFDS IOFB
    1300  F                                           KNUM       1
    1400  F                                           KID    ID
           RECORD FORMAT(S):  LIBRARY ASYNLIBRPG FILE CMNFILT.
                   EXTERNAL FORMAT ITMREQ RPG NAME ITMREQ
                   EXTERNAL FORMAT ITMREC RPG NAME ITMREC
                   EXTERNAL FORMAT INVITE RPG NAME INVITE

    1500  FQPRINT  O   F     132     OF     PRINTER

    1600  E*  ARRAYS
    1700  E                    FILERR  1   5 66              File Error Msgs.
    1800  E                    ITMERR  1   2 25              Item # Error Msgs
    1900  E                    ITMDES  1  20 25              Item Description
    2000  E                    ITMQNT  1  20  5 0            ITEM QUANTITY

A000000   INPUT  FIELDS FOR RECORD ITMREQ FILE CMNFILT FORMAT ITMREQ.
A000001                                       1   5 ITMNO
B000000   INPUT  FIELDS FOR RECORD ITMREC FILE CMNFILT FORMAT ITMREC.
B000001                                       1  25 ITMDSC
B000002                                      26  300ITMQTY
C000000   INPUT  FIELDS FOR RECORD INVITE FILE CMNFILT FORMAT INVITE.
           2
           *
    2100  IIOFB        DS
    2200  I*  I/O FEEDBACK AREA
    2300  I                                   *STATUS  STS
    2400  I                                    401 404 MAJMIN
    2500  I                                    401 402 MAJCOD
    2600  I                                    403 404 MINCOD
```

Figure F-9 (Part 1 of 5). RPG/400 Inquiry Example — Target Program

```
5738RG1 V2R1M0  910524            IBM AS/400 RPG/400              ASYNLIBRPG/ASYNCT       12/14/90  09:49:02      Page       3
SEQUENCE                                                                                IND    DO    LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE    NUM   UPDATE     LINE   ID
    2800  C*
    2900   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    3000   *   THIS 'WRITE' STATEMENT IS TO ISSUE AN 'INVITE' SO THAT THE
    3100   *   'READ' THAT FOLLOWS WILL HAVE AN OUTSTANDING 'INVITE' AND WILL
    3200   *   FUNCTION AS A 'READ-FROM-INVITED-DEVICES' STATEMENT.
    3300   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    3400   *  Indicator 98 tells you whether the WRITE command completed
    3500   *  successfully.  (OFF means the command completed successfully.)
    3600   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    3700   *
    3800  C                   WRITEINVITE               98                  2
    3900   *
    4000  C   98               TIME          TIME    60       GET TIME RECVD
    4100  C   98               MOVELFILERR,1 PART1   66
    4200  C   98               MOVELFILERR,2 PART2   66
    4300  C   98               GOTO ERROR
    4400   *
    4500   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    4600   *   THIS READ-FROM-INVITED-DEVICES IS TO RECEIVE THE 'ITMNO FROM THE
    4700   *   VALUE SPECIFIED ON THE WAITRCD PARAMETER OF THE ICF FILE)
    4800   *   PREVENTS WAITING INDEFINITELY ON THE 'READ' STATEMENT.
    4900   *   THE READ STATEMENT REFERENCES THE FILE NAME 'CMNFILS' TO BE ABLE
    5000   *   TO DO A 'READ-FROM-INVITED-DEVICES' WHICH IS REQUIRED TO GET THE
    5100   *   TIMER TO INTERRUPT THE 'READ'.  UPON SUCCESSFUL COMPLETION, THE
    5200   *   'READ' WILL PUT THE DATA INTO THE FIRST FORMAT IN THE ICF FILE
    5300   *   SINCE NO RECORD-IDENTIFYING CODES WERE USED FOR THE FORMATS ON
    5400   *   THE INPUT SPECIFICATIONS.
    5500   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    5600   *  Indicator 10 tells you whether you have reached the end of the
    5700   *  file (EOF).  (ON means EOF has been reached.)
    5800   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    5900   *  Indicator 98 tells you whether the READ command completed
    6000   *  successfully.  (OFF means the command completed successfully.)
    6100   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    6200   *
          3
    6300  C           NXTREC    TAG
    6400  C                     READ CMNFILT              9810                2 3
    6500  C                     TIME          TIME    60       GET TIME RECVD
    6600  C                     MOVE *BLANKS  PART2   66
    6700  C   98                MOVELFILERR,3 PART1   66
    6800  C   98 10              MOVELFILERR,4 PART2   66
    6900  C  N98 10              MOVELFILERR,4 PART1   66
    7000  C      10              GOTO ERROR
    7100  C   98       MAJMIN    CABGE'0310'   ERROR            CK INCL TIMER
    7200  C                     EXCPTRECVD                       PRINT RECVD LOG
    7300   *
    7400   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    7500   *   THIS IS THE ROUTINE TO CHECK THE 'ITMNO' FIELD FOR '99999' THAT
    7600   *   INDICATES THAT THE SOURCE SYSTEM HAS NO MORE REQUESTS.  IF NOT
    7700   *   THE END, THEN THE ARRAYS ARE PROCESSED TO RETRIEVE THE ITEM
    7800   *   DESCRIPTION AND QUANTITY TO BE SENT BACK TO THE SOURCE SYSTEM.
    7900   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
5738RG1 V2R1M0  910524            IBM AS/400 RPG/400              ASYNLIBRPG/ASYNCT       12/14/90  09:49:02      Page       4
SEQUENCE                                                                                IND    DO    LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE    NUM   UPDATE     LINE   ID
    8000   *   If the 'ITMNO' is not within the acceptable values for the
    8100   *   array index, the item number is placed in the quantity field
    8200   *   and an error meesage is placed in the description field.
    8300   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    8400   *
          4
    8500  C           ITMNO     CABEQ'99999'  END
    8600  C                     MOVE ITMNO    X       20
    8700   *
    8800  C           X         IFGT 00                        IF X > 0             B001
    8900  C           X         IFLE 20                        IF X <= 20           B002
    9000  C                     MOVE ITMDES,X ITMDSC                                 002
    9100  C                     MOVE ITMQNT,X ITMQTY                                 002
    9200  C                     ELSE                           IF X <=20 ELSE       X002
    9300  C                     MOVE ITMERR,2 ITMDSC                                 002
    9400  C                     MOVE X        ITMQTY                                 002
    9500  C                     ENDIF                          END: IF X <= 20      E002
    9600  C                     ELSE                           IF X > 0 ELSE        X001
    9700  C                     MOVE ITMERR,1 ITMDSC                                 001
    9800  C                     MOVE X        ITMQTY                                 001
    9900  C                     ENDIF                          END: IF X > 0        E001
   10000   *
```

Figure F-9 (Part 2 of 5). RPG/400 Inquiry Example — Target Program

```
    10100   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    10200   *   WRITE THE 'ITMREC' FORMAT THAT WILL SEND THE ITEM DESCRIPTION
    10300   *   AND QUANTITY TO THE SOURCE SYSTEM AND INVITE IT TO SEND.
    10400   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    10500   *  Indicator 98 tells you whether the WRITE command completed
    10600   *  successfully.  (OFF means the command completed successfully.)
    10700   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    10800   *
             5
    10900  C                   WRITEITMREC               98                  2
    11000  C                   TIME          TIME    60       GET TIME SENT
    11100  C   98              MOVELFILERR,5 PART1   66
    11200  C   98              MOVE *BLANKS  PART2   66
    11300  C   98      MAJCOD  CABEQ'  '     ERROR            CK FOR ERROR
    11400  C   98      MAJCOD  CABGE'03'     ERROR            CK FOR ERROR
    11500  C                   EXCPTSENT                      PRINT SENT LOG
    11600  C                   GOTO NXTREC
    11700   *
    11800   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    11900   *   ERROR - PRINT OUT THE MAJOR/MINOR RETURN CODES.
    12000   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    12100   *
             6
    12200  C           ERROR     TAG
    12300  C   10
    12400  COR 98              EXCPTERRDSC                    ERROR DESCRIPTN
    12500  C                   EXCPTERR
    12600   *
    12700   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    12800   *   END-OF-JOB.  TURN ON THE 'LR' INDICATOR.
    12900   * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    13000   *
             7
    13100  C           END       TAG
    13200  C                     SETON                     LR                  1
5738RG1 V2R1M0  910524            IBM AS/400 RPG/400              ASYNLIBRPG/ASYNCT       12/14/90  09:49:02      Page       5
SEQUENCE                                                                          IND    DO    LAST       PAGE   PROGRAM
NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7...*  USE    NUM   UPDATE     LINE   ID
    13400  O* - - - - - - - - - - *
    13500   *  PRINT HEADINGS   *
    13600   * - - - - - - - - - - *
    13700  OQPRINT  H  301   1P
    13800  O       OR        OF
    13900  O                                22 'TARGET TRANSACTION LOG'
    14000  O                          UDATE Y   60
    14100  O                                110 'PAGE'
    14200  O                          PAGE  J  116
    14300  O                                130 'ASYNCT'
    14400   * - - - - - - - - - - - - - - - *
    14500   *  PRINT RECEIVED TRANSACTION *
    14600   * - - - - - - - - - - - - - - - *
    14700  O       EF 1               RECVD
    14800  O                                22 'ITEM NUMBER RECEIVED -'
    14900  O                          ITMNO     29
    15000  O                                 90 'TIME -'
    15100  O                          TIME      99 '0 :  :  '
    15200  O                                130 'ASYNCT'
    15300   * - - - - - - - - - - - - - - *
    15400   *  PRINT SENT TRANSACTION   *
    15500   * - - - - - - - - - - - - - - *
    15600  O       EF 2               SENT
    15700  O                                 16 'SENT TO SOURCE :'
    15800  O                                 25 'ITMDSC -'
    15900  O                          ITMDSC    51
    16000  O                                 60 'ITMQTY -'
    16100  O                          ITMQTYJ   69
    16200  O                                 90 'TIME -'
    16300  O                          TIME      99 '0 :  :  '
    16400  O                                130 'ASYNCT'
    16500   * - - - - - - - - - - - - - - *
    16600   *  PRINT ERROR DESCRIPTION  *
    16700   * - - - - - - - - - - - - - - *
    16800  O       EF 1               ERRDSC
    16900  O                          PART1     66
    17000  O                          PART2    132
```

Figure F-9 (Part 3 of 5). RPG/400 Inquiry Example — Target Program

```
    17100   * - - - - - - - - - - - - - - - - - *
    17200   *  PRINT MAJOR/MINOR RETURN CODES *
    17300   * - - - - - - - - - - - - - - - - - *
    17400  O       EF 2            ERR
    17500  O                                  24 'MAJOR/MINOR RETURN CODE:'
    17600  O                       MAJCOD     27
    17700  O                                  28 '/'
    17800  O                       MINCOD     30
    17900  O                                  40 'STATUS -'
    18000  O                       STS        46
    18100  O                                  52 'ID -'
    18200  O                       ID         63
    18300  O                                  80 'JOB ENDED'
    18400  O                                  90 'TIME -'
    18500  O                       TIME       99 '0 :  :  '
    18600  O                                 130 'ASYNCT'

   D000000   OUTPUT FIELDS FOR RECORD ITMREC FILE CMNFILT FORMAT ITMREC.
   D000001                           ITMDSC    25  CHAR   25
   D000002                           ITMQTY    30  ZONE  5,0
   E000000   OUTPUT FIELDS FOR RECORD INVITE FILE CMNFILT FORMAT INVITE.


