SNA IMF Programmer's Reference Manual

HP 3000 MPE/iX Computer Systems

Edition 2

Manufacturing Part Number: 30293-90009
E0692

U.S.A. June 1992

Notice

The information contained in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability or fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for direct, indirect, special, incidental or consequential damages in connection with the furnishing or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

Restricted Rights Legend

Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013. Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19 (c) (1,2).

Acknowledgments

UNIX is a registered trademark of The Open Group.

Hewlett-Packard Company
3000 Hanover Street
Palo Alto, CA 94304 U.S.A.

Contents

Contents

Contents

4. Intrinsics Used with No-Wait I/O

Contents

Figures

Figures

Tables

Tables

Preface

This manual describes Hewlett-Packard's Systems Network Architecture Interactive Mainframe Facility (SNA IMF) for both the MPE V (SNA IMF/V) and MPE XL (SNA IMF/XL) operating systems. Systems Network Architecture (SNA) is IBM's specification for distributed data processing networks. This manual describes the features and uses of Hewlett-Packard's SNA IMF, which communicates in an SNA environment.

NOTE

In this manual, the term SNA IMF is used when the information being given is true for both SNA IMF/V and SNA IMF/XL. The terms SNA IMF/V and SNA IMF/XL are used when a distinction between the two systems is necessary.

MPE/iX, Multiprogramming Executive with Integrated POSIX, is the latest in a series of forward-compatible operating systems for the HP 3000 line of computers.

In HP documentation and in talking with HP 3000 users, you will encounter references to MPE XL, the direct predecessor of MPE/iX. MPE/iX is a superset of MPE XL. All programs written for MPE XL will run without change under MPE/iX. You can continue to use MPE XL system documentation, although it may not refer to features added to the operating system to support POSIX (for example, hierarchical directories).

Finally, you may encounter references to MPE V, which is the operating system for HP 3000s, not based on the PA-RISC architecture. MPE V software can be run on the PA-RISC (Series 900) HP 3000s in what is known as compatibility mode.

SNA IMF allows interactive and programmatic communications between an HP 3000 computer and an IBM host computer. SNA IMF emulates an IBM 3274 cluster controller, with attached 3278 display stations and 3287 printers, functioning as a Type 2 node in an SNA network.

This manual describes SNA IMF intrinsics, the programmatic interface to SNA IMF. SNA IMF intrinsics allow communication between applications on the HP 3000 and applications on the remote host.

For information on Pass Thru, SNA IMF's interactive interface, see *Using SNA IMF Pass Thru*.

For information on installing, configuring, managing, and troubleshooting SNA IMF, see the *SNA IMF/XL Node Manager's Guide* for SNA IMF/XL or *Installing and Troubleshooting SNA IMF* for SNA IMF/V.

Audience

This manual is for anyone involved in 3270-type data communications between an HP 3000 and an IBM host. It is for any of the following types of users:

- Application programmers, who develop applications that use SNA IMF intrinsics to communicate with applications running on a remote host.
- People responsible for training and supporting SNA IMF users.
- HP node managers or HP 3000 system managers responsible for HP 3000 data communications. The HP 3000 SNA node manager is responsible for overall HP-to-IBM data communications.

Some portions of this manual apply specifically to users of Asian terminals and printers with double-byte characters sets (DBCS).

Each of these audience types should be familiar with the pertinent operating characteristics of the host system in an SNA environment and have a working knowledge of Multiprogramming Executive (MPE), the operating system for the HP 3000.

Organization

This manual is divided into the following chapters and appendixes:

Chapter 1 , “Introducing SNA IMF,” gives an overview of SNA IMF, its capabilities, and its operating environment, including software and hardware requirements.

Chapter 2 , “Using SNA IMF Intrinsics,” explains how to use SNA IMF intrinsics to communicate programmatically with applications on a remote host.

Chapter 3 , “Intrinsics Used with Standard MPE I/O,” describes the SNA IMF intrinsics that are used with standard (wait) I/O.

Chapter 4 , “Intrinsics Used with No-Wait I/O,” explains how and when to use no-wait I/O, and it describes the SNA IMF intrinsics that are used with no-wait I/O.

Appendix A , “Intrinsic Result Codes,” Intrinsic Result Codes, lists all the result codes that can be returned by SNA IMF intrinsics, describes their probable causes, and recommends the actions you should take to resolve problems.

Appendix B , “SNA Character String (SCS) Support,” describes SCS control codes and support for LU.T1 printers.

Appendix C , “3270 Bit Assignment and Character Translation Tables,” provides bit assignments for 3270 field attribute characters and write control characters (WCC); lists the Attention ID codes generated by SNA IMF; gives the command codes for the IBM cluster controller; supplies 3270 buffer control orders; and discusses character sets, character translation tables, and Native Language Support.

Appendix D , “Differences Between IMF/3000 and SNA IMF/V,” compares the similarities and differences between the two IMF products. This appendix is useful if you are migrating from the IMF/3000 product to SNA IMF/V.

Appendix E , “HP and IBM Differences in DBCS Implementation,” describes how HP and IBM differ in their implementation of Asian Double-Byte Character Sets (DBCS).

Appendix F , “Sample Programs,” gives examples of programs that call SNA IMF intrinsics in transparent and non-transparent modes.

Appendix G , “Migrating Applications from SNA IMF/V to SNA IMF/XL,” describes the changes that must be made to an SNA IMF/V application before it will run on SNA IMF/XL.

Related Publications

You can find additional information about related topics in the following manuals:

- *Communicating With IBM*
- *Getting Started With SNA Node Management* (MPE V only)
- *SNA Link Services Reference Manual* (MPE V only)
- *SNA Link/XL Node Manager's Guide* (MPE XL only)
- *HP SNA Server/Access User's Guide*
- *Using the Node Management Services Utilities*
- *Installing and Troubleshooting SNA IMF* (MPE V only)
- *SNA IMF/XL Node Manager's Guide* (MPE XL only)
- *Using SNA IMF Pass Thru*
- *SNA IMF/XL Taiwanese User Support Guide*
- *SNA IMF/XL Japanese User Support Guide*
- *SNA IMF/XL Korean User Support Guide*
- *IMF/3000 User/Programmer Reference Manual* (for IMF/3000 on MPE V)
- *MPE V Intrinsics Reference Manual*
- *MPE V Commands Reference Manual*
- *MPE XL Intrinsics Reference Manual*
- *MPE XL Commands Reference Manual*
- *Native Language Support Reference Manual*
- *FORTRAN/3000 Reference Manual* (MPE V only)
- *FORTRAN 77/XL Reference Manual* (MPE XL only)
- *COBOL II/3000 Reference Manual* (MPE V only)
- *COBOL II Reference Manual* (MPE XL only)
- *HP Business BASIC Reference Manual* (MPE V only)
- *BASIC/XL Reference Manual* (MPE XL only)
- *Pascal/3000 Reference Manual* (MPE V only)
- *Pascal Reference Manual* (MPE XL only)
- *HP C/XL Reference Manual*
- *HP SNA Products: IBM Host System Programmer's Guides:*

— *HP SNA Products: Manager Guide*

— *HP SNA Products: ACF/NCP and ACF/VTAM Guide*

— *HP SNA Products: IMS Guide*

— *HP SNA Products: CICS Guide*

— *HP SNA Products: AS/400 Guide*

1 Introducing SNA IMF

This Chapter describes the SNA IMF/V and SNA IMF/XL products. It contains the following sections:

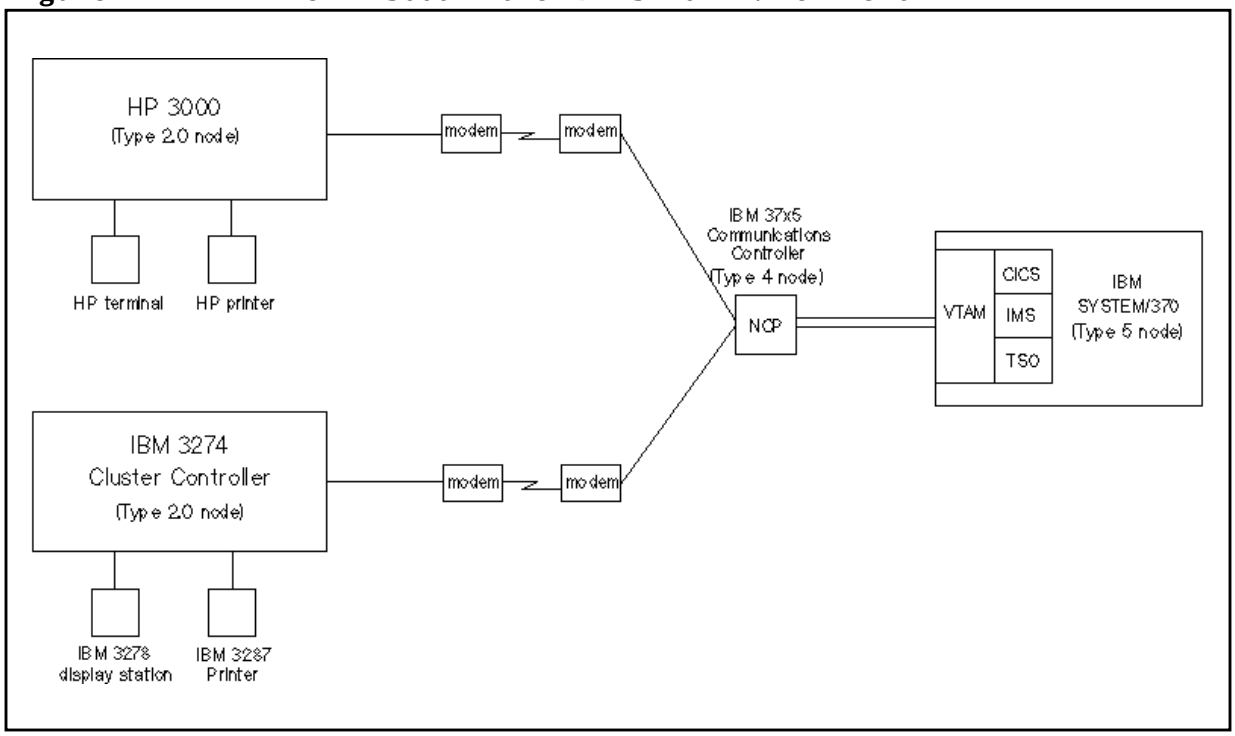
- Overview of SNA IMF — describes SNA IMF and its role in the SNA network.
- Programmatic Access and Pass Thru — describes the two modes in which you can use SNA IMF: programmatic mode and interactive (Pass Thru) mode.
- Hewlett-Packard's IMF Products — describes the three Interactive Mainframe Facility products available from Hewlett-Packard: SNA IMF/V, SNA IMF/XL, and IMF/3000.
- Features of SNA IMF — lists the capabilities and features available with SNA IMF.
- Features of Asian SNA IMF — lists the special features available for Asian users of SNA IMF, like Double-Byte Character Set support and localizable message and help text.
- IBM Host Applications — briefly describes the applications on the IBM host that you can access through SNA IMF.
- The Functional Layers of SNA — lists the architected layers of an SNA network.
- Structure of SNA IMF — describes the components of the SNA IMF and SNA link products.
- Operating Environment — lists the hardware and software required to run SNA IMF, the programming languages supported by SNA IMF, and the equipment you need to establish the data communications link with the IBM host.

Overview of SNA IMF

Systems Network Architecture Interactive Mainframe Facility (SNA IMF) is a Hewlett-Packard software product based on IBM's Systems Network Architecture (SNA). SNA IMF allows an HP 3000 to communicate with a remote host in an IBM 3270 environment. Using SNA IMF, the HP 3000 functions as a Type 2.0 node in an SNA network. Devices attached to the HP 3000 emulate the functions of IBM 3278 display stations and IBM 3287 printers.

The HP 3000 plays the same role in an SNA network that the IBM 3276 or 3274 Cluster Controller plays in a remote IBM environment. Figure 1-1 shows how the HP 3000 fits into the IBM 3270 environment.

Figure 1-1 The HP 3000 in the IBM 3270 Environment



SNA IMF supports the **base set** of IBM 3274 Cluster Controller functions. The following 3274 functions are not supported:

- Extended color
- Structured fields and attribute processing.
- APL/Text
- REQMS and RECFMS
- Operator-entry assist
- NetView support including the Alert function
- Magnetic-stripe readers and bar-code readers
- Operator type-ahead feature
- Dual session support

SNA IMF allows HP 3000 programs and attached devices to exchange data with application subsystems on an IBM host. These IBM application subsystems include the **Customer Information Control System (CICS**), **Information Management System (IMS)**, and **Time Sharing Option (TSO)**.

Programmatic Access and Pass Thru

SNA IMF supports two modes of communication between the HP 3000 and the host: **programmatic access mode** and **Pass Thru mode**.

SNA IMF's programmatic access mode consists of a set of subroutines called **intrinsics**. Programs on the HP 3000 call these intrinsics to establish communication and exchange data with application subsystems on the IBM host. SNA IMF intrinsics can be called from programs written in BASIC, COBOL, COBOL II, FORTRAN, Pascal, and SPL. In addition to these languages, SNA IMF/XL also supports C. An HP 3000 program using SNA IMF intrinsics appears to the IBM host as an IBM 3278 display station or an IBM 3287 printer.

Pass Thru mode is SNA IMF's interactive access mode. Through a program called Pass Thru, HP terminals and printers attached to your HP 3000 can look, to the IBM host, like IBM 3278 display stations or IBM 3287 printers attached to an IBM 3276 or 3274 Cluster Controller. The Pass Thru program calls the SNA IMF intrinsics that are used for programmatic access mode. You can use Pass Thru for direct access to IBM host application subsystems, without programming the HP 3000. Pass Thru helps you develop and debug application programs that call SNA IMF intrinsics.

Hewlett Packard's IMF Products

Hewlett-Packard offers three different IMF products: SNA IMF/V on MPE XL, and IMF/3000 on MPE V. SNA IMF/V and SNA IMF/XL use Synchronous Data Link Control (SDLC) protocol. IMF/3000, which runs only on MPE V, supports either Binary Synchronous Communications (BSC) or SDLC protocol. An HP 3000 running IMF/3000 functions as a Type 1 node.

You can migrate from IMF/3000 to SNA IMF/V (on MPE V) with minor modifications to your HP application programs. Appendix D , "Differences Between IMF/3000 and SNA IMF/V," of this manual describes the differences between IMF/3000 and SNA IMF/V.

NOTE IMF/3000 is currently supported only on MPE V.

This manual describes programmatic access mode for both SNA IMF/V and SNA IMF/XL. Pass Thru mode for SNA IMF/V and SNA IMF/XL is documented in *Using SNA IMF Pass Thru*. The IMF/3000 product is documented in the *IMF/3000 User/Programmer Reference Manual*.

NOTE In this manual, the term **SNA IMF** is used when the information being given is true for both SNA IMF/V and SNA IMF/XL. The terms **SNA IMF/V** and **SNA IMF/XL** are used when a distinction between the two products is necessary.

The term **IMF/3000** refers to the IMF product on MPE V that supports BSC protocol.

Features of SNA IMF

The SNA IMF product has the following features:

- Allows the HP 3000 to function as a cluster controller, or Type 2 node, in an IBM SNA network.
- Allows terminals and printers attached to the HP 3000 to emulate IBM 3278 display stations (LU.T2) and IBM 3287 printers (LU.T1 and LU.T3).
- Provides a set of callable routines, called intrinsics, that allow programs on an HP 3000 to exchange data with applications on a host computer.
- Allows interactive access to host applications from HP terminals and printers, using a program called Pass Thru.
- Shares the same data communications line with other SNA Services, such as SNA NRJE and LU 6.2 API.
- Allows the HP 3000 to share multipoint communication lines with other SNA systems such as the IBM AS/400, System/34, System/36, System/38, and the 8100 processors.
- Supports concurrent communication to multiple hosts or multiple lines to a single host (if you install multiple INP or PSI cards).
- Provides **Native Language Support (NLS)**, which allows application programmers to create local language applications for end users.
- Lets you migrate from IMF/3000 to SNA IMF/V with minor modification of your HP application programs (MPE V only).

Features of Asian SNA IMF

In addition to the features listed above, SNA IMF/XL has the following features for Asian users:

- Provides 16-bit Double-Byte Character Set (DBCS) support for several Asian countries: Japan, Taiwan, and Korea. SNA IMF/XL's DBCS feature will support 8-bit only, 16-bit only, or 8-bit and 16-bit mixed characters in SNA IMF/XL's data.
- Supports English language and the default system language.
- Allows HP Asian terminals and printers attached to the HP 3000 to emulate IBM's PS/55 Asian 3270 product, which supports LU.T1, LU.T2, and LU.T3.
- Allows 16-bit interactive and programmatic access to the Asian IBM host.
- Provides localizable error messages, online help text, and documentation in the country's native language.
- Supports DBCS characters (16-bit) with SNA IMF/XL's rolling softkeys.
- Provides 16-bit character mapping between HP (HP-15) and IBM (DBCS) data.

IBM Host Applications

Because SNA IMF supports interactive links between HP 3000s and IBM hosts, you can access transaction processing systems such as CICS and IMS, and interactive support programs such as TSO.

Transaction processing systems such as CICS are used for inquiry, inquiry with update capability, data entry, batch processing, and message switching applications.

IMS, another transaction processing system, can provide the following services: payroll and personnel file recording, manufacturing bill-of-materials control, inventory control, accounts receivable, and transaction processing.

Interactive support programs, such as TSO, provide interactive computing for large scale environments. TSO allows system programmers to maintain system libraries, catalogs, and procedure libraries. Application programmers use TSO to develop new applications and to maintain existing applications. TSO is used for batch and interactive communication and for data base/data communications (DB/DC).

Program librarians can use TSO to create, maintain, and control program libraries for development, support, and production. Through TSO, end users can access interactive programs, and problem solvers can use full operating system facilities.

The Functional Layers of SNA

An SNA network consists of a set of **Network Addressable Units (NAUs)** connected by a common path control network. The logical connection between NAUs is called a **session**. Each NAU is organized into functional layers. Each layer serves the next highest layer in its own node and relates to its peer layer on another node. Direct communication with another node occurs only at the lowest layer of a network. The functional layers of SNA implemented by the SNA IMF and SNA link products, beginning with the lowest level, are as follows:

- **Physical Control**, which sends and receives bits between nodes. It defines the mechanical and electrical interfaces and the bit-level data flow to the network.
- **Data Link Control**, which schedules and sends data across a link (physical connection) between two nodes and monitors errors that occur on the link.
- **Path Control,** which provides paths between end users (terminal operators, programs, or devices) and routes data between these end users.
- **Transmission Control**, which synchronizes and paces session-level data traffic, checks the sequence numbers of requests, and codes and decodes end user data.
- **Data Flow Control**, which monitors and controls the flow of data between two logically connected Network Addressable Units.
- **Presentation Services**, which formats data to be displayed or printed.
- **Application**, which provides services that directly support end users such as resource sharing, file transfers, remote file access, and data management on LU-LU sessions.