           * * * * *  E N D  O F  S O U R C E  * * * * *

          A d d i t i o n a l  D i a g n o s t i c  M e s s a g e s


 5738RG1 V2R1M0  910524            IBM AS/400 RPG/400              ASYNLIBRPG/ASYNCT       12/14/90  09:49:02      Page       6
SEQUENCE                                                                                                  LAST
 NUMBER    *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8          UPDATE


                  C o m p i l e - T i m e  T a b l e s

 Table/Array  . . . . . :   FILERR

    18800  Error occurred when issuing the initial WRITE.  The ICF file may h
    18900  ave been created without the proper value for ACQPGMDEV parameter.
    19000  Error occurred when issuing the READ CMNFILT command.  ... ...
    19100  EOF is indicated.  (Has the CMNFILT been ACQUIREd?)
    19200  Error occurred when issuing the WRITE ITMREC command.

 Table/Array  . . . . . :   ITMERR

    19400  ITEM NUMBER MUST BE > 00.
    19500  ITEM NUMBER MUST BE < 21.

 Table/Array  . . . . . :   ITMDES

    19700  THERE HAS TO BE A FIRST
    19800  ANYONE WANT SECONDS?
    19900  THREE'S A CROWD
    20000  FOURTH DOWN
    20100  PLEAD THE FIFTH
    20200  SIXTH SENSE
    20300  SEVENTH HEAVEN
    20400  EIGHT IS ENOUGH
    20500  CAT WITH NINE LIVES
    20600  THE TEN COMMANDMENTS
    20700  THE 11TH OF MARCH
    20800  CHEAPER BY THE DOZEN
    20900  THE BAKER'S DOZEN
    21000  FOURTEENTH OF JUNE
    21100  FIFTEEN FRENCH HORNS
    21200  SWEET SIXTEEN AND ???
    21300  SOUR SEVENTEEN
    21400  GRADUATE AT EIGHTEEN
    21500  FRESHMAN AGAIN AT 19
    21600  THE 20TH CENTURY
```

Figure F-9 (Part 4 of 5). RPG/400 Inquiry Example — Target Program

```
Table/Array . . . . . :   ITMQNT

   21800  00951
   21900  00375
   22000  00200
   22100  01027
   22200  00056
   22300  00450
   22400  05798
   22500  07731
   22600  09843
   22700  76360
   22800  01259
   22900  08399
   23000  00087
   23100  12004
   23200  00038
   23300  03867
   23400  18503
   23500  87370
   23600  26589
   23700  58217
```

```
5738RG1 V2R1M0  910524            IBM AS/400 RPG/400              ASYNLIBRPG/ASYNCT       12/14/90  09:49:02      Page       7
                          F i n a l   S u m m a r y
No errors found in source program.
Program Source Totals:
   Records . . . . . :   237
   Specifications  . . . . . :   96
   Table Records . . . . . :   47
   Comments  . . . . . :   88
PRM has been called.
Program ASYNCT is placed in library ASYNLIBRPG. 00 highest Error-Severity-Code.
        * * * * *  E N D  O F  C O M P I L A T I O N  * * * * *
```

Figure F-9 (Part 5 of 5). RPG/400 Inquiry Example — Target Program

C/400 Program Examples

The C/400 source program starts a session with a remote location and issues an evoke function, with no invite, to start the target program. The source program sends item numbers to the target program and then waits 30 seconds (the value specified by the WAITRCD parameter on the CRTICFF command) to receive an acknowledgment from the target program indicating that the evoke function completed successfully. If the source program receives a major return code equal to or greater than 03, the program goes to end-of-job.

In the following sample programs, the source program sends an item number to the target program requesting item information. The target program then sends the item information (description and quantity) to the source program. The source program sends the value 99999 to the target program, to indicate end-of-transaction. At this point, both programs go to end-of-job.

C/400 Program Descriptions

The following information describes the structure of the example programs in Figure F-11 on page F-36 and Figure F-13 on page F-46. The reference numbers in the figures correspond to those in the descriptions.

C/400 Source Program: The following describes the C/400 inquiry program that runs on the local system.

Program Files: The C/400 source program uses the following files:

ASYNICF An ICF file used to send records to and receive records from the target program.

QPRINT An AS/400 printer file that is used to print records, both sent and received, as well as major and minor ICF return codes.

DDS Source: The DDS used in the ICF file is shown in Figure F-10 on page F-34. QSYSPRT is a program-described file and does not require DDS.

```
SEQNBR  *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8  Date

   100       A*********************************************************************
   200       A*                                                                   *
   300       A* ICF communications file used by the Source System to              *
   400       A*   send/receive interactively with the Target System.              *
   500       A*                                                                   *
   600       A*********************************************************************
   700       A*
   800       A          R ITMREC
   900       A            ITMDSC        25A
  1000       A            ITMQTY         5S
  1100       A*
  1200       A          R EVOKE
  1300       A                                    EVOKE(&LIB/&PGM)
  1400       A                                    SECURITY( 2 &USRPWD +
  1500       A                                              3 &USRID)
  1600       A* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
  1700       A*                                                                   *
  1800       A*  *  The data placed in USERID and USRPWD must correspond to      *
  1900       A*     a user profile and password, respectively, on the Target      *
  2000       A*     System.                                                       *
  2100       A*                                                                   *
  2200       A*  *  The user in USRID must have authority to the device          *
  2300       A*     object (device description) being used on the Target          *
  2400       A*     System, as well as to the program and library indicated       *
  2500       A*     in PGM and LIB, respectively.                                 *
  2600       A*                                                                   *
  2700       A* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
  2800       A            PGM           10A  P
  2900       A            LIB           10A  P
  3000       A            USRID         10A  P
  3100       A            USRPWD        10A  P
  3200       A*
  3300       A          R ITMREQ
  3400       A                                    INVITE
  3500       A            ITMNO          5A

                        * * * * *  E N D  O F  S O U R C E  * * * * *

5738SS1 V2R1M0  910524                Data Description              ASYNLIBC/ASYNICF        1/02/91 11:21:46       Page    2

                                      Expanded Source
                                                                                    Field       Buffer position
SEQNBR  *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8 length       Out     In
   800                  R ITMREC
   900                    ITMDSC        25A  B                                       25           1       1
  1000                    ITMQTY         5S 0B                                       5          26      26
  1200                  R EVOKE                 EVOKE(&LIB/&PGM) +
  1400                                          SECURITY( 2 &USRPWD 3 &USRID)
  2800                    PGM           10A  P                                      10           1
  2900                    LIB           10A  P                                      10          11
  3000                    USRID         10A  P                                      10          21
  3100                    USRPWD        10A  P                                      10          31
  3300                  R ITMREQ                INVITE
  3500                    ITMNO          5A  B                                       5           1       1

              * * * * *  E N D  O F  E X P A N D E D  S O U R C E  * * * * *

5738SS1 V2R1M0  910524                Data Description              ASYNLIBC/ASYNICF        1/02/91 11:21:46       Page    3

                                      Message Summary

  Total          Informational          Warning        Error       Severe
                     (0-9)               (10-19)       (20-29)     (30-99)
      0                0                    0             0           0

* CPC7301      00              Message . . . . :   File ASYNICF created in library ASYNLIBC.

                      * * * * *  E N D  O F  C O M P I L A T I O N  * * * * *
```

Figure F-10. DDS Source for ICF File ASYNICF, C/400 Source Program

ICF File Creation and Program Device Entry Definition: The following command is needed to create the ICF file:

```
CRTICFF     FILE(ASYNLIBC/ASYNICF)
            SRCFILE(ASYNLIBC/QDDSSRC)
            ACQPGMDEV(*NONE)
            MAXPGMDEV(2) WAITRCD(30)
```

The following commands are needed to define the program device entry and to direct the target program to the proper ICF file:

```
OVRICFDEVE PGMDEV(ICF00)
            RMTLOCNAME(*REQUESTER)

OVRICFF     FILE(ASYNICF)
            TOFILE(ASYNLIBC/ASYNICF)
```

Print File Creation and Definition: If the print file, QPRINT, does not already exist on the target system, it will need to be created before running the C/400 program example. The command needed to create the print file is:

```
CRTPRTF     FILE(ASYNLIBC/QPRINT)
            TEXT('Printer File for
                 Asynchronous program examples.')
            REPLACE(*NO)
```

The output queue, ASYNLIBC/ASYNC, is created by the command:

```
CRTOUTQ     OUTQ(ASYNLIBC/ASYNC)
            TEXT('Target System''s asynchronous
                 ASYNC output queue.')
```

The following command is needed to direct the printer output to the proper print file and output queue:

```
OVRPRTF     FILE(QPRINT) TOFILE(ASYNLIBC/QPRINT)
            OUTQ(ASYNLIBC/ASYNC)
```

Program Explanation: The following describes the C/400 source program.

1 This section declares the structure for the record formats in the ICF file. CMNFILR is the name of the ICF file used to send records to and receive records from the target program.

The files used in the program are opened at the beginning of the program and the ICF program device is explicitly acquired since the ACQPGMDEV parameter was specified as *NONE on the CRTICFF command.

2 This section declares the structures for writing the following information to the printer file, QPRINT:

- Item number
- Description
- Quantity
- Major/minor return code

3 The routines are prototyped so the compiler knows the type of value returned and the type of parameters passed.

4 This section opens the ICF file, ASYNICF, for input/output and the printer file, QPRINT, for output.

5 This section sets the input/output feedback area pointer, and the program device, ICF00, is explicitly acquired.

6 The EVOKE_TARGET routine is called to start the target program.

7 Twenty records are sent to the target system.

8 Item number 99999 is sent to signify (to the target program) an end to the transaction with the target system.

9 The CL program, ASYNCTCCL, is started and contains the following:

```
ADDLIBLE    LIB(ASYNLIBC)
OVRICFDEVE PGMDEV(ICF00) RMTLOCNAME(*REQUESTER)
OVRPRTF     FILE(QPRINT) TOFILE(ASYNLIBC/QPRINT)
            OUTQ(ASYNLIBC/ASYNC)
CALL        PGM(ASYNLIBC/ASYNCTC)
```

10 The program device is retrieved from the input/output feedback area.

* * * * * S O U R C E * * * * *