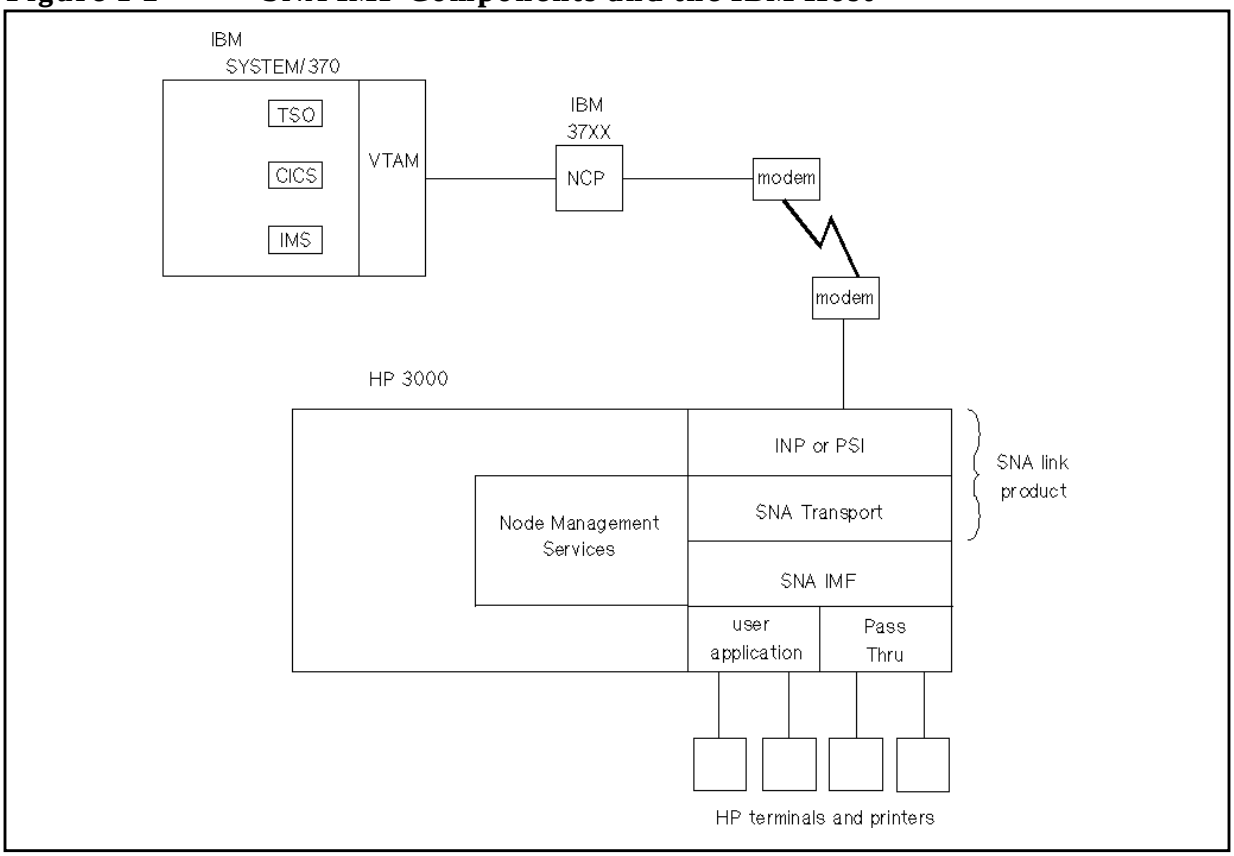
You can find introductory material about SNA and data communications in the *Communicating With IBM* primer and the *Getting Started with SNA Node Management* manual.

SNA IMF, along with the SNA link product, implements the seven functional layers of SNA.

Structure of SNA IMF

SNA IMF works with the SNA link product on the HP 3000. Figure 1-2 illustrates the components of SNA IMF on the HP 3000 and their relationship to the IBM host.

Figure 1-2 SNA IMF Components and the IBM Host



SNA IMF Product

SNA IMF is a software product that implements the upper three layers of SNA: the Data Flow Control, Presentation Services, and Application layers. In other documents you may see SNA IMF referred to as an **SNA Service**. An SNA Service is an HP data communications software product, like SNA IMF, SNA NRJE, or LU 6.2 API, which runs on top of the SNA link product.

SNA IMF stores data in the form of an internal screen image. The internal screen image contains the location and attributes of all the fields on the screen. It also contains any information that has been entered into the fields. Whenever an application or end user on the HP 3000 enters data to be transmitted to the IBM host, or whenever the IBM host sends data to the HP 3000, SNA IMF modifies the internal screen image.

You can run SNA IMF in **transparent mode** and **non-transparent mode**.

- In **transparent mode**, or **data stream mode**, you have access to the untranslated data stream. Untranslated data is in EBCDIC. SNA IMF stores the untranslated data stream in the internal screen image without interpreting any of the codes that indicate the cursor position, the location of fields on the screen, field attributes, and so on.
- In **non-transparent mode**, SNA IMF translates the data stream to ASCII and interprets all the codes that describe the appearance of a screen. In non-transparent mode, SNA IMF can locate the cursor, position all the fields on the screen, display the attributes of each field, and so on.

SNA Link Products

Hewlett-Packard offers three SNA link products: SNA Link/V, SNA/SDLC Link/XL, and SNA/X.25 Link/XL. SNA link products are bundled software and hardware products that permit a logical and physical connection from an HP 3000 into an SNA network. An SNA link product has two parts:

1. SNA Transport (software)
2. INP (MPE V hardware) or PSI (MPE XL hardware)

SNA Link/V has a third component: Node Management Services (software). On MPE XL, Node Management Services is part of the Fundamental Operating System (FOS) and is not bundled with SNA/SDLC Link/XL or SNA/X.25 Link/XL. The FOS is a collection of MPE programs, utilities, and subsystems bundled together for one price and supplied on a Master Installation Tape (MIT).

NOTE

In this manual, the term **SNA link product** is used when the information being given is true for all three SNA link products. The terms **SNA Link/V**, **SNA/SDLC Link/XL**, and **SNA/X.25 Link/XL** are used when a distinction between the SNA link products is necessary.

SNA Transport emulates an SNA Type 2 node. Through the Path Control and Transmission Control layers, it coordinates the communication sessions within the SNA network.

The hardware portion of the SNA link product is the **INP (Intelligent Network Processor)** card on MPE V, or the **PSI (Programmable Serial Interface)** card on MPE XL. The INP or PSI card implements the Physical Control and Data Link Control layers of SNA. It uses the SDLC protocol to control transmission over the communications line.

Node Management Services (NMS) is used by all the SNA Services (like SNA IMF and SNA NRJE) installed on the HP 3000. NMS handles

configuration, link and node level startup and shutdown, logging, tracing, and diagnostic functions.

In both Pass Thru mode and programmatic mode, SNA IMF intrinsics call SNA link intrinsics whenever data is sent to or received from the host. SNA IMF and the SNA link product together implement all architectural layers of SNA.

A separate INP or PSI card is required for each communications line from the HP 3000 to an IBM host. With multiple INP or PSI cards, you can connect the HP 3000 to multiple IBM hosts, or you can run multiple communications lines to a single IBM host.

The same INP or PSI can be used by multiple SNA Services.

The PSI card can also be used to run Network Services (NS). The same PSI card can be used for both NS and SNA communications, but NS and SNA cannot be run concurrently on the same PSI card.

NOTE The SNA link products are not supported as separate products independent of the SNA Services. Therefore, you must order and use SNA link products only with SNA Services.

Operating Environment

This section describes the hardware and software, on the IBM host and the HP 3000, required to run SNA IMF.

IBM Host Hardware Requirements

SNA IMF requires the following IBM host hardware:

- An IBM host can be any IBM System/370-compatible computer that supports the IBM 3270 family of terminals and printers, such as a System/370, 30xx, 43xx, or compatible processor.

  An IBM host can also be an IBM AS/400, System/36, or System/38.

NOTE When you configure an AS/400, System/36, or System/38 to communicate with SNA IMF, you must configure the device type for LU.T2 sessions as 3277, not 3278.

Also, bit 11 of the *flags* parameter of the OPEN3270 intrinsic (the UNBIND option) must be set to 0.

- An IBM 37xx Communications Controller that supports an IBM 3274 Cluster Controller with attached IBM 3278 display stations and IBM 3287 printers. The 37xx must support an SNA line.

IBM Host Software Requirements

SNA IMF requires the following IBM host software:

- Multiple Virtual Storage (MVS) or Disk Operating System/Virtual Storage Extended (DOS/VSE) operating system.
- Advanced Communications Function/Virtual Telecommunications Access Method (ACF/VTAM).
- Advanced Communications Function/Network Control Program (ACF/NCP).
- IBM 3270 host applications such as TSO, CICS, and IMS.

NOTE HP supports certain versions, releases, modifications, and Program Temporary Fix (PTF) levels of the above software. Your HP representative can verify whether SNA IMF can be supported with your particular configuration.

HP 3000 Hardware Requirements

SNA IMF requires the following HP 3000 hardware:

- For SNA IMF/XL: An HP 3000 Series 9xx computer system with a PSI card.

  For SNA IMF/V: an HP 3000 Series 37, 39, 40, 42, 44, 48, 58, 64, 68, or 70 computer system with an INP card and the following memory requirements: nine Segmented Library (SL) segments, including six Code Segment Table (CST) entries. Your HP representative can help you estimate your system memory requirements.

  The SNA IMF/V memory requirement is a superset of the IMF/3000 memory requirement. IMF/3000 requires four SL segments, including one CST entry. If you are migrating from IMF/3000 to SNA IMF/V, be sure to consider the SL segments and CST entries no longer required for IMF/3000 when calculating your SNA IMF/V memory requirement.

- The HP terminals and printers used for emulation are connected through one of the following:

  Datacommunications and Terminal Controller (DTC) (MPE XL only) or

  Asynchronous Data Communications Controller (ADCC) (MPE V only) or

  Asynchronous Terminal Processor (ATP) (MPE V only) or

  Multipoint Terminal Software (MTS) communication line (MPE V only) or

  X.25 Packet Assembler/Disassembler (PAD) or

  DS Network Services HP 3000 communications link (HP 3000 to HP 3000 only) or

  HP Network Services (NS 3000/V) communications link (HP 3000 to HP 3000 only).

NOTE For SNA IMF/XL, Pass Thru is supported only on terminals connected to a Datacommunications and Terminal Controller (DTC).

- A data communications line (switched or leased) between the HP 3000 and the host.
- A block mode terminal for Node Management Services. See the *SNA Link Services Reference Manual* or the *SNA Link/XL Node Manager's Guide* for descriptions of NMS screens produced by VPLUS.

HP 3000 Software Requirements

SNA IMF requires the following HP 3000 software:

- The HP 3000 Multiprogramming Executive (MPE V or MPE/XL) operating system.
- SNA IMF/V or SNA IMF/XL.
- SNA Link/V.

Language Support

Application programs that call SNA IMF intrinsics can be coded in the following languages:

- BASIC
- COBOL
- COBOL II
- FORTRAN
- Pascal
- SPL
- C (SNA IMF/XL only)

When using no-wait MPE I/O, application programs must be coded in COBOL II, FORTRAN, Pascal, SPL, or C. Some of the intrinsics used with no-wait I/O return a functional value, pass parameters by value, allow a variable number of parameters, and set condition codes. Neither COBOL nor BASIC has these capabilities.

NOTE On MPE XL, your application programs that call SNA IMF intrinsics may be in either native mode or compatibility mode. However, if your applications are written in BASIC or COBOL, the SNA IMF intrinsic calls in native mode are different from those in compatibility mode. See Chapter 2 , “Using SNA IMF Intrinsics,” for more information on native mode and compatibility mode.

The Communications Link

Data communications occurs over either a switched or a non-switched line. The line protocol is SDLC multipoint. Connection to the line is through a modem that you must obtain and install. You must ensure that modems at the HP 3000 end and at the IBM host end are compatible with each other and properly configured.

SNA IMF can use the same modem and INP or PSI as other SNA Services. Also, multiple services can run concurrently using the same communications line. The communications line between the HP 3000

and the host is configured on the mainframe end for a Type 2 node. You can find configuration information in the *HP SNA Products: Manager's Guide*.

Transmission can occur at speeds of up to 64K bps on MPE XL, or up to 56K bps on MPE V.

NOTE Only one SNA IMF control unit at a time may be configured to operate over a point-to-point line. However, multipoint lines can support multiple control units (one control unit per remote drop).

CAUTION Hewlett-Packard has not verified the SNA IMF product with all possible combinations of host system software releases. We are continually certifying with new host system software releases. Check with your Hewlett-Packard representative for the most up-to-date list.

Hewlett-Packard does not require that the customer run on one of the verified versions. However, not doing so applies limits to Hewlett-Packard's liability in addition to the normal limits. Because of the certification problems imposed by the host system versions, Hewlett-Packard must restrict its support policy in the following ways. The resolution of all problems will fall into one of the categories listed below:

- If a problem is caused by incorrect operation of the Hewlett-Packard product, Hewlett-Packard will repair the product.
- If a problem is caused by incorrect operation at the host, Hewlett-Packard may require that the user change to a known working version of the software or may elect to change the Hewlett-Packard product to conform to the situation.
- As always, Hewlett-Packard will in good faith attempt to solve the problem with the customer.

2 Using SNA IMF Intrinsics

This Chapter discusses SNA IMF's programmatic access mode, which allows your programs on the HP 3000 to communicate with programs on an IBM host. SNA IMF's programmatic access mode consists of a set of intrinsics, or subroutines, that you can call from HP 3000 programs written in the following languages: BASIC, COBOL, COBOL II, FORTRAN, Pascal, and SPL. In addition to these languages, SNA IMF/XL also supports C.

SNA IMF intrinsics allow your programs to read IBM 3270-type data stored in an internal screen image within the HP 3000. By communicating through intrinsics, your program looks to the IBM host like an IBM 3278 display station or IBM 3287 printer connected to a remote IBM cluster controller.

This Chapter contains the following sections:

- **Types of Intrinsics: Wait and No-Wait I/O** describes the two types of MPE I/O and lists the intrinsics used with each type.
- **Transparent and Non-Transparent Modes** describes the transparent and non-transparent modes of running SNA IMF.
- **Native Mode and Compatibility Mode** describes the two modes in which an application can be compiled on MPE XL. It explains the differences between the two modes when compiling an SNA IMF application.
- **Understanding Intrinsic Result Codes** explains how to handle the *result* parameter returned by SNA IMF intrinsics.
- **Understanding Host Screen Formats** explains how to handle the different screen formats when communicating with IBM host applications.
- **IBM 3278 Screen Layout** explains how the layout of the SNA IMF internal screen image differs from the layout of an IBM 3278 display station screen.
- **Intrinsic Calling Sequences** gives some example calling sequences for SNA IMF intrinsics.

Chapter 3 , "Intrinsics Used with Standard MPE I/O," and Chapter 4 , "Intrinsics Used with No-Wait I/O," describe the SNA IMF intrinsics in detail.

In addition to programmatic access mode, you can use **Pass Thru** mode to communicate interactively with an IBM host and its application programs. See *Using SNA IMF Pass Thru* for more information about Pass Thru.

NOTE On MPE XL, your application programs that call SNA IMF intrinsics may be in either native mode or compatibility mode. However, if your applications are written in BASIC or COBOL, the SNA IMF intrinsic calls in native mode are different from those in compatibility mode. See “Native Mode and Compatibility Mode,” later in this Chapter, for more information on native mode and compatibility mode.

Types of Intrinsics: Wait and No-Wait I/O

SNA IMF intrinsics can be divided into two categories: intrinsics used with standard (wait) MPE I/O, and intrinsics used with no-wait MPE I/O. With standard MPE I/O, a program that has issued an I/O request must suspend activity until the I/O request has finished. With no-wait I/O, or unblocked I/O, a program that has issued an I/O request may continue processing and issue additional I/O requests without waiting for previous I/O requests to finish. No-wait I/O overlaps the processing of multiple I/O requests, overlaps I/O with CPU processing, and responds to the completion of the first of several outstanding I/O requests.

Standard MPE Wait I/O

A program using standard MPE wait I/O must suspend activity after issuing an I/O request until the request has finished. It cannot issue another I/O request until the previous request has finished. Table 2-1 shows the intrinsics that can be used with standard (wait) MPE I/O. These intrinsics can also be used with no-wait MPE I/O.

Table 2-1 SNA IMF Intrinsics, Wait and No-Wait I/O

Intrinsic	Function
ACQUIRE3270	Allows your program to start Pass Thru on your own terminal or on any free terminal or printer that is not under MPE control.
ATTRLIST	Returns the locations of attribute characters within a specified subsection or within all subsections of a screen.
CLOSE3270	Emulates turning off a specified device.
ERR3270	Returns the error message associated with a given SNA IMF intrinsic error number.
EXTFIELDATTR	(for Asian users) Returns information about the attributes of a specified field, including the DBCS attribute.
FIELDATTR	Returns information about the attributes of a specified field.
OPEN3270	Emulates turning on the power of an emulated IBM 3278 display station or 3287 printer. The OPEN3270 intrinsic allocates the internal screen image used by SNA IMF.
PRINT3270	Produces a hard copy of the internal screen image.
READFIELD	Reads a field of data from the internal screen image and returns the data to an HP application program.
READSCREEN	Reads all or a specified part of the internal screen image and returns the data to an HP application program.

Table 2-1 SNA IMF Intrinsics, Wait and No-Wait I/O

Intrinsic	Function
READSTREAM	Reads all or part of the untranslated host data stream.
RECV3270	Allows your program to receive the screen after modification by the host. Also informs your program that the host has just sent data.
RESET3270	Emulates pressing the **[RESET]** key on an IBM 3278 display station keyboard.
SCREENATTR	Returns information about the attributes of the internal screen image.
STREAM3270	Emulates typing a series of keystrokes on an IBM 3278 display station keyboard. Allows you to do special function key operations and to update multiple data fields with one intrinsic call.
TRAN3270	Emulates pressing one of the transmit keys on the 3278 display station keyboard. TRAN3270 sends modified data to the host the next time the host polls the device.
VER3270	Returns the number of the version of SNA IMF that is being executed on the HP 3000.
WRITEFIELD	Writes data from the HP application program into an unprotected field of the internal screen image.
WRITESTREAM	Creates the data stream that an HP application program sends to the host.