```
Line  STMT                                                                                    SEQNBR  INCNO
          *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
    1     |/*-------------------------------------------------------------------*/                    1
    2     |/* This is an ICF send/receive program that uses an array (itm#) to  */                    2
    3     |/* simulate the retrieving of an item number from a data file and    */                    3
    4     |/* then sends that item number to a target system in order to retrieve */                   4
    5     |/* an item description and a quantity from the target system's data file*/                  5
    6     |/* on a subsequent read to the ICF file.                            */                    6
    7     |/*-------------------------------------------------------------------*/                    7
    8     |                                                                                          8
    9     |#define NOERROR 0                          /* No error occured */                        9
   10     |#define ERROR 1                            /* An error occured */                        10
   11     |                                                                                         11
   12     |#include <recio.h>                         /* Record I/O header */                       12
   13     |#include <stdio.h>                         /* Standard I/O header */                     13
   14     |#include <stddef.h>                        /* Standard definitions */                    14
   15     |#include <stdlib.h>                        /* General utilities */                       15
   16     |#include <string.h>                        /* String handling utilities */               16
   17     |#include <xxfdbk.h>                        /* Feedback area structures */                17
   18     |#include <xxasio.h>                        /* Indicator area structure */                18
   19     |                                                                                         19
    1
   20     |/*-------------------------------------------------------------------*/                    20
   21     |/* Define the ICF file's structure.                                  */                    21
   22     |/*-------------------------------------------------------------------*/                    22
   23     |                                                                                         23
   24     |struct {                                                                                 24
   25     |    char itmdsc??(25??);                                                                 25
   26     |    char itmqty??(5??);                                                                  26
   27     |} itmrec_icf_i;                                                                          27
   28     |                                                                                         28
   29     |struct {                                                                                 29
   30     |    char pgm??(10??);          /* Program Name on Target System */                       30
   31     |    char lib??(10??);          /* Library Name on Target System */                       31
   32     |    char usr??(10??);          /* User-ID on Target System */                            32
   33     |    char pwd??(10??);          /* Password on Taget System */                            33
   34     |} evoke_icf_o;                                                                          34
   35     |                                                                                         35
   36     |struct {                                                                                 36
   37     |    char itmno??(5??);                                                                   37
   38     |} itmreq_icf_o;                                                                         38
   39     |                                                                                         39
    2
   40     |/*-------------------------------------------------------------------*/                    40
   41     |/* Define structures used to write to the print file.                */                    41
   42     |/*-------------------------------------------------------------------*/                    42
   43     |                                                                                         43
   44     |struct {                                                                                 44
   45     |    char filler1??(37??);                                                                45
   46     |    char filler2??(36??);                                                                46
   47     |} blank_line;                                                                            47
   48     |                                                                                         48
   49     |struct {                                                                                 49
   50     |    char report_type??(22??); /* Report Type (ie. Source Tranaction Log ) */            50
   51     |    char      spaces??(15??);                                                            51
   52     |    char  main_title??(27??); /* Main Title */                                           52
```



```
Line  STMT                                                                                    SEQNBR  INCNO
          *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
   53     |    char       filler??(9??);                                                            53
   54     |} heading_one;                                                                           54
   55     |                                                                                         55
   56     |struct {                                                                                 56
   57     |    char     spaces??(37??);                                                             57
   58     |    char  sub_title??(27??); /* Sub-title */                                            58
   59     |    char      filler??(9??);                                                             59
   60     |} heading_two;                                                                           60
   61     |                                                                                         61
   62     |struct {                                                                                 62
   63     |    char pgmnam??(10??);     /* Program Name */                                          63
   64     |    char sndmsg??(30??);     /* Send Message Title */                                    64
   65     |    char itmnum??(05??);     /* Item Number */                                           65
   66     |    char filler??(28??);                                                                 66
   67     |} send_message;                                                                          67
```

Figure F-11 (Part 1 of 8). C/400 Inquiry Example — Source Program

```
  68       |                                                                                                    |    68
  69       |struct {                                                                                            |    69
  70       |    char pgmnam??(10??);     /* Program Name */                                                     |    70
  71       |    char recmsg??(30??);     /* Received Message Title */                                           |    71
  72       |    char itmdsc??(25??);     /* Item Description */                                                 |    72
  73       |    char spaces??(03??);                                                                            |    73
  74       |    char itmqty??(05??);     /* Item Quantity */                                                    |    74
  75       |} receive_message;                                                                                  |    75
  76       |                                                                                                    |    76
  77       |struct {                                                                                            |    77
  78       |    char pgmnam??(10??);     /* Program Name */                                                     |    78
  79       |    char rtnmsg??(30??);     /* Return Code Message Title */                                        |    79
  80       |    char  major??(02??);     /* Major Code */                                                       |    80
  81       |    char  slash??(01??);     /* slash */                                                            |    81
  82       |    char  minor??(02??);     /* Minor Code */                                                       |    82
  83       |    char spaces??(05??);                                                                            |    83
  84       |    char lparen??(01??);     /* left parenthesis */                                                 |    84
  85       |    char pgmdev??(10??);     /* Program Device, from feedback area */                               |    85
  86       |    char rparen??(01??);     /* right parenthesis */                                                |    86
  87       |    char filler??(11??);                                                                            |    87
  88       |} return_code;                                                                                      |    88
  89       |                                                                                                    |    89
  90       |struct {                                                                                            |    90
  91       |    char endmsg??(37??);     /* Ending Message Title */                                             |    91
  92       |    char spaces??(21??);                                                                            |    92
  93       |    char  rsnhdg??(9??);     /* Reason Heading */                                                   |    93
  94       |    char reason??(06??);     /* Reason (Normal or ERROR) */                                         |    94
  95       |} ending_message;                                                                                   |    95
  96       |                                                                                                    |    96
  97       |/*-----------------------------------------------------------------------*/                         |    97
  98       |/* Declare the array that contains the item numbers to be sent to the   */                         |    98
  99       |/* target system.                                                        */                         |    99
 100       |/*-----------------------------------------------------------------------*/                         |   100
 101       |                                                                                                    |   101
 102       |static char *item??(20??) = {"00001", "00002", "00003", "00004", "00005",                           |   102
 103       |                             "00006", "00007", "00008", "00009", "00010",                           |   103
 104       |                             "00011", "00012", "00013", "00014", "00015",                           |   104
 105       |                             "00016", "00017", "00018", "00019", "00020"                            |   105
 106       |};                                                                                                  |   106

5738CX1 V2R1M0  910524   IBM SAA C/400          ASYNLIBC/ASYNCSC                RCH38321       01/03/91 14:21:05     Page     4

 Line  STMT                                                                                                    SEQNBR   INCNO
            *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.........
  107       |                                                                                                  |   107
     3
  108       |int evoke_target(_RFILE *);                                                                       |   108
  109       |int send_item_number(_RFILE *);                                                                   |   109
  110       |int get_item_info(_RFILE *);                                                                      |   110
  111       |int check_error(void);                                                                            |   111
  112       |int check_timeout(void);                                                                          |   112
  113       |void end_error(_RFILE *, _RFILE *, XXIOFB_T *);                                                   |   113
  114       |void initialize_print_fields(_RFILE *);                                                           |   114
  115       |void print_heading(_RFILE *);                                                                     |   115
  116       |void print_request(_RFILE *);                                                                     |   116
  117       |void print_received(_RFILE *);                                                                    |   117
  118       |void print_error(_RFILE *, XXIOFB_T *);                                                           |   118
  119       |                                                                                                  |   119
  120       |main()                                                                                            |   120
  121       |{                                                                                                 |   121
```

Figure F-11 (Part 2 of 8). C/400 Inquiry Example — Source Program

```
122       |    XXIOFB_T *comm_fdbk;                  /* Ptr to common I/O feedback */                |    122
123       |    _RFILE *icffptr;                      /* Ptr to ICF file */                           |    123
124       |    _RFILE *prtfptr;                      /* Ptr to print file */                         |    124
125       |    int i;                                                                                |    125
126       |                                                                                          |    126
127       |  /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */          |    127
128       |  /*  Open a binary file for input and output.  Writes (output) to   */                  |    128
129       |  /*  the file occur at the end of the file (append).               */                  |    129
130       |  /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */          |    130
   4
131     1 |    if ((icffptr = _Ropen("ASYNICF", "ar+")) == NULL)                                     |    131
132     2 |        exit(ERROR);                                                                      |    132
133       |                                                                                          |    133
134       |  /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */          |    134
135       |  /*  Create a binary file for writing or clear the existing file.   */                  |    135
136       |  /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */          |    136
137       |  /*  Open printer file:                                              */                  |    137
138       |  /*   Keyword parameter, type, must be "record".                     */                  |    138
139       |  /*   Keyword parameter, lrecl, is required for program-described   */                  |    139
140       |  /*        printer files.                                            */                  |    140
141       |  /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */          |    141
142     3 |    if ((prtfptr = _Ropen("QPRINT", "wr lrecl=132")) == NULL) {                           |    142
143     4 |        _Rclose(icffptr);                                                                 |    143
144     5 |        exit(ERROR);                                                                      |    144
145       |    }                                                                                     |    145
146       |                                                                                          |    146
147     6 |    initialize_print_fields(prtfptr);                                                     |    147
148       |                                                                                          |    148
   5
149     7 |    comm_fdbk = _Riofbk(icffptr);                                                         |    149
150     8 |    _Racquire(icffptr, "ICF00     ");                                                    |    150
151       |                                                                                          |    151
152     9 |    if (check_error() == ERROR) {                                                         |    152
153    10 |        end_error(icffptr, prtfptr, comm_fdbk);                                           |    153
154    11 |        exit(ERROR);                                                                      |    154
155       |    }                                                                                     |    155
156    12 |    print_heading(prtfptr);                                                               |    156
   6
157    13 |    if (evoke_target(icffptr) == ERROR) {                                                 |    157
158    14 |        end_error(icffptr, prtfptr, comm_fdbk);                                           |    158
159    15 |        exit(ERROR);                                                                      |    159
160       |    }                                                                                     |    160

5738CX1 V2R1M0  910524   IBM SAA C/400          ASYNLIBC/ASYNCSC                RCH38321       01/03/91 14:21:05     Page     5

Line  STMT                                                                                       SEQNBR   INCNO
          *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.........
   7
161    16 |    for (i = 0 ; i < 20 ; i++) {                                                        |    161
162    17 |        strncpy(itmreq_icf_o.itmno, item??(i??), 5);                                     |    162
163    18 |        if (send_item_number(icffptr) == NOERROR) {                                      |    163
164    19 |            print_request(prtfptr);                                                      |    164
165    20 |            if (get_item_info(icffptr) == NOERROR)                                       |    165
166    21 |                print_received(prtfptr);                                                 |    166
167       |            else {                                                                       |    167
168    22 |                end_error(icffptr, prtfptr, comm_fdbk);                                  |    168
169    23 |                exit(ERROR);                                                             |    169
170       |            }                                                                            |    170
171       |        }                                                                                |    171
172       |        else {                                                                           |    172
173    24 |            end_error(icffptr, prtfptr, comm_fdbk);                                      |    173
174    25 |            exit(ERROR);                                                                 |    174
175       |        }                                                                                |    175
176       |    }                                                                                    |    176
```

Figure F-11 (Part 3 of 8). C/400 Inquiry Example — Source Program