No-Wait MPE I/O

No-wait MPE I/O requires MPE privileged mode (PM) capability. However, you do not need PM capability to write an SNA IMF application that uses no-wait I/O, because the SNA IMF intrinsics themselves have PM capability. When your application calls the SNA IMF OPEN3270 intrinsic with bit 15 of the *flags* parameter set to 1 (the no-wait I/O option), your program can call any of the SNA IMF intrinsics with no-wait I/O, whether or not you have PM capability.

If you do not have PM capability, SNA IMF intrinsics are the only intrinsics you can call with no-wait I/O. If you call, for example, the MPE FREAD intrinsic, your program will suspend processing until the I/O request has finished, even though you specified no-wait I/O in your OPEN3270 intrinsic call.

If your program will use no-wait MPE I/O, it must be written in COBOL II, FORTRAN, Pascal, C, or SPL. The IOWAIT and IODONTWAIT intrinsics, which are used with no-wait I/O, return a functional value, pass parameters by value, allow a variable number of parameters, and set condition codes. Neither COBOL nor BASIC has these capabilities.

Table 2-2 shows the intrinsics used only with no-wait MPE I/O. Two of these intrinsics, IOWAIT and IODONTWAIT, are actually MPE

intrinsics. You can call them directly using SNA IMF/V, but if you are using SNA IMF/XL, you must use the SNA IMF/XL intrinsics IOWAIT3270 and IODONTWAIT3270, which in turn call the MPE intrinsics IOWAIT and IODONTWAIT.

In addition to the intrinsics listed in Table 2-2, all of the intrinsics listed in Table 2-1 can be used with no-wait MPE I/O.

Table 2-2 SNA IMF Intrinsics, No-Wait I/O Only

Intrinsic	Function
ABORT3270	Aborts an outstanding no-wait RECV3270 or TRAN3270 request.
IODONTWAIT	(SNA IMF/V only) Informs a user program that a previous I/O operation has completed.
IODONTWAIT3270	(SNA IMF/XL only) Informs a user program that a previous I/O operation has completed.
IOWAIT	(SNA IMF/V only) Waits for a previous no-wait I/O request to complete.
IOWAIT3270	(SNA IMF/XL only) Waits for a previous no-wait I/O request to complete.

WARNING

Privileged mode bypasses the normal checks and restrictions that apply to standard users (those using blocked I/O). It is possible for a privileged mode program to destroy file integrity, including the MPE operating system. On request, Hewlett-Packard will investigate and attempt to resolve problems resulting from using privileged mode. Although this service is not available under the standard service contract, it is available on a time and materials basis. Hewlett-Packard does not support, under any type of contract, corrections or changes to the MPE operating system.

Transparent and Non-Transparent Modes

SNA IMF uses a data handling technique called the internal screen image. Whenever you call the OPEN3270 intrinsic, an internal screen image is created. The data stream is stored in the internal screen image, in SNA Character String (SCS) format for LU.T1 sessions, or in 3270 format for LU.T2 and LU.T3 sessions.

In **transparent mode**, or **data stream mode**, your application has access to the untranslated data stream. Untranslated data is in EBCDIC. In transparent mode, SNA IMF stores the untranslated data stream in the internal screen image without interpreting any of the codes that indicate the cursor position, the location of fields on the screen, field attributes, and so on.

In **non-transparent mode**, SNA IMF translates the data stream to ASCII and interprets all the codes that describe the appearance of a screen. In non-transparent mode, SNA IMF can locate the cursor, position all the fields on the screen, display the attributes of each field, and so on.

Transparent (Data Stream) Mode

The —transparent (data stream) mode of SNA IMF allows your application program to obtain the untranslated data stream in either no-wait or wait I/O. In transparent mode, SNA IMF does not interpret any of the control information in the data stream. It does not differentiate between the data contained in a field and the control information that describes the field and locates it on the screen. Therefore, in transparent mode, you cannot call non-transparent mode SNA IMF intrinsics that access individual fields in the internal screen image (such as READFIELD or WRITEFIELD).

In transparent mode, you obtain data from the host by calling RECV3270 and then READSTREAM. To send LU.T2 data to the host, call WRITESTREAM and then TRAN3270. To send LU.T1 data to the host, you need to call only TRAN3270. You cannot call READSTREAM and WRITESTREAM in non-transparent mode, because the original data stream is not preserved after it is translated.

Use transparent mode only for LU.T1 emulation or for communicating with host applications designed specifically for transferring files without translation. You must specify transparent mode for LU.T1 sessions, which communicate with host SCS applications.

To specify transparent mode, you open a device by calling the OPEN3270 intrinsic with bit 14 of the *flags* parameter set to 1 (the transparent mode option).

An option of the OPEN3270 intrinsic allows a printer to accept both LU.T1 and LU.T3 printer sessions. When you set up a printer that way, you can run your LU.T1 sessions in transparent mode and your LU.T3 sessions in non-transparent mode. See the description of the OPEN3270 intrinsic, in Chapter 3 , "Intrinsics Used with Standard MPE I/O," for more information.

Only the following intrinsics can be used in transparent mode:

ABORT3270	
CLOSE3270	
ERR3270	
IODONTWAIT	(SNA IMF/V only)
IODONTWAIT3270	(SNA IMF/XL only)
IOWAIT	(SNA IMF/V only)
IOWAIT3270	(SNA IMF/XL only)
OPEN3270	
READSTREAM	
RECV3270	
TRAN3270	
VERS3270	
WRITESTREAM	(for LU.T2 sessions only)

Binary data may be transmitted in transparent mode because the SDLC protocol allows transmission of transparent data. Because SNA IMF does not attempt to interpret the data, all possible values of data may be transmitted and received in transparent mode.

While in transparent mode, you are responsible for any character code translation. Use the Native Language Support (NLS) tables to perform the conversion. See the *Native Language Support Reference Manual* for more information.

When a file is transferred from the host system to the HP 3000, the entire file is sent in sequential blocks known as request units (RUs). Each LU.T2 or LU.T3 RU (or each RU chain, if chaining is used from the host contains a 3270 data stream write command. LU.T1 RUs contain SCS control codes instead of 3270 data stream commands.

The received data stream is passed unaltered through the SNA IMF software. The HP 3000 software transfers the data stream directly into the internal screen image untranslated. The LU.T2 and LU.T3 data returned to the user is a 3270 data stream consisting of a write command byte and transparent data. SNA IMF processes and removes

both the SNA controls and the SDLC protocol information before passing the received data to the user.

When transmitting data from the HP 3000 to the host system, the internal screen image used by the WRITESTREAM intrinsic need contain only the data to be transferred. The SNA IMF software partitions the data to be transmitted into RUs and adds the necessary SDLC and SNA protocol information. The SDLC protocol and SNA controls provide coordination between the host system and the HP 3000; however, you must establish the interprocess controls between the host application and the HP application. For example, you must identify the start or end of a file.

The AID and cursor address values specified in your intrinsic call are optionally inserted at the beginning of the first frame. An AID (attention identifier) is the numeric code indicating that you have pressed a key. The AID value and cursor address can be suppressed in the TRAN3270 call.

If AID and cursor suppression are not used, and you are sending a **CLEAR** or **PA** key during an LU.T2 session, call WRITESTREAM with an **outbuflen** of zero before calling TRAN3270. In transparent mode, SNA IMF will not check the **AID** key to determine whether it is a **CLEAR** or **PA** key. In non-transparent mode, no data is sent with the **CLEAR** and **PA** keys.

It is important to note that the lack of a true screen image invalidates using the host READ BUFFER and READ MODIFIED commands. These commands will not be executed by SNA IMF. If SNA IMF receives a READ BUFFER or READ MODIFIED command from the host, it will reject the command with sense code X'1003' (unsupported function). Thus, for the host to obtain data from a transparent mode device, this device must have a TRAN3270 intrinsic request outstanding. Also, the host must be indicating that it is ready to receive data. All other commands are accepted and passed on to your program along with any associated data. SNA IMF performs no other actions that normally might be associated with a specific command.

The size of the internal screen image determines the amount of data that can be transmitted or received. For LU.T2 and LU.T3 sessions, a screen size of 480, 1920, or 3440 characters will have a buffer limit of 540, 2160, and 3870 bytes, respectively. If data received from the host system is larger than the buffer limit, the buffer is filled to the limit and the rest of the data is lost. When the user issues the next READSTREAM call, a result code of 52 will be returned, indicating that the entire data stream could not be received. For LU.T1 sessions, SNA IMF can support an internal screen image of only 3870 bytes and an RU size of only 3584 bytes. If you exceed this RU limit, the internal screen image may be overwritten if the host application sends a split order. A split order occurs when SCS control codes or parameters span RU boundaries within an RU chain.

If a host writes to a transparent mode device that already has data in its internal screen image, SNA IMF will reject the data with sense code X'082D' (device busy). When you indicate that you are ready for more data by issuing another RECV3270 call, SNA IMF sends an LUSTAT to the host, indicating that the device is now available (X'00010000' for LU.T1 and LU.T3, X'0001D000' for LU.T2). This intervention prevents the host from writing over a data stream not yet obtained by the user.

Non-Transparent Mode

In non-transparent mode, the data stream is translated into the image of a terminal screen. Once the data is translated, SNA IMF can locate the cursor on the screen, locate each field on the screen, and identify the attributes of each field. Data that has been entered into the fields can be differentiated from control information in the data stream that locates and describes the fields on the screen.

In non-transparent mode, the internal screen image is maintained in ASCII character code, and the data is converted to EBCDIC character code only before being transmitted to the host system. Data received in EBCDIC character code is translated to ASCII character code before being placed in the internal screen image.

You cannot call READSTREAM and WRITESTREAM in non-transparent mode, because the original data stream is not preserved after it is translated.

Native Mode and Compatibility Mode

Native mode is the normal processing mode of the MPE XL operating system. Native mode applications are those that have been compiled into the native instruction set of the HP 3000 Series 900 computer.

Compatibility mode is a processing mode on HP 3000 Series 900 computers that allows MPE V applications to run on MPE XL machines without changes or recompilation.

Applications in native mode run faster than applications in compatibility mode. If you are running SNA IMF/XL, use the native mode SNA IMF intrinsic calls, and compile your SNA IMF applications in native mode for better performance.

Native mode intrinsic calls differ from compatibility mode intrinsic calls only in BASIC and COBOL. In FORTRAN, SPL, Pascal, and C, native mode intrinsic calls are identical to compatibility mode intrinsic calls.

In COBOL, native mode intrinsic calls differ from compatibility mode intrinsic calls in two ways:

1. The name of the intrinsic in a compatibility mode intrinsic call begins with a “C” for COBOL. The names of intrinsics in native mode have no initial “C.”
2. The word “INTRINSIC” appears in every COBOL intrinsic call in native mode. Compatibility mode intrinsic calls do not contain the word “INTRINSIC.”

In BASIC, native mode intrinsic calls differ from compatibility mode intrinsic calls in only one way: The name of the intrinsic in a compatibility mode intrinsic call begins with a “B” for BASIC. The names of intrinsics in native mode have no initial “B.”

The “Summary of Intrinsic Calls,” at the end of Chapter 3 , “Intrinsics Used with Standard MPE I/O,” lists all the intrinsic calls for native mode and compatibility mode.

For information on migrating applications from compatibility mode to native mode, see the *Migration Process Guide* and *Introduction to MPE XL for MPE V Programmers: Skills Migration Guide*.

Understanding Intrinsic Result Codes

All the SNA IMF intrinsics except VERS3270 return a code in the *result* parameter. Your program should always check the *result* parameter after it calls an intrinsic. A *result* parameter value of zero indicates that the intrinsic executed successfully.

You might want to check for specific non-zero *result* values after certain intrinsic calls. The *result* values that each intrinsic can return are listed in the intrinsic descriptions in Chapter 3 , “Intrinsics Used with Standard MPE I/O,” and Chapter 4 , “Intrinsics Used with No-Wait I/O,” If any non-zero *result* values are important to your program, check for them after the *result* parameter is returned.

For example, the RECV3270 intrinsic returns *result = 24* if the receive timer expires before the host sends data. If your program calls RECV3270 and receives *result = 24*, it knows that the internal screen image has not been updated, and there is no new data from the host. Your program should specifically check whether *result = 24* after a call to RECV3270 before it calls any intrinsics that read data from the internal screen image.

If the *result* parameter returns a value that has no meaning at the current point in your program, regard it as an error and branch to your error handling routine.

See Appendix F , “Sample Programs,” for examples of error checking and error handling using the *result* parameter.

Understanding Host Screen Formats

When you write an HP 3000 program designed to communicate with a host application that will not be modified, you need to know what the host screen formats are and how the host program works. You must know when the host program requires input, when it sends data, and how to determine when the host has finished sending a particular internal screen image. You should also know if a single screen of data will be transmitted in more than one data transmission.

You can use the automatic print feature in Pass Thru to show when and how the host program writes to your screen. Start Pass Thru specifying *format = 3* in the info string of the RUN command, and manually access your host program. Your internal screen image will be printed after every call to either the TRAN3270 or the RECV3270 intrinsic. The printed screen images show you the screens sent by the host, with attribute and null characters differentiated from blank characters. If the host sends a single screen in more than one transmission, you will be able to determine that from the printed images. See *Using SNA IMF Pass Thru* for more information on Pass Thru and its automatic print feature.

Because your HP program emulates a 3278 display station or 3287 printer, you must design your program to respond to the host program whenever it modifies your internal screen image. Your program must know what the host system will do in a particular situation. If the host may do one of several things at a particular time, your program must be written to handle variable situations.

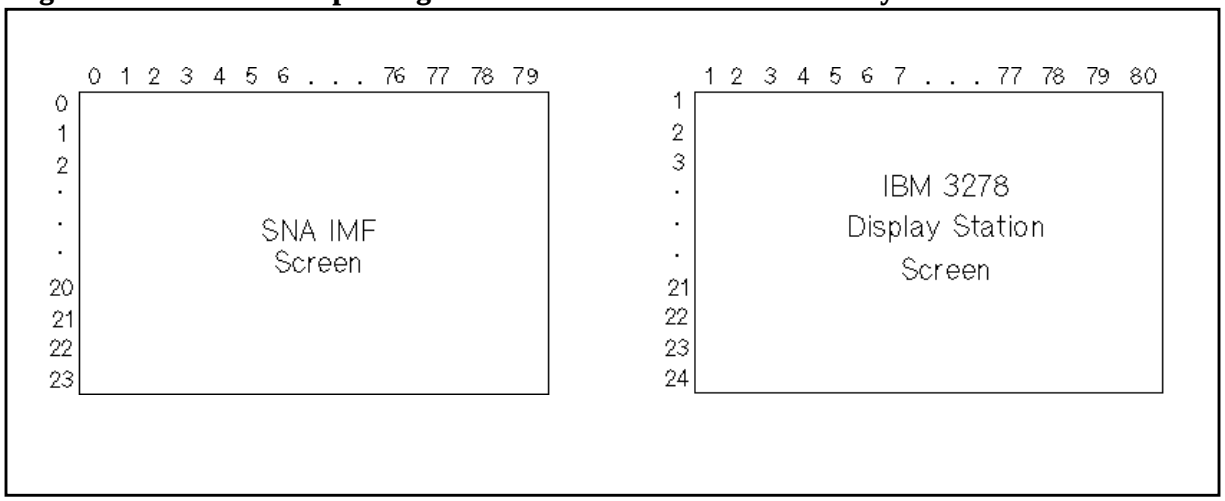
When you can modify existing host applications, or when you can write local and host applications in tandem, you have the opportunity to write simpler programs. Screens do not need operator instructions or even much structure. In fact, you can use an unformatted screen with no fields at all. Unformatted screens are often more efficient for new applications. You can write the programs to suit your local needs, possibly with the HP 3000 application controlling the host program.

IBM 3278 Screen Layout

On an IBM 3278 display station, data fields extend horizontally across the screen and are separated from one another by an attribute character. Data fields may occupy one or more contiguous lines on the screen. SNA IMF intrinsics read a screen one row at a time, beginning at the upper left corner and moving along each row from left to right, looking for a field attribute character. The first field attribute character found causes the following field to be labeled field number one; the field following the next attribute character is labeled field number two, and so on as SNA IMF scans down the screen. If an attribute character is not in the very first character position on the screen, then the first characters of data belong to the last field on the screen; that is, the last field wraps the screen.

SNA IMF intrinsics number rows in the screen 0 through 23 and columns 0 through 79. The IBM screen layout numbers rows 1 through 24 and columns 1 through 80. Figure 2-1 compares the way SNA IMF and IBM number the rows and columns on a screen.

Figure 2-1 Comparing SNA IMF and IBM Screen Layouts



An attribute is not considered part of the data field. The field-oriented intrinsics never return the attribute when they return field data. The data field begins with the first position following the attribute character. This position is defined as the location of the first data character in the field; here, *offset = 0*. When the READFIELD intrinsic returns data from a field, embedded and leading null characters are returned with data characters, but trailing null characters are not returned. This means that the actual field length parameter count does not include trailing null characters.

Intrinsic Calling Sequences

The order in which you call SNA IMF intrinsics depends on whether your device or the host system will be the first to send data after a communication path is opened. If the host has a screen of data waiting for you, the host will want to send the screen before allowing you to send any data. If the host system requires you to log on or start the host application, your device must send the first screen of data to the host system.

You can simplify your program by developing routines that consist of commonly used sequences of intrinsics. Be sure these routines check the value returned in the *result* parameter of each intrinsic. Appendix F , "Sample Programs," contains complete sample programs.

Following are suggested intrinsic sequences, presented to help you get started writing your programs. You may want to run Pass Thru with the *format* parameter in the info string set to three or four (the automatic print option), to verify the correct intrinsic calling sequence. See *Using SNA IMF Pass Thru* for more information.

Host Transmits Data First

If you expect the host to send data to your device first, use intrinsics in a sequence like this:

Establishing communications with the host:

OPEN3270	Establish communications with the host system.
RECV3270	Check to see if the data has been received.
RECV3270	Accept an initial screen of data from the host.
SCREENATTR	Determine the number of fields in the screen and update the cursor variables containing the coordinates of the cursor position.

Reading data, from a specified field:

FIELDATTR	Determine the attribute values for a specified field.
READFIELD	Read data from a specified field in the screen.
	(Repeat the FIELDATTR/READFIELD sequence as many times as necessary.)

Reading data, from part or all of the screen:

ATTRLIST	Return the locations of the attribute characters.
READSCREEN	Read data from part or all of the screen.

Writing data, into one field at a time:

`WRITEFIELD` Write data into an unprotected field.

Writing data, into several fields at a time:

`SCREENATTR` Update the cursor (row/column) variables again.

`STREAM3270` Write data into several fields at one time.

`TRAN3270` Send a data screen to the host system.

Receiving another screen of data from the host:

`RECV3270` Accept a new data screen from the host.

`SCREENATTR` Determine the number of fields in the screen and update the cursor variables containing the coordinates of the cursor position.

`FIELDATTR` Determine the attributes of the fields specified.

Closing a session with the host:

`CLOSE3270` Disassociate your terminal from an SSCP-LU session.

Terminal Transmits Data First

If you expect to send data to the host system first, use the intrinsics in a sequence like this:

Sending data to the host first:

`OPEN3270` Establish communications with the host.

`RECV3270` Accept the SNA IMF banner message.

`TRAN3270` Send the **[SYS REQ]** key, which puts you into the LU-SSCP session and allows you to send data to the host.

`RECV3270` Receive a blank SSCP-LU screen.

`WRITEFIELD` Write to the LU-SSCP screen.

`TRAN3270` Send the data screen to the host.

`RECV3270` Accept a new screen of data from the host.

`SCREENATTR` Determine the number of fields in the screen and update the variables containing the coordinates of the cursor position.

Reading data, from a specified field:

`FIELDATTR` Determine the field attribute values for a specified field.

READFIELD	Read data from a specified field in the screen. (Repeat the FIELDATTR/READFIELD sequence as many times as necessary.)

Reading data, from part or all of the screen:

ATTRLIST	Return the locations of the attribute characters.
READSCREEN	Read data from part or all of the screen.

Closing a session with the host:

CLOSE3270	End communications with the host.

NOTE Be sure to log off from the host application you are using before you issue the CLOSE3270 intrinsic. The CLOSE3270 intrinsic terminates your session with the host system, but it does not terminate the host application. If you call CLOSE3270 before you log off from the host application, the host application is left running, and you have no way of terminating it, because your session has been closed.

For complete example programs, in non-transparent and transparent mode, see Appendix F , “Sample Programs.”

3 Intrinsics Used with Standard MPE I/O

This chapter describes the intrinsics used with standard (wait) MPE I/O. For a description of MPE wait and no-wait I/O, see Chapter 2 , “Using SNA IMF Intrinsics.”

The intrinsic descriptions are listed in alphabetical order. Each description includes parameter definitions, a list of possible return codes that the intrinsic can generate, calling sequences in BASIC, COBOL, COBOL II, FORTRAN, Pascal, SPL, and C, and programming excerpts from Pascal programs.

Output parameters are bold. An abbreviation above each parameter in the syntax description indicates the data type of the parameter. The abbreviations are as follows:

`I`	Integer
`CA`	Character Array
`IA`	Integer Array

All parameters are passed by reference. All parameters are required.

NOTE References to COBOL in this section apply to COBOL II as well as COBOL. References to COBOL II apply only to COBOL II.

See Appendix A , “Intrinsic Result Codes,” for a complete list of all result codes returned in the *`result`* parameter of SNA IMF intrinsics.

See the “Summary of Intrinsic Calls,” later in this Chapter, for a list of the intrinsic calls in each supported language.

ACQUIRE3270

ACQUIRE3270 on a terminal or printer that is not under MPE control.

Syntax (for SNA IMF/V)

```
                    CA          I        I      I         I
ACQUIRE3270    (snalnkinfo, devicenum, ldev, enhance, priority,

                    I       I       I      I
                blanks, format, flags, result
```

Syntax (for SNA IMF/XL)

```
                    CA          I        I      I         I
ACQUIRE3270    (snalnkinfo, devicenum, ldev, enhance, priority,

                    I       I       I      I      CA     I
                blanks, format, flags, options, pfn, result
```

Parameters

snalnkinfo (input)

Character array that represents the name of the SNA node and the SNA class. For SNA IMF/V, the format of *snalnkinfo* is *snanode, #snaclassname*. For SNA IMF/XL, the format of *snalnkinfo* is *snanode#security classname*. A pound sign (#) must separate the array items. The *snalnkinfo* parameter identifies the section of the Node Management Services configuration file (NMCONFIG.PUB.SYS) used by the SNA link product. The *snalnkinfo* parameter must end with a non-alphanumeric character. The following characters cannot be used as terminators:# / [ ] . , :

devicenum (input)

Integer identifying the device to be emulated. The *devicenum* parameter must be -2 for a terminal, -1 for an LU.T1 printer, or -3 for an LU.T3 printer.

ldev (input)

Integer specifying the MPE logical device number of an HP 3000 terminal or printer to be used to run Pass Thru.

enhance (input)

Integer specifying the terminal screen display enhancements to be used. Table 3-1lists the values of the *enhance* parameter and their effects.

Table 3-1 Display Enhancement Options

Value	3278 normal intensity converted to	3278 high intensity converted to
0	264x half-bright or 23xx/262x normal (full-bright)	23xx/262x/264x normal (full-bright)
1	23xx/262x/264x normal (full-bright)	23xx/262x/264x underline
2	23xx/262x/264x normal (full-bright)	23xx/262x/264x inverse video
3	23xx/262x/264x inverse video	23xx/262x/264x normal (full-bright)

If you are using an HP 264x terminal, you must have the Display Enhancements option installed, because half-bright and underline are not standard features. Without the Display Enhancements option, if you specify 0 in the *enhance* parameter, all characters appear at normal intensity. Half-bright is not available on the 23xx or 262x terminals.

You cannot turn off inverse video for part of a field on an HP terminal. Therefore, if you choose inverse video, the white strip of the inverse video occupies the entire field, regardless of where the end of your data falls in the field.

priority (input)

Integer specifying the output priority for Pass Thru. *priority* must be from 1 through 13. This parameter will be used by Pass Thru to determine the priority of the LOGIMF file, which contains the output of the **PRINT** softkey used to print the internal screen image. The default value is 7.

For more information about priority, see the description of the FOPEN intrinsic in the MPE V Intrinsic Reference Manual. This *priority* parameter is passed to an FOPEN intrinsic call as the output priority value in the *numbuffers* parameter. You may override *priority* by

specifying a file equa3ion before starting your program. The name of the output file is LOGIMF.

blanks (input)

Integer that determines whether leading blank input characters in an unprotected field are converted to nulls before being transmitted to the host. For more information, see *Using SNA IMF Pass Thru* manual.

0	Pass Thru converts leading blank input characters in an unprotected field to null characters. This option performs the same function as specifying *LB = NO* in the PTCONFIG file for Pass Thru sessions.
1	If leading blanks exist in fields read from the HP terminal screen, Pass Thru transmits the leading blanks to the host. This option performs the same function as specifying *LB = YES* in the PTCONFIG file for Pass Thru sessions.

format (input)

Integer that tells Pass Thru which form of screen printing to use and when to print the internal screen (not used for LU.T1 emulation):

1	Print the internal screen image and the location and characteristic of each attribute character.
2	Print the internal screen image the way it appears on the terminal, with attribute and null characters appearing as blanks.
3	Automatically print the internal screen image and the location and characteristic of each attribute character whenever Pass Thru calls the TRAN3270 or RECV3270 intrinsic.
4	Print the internal screen image the same way it appears on the terminal with attribute and null characters appearing as blanks, whenever Pass Thru calls the TRAN3270 or RECV3270 intrinsic.

flags (input)

Integer indicating one of the following:

0	The Pass Thru session is started as a child process of your program, and your program continues to execute. With this option, your program can start Pass Thru sessions on multiple devices. Be careful: all Pass Thru sessions started by your program will be aborted if your program should terminate, either normally or abnormally.
1	The Pass Thru session is activated as the child process of your program, and your program is suspended until the Pass Thru session ends. Your program is reactivated when the Pass Thru session ends.
3	The Pass Thru session activated by your program becomes the child process of the Node Management Monitor (NMMON). Your program continues to execute after Pass Thru has started. With this option, your program can start Pass Thru sessions on multiple devices, and these Pass Thru sessions can remain active after your program has terminated.

options (input)

SNA IMF/XL only. A 16-bit word that indicates the trace, spooler priority, read timeout, and LaserJet II options. The bit groups are listed using the standard SPL notation: (15:1) indicates bit 15, and (10:3) indicates bits 10, 11, and 12.

Bit (15:1)	LaserJet II mode: 0 = LaserJet II option is off. Printer output files will go to the spooled printer. 1 = LaserJet II option is on. Printer output files will be redirected to an HP LaserJet Series II or Series III printer. The LaserJet II option can be used only in an LU.T1 or LU.T3 printer session.

	Bits (6:9)	Read timeout value: Terminal timeout value in seconds. The value must be an integer between 10 and 255. The default value for Pass Thru is 10 seconds. You may have to increase this value if your terminal's response time is longer than 10 seconds.
	Bits (2:4)	Spooler priority: This value must be an integer between 1 and 13.
	Bit (1:1)	Trace mode: 0 = SNA IMF internal tracing is off. 1 = SNA IMF internal tracing is on. Tracing should be turned on only at the request of your HP representative, to collect data for problem determination. A trace file with the name `IMFnnTxx`, where `nn` is the NAU number and `xx` is a unique two-digit identifier, will be created during the Pass Thru session. Note that tracing degrades performance.
	Bit (0:1)	Reserved.

pfn (input)

SNA IMF/XL only. Optional character array specifying a name for the spooler file. A legal filename is an alphanumeric string of up to 8 characters. The first character must be alphabetic. If *pfn* is not specified, the default file name is a field of blanks.

NOTE

If an illegal spooler file name is passed as the *pfn* parameter, no error will be generated. Illegal filenames will be treated as follows:

1. A file name of more than 8 characters will be truncated after the eighth character.
2. If the first character of an illegal file name is alphabetic, but the filename contains a non-alphanumeric character, the filename will be truncated at the illegal character.
3. If the illegal file name begins with a non-alphabetic character, the default printer file name (a field of blanks) will be used.

result (output)

The following values can be generated by the ACQUIRE3270 intrinsic:

0 = Successful completion.

22 = BASIC calling sequence error has occurred.

25 = Intrinsic call made while in split stack mode.

26 = Intrinsic call made with the parameter value out of bounds.

27 = Could not open device. Insufficient virtual memory was available.

28 = Could not open device. Insufficient real memory was available.

30 = Internal error occurred in IMF intrinsic.

43 = Transparent mode not requested for LU.T1 emulation.

69 = Unable to find snaclassname.

70 = Invalid LDEV specified or LDEV is already in use.

71 = The ENHANCE parameter must be an integer between 0 and 3.

72 = The PRIORITY parameter must be an integer between 1 and 13.

73 = The BLANKS parameter must be either 0 or 1.

74 = The FORMAT parameter must be an integer between 1 and 4.

75 = Invalid flags parameter.

76 = TTSSON.PUB.SYS is missing.

77 = An HP LAN nodename must begin with an alphabetic character.

78 = The SNA Server must be installed properly to use this feature.

79 = Fewer than three characters were specified for snanode#snaclassname.

83 = Unable to find HP LAN nodename.

84 = An HP LAN nodename must contain between one and eight alphanumeric characters.

85 = Unable to find end of user and account string(]).

86 = SNA nodename must begin with an alphabetic character.

87 = An SNA classname must begin with an alphabetic character.

88 = A user.account is required for 3287 printer emulation over a LAN.

89 = Fewer than three characters were specified for user.account.

251 = could not open NMCONFIG.PUB.SYS. (MPE XL only)

252 = Could not read from NMCONFIG.PUB.SYS. MPE XL only)

253 = Could not close NMCONFIG.PUB.SYS. (MPE XL only)

254 = Invalid SNAnode name. (MPE XL only)

255 = Invalid security class name. (MPE XL only)

256 = Security class not properly configured. (MPE XL only)

257 = Program not authorized to use this security class. (MPE XL only)

258 = User is not authorized to use this security class. (MPE XL only)

304 = Security violation. (MPE V only)

308 = Session is inactive.

315 = Internal Error.

321 = Invalid class name. (MPE V only)

324 = Invalid configuration access. (MPE V only)

326 = No available NAU.

329 = Inactive node or invalid node name.

Description

The ACQUIRE3270 intrinsic allows you to start Pass Thru programmatically on a terminal or printer attached to an HP 3000. The parameters of the ACQUIRE3270 intrinsic correspond to the info string parameters in the RUN command that starts Pass Thru. For more information, see *Using SNA IMF Pass Thru*.

In most cases, ACQUIRE3270 may start Pass Thru on any free terminal or printer not under MPE control. However, you can call ACQUIRE3270 to gain control over a device that is already in use if you specify a value of 1 in the *flags* parameter. You must exit Pass Thru before the device can again be used for normal MPE activity. An error occurs, and a result code of 70 is returned, if you specify a *flags* parameter value of 0 and attempt to start Pass Thru on a device that is in use.

If Pass Thru detects an error condition (for example, a line to the host system has not been established), an error message is displayed on both the HP 3000 terminal being acquired and the system console, and the error number is returned in the ACQUIRE3270 *result* parameter. You can call the ERR3270 intrinsic to get the message associated with the result code.

NOTE If you are using the HP SNA Server/Access products on a LAN, some of the ACQUIRE3270 parameters are different from those described here. See the *HP SNA Server/Access User's Guide* for more specific information.

Although similar to the IMF/3000 ACQUIRE3270 intrinsic, the SNA IMF ACQUIRE3270 intrinsic has a different interface. For example, the SNA IMF *snalnkinfo* parameter replaces the IMF/3000 *confile* parameter, and the SNA IMF *devicenum* parameter replaces the IMF/3000 *deviceid* parameter. See Appendix A , "Intrinsic Result Codes," for details.

Call ACQUIRE3270 in non-transparent mode.

COBOL Calling Sequence

```
CALL "CACQUIRE3270" USING SNALNKINFO DEVICENUM LDEV ENHANCE
PRIORITY BLANKS FORMAT FLAGS RESULT. (on MPE V)

CALL "CACQUIRE3270" USING SNALNKINFO DEVICENUM LDEV ENHANCE
PRIORITY BLANKS FORMAT FLAGS OPTIONS PFN RESULT. (on MPE XL
in compatibility mode)

CALL INTRINSIC "ACQUIRE3270" USING SNALNKINFO DEVICENUM
LDEV ENHANCE PRIORITY BLANKS FORMAT FLAGS OPTIONS PFN
RESULT. (on MPE XL in native mode)
```

All parameters are numeric data items except *SNALNKINFO* and *PFN*, which are alphanumeric data items.

FORTRAN Calling Sequence

CALL ACQUIRE3270 (SNALNKINFO, DEVICENUM, LDEV, ENHANCE, PRIORITY, BLANKS, FORMAT, FLAGS, RESULT) (on MPE V)

CALL ACQUIRE3270 (SNALNKINFO, DEVICENUM, LDEV, ENHANCE, PRIORITY, BLANKS, FORMAT, FLAGS, OPTIONS, PFN, RESULT) (on MPE XL)

All parameters are integer variables except *SNALNKINFO* and *PFN*, which are character variables.

BASIC Calling Sequence

CALL BACQUIRE3270 (C$, D1, L0, E1, P2, B1, F3, F, R) (on MPE V)

CALL BACQUIRE3270 (C$, D1, L0, E1, P2, B1, F3, F, O4, P$, R) (on MPE XL in compatibility mode)

CALL ACQUIRE3270 (C$, D1, L0, E1, P2, B1, F3, F, O4, P$, R) (on MPE XL in native mode)

All parameters are integer variables except C$and P$, which are string variables.

SPL Calling Sequence

ACQUIRE3270 (SNALNKINFO, DEVICENUM, LDEV, ENHANCE, PRIORITY, BLANKS, FORMAT, FLAGS, RESULT) (on MPE V)

ACQUIRE3270 (SNALNKINFO, DEVICENUM, LDEV, ENHANCE, PRIORITY, BLANKS, FORMAT, FLAGS, OPTIONS, PFN, RESULT) (on MPE XL)

All parameters are integer variables except *SNALNKINFO* and *PFN*, which are byte arrays.

Pascal Calling Sequence

ACQUIRE3270 (SNALNKINFO, DEVICENUM, LDEV, ENHANCE, PRIORITY, BLANKS, FORMAT, FLAGS, RESULT); (on MPE V)

ACQUIRE3270 (SNALNKINFO, DEVICENUM, LDEV, ENHANCE, PRIORITY, BLANKS, FORMAT, FLAGS, OPTIONS, PFN, RESULT); (on MPE XL)

All parameters are short integer variables except *SNALNKINFO* and *PFN*, which are packed arrays of char.

C/XL Calling Sequence

ACQUIRE3270 (SNALINKINFO, &DEVICENUM, &LDEV, &ENHANCE, &PRIORITY, &BLANKS, &FORMAT, &FLAGS, &OPTIONS, PFN, &RESULT);

All parameters are of type short, except *SNALINKINFO* and *PFN*, which are character arrays (pointers to characters).

Pascal Program Excerpts

Following are excerpts from a Pascal program that calls SNA IMF intrinsics. For examples of complete Pascal programs in non-transparent and transparent modes, see Appendix F , "Sample Programs."

```
{*** NOTE: This example is for SNA IMF/XL only, because it uses the    ***}
{*** 'options' and 'pfn' parameters, which SNA IMF/V does not support. ***}

{************************* Global Declarations *************************}
type
   shortint = -32768..32767; { global type, two bytes (half word) }

   options_type  = packed record        { options_type may be global }
      filler     : 0..1;                { or local. It's global in this }
      int_trace  : 0..1;                { example. Two bytes (half word) }
      spriority  : 0..13;               { Note: readto is defined as a }
      readto   : 10..256;               { value from 10 through 256, but }
      LJ2        : 0..1;                { the highest bit is reserved and }
   end;                                 { must be zero, so 255 is the }
                                      { highest allowable value for readto. }
.
.
.
var
   result        : shortint;
.
.
.
procedure ACQUIRE3270;  intrinsic;
.
.
.
```

```
{************************* Local Declarations *************************}
var
   snalnkinfo  : packed array[1..18] of char;   { All ACQUIRE3270 variables }
   devicenum   : shortint;                      { except result are }
   ldev        : shortint;                      { declared as local. }
   enhance     : shortint;
   priority    : shortint;
   blanks      : shortint;
   format      : shortint;
   flags       : shortint;
   options     : options_type;
   pfn         : packed array[1..9] of char;
.
.
.
{************** Variable Initialization and Intrinsic Call **************}
snalnkinfo := 'SNANODE#SNACLASS ';
devicenum := -3;
ldev := 6;
enhance := 0;
priority := 6;
blanks := 0;
format := 3;
flags := 2;

with options do
begin
   filler := 0;
   int_trace := 0;
   spriority := 8;
   readto := 10;
   LJ2 := 1;
end;

ACQUIRE3270 (snalnkinfo, devicenum, ldev, enhance, priority, blanks,
             format, flags, options, pfn, result);
```

ATTRLIST

ATTRLIST returns the locations of attribute characters in the internal screen image.