```
    8
177    26 |    strncpy(itmreq_icf_o.itmno, "99999", 5);                                                    |    177
178    27 |    if (send_item_number(icffptr) == NOERROR) {                                                |    178
179    28 |        print_request(prtfptr);                                                                |    179
180    29 |        strncpy(ending_message.reason, "Normal", 6);                                           |    180
181    30 |        _Rwrite (prtfptr, &blank_line, sizeof(blank_line));                                    |    181
182    31 |        _Rwrite (prtfptr, &ending_message, sizeof(ending_message));                            |    182
183    32 |        _Rclose(icffptr);                                                                      |    183
184    33 |        _Rclose(prtfptr);                                                                      |    184
185    34 |        exit(NOERROR);                                                                         |    185
186       |    }                                                                                          |    186
187       |    else {                                                                                     |    187
188    35 |        end_error(icffptr, prtfptr, comm_fdbk);                                                |    188
189    36 |        exit(ERROR);                                                                           |    189
190       |    }                                                                                          |    190
191       |}                                                                                              |    191
192       |                                                                                               |    192
193       |                                                                                               |    193
    9
194       |/*-------------------------------------------------------------------------*/                  |    194
195       |/* Evoke the target program on the target system identified by the      */                  |    195
196       |/* program named in "evoke_icf_o.pgm" and in the library named in       */                  |    196
197       |/* "evoke_icf_o.lib".  The user-id and password used to gain access     */                  |    197
198       |/* to the target system is contained in "evoke_icf_0.usr" and           */                  |    198
199       |/* "evoke_icf_o.pwd", respectively.  If an error occurs, this program   */                  |    199
200       |/* will end.                                                            */                  |    200
201       |/*-------------------------------------------------------------------------*/                  |    201
202       |                                                                                               |    202
203       |evoke_target(_RFILE *icffptr)                                                                  |    203
204       |{                                                                                              |    204
205    1  |    strncpy(evoke_icf_o.pgm, "ASYNCTCCL ", 10);                                                |    205
206    2  |    strncpy(evoke_icf_o.lib, "ASYNLIBC  ", 10);                                                |    206
207    3  |    strncpy(evoke_icf_o.usr, "ASYNCUSR  ", 10);                                                |    207
208    4  |    strncpy(evoke_icf_o.pwd, "ASYNCPWD  ", 10);                                                |    208
209       |                                                                                               |    209
210    5  |    _Rformat(icffptr, "EVOKE     ");                                                          |    210
211    6  |    _Rpgmdev(icffptr, "ICF00     ");                                                          |    211
212    7  |    _Rwrite (icffptr, &evoke_icf_o, sizeof(evoke_icf_o));                                      |    212
213    8  |    return(check_error());                                                                     |    213
214       |}                                                                                              |    214

5738CX1 V2R1M0  910524   IBM SAA C/400          ASYNLIBC/ASYNCSC                  RCH38321        01/03/91 14:21:05     Page      6

 Line  STMT                                                                                          SEQNBR   INCNO
          *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.........
215       |                                                                                               |    215
216       |                                                                                               |    216
217       |/*-------------------------------------------------------------------------*/                  |    217
218       |/* This routine processes an array to simulate the reading of a data    */                  |    218
219       |/* base or display file to get the item number to send to the target    */                  |    219
220       |/* system.  The write will send the item number (itmno) to the target   */                  |    220
221       |/* system and then invite it to send.                                   */                  |    221
222       |/*-------------------------------------------------------------------------*/                  |    222
223       |                                                                                               |    223
224       |send_item_number(_RFILE *icffptr)                                                              |    224
225       |{                                                                                              |    225
226    1  |    _Rformat(icffptr, "ITMREQ    ");                                                          |    226
227    2  |    _Rwrite(icffptr, &itmreq_icf_o, sizeof(itmreq_icf_o));                                     |    227
228    3  |    return(check_error());                                                                     |    228
229       |}                                                                                              |    229
230       |                                                                                               |    230
231       |                                                                                               |    231
```

Figure F-11 (Part 4 of 8). C/400 Inquiry Example — Source Program

```
232       |/*-------------------------------------------------------------------*/                    |  232
233       |/* This read-from-invited-devices is to receive the item description   */                    |  233
234       |/* and quantity.  Using a read-from-invited-devices (and the time value */                    |  234
235       |/* specified on the waitrcd parameter) prevents waiting indefinitely on */                    |  235
236       |/* the read statement.  The 'read' statement refers to the file name   */                    |  236
237       |/* 'cmnfils' to be able to do a 'read-from-invited-devices' which is   */                    |  237
238       |/* required toget the timer to interrupt the 'read'.  Since record-    */                    |  238
239       |/* identifying characters were not used for the formats in the file,   */                    |  239
240       |/* the 'itmrec' format was placed first in the file and will receive   */                    |  240
241       |/* the input from the 'read cmnfils' statement.                        */                    |  241
242       |/*-------------------------------------------------------------------*/                    |  242
243       |                                                                                          |  243
244       |get_item_info(_RFILE *icffptr)                                                            |  244
245       |{                                                                                         |  245
246     1 |    _Rreadindv(icffptr, &itmrec_icf_i, sizeof(itmrec_icf_i), __DFT);                      |  246
247     2 |    return(check_timeout());                                                              |  247
248       |}                                                                                         |  248
249       |                                                                                          |  249
250       |                                                                                          |  250
251       |/*-------------------------------------------------------------------*/                    |  251
252       |/* Check for terminating error.  If the major return code is greater   */                    |  252
253       |/* than or equal to 03, then the program ends.                         */                    |  253
254       |/*-------------------------------------------------------------------*/                    |  254
255       |                                                                                          |  255
256       |check_error()                                                                             |  256
257       |{                                                                                         |  257
258     1 |    if (strncmp(_Maj_Min_rc.major_rc, "03", 2) == -1)         /* Major < 03 */            |  258
259     2 |        return(NOERROR);                                                                  |  259
260       |    else                                                      /* Major >= 03 */           |  260
261     3 |        return(ERROR);                                                                    |  261
262       |}                                                                                         |  262
263       |                                                                                          |  263
264       |                                                                                          |  264
265       |/*-------------------------------------------------------------------*/                    |  265
266       |/* Check for timeout on read-from-invited-program-devices operation.   */                    |  266
267       |/* If a return code greater than or equal to 0310 was received, then   */                    |  267
268       |/* the program ends.                                                   */                    |  268
```

```
5738CX1 V2R1M0  910524   IBM SAA C/400          ASYNLIBC/ASYNCSC                RCH38321        01/03/91 14:21:05     Page     7

 Line  STMT                                                                                        SEQNBR   INCNO
          *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
  269       |/*-------------------------------------------------------------------*/                    |  269
  270       |                                                                                          |  270
  271       |check_timeout()                                                                           |  271
  272       |{                                                                                         |  272
  273     1 |    if (strncmp(_Maj_Min_rc.major_rc, "03", 2) == 1)          /* Major > 03 */            |  273
  274     2 |        return(ERROR);                                                                    |  274
  275       |    else                                                                                  |  275
  276     3 |       if (strncmp(_Maj_Min_rc.major_rc, "03", 2) == 0)       /* Major = 03 */            |  276
  277     4 |           if (strncmp(_Maj_Min_rc.minor_rc, "10", 2) == -1)  /* Minor < 10 */            |  277
  278     5 |               return(NOERROR);                                                          |  278
  279       |           else                                               /* Minor >= 10 */           |  279
  280     6 |               return(ERROR);                                                            |  280
  281       |       else                                                   /* Major < 03 */            |  281
  282     7 |           return(NOERROR);                                                              |  282
  283       |}                                                                                         |  283
```

Figure F-11 (Part 5 of 8). C/400 Inquiry Example — Source Program

```
284       |                                                                                                  |  284
285       |                                                                                                  |  285
286       |/*-------------------------------------------------------------------*/                           |  286
287       |/* Print error message, close the files, and end the program.       */                           |  287
288       |/*-------------------------------------------------------------------*/                           |  288
289       |                                                                                                  |  289
290       |void end_error(_RFILE *icffptr, _RFILE *prtfptr, XXIOFB_T *comm_fdbk)                             |  290
291       |{                                                                                                 |  291
292     1 |    print_error(prtfptr, comm_fdbk);                                                              |  292
293     2 |    strncpy(ending_message.reason, "ERROR!", 6);                                                  |  293
294     3 |    _Rwrite (prtfptr, &blank_line, sizeof(blank_line));                                           |  294
295     4 |    _Rwrite (prtfptr, &ending_message, sizeof(ending_message));                                   |  295
296     5 |    _Rclose(icffptr);                                                                             |  296
297     6 |    _Rclose(prtfptr);                                                                             |  297
298       |}                                                                                                 |  298
299       |                                                                                                  |  299
300       |void initialize_print_fields(_RFILE *prtfptr)                                                     |  300
301       |{                                                                                                 |  301
302     1 |    strncpy(blank_line.filler1, "                                      ", 37);                    |  302
303     2 |    strncpy(blank_line.filler2, blank_line.filler1,  36);                                         |  303
304       |                                                                                                  |  304
305     3 |    strncpy(heading_one.report_type, "Source Transaction Log", 22);                               |  305
306     4 |    strncpy(heading_one.main_title,  "   C/400 Program Example   ",  27);                        |  306
307     5 |    strncpy(heading_one.spaces, blank_line.filler1, 15);                                          |  307
308     6 |    strncpy(heading_one.filler, blank_line.filler1,  9);                                          |  308
309       |                                                                                                  |  309
310     7 |    strncpy(heading_two.sub_title,   "Asynchronous Communications",  27);                        |  310
311     8 |    strncpy(heading_two.spaces, blank_line.filler1, 37);                                          |  311
312     9 |    strncpy(heading_two.filler, blank_line.filler1,  9);                                          |  312
313       |                                                                                                  |  313
314    10 |    strncpy(send_message.pgmnam, "ASYNCSC - ", 10);                                               |  314
315    11 |    strncpy(send_message.sndmsg, "Item Number SENT to Target:   ", 30);                           |  315
316    12 |    strncpy(send_message.filler, blank_line.filler1, 28);                                         |  316
317    13 |    strncpy(send_message.itmnum, "-----",  5);                                                    |  317
318       |                                                                                                  |  318
319    14 |    strncpy(receive_message.pgmnam, send_message.pgmnam, 10);                                     |  319
320    15 |    strncpy(receive_message.recmsg, "Info. RECEIVED from Target:   ", 30);                        |  320
321    16 |    strncpy(receive_message.spaces, "   ", 3);                                                   |  321
322    17 |    strncpy(receive_message.itmdsc, "-------------------------", 25);                             |  322
```

```
5738CX1 V2R1M0  910524   IBM SAA C/400          ASYNLIBC/ASYNCSC                  RCH38321        01/03/91 14:21:05     Page     8

 Line  STMT                                                                                                SEQNBR   INCNO
          *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
  323    18 |    strncpy(receive_message.itmqty, "-----",  5);                                             |  323
  324       |                                                                                               |  324
  325    19 |    strncpy(return_code.pgmnam, send_message.pgmnam, 10);                                      |  325
  326    20 |    strncpy(return_code.rtnmsg, "** Major/Minor Return Code:   ", 30);                         |  326
  327    21 |    strncpy(return_code.slash,  "/", 1);                                                       |  327
  328    22 |    strncpy(return_code.spaces, "    ", 5);                                                    |  328
  329    23 |    strncpy(return_code.lparen, "(", 1);                                                       |  329
  330    24 |    strncpy(return_code.rparen, ")", 1);                                                       |  330
  331    25 |    strncpy(return_code.filler, " **       ", 11);                                             |  331
  332    26 |    strncpy(return_code.major,  "--", 2);                                                      |  332
  333    27 |    strncpy(return_code.minor,  "--", 2);                                                      |  333
  334    28 |    strncpy(return_code.pgmdev, "----------", 10);                                             |  334
  335       |                                                                                               |  335
  336    29 |    strncpy(ending_message.endmsg, "******* Source Program Ended *******", 36);                |  336
  337    30 |    strncpy(ending_message.rsnhdg, "Reason = ", 9);                                             |  337
  338    31 |    strncpy(ending_message.spaces, blank_line.filler1, 22);                                     |  338
  339    32 |    strncpy(ending_message.reason, "------", 6);                                                |  339
  340       |}                                                                                              |  340
```

Figure F-11 (Part 6 of 8). C/400 Inquiry Example — Source Program