Syntax

```
                I          I            I           I
ATTRLIST      (terminalid, offset, subscreensize, maxlistlen,
                I          IA           I           I
 fieldnum, offsetlist, actlistlen, result)
```

Parameters

terminalid (input)

Integer identifying the terminal. The *terminalid* is returned in a call to the OPEN3270 intrinsic.

offset (input)

Integer specifying the offset, in characters, into the internal screen image. The first character position on the screen has an offset of zero. The search for attribute characters starts at the location you specify with *offset*. The *offset* parameter must be zero when the DBCS option to the OPEN3270 intrinsic is enabled.

subscreensize (input)

Integer specifying the size of screen, in characters, to be searched for attributes starting at *offset*. The *subscreensize* parameter must be a positive integer and is limited to the length of the screen.

maxlistlen (input)

Integer specifying the maximum number of attribute character locations to return in the array *offsetlist*. The *maxlistlen* parameter must be a positive integer.

fieldnum (output)

Integer identifying the number of the field associated with the first attribute character within the screen area specified by *offset* and *subscreensize*. The value of *fieldnum* is relative to the full internal screen image.

offsetlist (output)

Integer array containing the offset locations of the attribute characters within the screen area defined by *offset* and *subscreensize*. The array must be large enough to accept *maxlistlen* number of locations where each offset location occupies one word of the array. If the array is too small, it will be filled, and the actual number of attribute characters within the specified screen subsection will be returned in *actlistlen*.

actlistlen (output)

Integer indicating the actual number of attributes in the screen subsection defined by *offset* and *subscreensize*. If the actual number of attributes is less than or equal to *maxlistlen*, the count returned in *actlistlen* is equal to the number of attributes written to *offsetlist*. If the number of attributes is greater than *maxlistlen*, the count returned in *actlistlen* is greater than the number of elements in *offsetlist*. The parameter *actlistlen* tells you the number of attribute characters in a subsection.

result (output)

The following values can be generated by the ATTRLIST intrinsic:

0 = Successful completion

1 = Device not open.

9 = Host modified screen since the last receive request. (MPE V only)

21 = Field offset specified is out of range.

22 = BASIC calling sequence error has occurred.

25 = Intrinsic call made while in split stack mode.

26 = Intrinsic call made with the parameter value out of bounds.

29 = Called intrinsic with a request already outstanding. (No-wait I/O only)

30 = Internal error occurred in IMF intrinsic.

42 = Specified MAXINBUFLEN parameter is too large.

53 = Invalid intrinsic called for data stream mode device.

100 = A bad (non-zero) offset was specified in the intrinsic call

Description

The ATTRLIST intrinsic returns an array containing the locations of the attribute characters within any subsection or all of the internal screen image. The ATTRLIST intrinsic is intended to complement the READSCREEN intrinsic. After receiving a subsection of the internal screen image from READSCREEN, you can call ATTRLIST to find out where attribute characters are located, and thus where fields are located, within the subsection.

The *offset* and *subscreensize* parameters define the screen subsection in the same manner for ATTRLIST as the parameters *offset* and *maxinbuflen* do for READSCREEN. The array *offsetlist* is filled with all of the locations of attribute characters within the subsection, up to *maxlistlen*. These locations are expressed as offsets into the full screen image (not the screen subsection).

The *fieldnum* parameter returns the number of the field associated with the first attribute character within the screen subsection.

The *actlistlen* parameter contains the actual number of attribute characters in the screen subsection. This number will reflect the number of elements in *offsetlist* only if the *offsetlist* array is large enough to hold all the locations. If you specified a *maxlistlen* value smaller than the actual number of attribute characters in the screen subsection, the *actlistlen* value will be larger than the number of elements in the *offsetlist* parameter.

If you have called the SCREENATTR intrinsic and determined the number of fields in an internal screen image (from the *numfields* parameter), you can locate the attribute characters within any area or within all of the internal screen image. You do not need to use the ATTRLIST intrinsic if you call the READFIELD intrinsic and work with field numbers instead of locations. However, you must use the ATTRLIST intrinsic if you have used the READSCREEN intrinsic, because you must be able to distinguish attribute characters from data characters.

Use the ATTRLIST intrinsic in non-transparent mode.

COBOL Calling Sequence

CALL “CATTRLIST” USING TERMINALID OFFSET SUBSCREENSIZE MAXLISTLEN FIELDNUM OFFSETLIST ACTLISTLEN RESULT. (on MPE V and in compatibility mode on MPE XL)

```
CALL INTRINSIC "ATTRLIST" USING TERMINALID OFFSET
SUBSCREENSIZE MAXLISTLEN FIELDNUM OFFSETLIST ACTLISTLEN
RESULT.
```
(in native mode on MPE XL)

All parameters are numeric data items except *OFFSETLIST*, which is an integer table.

FORTRAN Calling Sequence

```
CALL ATTRLIST (TERMINALID, OFFSET, SUBSCREENSIZE,
MAXLISTLEN, FIELDNUM, OFFSETLIST, ACTLISTLEN, RESULT)
```

All parameters are numeric data items except *OFFSETLIST*, which is an integer array.

BASIC Calling Sequence

```
CALL BATTRLIST (T, O1, L1, L2, N, O2(*), L3, R)
```
(on MPE V and in compatibility mode on MPE XL)

```
CALL ATTRLIST (T, O1, L1, L2, N, O2(*), L3, R)
```
(in native mode on MPE XL)

All parameters are integer variables except *O2(*)*, which is an integer array.

SPL Calling Sequence

```
ATTRLIST (TERMINALID, OFFSET, SUBSCREENSIZE, MAXLISTLEN,
FIELDNUM, OFFSETLIST, ACTLISTLEN, RESULT)
```

All parameters are integer variables except *OFFSETLIST*, which is an integer array.

Pascal Calling Sequence

```
ATTRLIST (TERMINALID, OFFSET, SUBSCREENSIZE, MAXLISTLEN,
FIELDNUM, OFFSETLIST, ACTLISTLEN, RESULT);
```

All parameters are short integer variables except *OFFSETLIST*, which is an array of short integer variables.

C/XL Calling Sequence

```
ATTRLIST (&TERMINALID, &OFFSET, &SUBSCREENSIZE,
&MAXLISTLEN, &FIELDNUM, OFFSETLIST, &ACTLISTLEN, &RESULT);
```

All parameters are of type short, except *OFFSETLIST*, which is an array of 2 short integer variables (a pointer to a short).

Pascal Program Excerpts

Following are excerpts from a Pascal program that calls SNA IMF intrinsics. For examples of complete Pascal programs in

non-transparent and transparent modes, see Appendix F , “Sample Programs.”

```
{************************* Global Declarations *************************}
type
   shortint      = -32768..32767;       { global type, two bytes (half word) }
.
.
.
var
   terminalid    : shortint;          { value returned by OPEN3270 intrinsic }
   result        : shortint;
.
.
.
procedure ATTRLIST;  intrinsic;
.
.
.
************************* Local Declarations *************************}
var
   offset        : shortint;         { All ATTRLIST variables except }
   subscreensize : shortint;         { terminalid and result are local. }
   maxlistlen    : shortint;
   fieldnum      : shortint;
   offsetlist    : packed array[1..128] of shortint;
   actlistlen    : shortint;
{************** Variable Initialization and Intrinsic Call **************}
offset := 0;
subscreensize := 1920;
maxlistlen := 128;

ATTRLIST (terminalid, offset, subscreensize, maxlistlen, fieldnum,
          offsetlist, actlistlen, result);
```

CLOSE3270

CLOSE3270 emulates turning off a specified device.

Syntax

```
                I          I
CLOSE3270       (terminalid, result)
```

Parameters

terminalid (input)

Integer identifying the terminal. The *terminalid* is returned in a call to the OPEN3270 intrinsic.

result (output)

The following values can be generated by the CLOSE3270 intrinsic:

0 = Successful completion.

1 = Device not open.

22 = BASIC calling sequence error has occurred.

26 = Intrinsic call made with the parameter value out of bounds.

30 = Internal error occurred in IMF intrinsic.

301 = Illegal DB register.

302 = Invalid session.

305 = Parameter bounds violation.

315 = Internal Error.

340 = No stack space.

Description

The CLOSE3270 intrinsic is equivalent to turning off a particular device. CLOSE3270 releases the internal screen image created by the OPEN3270 intrinsic and frees the LU-LU session for other users.

Use the CLOSE3270 intrinsic in either transparent or non-transparent mode.

NOTE Be sure to log off from the host application you are using before you issue the CLOSE3270 intrinsic. The CLOSE3270 intrinsic terminates your session with the host system, but it does not terminate the host application. If you call CLOSE3270 before you log off from the host application, the host application is left running, and you have no way of terminating it, because your session has been closed.

COBOL Calling Sequence

CALL "CCLOSE3270" USING TERMINALID RESULT. (on MPE V and in compatibility mode on MPE XL)

CALL INTRINSIC "CLOSE3270" USING TERMINALID RESULT. (in native mode on MPE XL)

Both parameters are numeric data items.

FORTRAN Calling Sequence

CALL CLOSE3270 (TERMINALID, RESULT)

Both parameters are integer variables.

BASIC Calling Sequence

CALL BCLOSE3270 (T, R) (on MPE V and in compatibility mode on MPE XL)

CALL CLOSE3270 (T, R) (in native mode on MPE XL)

Both parameters are integer variables.

SPL Calling Sequence

CLOSE3270 (TERMINALID, RESULT)

Both parameters are integer variables.

Pascal Calling Sequence

CLOSE3270 (TERMINALID, RESULT)

Both parameters are short integer variables.

C/XL Calling Sequence

CLOSE3270 (&TERMINALID, &RESULT)

Both parameters are of type short.

Pascal Program Excerpts

Following are excerpts from a Pascal program that calls SNA IMF intrinsics. For examples of complete Pascal programs in non-transparent and transparent modes, see Appendix F , “Sample Programs.”

```
{************************* Global Declarations *************************}
type
   shortint      = -32768..32767;       { global type, two bytes (half word) }
.
.
.
var
   terminalid    : shortint;          { value returned by OPEN3270 intrinsic }
   result        : shortint;
.
.
.
procedure CLOSE3270;  intrinsic;
.
.
.
{************************* Intrinsic Call *************************}
CLOSE3270 (terminalid, result);
```

ERR3270

ERR3270 returns the error message associated with a given intrinsic result code.

Syntax

```
                I       CA      I       I
ERR3270      (errorcode, msgbuf, msglen, result)
```

Parameters

errorcode (input)

An integer value returned in the *result* parameter of a previous SNA IMF intrinsic call.

msgbuf (output)

Character array where the message associated with the *errorcode* will be returned. This string must be large enough to hold up to 144 characters. If a bounds violation occurs while you are using *msgbuf*, you will receive a *result* of 26, and you will not receive a message in *msgbuf*.

msglen (output)

Integer indicating the number of characters actually returned in *msgbuf*. If an error occurs while you are using *msglen*, both a message and a message length are returned to your program. See the *result* parameter for ERR3270. If a bounds violation occurs while you are using *msglen*, you will not receive a value for *msglen*.

result (output)

The following values can be generated by the ERR3270 intrinsic:

<0 =	(Less than zero) Indicates this result code originated with the MPE V intrinsic, GENMESSAGE, or with the MPE XL intrinsic CATREAD. Look up the absolute value of the result code under GENMESSAGE in the *MPE V Intrinsics Reference Manual*, or under CATREAD in the *MPE XL Intrinsics Reference Manual*.

0 = Successful completion.

22 = BASIC calling sequence error has occurred.

25 = Intrinsic call made while in split stack mode.

26 = Intrinsic call made with the parameter value out of bounds.

30 = Internal error occurred in IMF intrinsic.

Description

ERR3270 takes the result code from a previous intrinsic call and returns the message that corresponds to that result code. You pass the result code to the ERR3270 intrinsic in the *errorcode* parameter, and the corresponding message is returned in the *msgbuf* parameter.

Appendix A , "Intrinsic Result Codes," contains a list of all intrinsic result codes, their corresponding messages, their probable causes, and the recommended actions for resolving problems.

SNA IMF messages are kept in a message catalog named CATIMF.PUB.SYS. With the appropriate MPE file access capabilities, you may change the messages in CATIMF to suit your needs. The procedure for changing diagnostic messages is explained in the *MPE V Intrinsics Reference Manual*.

Use the ERR3270 intrinsic in transparent or non-transparent mode.

COBOL Calling Sequence

CALL "CERR3270" USING ERRORCODE MSGBUF MSGLEN RESULT. (on MPE V and in compatibility mode on MPE XL)

CALL INTRINSIC "ERR3270" USING ERRORCODE MSGBUF MSGLEN RESULT. (in native mode on MPE XL)

All parameters are numeric data items except *MSGBUF*, which is an alphanumeric data item.

FORTRAN Calling Sequence

CALL ERR3270 (ERRORCODE, MSGBUF, MSGLEN, RESULT)

All parameters are integer variables except *MSGBUF*, which is a character array.

BASIC Calling Sequence

CALL BERR3270 (E, M$, L4, R) (on MPE V and in compatibility mode on MPE XL)

CALL ERR3270 (E, M$, L4, R) (in native mode on MPE XL)

All parameters are integer variables except *M$*, which is a string variable.

SPL Calling Sequence

```
ERR3270 (ERRORCODE, MSGBUF, MSGLEN, RESULT)
```

All parameters are integer variables except *MSGBUF*, which is a byte array.

Pascal Calling Sequence

```
ERR3270 (ERRORCODE, MSGBUF, MSGLEN, RESULT);
```

All parameters are short integers except *MSGBUF*, which is a packed array of character.

C/XL Calling Sequence

```
ERR3270 (&ERRORCODE, MSGBUF, &MSGLEN, &RESULT);
```

All parameters are of type short except *MSGBUF*, which is an array of characters (a pointer to a char).

Pascal Program Excerpts

Following are excerpts from a Pascal program that calls SNA IMF intrinsics. For examples of complete Pascal programs in non-transparent and transparent modes, see Appendix F , “Sample Programs.”

```
{************************* Global Declarations *************************}
type
   shortint      = -32768..32767;       { global type, two bytes (half word) }
.
.
.
var
   result        : shortint;
.
.
.
procedure ERR3270;  intrinsic;
.
.
.
{************************* Local Declarations **************************}
var
   errorcode  : shortint;                      { All ERR3270 variables }
   msgbuf     : packed array[1..160] of char;  { except result }
   msglen     : shortint;                      { are local. }
.
.
.
{************** Variable Initialization and Intrinsic Call **************}
errorcode := result;
msgbuf := ' ';
                ERR3270 (errorcode, msgbuf, msglen, result);
```

EXTFIELDATTR

For Asian users. **EXTFIELDATTR** returns information about the attributes of a specified field. The EXTFIELDATTR intrinsic is equivalent to the FIELDATTR intrinsic, except that it returns the Double-Byte Character Set (DBCS) attribute.

Syntax

```
                I          I          I            I
EXTFIELDATTR   (terminalid, fieldnum, fieldrow, fieldcolumn,

                I             I             I        I
 protectedattr, numericattr, displayattr, mdt,

                I             I              I         I
 dbcsattr, currentfieldlen, maxfieldlen, result)
```

Parameters

terminalid (input)

Integer identifying the terminal. The *terminalid* is returned in a call to the OPEN3270 intrinsic.

fieldnum (input)

Integer specifying the number of the field whose attributes you are requesting. Put a zero in *fieldnum* if the screen is unformatted (there are no attribute characters).

fieldrow (output)

Integer indicating the row that is the location of the first data character in the field. Possible values are as follows:

0 through 11 (480-character screen)

0 through 23 (1920-character screen)

0 through 42 (3440-character screen)

fieldcolumn (output)

Integer indicating the column that is the location of the first data character in the field. Possible values are as follows:

0 through 39 (480-character screen)

0 through 79 (1920-character screen)

0 through 79 (3440-character screen)

protectedattr (output)

Integer indicating whether the field specified by *fieldnum* is protected or unprotected. The *protectedattr* parameter returns 0 if the screen is unformatted.

0 = Unprotected

1 = Protected

numericattr (output)

Integer indicating whether the field specified by *fieldnum* is alphanumeric or numeric. The *numericattr* parameter returns 0 if the screen is unformatted.

0 = Alphanumeric

1 = Numeric

displayattr (output)

Integer indicating how the field specified by *fieldnum* is displayed on the terminal screen. The *displayattr* parameter returns 0 if the screen is unformatted.

0 = Normal display, light pen not detectable

1 = Normal display, light pen detectable

2 = Intensified display

3 = No display and no print

HP terminals do not support light pens.

mdt (output)

Integer indicating whether the Modified Data Tag is set for the field specified by *fieldnum*. The *mdt* parameter returns 0 if the screen is unformatted.

0 = Modified Data Tag not set

1 = Modified Data Tag set

dbcsattr (output)

Double-Byte Character Set attribute. Integer indicating whether 8-bit data, 16-bit data, or a mixture of 8-bit and 16-bit data is allowed in the field specified by *fieldnum*. The *dbcsattr* parameter returns 0 if the screen is unformatted.

0 = Only 16-bit characters are allowed in the field.

1 = Only 8-bit characters are allowed in the field.

2 = Mixed 8-bit and 16-bit data are allowed. SO/SI control characters may be entered to shift between 8-bit and 16-bit character modes.

3 = Mixed 8-bit and 16-bit data allowed. 16-bit characters must be entered between existing SO/SI pairs. No new SO/SI pairs may be entered.

currentfieldlen (output)

Integer indicating the current length of the field in the internal buffer. Length includes all characters plus embedded and leading nulls; trailing nulls are not included.

maxfieldlen (output)

Integer indicating the maximum number of characters the field can contain. This is the number of character positions between the current field attribute character and the next field attribute character.

result (output)

The following values can be generated by the EXTFIELDATTR intrinsic:

0 = Successful completion.

1 = Device not open.

9 = Host modified screen since the last receive request. (MPE V only)

11 = Non-existent field number specified.

22 = BASIC calling sequence error has occurred.

25 = Intrinsic call made while in split stack mode.

26 = Intrinsic call made with the parameter value out of bounds.

29 = Called intrinsic with a request already outstanding. (No-wait I/O only)

30 = Internal error occurred in IMF intrinsic.

53 = Invalid intrinsic called for data stream mode device.

99 = Invalid intrinsic called while DBCS option is not set.

Description

The EXTFIELDATTR intrinsic returns information about a specified field of the internal screen image. Your program passes the terminal ID and the field number. The intrinsic returns the row and column address of the field in the buffer, the attributes of the field, the maximum number of characters in the field, the current number of characters in the field, and a result parameter indicating success or the reason for failure.

The EXTFIELDATTR intrinsic is identical to the FIELDATTR intrinsic except for one parameter: the *dbcsattr* parameter, which tells your program whether 8-bit data, 16-bit data, or a mixture of 8-bit and 16-bit data is allowed in the specified field. It also indicates whether SO/SI control characters may be entered into the field, or whether DBCS data must be entered between existing SO/SI pairs.

Use the EXTFIELDATTR intrinsic in non-transparent mode.

COBOL Calling Sequence

CALL "CEXTFIELDATTR" USING TERMINALID FIELDNUM FIELDROW FIELDCOLUMN PROTECTEDATTR NUMERICATTR DISPLAYATTR MDT DBCSATTR CURRENTFIELDLEN MAXFIELDLEN RESULT. (on MPE V and in compatibility mode on MPE XL)

CALL INTRINSIC "EXTFIELDATTR" USING TERMINALID FIELDNUM FIELDROW FIELDCOLUMN PROTECTEDATTR NUMERICATTR DISPLAYATTR MDT DBCSATTR CURRENTFIELDLEN MAXFIELDLEN RESULT. (in native mode on MPE XL)

All parameters are numeric data items.

FORTRAN Calling Sequence

CALL EXTFIELDATTR (TERMINALID, FIELDNUM, FIELDROW, FIELDCOLUMN, PROTECTEDATTR, NUMERICATTR, DISLAYATTR, MDT, DBCSATTR, CURRENTFIELDLEN, MAXFIELDLEN, RESULT)

All parameters are integer variables.

BASIC Calling Sequence

```
CALL BEXTFIELDATTR (T, N, R0, C0, A1, A2, A3, A4, A5, L5,
L6, R)
```
(on MPE V and in compatibility mode on MPE XL)

```
CALL EXTFIELDATTR (T, N, R0, C0, A1, A2, A3, A4, A5, L5, L6,
R)
```
(in native mode on MPE XL)

All parameters are integer variables.

SPL Calling Sequence

```
EXTFIELDATTR (TERMINALID, FIELDNUM, FIELDROW, FIELDCOLUMN,
PROTECTEDATTR, NUMERICATTR, DISPLAYATTR, MDT, DBCSATTR,
CURRENTFIELDLEN, MAXFIELDLEN, RESULT)
```

All parameters are integer variables.

Pascal Calling Sequence

```
EXTFIELDATTR (TERMINALID, FIELDNUM, FIELDROW, FIELDCOLUMN,
PROTECTEDATTR, NUMERICATTR, DISPLAYATTR, MDT, DBCSATTR,
CURRENTFIELDLEN, MAXFIELDLEN, RESULT);
```

All parameters are short integer variables.

C/XL Calling Sequence

```
EXTFIELDATTR (&TERMINALID, &FIELDNUM, &FIELDROW,
&FIELDCOLUMN, &PROTECTEDATTR, &NUMERICATTR, &DISPLAYATTR,
&MDT, &DBCSATTR, &CURRENTFIELDLEN, &MAXFIELDLEN, &RESULT);
```

All parameters are of type short.

Pascal Program Excerpts

Following are excerpts from a Pascal program that calls SNA IMF intrinsics. For examples of complete Pascal programs in non-transparent and transparent modes, see Appendix F , “Sample Programs.”

```
{************************* Global Declarations *************************}
type
   shortint      = -32768..32767;       { global type, two bytes (half word) }
.
.
.
var
   terminalid    : shortint;          { value returned by OPEN3270 intrinsic }
   result        : shortint;
.
.
.
procedure EXTFIELDATTR;  intrinsic;
.
.
.
{************************* Local Declarations *************************}
var
   fieldnum          : shortint;         { All EXTFIELDATTR variables }
   fieldrow          : shortint;         { except terminalid and result }
   fieldcolumn       : shortint;         { are local. }
   protectedattr     : shortint;
   numericattr       : shortint;
   displayattr       : shortint;
   mdt               : shortint;
   dbcsattr          : shortint;
   currentfieldlen   : shortint;
   maxfieldlen       : shortint;
.
.
.
{************** Variable Initialization and Intrinsic Call **************}
fieldnum := 2;

EXTFIELDATTR (terminalid, fieldnum, fieldrow, fieldcolumn, protectedattr,
numericattr, displayattr, mdt, dbcsattr, currentfieldlen, maxfieldlen,
result);
```

FIELDATTR

FIELDATTR returns information about the attributes of a specified field.

Syntax

```
                I          I          I          I
FIELDATTR       (terminalid, fieldnum, fieldrow, fieldcolumn,

                  I           I            I         I
                protectedattr, numericattr, displayattr, mdt,

                   I             I         I
                currentfieldlen, maxfieldlen, result)
```

Parameters

terminalid (input)

Integer identifying the terminal. The *terminalid* is returned in a call to the OPEN3270 intrinsic.

fieldnum (input)

Integer specifying the number of the field whose attributes you are requesting. Put a zero in *fieldnum* if the screen is unformatted (there are no attribute characters).

fieldrow (output)

Integer indicating the row that is the location of the first data character in the field. Possible values are as follows:

0 through 11 (480-character screen)

0 through 23 (1920-character screen)

0 through 42 (3440-character screen)

fieldcolumn (output)

Integer indicating the column that is the location of the first data character in the field. Possible values are as follows:

0 through 39 (480-character screen)

0 through 79 (1920-character screen)

0 through 79 (3440-character screen)

protectedattr (output)

Integer indicating whether the field specified by *fieldnum* is protected or unprotected. The *protectedattr* parameter returns 0 if the screen is unformatted.

0 = Unprotected

1 = Protected

numericattr (output)

Integer indicating whether the field specified by *fieldnum* is alphanumeric or numeric. The *numericattr* parameter returns 0 if the screen is unformatted.

0 = Alphanumeric

1 = Numeric

displayattr (output)

Integer indicating how the field specified by *fieldnum* is displayed on the terminal screen. The *displayattr* parameter returns 0 if the screen is unformatted.

0 = Normal display, light pen not detectable

1 = Normal display, light pen detectable

2 = Intensified display

3 = No display and no print

HP terminals do not support light pens.

mdt (output)

Integer indicating whether the Modified Data Tag is set for the field specified by *fieldnum*. The *mdt* parameter returns 0 if the screen is unformatted.

0 = Modified data tag not set

1 = Modified data tag set

currentfieldlen (output)

Integer indicating the current length of the field in the internal buffer. Length includes all characters plus embedded and leading nulls; trailing nulls are not included.

maxfieldlen (output)

Integer indicating the maximum number of characters the field can contain. This is the number of character positions between the current field attribute character and the next field attribute character.

result (output)

The following values can be generated by the FIELDATTR intrinsic:

0 = Successful completion.

1 = Device not open.

9 = Host modified screen since the last receive request. (MPE V only)

11 = Non-existent field number specified.

22 = BASIC calling sequence error has occurred.

25 = Intrinsic call made while in split stack mode.

26 = Intrinsic call made with the parameter value out of bounds.

29 = Called intrinsic with a request already outstanding. (No-wait I/O only)

30 = Internal error occurred in IMF intrinsic.

53 = Invalid intrinsic called for data stream mode device.

Description

The FIELDATTR intrinsic returns information about a specified field of the internal screen image. Your program passes the terminal ID and the field number. The intrinsic returns the row and column address of the field in the buffer, the attributes of the field, the maximum number of characters in the field, the current number of characters in the field, and a result parameter indicating success or the reason for failure.

Use the FIELDATTR intrinsic in non-transparent mode.

COBOL Calling Sequence

CALL “CFIELDATTR” USING TERMINALID FIELDNUM FIELDROW FIELDCOLUMN PROTECTEDATTR NUMERICATTR DISPLAYATTR MDT CURRENTFIELDLEN MAXFIELDLEN RESULT. (on MPE V and in compatibility mode on MPE XL)

```
CALL INTRINSIC "FIELDATTR" USING TERMINALID FIELDNUM
FIELDROW FIELDCOLUMN PROTECTEDATTR NUMERICATTR DISPLAYATTR
MDT CURRENTFIELDLEN MAXFIELDLEN RESULT.
```
(in native mode on MPE XL)

All parameters are numeric data items.

FORTRAN Calling Sequence

```
CALL FIELDATTR (TERMINALID, FIELDNUM, FIELDROW,
FIELDCOLUMN, PROTECTEDATTR, NUMERICATTR, DISLAYATTR, MDT,
CURRENTFIELDLEN, MAXFIELDLEN, RESULT)
```

All parameters are integer variables.

BASIC Calling Sequence

```
CALL BFIELDATTR (T, N, R0, C0, A1, A2, A3, A4, L5, L6, R)
```
(on MPE V and in compatibility mode on MPE XL)

```
CALL FIELDATTR (T, N, R0, C0, A1, A2, A3, A4, L5, L6, R)
```
(in native mode on MPE XL)

All parameters are integer variables.

SPL Calling Sequence

```
FIELDATTR (TERMINALID, FIELDNUM, FIELDROW, FIELDCOLUMN,
PROTECTEDATTR, NUMERICATTR, DISPLAYATTR, MDT,
CURRENTFIELDLEN, MAXFIELDLEN, RESULT)
```

All parameters are integer variables.

Pascal Calling Sequence

```
FIELDATTR (TERMINALID, FIELDNUM, FIELDROW, FIELDCOLUMN,
PROTECTEDATTR, NUMERICATTR, DISPLAYATTR, MDT,
CURRENTFIELDLEN, MAXFIELDLEN, RESULT);
```

All parameters are short integer variables.

C/XL Calling Sequence

```
FIELDATTR (&TERMINALID, &FIELDNUM, &FIELDROW, &FIELDCOLUMN,
&PROTECTEDATTR, &NUMERICATTR, &DISPLAYATTR, &MDT,
&CURRENTFIELDLEN, &MAXFIELDLEN, &RESULT);
```

All parameters are of type short.

Pascal Program Excerpts

Following are excerpts from a Pascal program that calls SNA IMF intrinsics. For examples of complete Pascal programs in non-transparent and transparent modes, see Appendix F, "Sample Programs."

```
{************************* Global Declarations *************************}
type
   shortint      = -32768..32767;       { global type, two bytes (half word) }
.
.
.
var
 terminalid    : shortint;           { value returned by OPEN3270 intrinsic }
   result        : shortint;
.
.
.
procedure FIELDATTR;  intrinsic;
.
.
.
{************************* Local Declarations *************************}
var
   fieldnum          : shortint;         { All FIELDATTR variables }
   fieldrow          : shortint;         { except terminalid and result }
   fieldcolumn       : shortint;         { are local. }
   protectedattr     : shortint;
   numericattr       : shortint;
   displayattr       : shortint;
   mdt               : shortint;
   currentfieldlen   : shortint;
   maxfieldlen       : shortint;
.
.
.
{************** Variable Initialization and Intrinsic Call **************}
fieldnum := 2;
FIELDATTR (terminalid, fieldnum, fieldrow, fieldcolumn, protectedattr,
numericattr, displayattr, mdt, currentfieldlen, maxfieldlen, result);
```

OPEN3270

OPEN3270 emulates turning on the power to a 3278 display station or 3287 printer.

Syntax

```
                 I          CA        I         I         I
OPEN3270         (devicenum, snalnkinfo, flags, terminalid, devtype,

                    I         I         IA       I
                 ffindex, screensize, timeout, result)
```

Parameters

devicenum (input)

Integer identifying the device to be emulated. The *devicenum* parameter must be -2 for a terminal, -1 for an LU.T1 printer, or -3 for an LU.T3 printer.

snalnkinfo (input)

Character array that represents the name of the SNA node and the SNA class. For SNA IMF/V, the format of *snalnkinfo* is *snanode#snaclassname*. For SNA IMF/XL, the format of *snalnkinfo* is *snanode#security classname*. A pound sign (#) must separate the array items. The *snalnkinfo* parameter identifies the section of the Node Management Services configuration file (NMCONFIG.PUB.SYS) used by the SNA link product. The *snalnkinfo* parameter must end with a non-alphanumeric character. The following characters cannot be used as terminators:

/ [ ] . , :

flags (input)

A word whose bit groups specify the following options: wait or no-wait I/O, transparent or non-transparent mode, internal tracing, LU support, UNBIND option, and DBCS option. Bit groups are listed using the standard SPL notation: (15:1) indicates bit 15, and (10:3) indicates bits 10, 11, and 12. See Table 3-2, following the *flags* parameter description, for OPEN3270 configuration options for setting up LU.T1 or LU.T3 emulation.

Bit (15:1)	Input/Output (I/O) mode. (See Chapter 2 , “Using SNA IMF Intrinsics,” for information on wait and no-wait I/O.) 0 = Standard (wait) I/O. Your program will be suspended during I/O processing. 1 = No-wait I/O. I/O requests are overlapped, and your program continues to execute while I/O requests are outstanding.
Bit (14:1)	Transparent/Non-Transparent mode. (See Chapter 2 , “Using SNA IMF Intrinsics,” for information on transparent and non-transparent modes.) 0 = Non-transparent mode. The data stream is translated into an image of the terminal screen. 1 = Transparent mode. Your program has access to the untranslated data stream.
Bit (13:1)	Internal tracing. 0 = Internal tracing is off. 1 = Internal tracing is on. SNA IMF internal tracing should be used only at the request of your HP representative to collect data for problem determination. Enabling the tracing facility degrades performance. The trace file is named `IMFnnTxx`, where `nn` is the NAU number and `xx` is a unique two-digit identifier.
Bit (12:1)	LU.T1/LU.T3 support for IBM 3287 printer emulation. The effect of this option depends on the value in bit 14 and the value in the *`devicenum`* parameter. See Table 3-2 for more information. 0 = This option supports either LU.T1 or LU.T3 emulation but not both. Which LU type is emulated depends on the value of the *`devicenum`* parameter. If *`devicenum = -1`*, bit 14 of the *`flags`* parameter must also be set to 1. When

in LU.T1 sessions, the **PA1**, **PA2**, and **CANCEL** keys are accepted by.

1 = This option allows a printer to accept both LU.T1 and LU.T3 sessions. LU.T1 sessions are in transparent mode, and LU.T3 sessions are in non-transparent mode. If *devicenum = -1*, bit 14 of the *flags* parameter must also be set to 1. When in LU.T1 sessions, the **PA1**, **PA2**, **CANCEL**, **HOLD**, and **ENABLE** keys are accepted by TRAN3270, and additional RECV3270 result codes 91 through 95 may be returned. When in LU.T3 sessions, the printer keys cannot be used and the additional RECV3270 result codes will not be returned.

Bit (11:1) UNBIND option.

0 = UNBIND option is disabled. SNA IMF will send the LUSTAT X'0831' if closing while an LU-LU session is active.

1 = UNBIND option is enabled. SNA IMF will send UNBIND instead of LUSTAT if closing while an LU-LU session is active.

Bit (10:1) DBCS option.

0 = Double-Byte Character Set option is disabled. SNA IMF/XL will process 8-bit data only.

1 = Double-Byte Character Set option is enabled. SNA IMF/XL will process 8-bit and 16-bit data.

Bits (0:10) Reserved

Table 3-2 shows how bits 12 and 14 of the flags parameter work with the value in the devicenum parameter to set up SNA IMF for printer emulation.

Table 3-2 Setting Up LU.T1 or LU.T3 Emulation

LU.T1/LU.T3 (bit 12)	devicenum parameter	Transparent Mode (bit 14)	Effect
0	-1	1	LU.T1 is the initial setup. Only LU.T1 sessions are accepted. Uses transparent mode intrinsics. **PA1**, **PA2**, and **CANCEL** keys accepted.
1	-1	1	LU.T1 is the initial setup. Both LU.T1 and LU.T3 sessions are accepted. LU.T1 sessions are in transparent mode, LU.T3 sessions are in non-transparent mode. All printer keys are accepted for LU.T1. New result codes returned (RECV3270).
0	-3	1	LU.T3 is the initial setup. Only LU.T3 sessions are accepted. Uses transparent mode intrinsics. (No printer keys accepted for LU.T3.)
1	-3	1	LU.T3 is the initial setup. Both LU.T1 and LU.T3 sessions are accepted. LU.T1 sessions are in transparent mode, LU.T3 sessions are in non-transparent mode. All printer keys are accepted for LU.T1. New result codes returned (RECV3270).
0	-1	0	Error: Non-transparent mode with LU.T1 was specified.
1	-1	0	Error: Non-transparent mode with LU.T1 was specified.
0	-3	0	LU.T3 is the initial setup. Only LU.T3 sessions are accepted. Uses non-transparent mode intrinsics. (No printer keys accepted for LU.T3.)
1	-3	0	LU.T3 is the initial setup. Both LU.T1 and LU.T3 sessions are accepted. LU.T1 sessions are in transparent mode, LU.T3 sessions are in non-transparent mode. All printer keys are accepted for LU.T1. New result codes returned (RECV3270).

terminalid (output)

Integer identifying the terminal. This terminal identifier returned by OPEN3270 is used by other SNA IMF intrinsics to identify a device. The *terminalid* parameter is also used by IOWAIT and IODONTWAIT when doing no-wait I/O (*terminalid=filenum*).

devtype (output

Integer identifying the device type. Possible values are as follows:

2 = 3278 display station/keyboard

8 = 3287 printer with 3274/3276 attachment

ffindex (input)

This parameter is not used by SNA IMF, but it is included for backward compatibility. The application calling OPEN3270 is responsible for managing the number of lines per page for printers.

screensize (output)

This parameter is always defined as 1920 characters until the BIND is received from the host. The BIND may specify a buffer size of 480 or 3440 characters.

timeout (input)

An integer array of 2 words that contains timeout values. The first word is the keyboard enable timeout value, and the second word is the transmit/receive timeout value. The keyboard enable timer controls the RECV3270 intrinsic. The transmit/receive value controls both the TRAN3270 and RECV3270 intrinsics. Each integer in the array specifies a timeout value in seconds, from zero through 28800 (0 seconds to 8 hours). Any numbers greater than 28800 and any negative numbers are treated as 28800. A value of zero in a word disables the timer.

The values used in *timeout* usually are treated as unsigned logical 16-bit positive integers. For COBOL and BASIC, the values are treated as signed integers.

If you specify a value for the transmit/receive timeout, SNA IMF sets two timers: a receive timer and a transmit timer. The receive timeout is the maximum time interval that a RECV3270 request may remain outstanding. By specifying a maximum time interval, you can prevent an SNA IMF application from suspending indefinitely (hanging) when the host does not send any data to your device. The transmit timeout is the maximum time interval that a TRAN3270 intrinsic request may remain outstanding. By specifying a maximum time interval, you can prevent an SNA IMF application from hanging when the host system does not allow your device to send data, or when the host goes down before you can send your data to the host.