```
341       |                                                                                                    |  341
342       |                                                                                                    |  342
343       |/*------------------------------------------------------------------------*/                        |  343
344       |/* Print heading to print file.                                          */                        |  344
345       |/*------------------------------------------------------------------------*/                        |  345
346       |                                                                                                    |  346
347       |void print_heading(_RFILE *prtfptr)                                                                 |  347
348       |{                                                                                                   |  348
349     1 |    _Rwrite (prtfptr, &heading_one, sizeof(heading_one));                                           |  349
350     2 |    _Rwrite (prtfptr, &heading_two, sizeof(heading_two));                                           |  350
351     3 |    _Rwrite (prtfptr, &blank_line,  sizeof(blank_line));                                            |  351
352       |}                                                                                                   |  352
353       |                                                                                                    |  353
354       |                                                                                                    |  354
355       |/*------------------------------------------------------------------------*/                        |  355
356       |/* Print request transaction to print file.                               */                        |  356
357       |/*------------------------------------------------------------------------*/                        |  357
358       |                                                                                                    |  358
359       |void print_request(_RFILE *prtfptr)                                                                 |  359
360       |{                                                                                                   |  360
361     1 |    strncpy(send_message.itmnum, itmreq_icf_o.itmno, 5);                                            |  361
362     2 |    _Rwrite (prtfptr, &send_message, sizeof(send_message));                                         |  362
363       |}                                                                                                   |  363
364       |                                                                                                    |  364
365       |                                                                                                    |  365
366       |/*------------------------------------------------------------------------*/                        |  366
367       |/* Print received transaction to print file.                              */                        |  367
368       |/*------------------------------------------------------------------------*/                        |  368
369       |                                                                                                    |  369
370       |void print_received(_RFILE *prtfptr)                                                                |  370
371       |{                                                                                                   |  371
372     1 |    strncpy(receive_message.itmdsc, itmrec_icf_i.itmdsc, 25);                                       |  372
373     2 |    strncpy(receive_message.itmqty, itmrec_icf_i.itmqty,  5);                                       |  373
374     3 |    _Rwrite (prtfptr, &receive_message, sizeof(receive_message));                                   |  374
375     4 |    _Rwrite (prtfptr, &blank_line,  sizeof(blank_line));                                            |  375
376       |}                                                                                                   |  376
```

```
5738CX1 V2R1M0  910524   IBM SAA C/400          ASYNLIBC/ASYNCSC                RCH38321        01/03/91 14:21:05     Page     9

 Line  STMT                                                                                                    SEQNBR   INCNO
          *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
  377       |                                                                                                  |  377
  378       |                                                                                                  |  378
  379       |/*------------------------------------------------------------------------*/                      |  379
  380       |/* Print error information to print file.                                */                      |  380
  381       |/*------------------------------------------------------------------------*/                      |  381
  382       |                                                                                                  |  382
  383       |void print_error(_RFILE *prtfptr, XXIOFB_T *comm_fdbk)                                            |  383
  384       |{                                                                                                 |  384
  385     1 |    strncpy(return_code.major, _Maj_Min_rc.major_rc, 2);                                          |  385
  386     2 |    strncpy(return_code.minor, _Maj_Min_rc.minor_rc, 2);                                          |  386
      10
  387     3 |    strncpy(return_code.pgmdev, comm_fdbk->dev_name, 10);                                         |  387
  388     4 |    _Rwrite (prtfptr, &return_code, sizeof(return_code));                                         |  388
  389       |}                                                                                                 |  389

                                        * * * * *  E N D  O F  S O U R C E  * * * * *
```

Figure F-11 (Part 7 of 8). C/400 Inquiry Example — Source Program

```
5738CX1 V2R1M0  910524   IBM SAA C/400          ASYNLIBC/ASYNCSC                RCH38321       01/03/91 14:21:05     Page    10

                                        *****  I N C L U D E S  *****

INCNO   Include Name                    Last change        Actual Include Name
    1   recio.h                         90/12/04 17:13:54  QCC/H/RECIO
    2   xxasio.h                        90/12/04 17:14:01  QCC/H/XXASIO
    3   stdio.h                         90/12/04 17:13:58  QCC/H/STDIO
    4   stddef.h                        90/12/04 17:13:56  QCC/H/STDDEF
    5   errno.h                         90/12/04 17:13:50  QCC/H/ERRNO
    6   signal.h                        90/12/04 17:13:55  QCC/H/SIGNAL
    7   ctype.h                         90/12/04 17:13:49  QCC/H/CTYPE
    8   stdarg.h                        90/12/04 17:13:56  QCC/H/STDARG
    9   xxfdbk.h                        90/12/04 17:14:04  QCC/H/XXFDBK
   10   stdio.h                         90/12/04 17:13:58  QCC/H/STDIO
   11   stddef.h                        90/12/04 17:13:56  QCC/H/STDDEF
   12   stdlib.h                        90/12/04 17:13:58  QCC/H/STDLIB
   13   string.h                        90/12/04 17:13:59  QCC/H/STRING
   14   xxfdbk.h                        90/12/04 17:14:04  QCC/H/XXFDBK
   15   xxasio.h                        90/12/04 17:14:01  QCC/H/XXASIO

                                    *****  E N D  O F  I N C L U D E S  *****

5738CX1 V2R1M0  910524   IBM SAA C/400          ASYNLIBC/ASYNCSC                RCH38321       01/03/91 14:21:05     Page    11

                                        *****  M E S S A G E  S U M M A R Y  *****

     Total     Info(0-4)          Warning(5-19)      Error(20-29)       Severe(30-39)      Terminal(40-99)
        0          0                  0                  0                  0                  0

                                *****  E N D  O F  M E S S A G E  S U M M A R Y  *****

5738CX1 V2R1M0  910524   IBM SAA C/400          ASYNLIBC/ASYNCSC                RCH38321       01/03/91 14:21:05     Page  12

 ROUTINE          BLOCK NUMBER  SCOPE  TYPE
 <MAIN>                  2      LOCAL  MAIN-PROGRAM
 QXXPGMDEV              88      LOCAL  PROCEDURE
 QXXACQUIRE             89      LOCAL  PROCEDURE
 QXXFORMAT              91      LOCAL  PROCEDURE
 QXXREADINVDEV          93      LOCAL  PROCEDURE
 __reads               117      LOCAL  PROCEDURE
 __rwrite              120      LOCAL  PROCEDURE
 __rfmt                127      LOCAL  PROCEDURE
 __strncmp             192      LOCAL  PROCEDURE
 __strncpy             194      LOCAL  PROCEDURE
 evoke_target          215      ENTRY   PROCEDURE
 send_item_number      216      ENTRY   PROCEDURE
 get_item_info         217      ENTRY   PROCEDURE
 check_error           218      ENTRY   PROCEDURE
 check_timeout         219      ENTRY   PROCEDURE
 end_error             220      ENTRY   PROCEDURE
 initialize_print_fields
                       221      ENTRY   PROCEDURE
 print_heading         222      ENTRY   PROCEDURE
 print_request         223      ENTRY   PROCEDURE
 print_received        224      ENTRY   PROCEDURE
 print_error           225      ENTRY   PROCEDURE
 main                  226      ENTRY   PROCEDURE

Program ASYNCSC was created in library ASYNLIBC.

                                        *****  E N D  O F  C O M P I L A T I O N  *****
```

Figure F-11 (Part 8 of 8). C/400 Inquiry Example — Source Program

C/400 Target Program: The following describes a C/400 asynchronous target program.

Program Files: The C/400 target program uses the following files:

CMNFILR An ICF file used to send records to and receive records from the source program.

QPRINT An AS/400 printer file that is used to print records, both sent and received, as well as major and minor ICF return codes.

DDS Source: Figure F-12 on page F-44 shows the DDS keywords that are used in the ICF file. (QSYSPRT is a program-described file; no DDS is required.)

```
                                Data Description Source

SEQNBR  *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8  Date

   100       A*********************************************************************
   200       A*                                                                   *
   300       A* ICF communications file used by the Target System to             *
   400       A*   send/receive interactively with the Source System.              *
   500       A*                                                                   *
   600       A*********************************************************************
   700       A*
   800       A          R ITMREQ
   900       A            ITMNO          5A
  1000       A*
  1100       A          R ITMREC
  1200       A                                    INVITE
  1300       A            ITMDSC        25A
  1400       A            ITMQTY         5S
  1500       A*
  1600       A          R INVITE
  1700       A                                    INVITE

                       * * * * *  E N D  O F  S O U R C E  * * * * *


5738SS1 V2R1M0  910524                Data Description              ASYNLIBC/CMNFILR        1/02/91 11:22:42        Page    2

                                      Expanded Source
                                                                                  Field       Buffer position
SEQNBR  *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8 length        Out     In
   800                   R ITMREQ
   900                     ITMNO          5A  B                                     5           1       1
  1100                   R ITMREC                    INVITE
  1300                     ITMDSC        25A  B                                   25           1       1
  1400                     ITMQTY         5S 0B                                    5          26      26
  1600                   R INVITE                    INVITE

                 * * * * *  E N D  O F  E X P A N D E D  S O U R C E  * * * * *


5738SS1 V2R1M0  910524                Data Description              ASYNLIBC/CMNFILR        1/02/91 11:22:42        Page    3

                                      Message Summary

    Total          Informational          Warning        Error       Severe
                       (0-9)              (10-19)       (20-29)      (30-99)
       0                 0                   0             0            0

* CPC7301      00              Message . . . . :   File CMNFILR created in library ASYNLIBC.

                       * * * * *  E N D  O F  C O M P I L A T I O N  * * * * *
```

Figure F-12. DDS Source for ICF File CMNFILR, C/400 Target Program

ICF File Creation and Program Device Entry Definition: The command needed to create the ICF file is:

```
CRTICFF     FILE(ASYNLIBC/CMNFILR)
            SRCFILE(ASYNLIBC/QDDSSRC)
            ACQPGMDEV(*NONE)
            MAXPGMDEV(2) WAITRCD(30)
```

The following commands are needed to define the program device entry and to direct the target program to the proper ICF file:

```
OVRICFDEVE PGMDEV(ICF00)
           RMTLOCNAME(*REQUESTER)

OVRICFF    FILE(CMNFILR)
           TOFILE(ASYNLIBC/CMNFILR)
```

Print File Creation and Definition: If the print file, QPRINT, does not already exist on the target system, it will need to be created before running the C/400 program example. The command needed to create the print file is:

```
CRTPRTF    FILE(ASYNLIBC/QPRINT)
           TEXT('Printer File for Asynchronous
                program examples.')
           REPLACE(*NO)
```

The output queue, ASYNLIBC/ASYNC, is created by the command:

```
CRTOUTQ    OUTQ(ASYNLIBC/ASYNC)
           TEXT('Target System''s asynchronous
                ASYNC output queue.')
```

The following command is needed to direct the printer output to the proper print file and output queue:

```
OVRPRTF    FILE(QPRINT) TOFILE(ASYNLIBC/QPRINT)
           OUTQ(ASYNLIBC/ASYNC)
```

Program Explanation: The following describes the structure of the program example shown in Figure F-13 on page F-46. The ICF file used in this example uses externally described data formats (DDS) defined by the user. The reference letters in the figure correspond to those in the following description.

1 This section declares the structure for the record formats in the ICF file. CMNFILR is the name of the ICF file used to send records to and receive records from the target program.

The files used in the program are opened at the beginning of the program and the ICF program device is explicitly acquired since the ACQPGMDEV parameter was specified as *NONE on the CRTICFF command.

2 This section declares the structures for writing the following information to the printer file, QPRINT:

- Item number
- Description
- Quantity
- Major/minor return code
- Program device

3 This section opens the ICF file, CMNFILR, for input/output and the printer file, QPRINT, for output.

4 This section sets the input/output feedback area pointer, and the program device, ICF00, is explicitly acquired.

5 The program device, ICF00, is invited.

6 Item numbers are read until an error occurs or the number 99999 is received, signifying the source program is finished with the transaction.

7 The program device is retrieved from the input/output feedback area.

```
5738CX1 V2R1M0  910524   IBM SAA C/400          ASYNLIBC/ASYNCTC                RCH38321       01/02/91 11:22:45     Page     2

                                        * * * * *  S O U R C E  * * * * *

 Line  STMT                                                                                                        SEQNBR   INCNO
          *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
    1     |/*-------------------------------------------------------------------*/                                    |     1
    2     |/* This is the target program that will perform the ICF receive/send   */                                |     2
    3     |/* function with the source system's program.  It will use arrays to   */                                |     3
    4     |/* simulate the retrieval of data from a data base file, using the     */                                |     4
    5     |/* 'itmno' sent from the source system to index into the arrays to     */                                |     5
    6     |/* retrieve the appropriate item description and quantity.            */                                |     6
    7     |/*-------------------------------------------------------------------*/                                    |     7
    8     |                                                                                                         |     8
    9     |#define NOERROR 0                          /* No error occured */                                       |     9
   10     |#define ERROR 1                            /* An error occured */                                       |    10
   11     |                                                                                                         |    11
   12     |#include <recio.h>                         /* Record I/O header */                                      |    12
   13     |#include <stdio.h>                         /* Standard I/O header */                                    |    13
   14     |#include <stddef.h>                        /* Standard definitions */                                   |    14
   15     |#include <stdlib.h>                        /* General utilities */                                      |    15
   16     |#include <string.h>                        /* String handling utilities */                              |    16
   17     |#include <xxfdbk.h>                        /* Feedback area structures */                               |    17
   18     |#include <xxasio.h>                        /* Indicator area structure */                               |    18
   19     |                                                                                                         |    19
    1
   20     |/*-------------------------------------------------------------------*/                                    |    20
   21     |/* Define the ICF file's structure.                                  */                                |    21
   22     |/*-------------------------------------------------------------------*/                                    |    22
   23     |                                                                                                         |    23
   24     |struct {                                                                                                 |    24
   25     |    char itmno??(5??);                                                                                   |    25
   26     |} itmreq_icf_i;                                                                                          |    26
   27     |                                                                                                         |    27
   28     |struct {                                                                                                 |    28
   29     |    char itmdsc??(25??);                                                                                 |    29
   30     |    char itmqty??(5??);                                                                                  |    30
   31     |} itmrec_icf_o;                                                                                          |    31
   32     |                                                                                                         |    32
    2
   33     |/*-------------------------------------------------------------------*/                                    |    33
   34     |/* Define structures used to write to the print file.                */                                |    34
   35     |/*-------------------------------------------------------------------*/                                    |    35
   36     |                                                                                                         |    36
   37     |struct {                                                                                                 |    37
   38     |    char filler1??(37??);                                                                                |    38
   39     |    char filler2??(36??);                                                                                |    39
   40     |} blank_line;                                                                                            |    40
   41     |                                                                                                         |    41
   42     |struct {                                                                                                 |    42
   43     |    char report_type??(22??); /* Report Type (ie. Target Tranaction Log ) */                            |    43
   44     |    char      spaces??(14??);                                                                            |    44
   45     |    char  main_title??(27??); /* Main Title */                                                           |    45
   46     |    char      filler??(10??);                                                                            |    46
   47     |} heading_one;                                                                                           |    47
   48     |                                                                                                         |    48
   49     |struct {                                                                                                 |    49
   50     |    char     spaces??(36??);                                                                             |    50
   51     |    char  sub_title??(27??); /* Sub-title */                                                           |    51
   52     |    char     filler??(10??);                                                                             |    52

5738CX1 V2R1M0  910524   IBM SAA C/400          ASYNLIBC/ASYNCTC                RCH38321       01/02/91 11:22:45     Page     3

 Line  STMT                                                                                                        SEQNBR   INCNO
          *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
   53     |} heading_two;                                                                                          |    53
   54     |                                                                                                         |    54
   55     |struct {                                                                                                 |    55
   56     |    char pgmnam??(10??);     /* Program Name */                                                          |    56
   57     |    char recmsg??(30??);     /* Received Message Title */                                                |    57
   58     |    char itmnum??(05??);     /* Item Number */                                                           |    58
   59     |    char filler??(28??);                                                                                 |    59
   60     |} receive_message;                                                                                       |    60
   61     |                                                                                                         |    61
```

Figure F-13 (Part 1 of 8). C/400 Inquiry Example — Target Program

```
   62       |struct {                                                                                    |    62
   63       |    char pgmnam??(10??);     /* Program Name */                                             |    63
   64       |    char sndmsg??(30??);     /* Send Message Title */                                       |    64
   65       |    char itmdsc??(25??);     /* Item Description */                                         |    65
   66       |    char spaces??(03??);                                                                    |    66
   67       |    char itmqty??(05??);     /* Item Quantity */                                            |    67
   68       |} send_message;                                                                             |    68
   69       |                                                                                            |    69
   70       |struct {                                                                                    |    70
   71       |    char pgmnam??(10??);     /* Program Name */                                             |    71
   72       |    char rtnmsg??(30??);     /* Return Code Message Title */                                |    72
   73       |    char  major??(02??);     /* Major Code */                                               |    73
   74       |    char  slash??(01??);     /* slash */                                                    |    74
   75       |    char  minor??(02??);     /* Minor Code */                                               |    75
   76       |    char spaces??(05??);                                                                    |    76
   77       |    char lparen??(01??);     /* left parenthesis */                                         |    77
   78       |    char pgmdev??(10??);     /* Program Device, from feedback area */                       |    78
   79       |    char rparen??(01??);     /* right parenthesis */                                        |    79
   80       |    char filler??(11??);                                                                    |    80
   81       |} return_code;                                                                              |    81
   82       |                                                                                            |    82
   83       |struct {                                                                                    |    83
   84       |    char endmsg??(36??);     /* Ending Message Title */                                     |    84
   85       |    char spaces??(20??);                                                                    |    85
   86       |    char rsnhdg??(11??);     /* Reason Heading */                                           |    86
   87       |    char reason??(06??);     /* Reason (Normal or ERROR) */                                 |    87
   88       |} ending_message;                                                                           |    88
   89       |                                                                                            |    89
   90       |/*------------------------------------------------------------------------*/               |    90
   91       |/* Declare array that contains the description of the items.            */               |    91
   92       |/*------------------------------------------------------------------------*/               |    92
   93       |                                                                                            |    93
   94       |static char *itmdes??(20??) = {                                                             |    94
   95       |         "There has be to a FIRST. ",                                                       |    95
   96       |         "Anyone for SECONDS?      ",                                                       |    96
   97       |         "THREE's a crowd.         ",                                                       |    97
   98       |         "FOURTH down ...          ",                                                       |    98
   99       |         "Plead the FIFTH.         ",                                                       |    99
  100       |         "SIXTH sense ...          ",                                                       |   100
  101       |         "SEVENTH Heaven ...       ",                                                       |   101
  102       |         "EIGHT is enough.         ",                                                       |   102
  103       |         "Cat with NINE lives ...  ",                                                       |   103
  104       |         "The TEN Commandments ... ",                                                       |   104
  105       |         "The ELEVENTH of March ...",                                                       |   105
  106       |         "Cheaper by the DOZEN ... ",                                                       |   106
```

```
5738CX1 V2R1M0  910524   IBM SAA C/400          ASYNLIBC/ASYNCTC                RCH38321        01/02/91 11:22:45     Page     4

 Line  STMT                                                                                                SEQNBR   INCNO
          *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.....
  107       |         "The BAKER'S DOZEN ...    ",                                                       |   107
  108       |         "FOURTEENTH of June ...   ",                                                       |   108
  109       |         "FIFTEEN French horns ... ",                                                       |   109
  110       |         "Sweet SIXTEEN and ???    ",                                                       |   110
  111       |         "Sour SEVENTEEN ...       ",                                                       |   111
  112       |         "Graduate at EIGHTEEN.    ",                                                       |   112
  113       |         "Freshman again at 19.    ",                                                       |   113
  114       |         "The TWENTIETH Century ...",                                                       |   114
  115       |};                                                                                          |   115
  116       |                                                                                            |   116
```

Figure F-13 (Part 2 of 8). C/400 Inquiry Example — Target Program

```
117       |/*--------------------------------------------------------------------*/                    |  117
118       |/* Declare array that contains the quantity of the relative item in the */                   |  118
119       |/* itmdes array.                                                      */                    |  119
120       |/*--------------------------------------------------------------------*/                    |  120
121       |                                                                                           |  121
122       |static char *itmqnt??(20??) = {"00951", "00375", "00200", "01027", "00056",                |  122
123       |                               "00450", "05798", "07731", "09843", "76360",                |  123
124       |                               "01259", "08399", "00087", "12004", "00038",                |  124
125       |                               "03867", "18503", "87370", "26589", "58217"                 |  125
126       |};                                                                                         |  126
127       |                                                                                           |  127
128       |int get_item_number(_RFILE *);                                                             |  128
129       |int send_item_info(_RFILE *);                                                              |  129
130       |int check_error(void);                                                                     |  130
131       |int check_timeout(void);                                                                   |  131
132       |void initialize_print_fields(_RFILE *);                                                    |  132
133       |void print_heading(_RFILE *);                                                              |  133
134       |void print_received(_RFILE *);                                                             |  134
135       |void print_sent(_RFILE *);                                                                 |  135
136       |void print_error(_RFILE *, XXIOFB_T *);                                                    |  136
137       |                                                                                           |  137
138       |main()                                                                                     |  138
139       |{                                                                                          |  139
140       |    XXIOFB_T *comm_fdbk;                  /* Ptr to common I/O feedback */                 |  140
141       |    _RFILE *icffptr;                      /* Ptr to ICF file */                            |  141
142       |    _RFILE *prtfptr;                      /* Ptr to print file */                          |  142
143       |                                                                                           |  143
144       |  /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */                    |  144
145       |  /*  Open a binary file for input and output.  Writes (output) to   */                    |  145
146       |  /*  the file occur at the end of the file (append).               */                    |  146
147       |  /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */                    |  147
    3
148     1 |    if ((icffptr = _Ropen("CMNFILR", "ar+")) == NULL)                                     |  148
149     2 |        exit(ERROR);                                                                       |  149
150       |                                                                                           |  150
151       |  /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */                    |  151
152       |  /*  Create a binary file for writing or clear the existing file.   */                    |  152
153       |  /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */                    |  153
154       |  /*  Open printer file:                                            */                    |  154
155       |  /*    Keyword parameter, type, must be "record".                  */                    |  155
156       |  /*    Keyword parameter, lrecl, is required for program-described */                    |  156
157       |  /*        printer files.                                          */                    |  157
158       |  /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */                    |  158
159     3 |    if ((prtfptr = _Ropen("QPRINT", "wr lrecl=132")) == NULL) {                            |  159
160     4 |        _Rclose(icffptr);                                                                  |  160
```

```
5738CX1 V2R1M0  910524   IBM SAA C/400          ASYNLIBC/ASYNCTC                RCH38321       01/02/91 11:22:45     Page     5