If new data is not waiting for you (that is, the host has not yet modified your screen), the receive timer is started as soon as the RECV3270 intrinsic starts executing. The timer is turned off when SNA IMF receives data for your device. Likewise, the transmit timer is started when the TRAN3270 intrinsic begins execution. The transmit timer is turned off when SNA IMF successfully completes sending data to the host. If either timer expires, your I/O request (your call to TRAN3270 or RECV3270) is cancelled, and result code 24 is returned.

The keyboard enable timer simplifies writing SNA IMF applications when the keyboard enable sent by the host is always in the last block of data. The keyboard enable timer signals SNA IMF to wait for the keyboard enable before completing your RECV3270 intrinsic call. This timer is especially useful if the host sends a variable number of blocks to update your screen. A single RECV3270 intrinsic call lets the keyboard enable signal when the screen update is complete. If your device is expecting a variable number of blocks, issue a RECV3270 intrinsic call and then check the screen contents to see if the host has finished sending data. If it has not, then you must issue another RECV3270 intrinsic call and check the screen again.

You may use the keyboard enable timer either alone or with the receive timer to control the RECV3270 intrinsic. When you use the keyboard enable timer alone, the timer starts when the first block is received and restarts after each block without a keyboard enable. When you use the keyboard enable timer with the receive timer, the receive timer starts when the RECV3270 intrinsic begins executing. The receive timer is turned off when your device receives the first block of data. The keyboard timer, on the other hand, starts operating and continues to operate until your device receives a block of data that includes the keyboard enable, or until the timeout expires. If the keyboard enable timer expires, your RECV3270 I/O request is cancelled and result code 23 is returned in the *result* parameter of the intrinsic.

result (output)

The following values can be generated by the OPEN3270 intrinsic:

0 = Successful completion.

2 = Invalid devicenum parameter specified.

22 = BASIC calling sequence error has occurred.

25 = Intrinsic call made while in split stack mode.

26 = Intrinsic call made with the parameter value out of bounds.

27 = Could not open device. Insufficient virtual memory was available.

28 = Could not open device. Insufficient real memory was available.

30 = Internal error occurred in IMF intrinsic.

43 = Transparent mode not requested for LU.T1 emulation.

251 = Could not open NMCONFIG.PUB.SYS. MPE XL only)

252 = Could not read from NMCONFIG.PUB.SYS. (MPE XL only)

253 = Could not close NMCONFIG.PUB.SYS. (MPE XL only)

254 = Invalid SNAnode name. (MPE XL only)

255 = Invalid security class name. (MPE XL only)

256 = Security class not properly configured. (MPE XL only)

257 = Program not authorized to use this security class. (MPE XL only)

258 = User is not authorized to use this security class. (MPE XL only)

301 = Illegal DB register.

304 = Security violation. (MPE V only)

305 = Parameter bounds violation.

308 = Session is inactive.

309 = No available AFT entry.

315 = Internal Error.

317 = NAU is inactive.

321 = Invalid class name. (MPE V only)

322 = Invalid session type.

324 = Invalid configuration access. (MPE V only)

326 = No available NAU.

329 = Inactive node or invalid node name.

336 = Link shutdown occurred.

337 = Protocol shutdown requested.

338 = Quiesce shutdown requested.

340 = No stack space.

351 = Link Failure occurred.

352 = Transport Internal Error Shutdown.

353 = Hierarchical Shutdown.

Description

Calling the OPEN3270 intrinsic is equivalent to turning on an IBM display station or printer.

The *terminalid* number, returned by the intrinsic, must be used to reference the device in subsequent SNA IMF intrinsics.

When you call the OPEN3270 intrinsic in non-transparent mode, an internal screen image is created that contains a description of all the fields on terminal screen. In non-transparent mode, the internal screen image can be accessed only through the following intrinsics:

ATTRLIST

SCREENATTR

FIELDATTR

READFIELD

READSCREEN

STREAM3270

WRITEFIELD

In transparent mode, the internal screen image contains the untranslated data stream, so you cannot use any of the intrinsics that access specific fields or field attributes. In transparent mode, the internal screen image can be accessed only through the following intrinsics:

READSTREAM

WRITESTREAM

NOTE The WRITESTREAM intrinsic is used only for LU.T2 sessions. LU.T1 sessions use the TRAN3270 intrinsic to transmit printer keys and do not use the WRITESTREAM intrinsic. LU.T3 emulated devices cannot send data to the host.

After you call OPEN3270, your internal screen image initially contains the SNA IMF unowned screen. This screen is equivalent to the IBM unowned screen. When an IBM 3278 display station displays the unowned screen, a question mark (?) appears on the screen in the operator information area. An unowned screen image on an HP terminal contains the SNA IMF banner and is made up of one protected field.

Immediately after issuing the OPEN3270 intrinsic, your program should call the RECV3270 intrinsic to obtain this initial screen with the banner message. If the host sends data to the device before you call RECV3270, the internal screen image sent by the host replaces the unowned screen and is the first screen you see.

You can customize the unowned screen with your own message. To customize this screen, use an editor to modify the unowned screen portion of the CATIMF.PUB.SYS file (message set 11) on your HP 3000. After you have designed your customized screen, run the HP 3000 MAKECAT or GENCAT utility to incorporate your changes into the catalog. The MAKECAT utility is only on MPE V and on MPE XL releases prior to 2.1; the GENCAT utility is only on MPE XL release 2.1 and later releases.

Each time you call the OPEN3270 intrinsic after you run MAKECAT or GENCAT, your HP terminal will display your customized screen. However, if data is received immediately from the host, that data will replace the unowned screen, perhaps before you even see it.

After you receive the first unowned screen, call TRAN3270 to send the **[SYS REQ]** key to acquire the SSCP-LU screen. See Appendix D , “Differences Between IMF/3000 and SNA IMF/V,” for more information on display screen ownership.

The OPEN3270 intrinsic can be called in transparent or non-transparent mode.

Although similar to the IMF/3000 OPEN3270 intrinsic, the SNA IMF OPEN3270 intrinsic has a different interface. For example, the SNA IMF *snalnkinfo* parameter replaces the IMF/3000 *confile* parameter and the SNA IMF *devicenum* parameter replaces the IMF/3000 *deviceid* parameter. See Appendix D , “Differences Between IMF/3000 and SNA IMF/V,” for details.

When using the SNA Server/Access product, the *netinfo* parameter replaces the SNA IMF *snalnkinfo* parameter. See the *HP SNA Server/Access User's Guide* for more specific information.

COBOL Calling Sequence

CALL "COPEN3270" USING DEVICENUM SNALNKINFO FLAGS TERMINALID DEVICETYPE FFINDEX SCREENSIZE TIMEOUT RESULT. (on MPE V and in compatibility mode on MPE XL)

CALL INTRINSIC "OPEN3270" USING DEVICENUM SNALNKINFO FLAGS TERMINALID DEVICETYPE FFINDEX SCREENSIZE TIMEOUT RESULT. (in native mode on MPE XL)

All parameters are numeric data items except *SNALNKINFO* and *TIMEOUT*. *SNALNKINFO* is an alphanumeric data item. *TIMEOUT* is a two-element numeric table.

FORTRAN Calling Sequence

CALL OPEN3270 (DEVICENUM, SNALNKINFO, FLAGS, TERMINALID, DEVTYPE, FFINDEX, SCREENSIZE, TIMEOUT, RESULT)

All parameters are integer variables except *SNALNKINFO* and *TIMEOUT*. *SNALNKINFO* is a character array. *TIMEOUT* is an integer array of two elements.

BASIC Calling Sequence

CALL BOPEN3270 (D1, C$, F, T, D2, F1, B, T2(*), R) (on MPE V and in compatibility mode on MPE XL)

CALL OPEN3270 (D1, C$, F, T, D2, F1, B, T2(*), R) (in native mode on MPE XL)

All parameters are integer variables except C$ and T2(*). C$ is a string variable. T2(*) is a numeric array of two integers.

SPL Calling Sequence

OPEN3270 (DEVICENUM, SNALNKINFO, FLAGS, TERMINALID, DEVTYPE, FFINDEX, SCREENSIZE, TIMEOUT, RESULT)

All parameters are integers except *SNALNKINFO* and *TIMEOUT*. *SNALNKINFO* is a byte array. *TIMEOUT* is an integer array of two elements.

Pascal Calling Sequence

OPEN3270 (DEVICENUM, SNALNKINFO, FLAGS, TERMINALID, DEVTYPE, FFINDEX, SCREENSIZE, TIMEOUT, RESULT);

All parameters are short integer variables except *SNALNKINFO* and *TIMEOUT*. *SNALNKINFO* is a packed array of char. *TIMEOUT* is an array of two short integer variables.

C/XL Calling Sequence

```
OPEN3270 (&DEVICENUM, SNALNKINFO, &FLAGS, &TERMINALID,
&DEVTYPE, &FFINDEX, &SCREENSIZE, TIMEOUT, &RESULT);
```

All parameters are of type short except *SNALINKINFO* and *TIMEOUT*. *SNALINKINFO* is a character array (a pointer to a char), and *TIMEOUT* is a short integer array (a pointer to a short).

Pascal Program Excerpts

Following are excerpts from a Pascal program that calls SNA IMF intrinsics. For examples of complete Pascal programs in non-transparent and transparent modes, see Appendix F, "Sample Programs."

```
{************************ Global Declarations ************************}
type
shortint      = -32768..32767;       { global type, two bytes (half word) }

   flags_type    = packed record        { flags may be global or local. }
      filler     : 0..1023;             { It's global in this example. }
      dbcs       : 0..1;                { Two bytes (half word). }
      unbind_op  : 0..1;
      LU1_3sup   : 0..1;
      int_trace  : 0..1;
      trans_mode : 0..1;
      IO_mode    : 0..1;
   end;
.
.
.
var
   terminalid    : shortint;
   result        : shortint;
.
.
.
procedure OPEN3270;  intrinsic;
.
.
.
```

```
{************************* Local Declarations *************************}
var
   snalnkinfo  : packed array[1..18] of char;   { All OPEN3270 variables }
   devicenum   : shortint;                      { except terminalid and }
   flags       : flags_type;                    { result are local. }
   devtype     : shortint;
   ffindex     : shortint;
   screensize  : shortint;
   timeout     : packed array[1..2] of shortint;
.
.
.
{************** Variable Initialization and Intrinsic Call **************}
snalnkinfo := 'SNANODE#SNACLASS ';
devicenum := -2;

with flags do
begin
   IO_mode := 0;
   trans_mode := 1;
   int_trace := 0;
   LU1_3sup := 0;
   unbind_op := 1;
   dbcs := 0;
   filler := 0;
end;

ffindex := 0;
timeout[1] := 30;
timeout[2] := 30;

OPEN3270 (devicenum, snalnkinfo, flags, terminalid, devtype, ffindex,
          screensize, timeout, result);
```

PRINT3270

PRINT3270 produces a hard copy of the internal screen image. The PRINT3270 intrinsic cannot be used during LU.T1 emulation or while in transparent mode.

Syntax

```
                I         I         I        CA         I          I
PRINT3270       (terminalid, fileid, action, location, priority, result)
```

Parameters

terminalid (input)

Integer identifying the terminal. The *terminalid* is returned in a call to the OPEN3270 intrinsic.

fileid (output and then input)

An integer returned by PRINT3270 that uniquely identifies the MPE file to which the copies of the internal screen image are written. *fileid* is input to all subsequent calls to PRINT3270.

action (input)

Integer that determines the action of PRINT3270:

0 = Open output file.

1 = Print the internal screen image and the location and characteristic of each attribute character.

2 = Print the internal screen image as it appears on the terminal, with attribute and null characters appearing as blanks.

3 = Print the attribute characteristics banner.

4 = Close the file.

location (input)

40-character string that is printed on the hard copy listing to help you determine the origin of a particular PRINT3270 call. If your string is less than 40 characters long, pad it with blanks until it contains exactly 40 characters. Any non-printing character is printed as a blank. The contents of *location* is printed if *action* is set to 1 or 2.

priority (input)

Integer specifying the output priority, which controls the order in which files are produced when several files are waiting for the same device. The value of *priority* can be from 1 through 13. Printing is deferred if *priority* is less than the current system outfence set by the console operator. The *priority* parameter is especially useful for deferring or cancelling the output from PRINT3270.

The *priority* parameter works only when *action = 0*; if a value other than 0 is used, *priority* is not checked. The *priority* parameter is passed to the **FOPEN** intrinsic as the output priority value in the *numbuffers* parameter. For more information on priority, see the description of the FOPEN intrinsic in the *MPE V Intrinsics Reference Manual*.

You may override the value in the *priority* parameter by specifying a file equation before starting your program. The formal file designator is LOGIMF.

result (output)

The following values can be generated by the PRINT3270 intrinsic:

0 = Successful completion.

1 = Device not open.

9 = Host modified screen since the last receive request. (MPE V only)

22 = BASIC calling sequence error has occurred.

25 = Intrinsic call made while in split stack mode.

26 = Intrinsic call made with the parameter value out of bounds.

29 = Called intrinsic with a request already outstanding. (No-wait I/O only)

30 = Internal error occurred in IMF intrinsic.

53 = Invalid intrinsic called for data stream mode device.

60 = Invalid spool file priority. Value must be between 1 and 13 inclusive.

61 = Failed to open PRINT3270 spool file.

62 = Failed to write to PRINT3270 spool file.

63 = The action parameter must be an integer between 0 and 4.

64 = Wrong file type for PRINT3270 output file.

65 = Failed to close PRINT3270 output spool file.

66 = Failed to open CATIMF.PUB.SYS.

67 = GENMESSAGE failed to extract message.

68 = Out of stack space. Increase your maxdata size.

Description

The PRINT3270 intrinsic is a powerful program testing aid that you can call at any time unless a TRAN3270 or RECV3270 call is outstanding. The PRINT3270 intrinsic sends a copy of your current internal screen image to a spooled output file. An appropriate time to call PRINT3270 is after a RECV3270 call, when the host has modified your internal screen image. The printed output helps you verify exactly what has been changed.

The *action* parameter determines the output of the PRINT3270 intrinsic. The values for the *action* parameter, and their effects on the output of the PRINT3270 intrinsic, are as follows:

action = 0

Opens the output file. Your first call to PRINT3270 must specify *action = 0*.

action = 1

Prints the internal screen image. This format is designed for program testing and shows all attribute character locations as overprinted characters. The hexadecimal values of the first 16 attribute characters (per row) are printed to the right of the internal screen image. Null characters are printed as tildes (~) to distinguish them from blank characters. Tildes may appear as dashes (-) on some printers. Two screen images are printed per page with a border indicating column and row numbers. The cursor is represented by an underline character. The cursor location is written at the bottom of the internal screen image, along with the state of the keyboard (enabled or disabled).

action = 2

Prints the internal screen image. This format is a clean print, with attribute and null characters appearing as blank characters. This format resembles the internal screen image as it would appear on a terminal screen. You may wish to use this format to document host

transmissions. This format prints only one screen per page with no border and no cursor.

action = 3

Prints an initial page that defines all possible hexadecimal values for attribute characters and the meaning of each value. After opening your output file with *action = 0*, call the PRINT3270 intrinsic again with *action = 3*.

action = 4

Closes the output file.

NOTE A spooled file does not print until you call the CLOSE3270 intrinsic or until you call PRINT3270 with the *action* parameter set to 4.

Your first call to the PRINT3270 intrinsic must specify an *action* value of 0 to open the output file. You can use more than one spooled output file within the same program, because the *fileid* parameter provides a unique identifier each time PRINT3270 is called with. By opening more than one output file, you can segregate internal screen images into separate files based on the data they contain or the devices to which they are directed.

The formal file designator for the output file that PRINT3270 opens is LOGIMF. The device class of LOGIMF is "LP." If device class "LP" is spooled, the output of PRINT3270 will be sent to the spooler. The LOGIMF file can be routed to disk instead of to the spooler with the following file equation:

```
:FILE LOGIMF,NEW;DEV=DISC;REC=-133,,,ASCII;NOCCTL;SAVE
```

The LOGIMF file may be equated to another file, which must have a record size of 133 bytes and contain ASCII data. It may not be a KSAM file.

You can use the *location* parameter, which is a character string, to identify the origin of the printed screen images. If you use a narrow width paper you should set the *location* parameter to 40 blanks.

NOTE The HP 2608 printer characteristics prevent correct overprinting of attribute characters in the same row as the cursor. This affects all attribute characters in columns up to and including the column where the cursor is located. The overprint will appear as a capital "H" in this circumstance. The hexadecimal attribute value on the right will help to determine that the "H" is an attribute character rather than a capital "H" text character.

The PRINT3270 intrinsic cannot be used during LU.T1 emulation or while in transparent mode. Use the PRINT3270 intrinsic only in non-transparent mode, during LU.T2 and LU.T3 sessions.

COBOL Calling Sequence

```
CALL "CPRINT3270" USING TERMINALID FILEID ACTION LOCATION
PRIORITY RESULT.
```
(on MPE V and in compatibility mode on MPE XL)

```
CALL INTRINSIC "PRINT3270" USING TERMINALID FILEID ACTION
LOCATION PRIORITY RESULT.
```
(in native mode on MPE XL)

All parameters are numeric data items except `LOCATION`, which is an alphanumeric data item.

FORTRAN Calling Sequence

```
CALL PRINT3270 (TERMINALID, FILEID, ACTION, LOCATION,
PRIORITY, RESULT)
```

All parameters are integer variables except `LOCATION`, which is a character array.

BASIC Calling Sequence

`CALL BPRINT3270 (T, F2, 03, L$, P2, R)` (on MPE V and in compatibility mode on MPE XL)

`CALL PRINT3270 (T, F2, 03, L$, P2, R)` (in native mode on MPE XL)

All parameters are integer variables except `L$`, which is a string variable.

SPL Calling Sequence

```
PRINT3270 (TERMINALID, FILEID, ACTION, LOCATION, PRIORITY,
RESULT)
```

All parameters are integer variables except `LOCATION`, which is a byte array.

Pascal Calling Sequence

```
PRINT3270 (TERMINALID, FILEID, ACTION, LOCATION, PRIORITY,
RESULT);
```

All parameters are short integer variables except `LOCATION`, which is a packed array of char.

C/XL Calling Sequence

```
PRINT3270 (&TERMINALID, &FILEID, &ACTION, LOCATION,
&PRIORITY, &RESULT);
```

All parameters are of type short, except `LOCATION`, which is an array of characters (a pointer to a char).