 Line  STMT                                                                                        SEQNBR   INCNO
          *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.........
161       5 |        exit(ERROR);                                                                   |  161
162         |    }                                                                                  |  162
163         |                                                                                       |  163
164       6 |    initialize_print_fields(prtfptr);                                                  |  164
165         |                                                                                       |  165
    4
166       7 |    comm_fdbk = _Riofbk(icffptr);                                                      |  166
167       8 |    _Racquire(icffptr, "ICF00     ");                                                  |  167
168       9 |    if (check_error() == ERROR) {                                                      |  168
169      10 |        print_error(prtfptr, comm_fdbk);                                               |  169
170      11 |        strncpy(ending_message.reason, "Error-", 6);                                   |  170
171      12 |        _Rwrite (prtfptr, &ending_message, sizeof(ending_message));                    |  171
172      13 |        _Rclose(icffptr);                                                              |  172
173      14 |        _Rclose(prtfptr);                                                              |  173
174      15 |        exit(ERROR);                                                                   |  174
175         |    }                                                                                  |  175
```

Figure F-13 (Part 3 of 8). C/400 Inquiry Example — Target Program

```
176    16 |    print_heading(prtfptr);                                                    |    176
    5
177    17 |    _Rformat(icffptr, "INVITE    ");                                          |    177
178    18 |    _Rpgmdev(icffptr, "ICF00     ");                                          |    178
179    19 |    _Rwrite (icffptr, NULL, 0);                                               |    179
180    20 |    if (check_error() == NOERROR)                                             |    180
    6
181    21 |        while (1) {                                                           |    181
182    22 |            if (get_item_number(icffptr) == ERROR) {                          |    182
183    23 |                print_error(prtfptr, comm_fdbk);                               |    183
184    24 |                break;                                                         |    184
185       |            }                                                                  |    185
186       |            else {                                                            |    186
187    25 |                print_received(prtfptr);                                       |    187
188       |                                                                               |    188
189       |                /* - - - - - - - - - - - - - - - - - - - - - - - - */         |    189
190       |                /* When the 'itmno' field has a value of '99999',  */         |    190
191       |                /* the source system has no more requests.         */         |    191
192       |                /* - - - - - - - - - - - - - - - - - - - - - - - - */         |    192
193    26 |                if (strncmp(itmreq_icf_i.itmno, "99999", 5) == 0) {            |    193
194    27 |                    strncpy(ending_message.reason, "Normal", 6);               |    194
195    28 |                    break;                                                     |    195
196       |                }                                                              |    196
197       |                else                                                           |    197
198    29 |                    if (send_item_info(icffptr) == ERROR) {                    |    198
199    30 |                        print_error(prtfptr, comm_fdbk);                       |    199
200    31 |                        strncpy(ending_message.reason, "ERROR!", 6);           |    200
201    32 |                        break;                                                 |    201
202       |                    }                                                          |    202
203       |                    else                                                       |    203
204    33 |                        print_sent(prtfptr);                                   |    204
205       |            }                                                                  |    205
206       |        }                                                                      |    206
207       |    else                                                                       |    207
208    34 |        print_error(prtfptr, comm_fdbk);                                       |    208
209       |                                                                               |    209
210    35 |    _Rwrite (prtfptr, &blank_line,     sizeof(blank_line));                   |    210
211    36 |    _Rwrite (prtfptr, &ending_message, sizeof(ending_message));                |    211
212    37 |    _Rclose(icffptr);                                                          |    212
213    38 |    _Rclose(prtfptr);                                                          |    213
214       |}                                                                              |    214
```

```
5738CX1 V2R1M0  910524   IBM SAA C/400          ASYNLIBC/ASYNCTC                RCH38321        01/02/91 11:22:45     Page     6

 Line  STMT                                                                                                    SEQNBR   INCNO
          *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
  215      |                                                                                                    |    215
  216      |                                                                                                    |    216
  217      |/*------------------------------------------------------------------------*/                        |    217
  218      |/* This read-from-program-devices is to receive the itmno from the      */                        |    218
  219      |/* source system.  Using a read-from-invited-devices (and the timer     */                        |    219
  220      |/* value specified on the waitrcd parameter of the ICF file) prevents   */                        |    220
  221      |/* waiting indefinately on the 'read' statement.  The read statement    */                        |    221
  222      |/* refers to the file name 'cmnfilr' to be able to do a 'read-from-     */                        |    222
  223      |/* invited-device' which is required to get the timer interrupt on the  */                        |    223
  224      |/* 'read'.  On successful completion, the 'read' will put the data into */                        |    224
  225      |/* the first format in the ICF file since no record-identifying codes   */                        |    225
  226      |/* were used for the formats on the input specifications.               */                        |    226
  227      |/*------------------------------------------------------------------------*/                        |    227
  228      |                                                                                                    |    228
```

Figure F-13 (Part 4 of 8). C/400 Inquiry Example — Target Program

```
229       |get_item_number(_RFILE *icffptr)                                                  |  229
230       |{                                                                                 |  230
231     1 |    _Rreadindv(icffptr, &itmreq_icf_i, sizeof(itmreq_icf_i), __DFT);              |  231
232     2 |    return(check_timeout());                                                      |  232
233       |}                                                                                 |  233
234       |                                                                                  |  234
235       |                                                                                  |  235
236       |/*------------------------------------------------------------------------*/      |  236
237       |/* This is the routine processes the arrays to retrieve the item         */      |  237
238       |/* description and quantity to be sent back to the source system.  The    */      |  238
239       |/* item number received is converted to an integer and then decremented */      |  239
240       |/* by one to get the correct position in the arrays.                      */      |  240
241       |/*------------------------------------------------------------------------*/      |  241
242       |                                                                                  |  242
243       |send_item_info(_RFILE *icffptr)                                                   |  243
244       |{                                                                                 |  244
245       |    int item_number;                                                              |  245
246       |                                                                                  |  246
247     1 |    item_number = atoi(itmreq_icf_i.itmno) - 1;                                   |  247
248     2 |    strncpy(itmrec_icf_o.itmdsc, itmdes??(item_number??), 25);                    |  248
249     3 |    strncpy(itmrec_icf_o.itmqty, itmqnt??(item_number??), 5);                     |  249
250     4 |    _Rformat(icffptr, "ITMREC    ");                                             |  250
251     5 |    _Rwrite (icffptr, &itmrec_icf_o, sizeof(itmrec_icf_o));                      |  251
252     6 |    return(check_error());                                                        |  252
253       |}                                                                                 |  253
254       |                                                                                  |  254
255       |                                                                                  |  255
256       |/*------------------------------------------------------------------------*/      |  256
257       |/* Check for terminating error.  If the major return code is greater    */      |  257
258       |/* than or equal to 03, then the program ends.                           */      |  258
259       |/*------------------------------------------------------------------------*/      |  259
260       |                                                                                  |  260
261       |check_error()                                                                     |  261
262       |{                                                                                 |  262
263     1 |    if (strncmp(_Maj_Min_rc.major_rc, "03", 2) == -1)                             |  263
264     2 |        return(NOERROR);                                                          |  264
265       |    else                                                                          |  265
266     3 |        return(ERROR);                                                            |  266
267       |}                                                                                 |  267
268       |                                                                                  |  268
```

```
5738CX1 V2R1M0  910524   IBM SAA C/400          ASYNLIBC/ASYNCTC                RCH38321        01/02/91 11:22:45     Page      7

 Line  STMT                                                                                                    SEQNBR   INCNO
          *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9........
 269       |                                                                                  |  269
 270       |/*------------------------------------------------------------------------*/      |  270
 271       |/* Check for timeout on read-from-invited-program-devices operation.    */      |  271
 272       |/* If a return code greater than or equal to 0310 was received, then    */      |  272
 273       |/* the program ends.                                                     */      |  273
 274       |/*------------------------------------------------------------------------*/      |  274
 275       |                                                                                  |  275
 276       |check_timeout()                                                                   |  276
 277       |{                                                                                 |  277
 278     1 |    if (strncmp(_Maj_Min_rc.major_rc, "03", 2) == 1)          /* Major > 03 */    |  278
 279     2 |        return(ERROR);                                                            |  279
 280       |    else                                                                          |  280
 281     3 |        if (strncmp(_Maj_Min_rc.major_rc, "03", 2) == 0)      /* Major = 03 */    |  281
 282     4 |            if (strncmp(_Maj_Min_rc.minor_rc, "10", 2) == -1) /* Minor < 10 */    |  282
 283     5 |                return(NOERROR);                                                  |  283
 284       |            else                                              /* Minor >= 03 */   |  284
 285     6 |                return(ERROR);                                                    |  285
 286     7 |    return(NOERROR);                                          /* Major < 03 */    |  286
 287       |}                                                                                 |  287
 288       |                                                                                  |  288
```

Figure F-13 (Part 5 of 8). C/400 Inquiry Example — Target Program

```
      |
      |/*------------------------------------------------------------------*/
      |/* Initialize the various print file fields.                        */
      |/*------------------------------------------------------------------*/
      |
      |void initialize_print_fields(_RFILE *prtfptr)
      |{
    1 |    strncpy(blank_line.filler1, "                                     ", 37);
    2 |    strncpy(blank_line.filler2, blank_line.filler1,  36);
      |
    3 |    strncpy(heading_one.report_type, "Target Transaction Log", 22);
    4 |    strncpy(heading_one.main_title, "   C/400 Program Example   ",  27);
    5 |    strncpy(heading_one.spaces, blank_line.filler1, 14);
    6 |    strncpy(heading_one.filler, blank_line.filler1, 10);
      |
    7 |    strncpy(heading_two.sub_title,   "Asynchronous Communications",  27);
    8 |    strncpy(heading_two.spaces, blank_line.filler1, 36);
    9 |    strncpy(heading_two.filler, blank_line.filler1, 10);
      |
   10 |    strncpy(receive_message.pgmnam, "ASYNCTC --", 10);
   11 |    strncpy(receive_message.recmsg, " Item RECEIVED from Source:   ", 30);
   12 |    strncpy(receive_message.filler, blank_line.filler1, 28);
   13 |    strncpy(receive_message.itmnum, "-----",  5);
      |
   14 |    strncpy(send_message.pgmnam, receive_message.pgmnam, 10);
   15 |    strncpy(send_message.sndmsg, " Item Info. SENT to Source:   ", 30);
   16 |    strncpy(send_message.spaces, "   ", 3);
   17 |    strncpy(send_message.itmdsc, "-------------------------", 25);
   18 |    strncpy(send_message.itmqty, "-----",  5);
      |
   19 |    strncpy(return_code.pgmnam, receive_message.pgmnam, 10);
   20 |    strncpy(return_code.rtnmsg, "** Major/Minor Return Code:   ", 30);
   21 |    strncpy(return_code.slash,  "/", 1);
   22 |    strncpy(return_code.spaces, "     ", 5);

5738CX1 V2R1M0  910524   IBM SAA C/400          ASYNLIBC/ASYNCTC                RCH38321        01/02/91 11:22:45     Page    8

 Line  STMT                                                                                                 SEQNBR   INCNO
        *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.......
   23 |    strncpy(return_code.lparen, "(", 1);
   24 |    strncpy(return_code.rparen, ")", 1);
   25 |    strncpy(return_code.filler, " **        ", 11);
   26 |    strncpy(return_code.major,  "--", 2);
   27 |    strncpy(return_code.minor,  "--", 2);
   28 |    strncpy(return_code.pgmdev, "----------", 10);
      |
   29 |    strncpy(ending_message.endmsg, "******* Target Program Ended *******", 36);
   30 |    strncpy(ending_message.rsnhdg, "* Reason = ", 11);
   31 |    strncpy(ending_message.spaces, blank_line.filler1, 20);
   32 |    strncpy(ending_message.reason, "------", 6);
      |}
      |
      |
      |/*------------------------------------------------------------------*/
      |/* Print heading to print file.                                     */
      |/*------------------------------------------------------------------*/
      |
      |void print_heading(_RFILE *prtfptr)
      |{
    1 |    _Rwrite (prtfptr, &heading_one, sizeof(heading_one));
    2 |    _Rwrite (prtfptr, &heading_two, sizeof(heading_two));
    3 |    _Rwrite (prtfptr, &blank_line,  sizeof(blank_line));
      |}
      |
      |
```

Figure F-13 (Part 6 of 8). C/400 Inquiry Example — Target Program

```
  349       |/*--------------------------------------------------------------------*/                    |   349
  350       |/* Print received transaction to print file.                         */                    |   350
  351       |/*--------------------------------------------------------------------*/                    |   351
  352       |                                                                                          |   352
  353       |void print_received(_RFILE *prtfptr)                                                      |   353
  354       |{                                                                                         |   354
  355     1 |    strncpy(receive_message.itmnum, itmreq_icf_i.itmno, 5);                               |   355
  356     2 |    _Rwrite (prtfptr, &receive_message, sizeof(receive_message));                         |   356
  357       |}                                                                                         |   357
  358       |                                                                                          |   358
  359       |                                                                                          |   359
  360       |/*--------------------------------------------------------------------*/                    |   360
  361       |/* Print sent transaction to print file.                             */                    |   361
  362       |/*--------------------------------------------------------------------*/                    |   362
  363       |                                                                                          |   363
  364       |void print_sent(_RFILE *prtfptr)                                                          |   364
  365       |{                                                                                         |   365
  366     1 |    strncpy(send_message.itmdsc, itmrec_icf_o.itmdsc, 25);                                |   366
  367     2 |    strncpy(send_message.itmqty, itmrec_icf_o.itmqty,  5);                                |   367
  368     3 |    _Rwrite (prtfptr, &send_message, sizeof(send_message));                               |   368
  369     4 |    _Rwrite (prtfptr, &blank_line,   sizeof(blank_line));                                 |   369
  370       |}                                                                                         |   370
  371       |                                                                                          |   371
  372       |                                                                                          |   372
  373       |/*--------------------------------------------------------------------*/                    |   373
  374       |/* Print error information to print file.                            */                    |   374
  375       |/*--------------------------------------------------------------------*/                    |   375
  376       |                                                                                          |   376

5738CX1 V2R1M0  910524   IBM SAA C/400          ASYNLIBC/ASYNCTC                RCH38321       01/02/91 11:22:45     Page      9

 Line  STMT                                                                                        SEQNBR   INCNO
          *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.........
  377       |void print_error(_RFILE *prtfptr, XXIOFB_T *comm_fdbk)                                  |   377
  378       |{                                                                                       |   378
  379     1 |    strncpy(return_code.major, _Maj_Min_rc.major_rc, 2);                                |   379
  380     2 |    strncpy(return_code.minor, _Maj_Min_rc.minor_rc, 2);                                |   380
    7
  381     3 |    strncpy(return_code.pgmdev, comm_fdbk->dev_name, 10);                               |   381
  382     4 |    _Rwrite (prtfptr, &return_code, sizeof(return_code));                               |   382
  383       |}                                                                                       |   383

                                  * * * * *  E N D  O F  S O U R C E  * * * * *

5738CX1 V2R1M0  910524   IBM SAA C/400          ASYNLIBC/ASYNCTC                RCH38321       01/02/91 11:22:45     Page     10

                                       * * * * *  I N C L U D E S  * * * * *

INCNO   Include Name                  Last change        Actual Include Name
    1   recio.h                       90/12/04 17:13:54  QCC/H/RECIO
    2   xxasio.h                      90/12/04 17:14:01  QCC/H/XXASIO
    3   stdio.h                       90/12/04 17:13:58  QCC/H/STDIO
    4   stddef.h                      90/12/04 17:13:56  QCC/H/STDDEF
    5   errno.h                       90/12/04 17:13:50  QCC/H/ERRNO
    6   signal.h                      90/12/04 17:13:55  QCC/H/SIGNAL
    7   ctype.h                       90/12/04 17:13:49  QCC/H/CTYPE
    8   stdarg.h                      90/12/04 17:13:56  QCC/H/STDARG
    9   xxfdbk.h                      90/12/04 17:14:04  QCC/H/XXFDBK
   10   stdio.h                       90/12/04 17:13:58  QCC/H/STDIO
   11   stddef.h                      90/12/04 17:13:56  QCC/H/STDDEF
   12   stdlib.h                      90/12/04 17:13:58  QCC/H/STDLIB
   13   string.h                      90/12/04 17:13:59  QCC/H/STRING
   14   xxfdbk.h                      90/12/04 17:14:04  QCC/H/XXFDBK
   15   xxasio.h                      90/12/04 17:14:01  QCC/H/XXASIO

                                  * * * * *  E N D  O F  I N C L U D E S  * * * * *
```

Figure F-13 (Part 7 of 8). C/400 Inquiry Example — Target Program

```
5738CX1 V2R1M0  910524   IBM SAA C/400          ASYNLIBC/ASYNCTC                RCH38321       01/02/91 11:22:45     Page    11

                                    *****  M E S S A G E   S U M M A R Y  *****

     Total     Info(0-4)          Warning(5-19)       Error(20-29)        Severe(30-39)       Terminal(40-99)
       0           0                   0                   0                   0                   0

                               *****  E N D   O F   M E S S A G E   S U M M A R Y  *****

5738CX1 V2R1M0  910524   IBM SAA C/400          ASYNLIBC/ASYNCTC                RCH38321       01/02/91 11:22:45     Page  12

 ROUTINE          BLOCK NUMBER  SCOPE  TYPE
 <MAIN>                2       LOCAL  MAIN-PROGRAM
 QXXPGMDEV            88       LOCAL  PROCEDURE
 QXXACQUIRE           89       LOCAL  PROCEDURE
 QXXFORMAT            91       LOCAL  PROCEDURE
 QXXREADINVDEV        93       LOCAL  PROCEDURE
 __reads             117       LOCAL  PROCEDURE
 __rwrite            120       LOCAL  PROCEDURE
 __rfmt              127       LOCAL  PROCEDURE
 __strncmp           192       LOCAL  PROCEDURE
 __strncpy           194       LOCAL  PROCEDURE
 get_item_number     214       ENTRY   PROCEDURE
 send_item_info      215       ENTRY   PROCEDURE
 check_error         216       ENTRY   PROCEDURE
 check_timeout       217       ENTRY   PROCEDURE
 initialize_print_fields
                     218       ENTRY   PROCEDURE
 print_heading       219       ENTRY   PROCEDURE
 print_received      220       ENTRY   PROCEDURE
 print_sent          221       ENTRY   PROCEDURE
 print_error         222       ENTRY   PROCEDURE
 main                223       ENTRY   PROCEDURE

Program ASYNCTC was created in library ASYNLIBC.

                                    *****  E N D   O F   C O M P I L A T I O N  *****
```

Figure F-13 (Part 8 of 8). C/400 Inquiry Example — Target Program

Bibliography

The following AS/400 manuals contain additional information you may need when you use the AS/400 asynchronous communications support. The manuals are listed with their full title and order number. When these manuals are referred to in this manual, a shortened version of the title is used.

- *Communications: Intersystem Communications Function Programmer's Guide*, SC41-9590. Contains information about DDS specific to communications applications. **Short Title**: *ICF Programmer's Guide*.
- *Communications: Management Guide*, SC41-0024. Supplies management information relating to communications, specific work management, communications error handling, and performance. **Short Title**: *Communications Management Guide*.
- *Communications: Operating System/400* Communications Configuration Reference*, SC41-0001. Contains general configuration information, including detailed descriptions of network interface, line, controller, device, mode, and class-of-service descriptions, configuration lists, and connection lists. **Short Title**: *OS/400* Communications Configuration Reference*.
- *Communications: X.25 Network Guide*, SC41-0005. Contains information about the X.25 network interface and how to use it on the OS/400 operating system. **Short Title**: *X.25 Network Guide*.
- *Data Description Specifications Reference*, SC41-9620. Contains information about coding data description specifications. **Short Title**: *DDS Reference*.
- *Programming: Concepts and Programmer's Guide for the System/36 Environment*, SC41-9663. Identifies the differences in the applications process in the System/36 environment on the AS/400 system. **Short Title**: *Concepts and Programmer's Guide for the System/36 Environment*.
- *Programming: Control Language Programmer's Guide*, SC41-8077. Contains general information about control language programming. **Short Title**: *CL Programmer's Guide*.
- *Programming: Control Language Reference*, SC41-0030. Contains descriptions of all AS/400 control language (CL) commands, including syntax diagrams. **Short Title**: *CL Reference*.

The following manuals contain information on how to design, code, compile, run, and debug programs written in the languages supported for AS/400 communications:

- *Languages: Systems Application Architecture* AD/Cycle* COBOL/400* Reference*, SC09-1380. **Short Title**: *COBOL/400* Reference*.
- *Languages: Systems Application Architecture* AD/Cycle* COBOL/400* User's Guide*, SC09-1383. **Short Title**: *COBOL/400* User's Guide*.
- *Languages: Systems Application Architecture* AD/Cycle* RPG/400* Reference*, SC09-1349. **Short Title**: *RPG/400* Reference*.
- *Languages: Systems Application Architecture* AD/Cycle* RPG/400* User's Guide*, SC09-1348. **Short Title**: *RPG/400* User's Guide*.
- *Languages: Systems Application Architecture* C/400* Reference Summary*, SX09-1217. **Short Title**: *C/400* Reference Summary*.
- *Languages: Systems Application Architecture* C/400* User's Guide*, SC09-1347. **Short Title**: *C/400* User's Guide*.
- *Languages: Systems Application Architecture* FORTRAN/400* Reference*, SC41-9844. **Short Title**: *FORTRAN/400* Reference*.
- *Languages: Systems Application Architecture* FORTRAN/400* User's Guide*, SC41-9845. **Short Title**: *FORTRAN/400* User's Guide*.

Index

D

E

F

G

I

L

M

N

O

S

T

V

W

X

Reader's Comments

Application System/400™
Communications:
Asynchronous Communications
Programmer's Guide
Version 2

Publication No. SC41-9592-00

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Be sure to print your name and address below if you would like a reply.

Name

Address

Company or Organization

Phone No.

Reader's Comments
SC41-9592-00

IBM®

Cut o
Alon

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN DEPT 245
IBM CORPORATION
3605 HWY 52 N
ROCHESTER MN 55901-7899

Fold and Tape | **Please do not staple** | Fold and Tape

Cut
Alor

Reader's Comments

Application System/400™
Communications:
Asynchronous Communications
Programmer's Guide
Version 2

Publication No. SC41-9592-00

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Be sure to print your name and address below if you would like a reply.

Name

Address

Company or Organization

Phone No.

Reader's Comments
SC41-9592-00

IBM®

Cut a
Alon

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN DEPT 245
IBM CORPORATION
3605 HWY 52 N
ROCHESTER MN 55901-7899

Fold and Tape | **Please do not staple** | Fold and Tape

Cut a
Alon

Program Number: 5738-SS1

Printed in Denmark by Rosendahl · Esbjerg

SC41-9592-00