Pascal Program Excerpts

Following are excerpts from a Pascal program that calls SNA IMF intrinsics. For examples of complete Pascal programs in non-transparent and transparent modes, see Appendix F , “Sample Programs.”

```
{************************* Global Declarations *************************}
type
   shortint      = -32768..32767;       { global type, two bytes (half word) }
.
.
.
var
   fileid        : shortint;
   terminalid    : shortint;          { value returned by OPEN3270 intrinsic }
   result        : shortint;
.
.
.
procedure PRINT3270;  intrinsic;
.
.
.
{************************* Local Declarations **************************}
var
   action    : shortint;                    { All PRINT3270 variables except }
   location  : packed array[1..40] of char; { fileid, terminalid, and result }
   priority  : shortint;                  { are local. }
.
.
.
{************** Variable Initialization and Intrinsic Call **************}
action := 0;
location := 'SNAIMF Screen Image';
priority := 4;

PRINT3270 (terminalid, fileid, action, location, priority, result);
```

READFIELD

READFIELD reads a field of data from the internal screen image.

Syntax

```
                I          I       I         I
READFIELD       (terminalid, fieldnum, offset, maxinbuflen,

                  CA        I         I
                 inbuf, actinbuflen, result
```

Parameters

terminalid (input)

Integer identifying the terminal. The *terminalid* is returned in a call to the OPEN3270 intrinsic.

fieldnum (input)

Integer indicating the relative field number. Field number 1 is the field following the first attribute character in the internal screen image. Field numbers start at 1 and continue through *numfields* (returned by the SCREENATTR intrinsic), unless the screen is unformatted, which means that *numfields* and *fieldnum* are both zero. Put a zero in *fieldnum* if the screen is unformatted. (An unformatted screen has no attribute characters.)

offset (input)

Integer specifying the offset value, in characters, of the first character position read from the internal screen image. The first character position on the screen has an offset value of zero. Reading begins at the location you specify in *offset*. The *offset* parameter must be zero when the DBCS option to the OPEN3270 intrinsic is enabled.

maxinbuflen (input)

Integer specifying the maximum number of characters to return in *inbuf*.

inbuf (output)

Character array containing the contents of the field that was read. Trailing nulls are removed. Embedded and leading nulls are returned.

actinbuflen (output)

Integer indicating the actual number of characters returned in *inbuf*.

result (output)

The following values can be generated by the READFIELD intrinsic:

0 = Successful completion.

1 = Device not open.

9 = Host modified screen since the last receive request. (MPE V only)

11 = Non-existent field number was specified.

21 = Field offset specified is out of range.

22 = BASIC calling sequence error has occurred.

25 = Intrinsic call made while in split stack mode.

26 = Intrinsic call made with the parameter value out of bounds.

29 = Called intrinsic with a request already outstanding. (No-wait I/O only)

30 = Internal error occurred in IMF intrinsic.

53 = Invalid intrinsic called for data stream mode device.

100 = A bad (non-zero) offset was specified in the intrinsic call.

Description

The READFIELD intrinsic returns the contents of a specified field in the internal screen image. You pass the terminal ID, a field number, an offset within the field, and the maximum number of characters to transfer. The READFIELD intrinsic returns the contents of the field (unless it is non-display) and the actual length of the data returned. If the specified field is non-display, the *inbuf* parameter is filled with a number of nulls equal to the value in the *maxinbuflen* parameter.

By calling SCREENATTR before calling READFIELD, you can determine whether the screen is formatted, and if so, the number of fields in the screen. If the screen is unformatted (the SCREENATTR intrinsic has returned a *numfields* value of zero), you can read the screen by calling READFIELD with the *fieldnum* parameter set to zero.

If you specify a *maxinbuflen* of more characters than a field actually contains, READFIELD will return a number of characters equal to the number of characters in the field.

During printer emulation, you should routinely follow your last call to READFIELD (or to READSCREEN) with a call to RECV3270. The host cannot interrupt READFIELD with a data transmission during printer emulation, and you must call RECV3270 to indicate to the host that you are finished with the previous screen of data.

If your device is configured as an LU.T3 printer on the host side, your internal screen image may contain printer orders embedded in the data. Examples of printer orders are CR (carriage return) and NL (New Line). IBM 3287 printer models differ, so consult the *IBM 3270 Information Display System Data Stream Programmer's Reference* (IBM part number GA23-0059) for further information about printer orders.

NOTE When checking for LU.T3 printer orders in your screen data, remember that the internal screen image is in ASCII.

Use the READFIELD intrinsic in non-transparent mode.

COBOL Calling Sequence

CALL "CREADFIELD" USING TERMINALID FIELDNUM OFFSET MAXINBUFLEN INBUF ACTINBUFLEN RESULT. (on MPE V and in compatibility mode on MPE XL)

CALL INTRINSIC "READFIELD" USING TERMINALID FIELDNUM OFFSET MAXINBUFLEN INBUF ACTINBUFLEN RESULT. (in native mode on MPE XL)

All parameters are numeric data items except *INBUF*, which is an alphanumeric data item.

FORTRAN Calling Sequence

CALL READFIELD (TERMINALID, FIELDNUM, OFFSET, MAXINBUFLEN, INBUF, ACTINBUFLEN, RESULT)

All parameters are integer variables except *INBUF*, which is a character array.

BASIC Calling Sequence

`CALL BREADFIELD (T, N, O1, L7, I$, L8, R)` (on MPE V and in compatibility mode on MPE XL)

`CALL READFIELD (T, N, O1, L7, I$, L8, R)` (in native mode on MPE XL)

All parameters are integer variables except `I$`, which is a string variable.

NOTE Since BASIC cannot read a string of more than 255 characters, and since the length of a field on the internal screen may exceed 255 characters, the *maxinbuflen* parameter is provided to prevent the BASIC string variable from overflowing. The *offset* parameter permits you to choose which portion of the field to read. Multiple calls to READFIELD with different *offset* values can permit programs written in BASIC to read all of a field that is longer than 255 characters.

SPL Calling Sequence

```
READFIELD (TERMINALID, FIELDNUM, OFFSET, MAXINBUFLEN,
INBUF, ACTINBUFLEN, RESULT)
```

All parameters are integer variables except *INBUF*, which is a byte array.

Pascal Calling Sequence

```
READFIELD (TERMINALID, FIELDNUM, OFFSET, MAXINBUFLEN,
INBUF, ACTINBUFLEN, RESULT);
```

All parameters are short integer variables except *INBUF*, which is a packed array of char.

C/XL Calling Sequence

```
READFIELD (&TERMINALID, &FIELDNUM, &OFFSET, &MAXINBUFLEN,
INBUF, &ACTINBUFLEN, &RESULT);
```

All parameters are of type short, except *INBUF*, which is an array of characters (a pointer to a char).

Pascal Program Excerpts

Following are excerpts from a Pascal program that calls SNA IMF intrinsics. For examples of complete Pascal programs in non-transparent and transparent modes, see Appendix F , “Sample Programs.”

```
{************************* Global Declarations *************************}
type
   shortint      = -32768..32767;       { global type, two bytes (half word) }
.
.
.
var
   terminalid    : shortint;          { value returned by OPEN3270 intrinsic }
   result        : shortint;
.
.
.
procedure READFIELD;  intrinsic;
.
.
.
{************************* Local Declarations *************************}
var
   fieldnum      : shortint;            { All READFIELD variables }
   offset        : shortint;            { except terminalid and result }
   maxinbuflen   : shortint;            { are local. }
   inbuf         : packed array[1..1920] of char;
   actinbuflen   : shortint;
.
.
.
{************** Variable Initialization and Intrinsic Call **************}
fieldnum := 2;
offset := 0;
maxinbuflen := 80;
inbuf := ' ';

READFIELD (terminalid, fieldnum, offset, maxinbuflen, inbuf,
           actinbuflen, result);
```

READSCREEN

READSCREEN reads all or a specified part of the internal screen image.

Syntax

```
                I          I          I          CA
READSCREEN     (terminalid, offset, maxinbuflen, inbuf,

                I          I
  actinbuflen, result)
```

Parameters

terminalid (input)

Integer identifying the terminal. The *terminalid* is returned in a call to the OPEN3270 intrinsic.

offset (input)

Integer specifying the offset, in characters, into the internal screen image. The first character position on the screen has an offset value of zero. Reading begins at the location you specify in *offset*. The *offset* parameter must be zero when the DBCS option to the OPEN3270 intrinsic is enabled.

maxinbuflen (input)

Integer specifying the maximum number of characters to return in *inbuf*.

inbuf (output)

Character array containing the contents of the field that was read. Trailing nulls are removed. Embedded and leading nulls are returned.

actinbuflen (output)

Integer indicating the number of characters actually returned in *inbuf*.

result (output)

The following values can be generated by the READSCREEN intrinsic:

0 = Successful completion.

1 = Device not open.

9 = Host modified screen since the last receive request. (MPE V only)

21 = Field offset specified is out of range.

22 = BASIC calling sequence error has occurred.

25 = Intrinsic call made while in split stack mode.

26 = Intrinsic call made with the parameter value out of bounds.

29 = Called intrinsic with a request already outstanding. (No-wait I/O only)

30 = Internal error occurred in IMF intrinsic.

42 = Specified MAXINBUFLEN parameter is too large.

53 = Invalid intrinsic called for data stream mode device.

100 = A bad (non-zero) offset was specified in the intrinsic call.

Description

The READSCREEN intrinsic returns any portion or all of the internal screen image. Portions of the returned screen that are non-display fields will be filled with nulls.

READSCREEN is a position-oriented intrinsic; it reads a section of the screen beginning at a specified character position. READFIELD is a field-oriented intrinsic; it reads a specified field of the screen and selects the field by number rather than by offset into the internal screen image.

The image returned by READSCREEN begins at a location determined from the value of the *offset* parameter and extends across all columns and down each row, terminating at the last position in the last row and column. Therefore, if the last field on the screen wraps around to the top of the screen, and you ask for less than a full internal screen image, READSCREEN will not return all of the last field.

By calling SCREENATTR before calling READSCREEN, you can determine whether the screen is formatted, and if so, how many fields there are in the screen. If you call READSCREEN when the screen is formatted, field attributes will be interspersed with data in your *inbuf* string.

Field attribute characters can be located by calling the ATTRLIST intrinsic. The ATTRLIST intrinsic will return all the attribute character locations within all or part of the internal screen image. If the values you choose for the ATTRLIST parameters *offset* and *subscreensize* are the same as those used in the READSCREEN parameters *offset* and *maxinbuflen*, you will access the same screen subsection. If the *offsetlist* array in your call to ATTRLIST is too small, the array will be filled, and the actual number of attribute characters within the specified screen subsection will be returned in *actlistlen*.

Field attribute characters can also be located by calling the FIELDATTR intrinsic. FIELDATTR returns the location of a specified field and the information about that field, which is coded in its attribute character. If a field is non-display, the *displayattr* parameter of the FIELDATTR intrinsic will be 3.

During LU.T3 printer emulation, you should routinely call RECV3270 after you have finished reading the current internal screen image. Calling RECV3270 will cause SNA IMF to inform the host that your "printer" is ready to receive more data. The host must wait for a printer to finish before sending more data; so, your program, using a device configured as a printer, can control host transmissions through the RECV3270 intrinsic.

If your device is configured on the host as a printer, your internal screen image may contain printer orders embedded in the data. Examples of printer orders are CR (Carriage Return) and NL (New Line). IBM 3287 printer models differ, so consult the *IBM Information Display System Data Stream Programmer's Reference Manual* (IBM part number GA23-0059) for further information about printer orders.

NOTE When checking for LU.T3 printer orders in your screen data, remember that the internal screen image is in ASCII.

Call the READSCREEN intrinsic in non-transparent mode.

COBOL Calling Sequence

```
CALL "CREADSCREEN" USING TERMINALID OFFSET MAXINBUFLEN
INBUF ACTINBUFLEN RESULT.
```
(on MPE V and in compatibility mode on MPE XL)

```
CALL INTRINSIC "READSCREEN" USING TERMINALID OFFSET
MAXINBUFLEN INBUF ACTINBUFLEN RESULT.
```
(in native mode on MPE XL)

All parameters are numeric data items except *INBUF*, which is an alphanumeric data item.

FORTRAN Calling Sequence

```
CALL READSCREEN (TERMINALID, OFFSET, MAXINBUFLEN, INBUF,
ACTINBUFLEN, RESULT)
```

All parameters are integer variables except *INBUF*, which is a character array.

BASIC Calling Sequence

`CALL BREADSCREEN (T, O1, L7, I$, L8, R)` (on MPE V and in compatibility mode on MPE XL)

`CALL READSCREEN (T, O1, L7, I$, L8, R)` (in native mode on MPE XL)

All parameters are integer variables except `I$`, which is a string variable.

NOTE

Because BASIC cannot read a string of more than 255 characters, and because the length of a screen will exceed 255 characters, you can use the *maxinbuflen* parameter to prevent the BASIC string variable from overflowing. *offset* allows you to begin reading at the location you specify. Multiple calls to READSCREEN with different *offset* values can give BASIC programs the entire screen.

SPL Calling Sequence

```
READSCREEN (TERMINALID, OFFSET, MAXINBUFLEN, INBUF,
ACTINBUFLEN, RESULT)
```

All parameters are integer data items except *INBUF*, which is a byte array.

Pascal Calling Sequence

```
READSCREEN (TERMINALID, OFFSET, MAXINBUFLEN, INBUF,
ACTINBUFLEN, RESULT);
```

All parameters are short integers except for *INBUF*, which is a packed array of char.

C/XL Calling Sequence

```
READSCREEN (&TERMINALID, &OFFSET, &MAXINBUFLEN, INBUF,
&ACTINBUFLEN, &RESULT);
```

All parameters are of type short, except *INBUF*, which is a character array (a pointer to a char).

Pascal Program Excerpts

Following are excerpts from a Pascal program that calls SNA IMF intrinsics. For examples of complete Pascal programs in

non-transparent and transparent modes, see Appendix F , “Sample Programs.”

```
{************************* Global Declarations *************************}
type
shortint      = -32768..32767;       { global type, two bytes (half word) }
.
.
.
var
   terminalid    : shortint;          { value returned by OPEN3270 intrinsic }
   result        : shortint;
.
.
.
procedure READSCREEN;  intrinsic;
.
.
.
{************************* Local Declarations *************************}
var
   offset        : shortint;   { All READSCREEN variables except }
   maxinbuflen   : shortint;   { terminalid and result are local. }
   inbuf         : packed array[1..1920] of char;
   actinbuflen   : shortint;
.
.
.
{************** Variable Initialization and Intrinsic Call **************}
offset := 0;
maxinbuflen := 1920;
inbuf := ' ';

READSCREEN (terminalid, offset, maxinbuflen, inbuf, actinbuflen, result);
```

READSTREAM

READSTREAM reads all or part of the untranslated host data stream.

Syntax

```
                I          I           I         CA
READSTREAM     (terminalid, offset, maxinbuflen, inbuf,

                I          I
 actinbuflen, result)
```

Parameters

terminalid (input)

Integer identifying the terminal. The *terminalid* is returned in a call to the OPEN3270 intrinsic.

offset (input)

Integer indicating the offset, in characters, into the data stream. The first character in the data stream has an *offset* of zero. The data transfer begins at the location you specify in *offset*. The *offset* parameter must be zero when the DBCS option to the OPEN3270 intrinsic is enabled.

maxinbuflen (input)

Integer specifying the maximum number of characters to return in *inbuf*.

inbuf (output)

Character array containing the data stream.

actinbuflen (output

Integer indicating the number of characters actually returned in *inbuf*.

result (output)

The following values can be generated by the READSTREAM intrinsic:

0 = Successful completion.

1 = Device not open.

9 = Host modified screen since the last

receive request. (MPE V only)

21 = Field offset specified is out of range.

22 = BASIC calling sequence error has occurred.

25 = Intrinsic call made while in split stack mode.

26 = Intrinsic call made with the parameter value out of bounds.

29 = Called intrinsic with a request already outstanding. (No-wait I/O only)

30 = Internal error occurred in IMF intrinsic.

49 = Value specified for the INBUF parameter is too small to hold the entire data stream.

50 = Called READSTREAM without calling RECV3270 first.

52 = Data stream is too long.

54 = Device not opened in transparent mode.

100 = A bad (non-zero) offset was specified in the intrinsic call.

Description

To use the READSTREAM intrinsic, open a device in transparent mode by calling the OPEN3270 intrinsic with bit 14 of the *flags* parameter set to 1.

With SDLC protocol, data received from the host is contained in the Request Unit (RU). The RU is preceded by some header bytes known as the Request Header (RH). The RU contains the data stream commands and the data. Each LU.T2 and LU.T3 RU (or each RU chain, if RU chaining is used) from the host contains a 3270 data stream command. LU.T1 RUs contain SCS control codes instead of 3270 data stream commands.

SNA IMF removes the RH bytes and stores the rest of the RU in the internal screen image. For LU.T2 and LU.T3 sessions, the first byte in the internal screen image is the 3270 data stream write command.

To obtain host data, first call the RECV3270 intrinsic, and then call the READSTREAM intrinsic. After calling RECV3270, the stream of data received from the host is buffered in the internal screen image. Calling READSTREAM retrieves the data from the internal screen image. You can access all or any part of the RU through the READSTREAM intrinsic.

If the host attempts to write more data to your internal screen image before you have called RECV3270 to retrieve the data already stored there, SNA IMF rejects the command with sense code X'0820', "device busy." This rejection prevents the host from writing over a data stream that you have not received. You should call RECV3270 and READSTREAM for every RU received.

The READSTREAM intrinsic can be called only in transparent mode. See Chapter 2 , "Using SNA IMF Intrinsics," for more information about transparent mode.

COBOL Calling Sequence

CALL "CREADSTREAM" USING TERMINALID OFFSET MAXINBUFLEN INBUF ACTINBUFLEN RESULT. (on MPE V and in compatibility mode on MPE XL)

CALL INTRINSIC "READSTREAM" USING TERMINALID OFFSET MAXINBUFLEN INBUF ACTINBUFLEN RESULT. (in native mode on MPE XL)

All parameters are numeric data items except *INBUF*, which is an alphanumeric data item.

FORTRAN Calling Sequence

CALL READSTREAM (TERMINALID, OFFSET, MAXINBUFLEN, INBUF, ACTINBUFLEN, RESULT)

All parameters are integer variables except *INBUF*, which is a character array.

BASIC Calling Sequence

CALL BREADSTREAM (T, O1, L7, I$, L8, R) (on MPE V and in compatibility mode on MPE XL)

CALL READSTREAM (T, O1, L7, I$, L8, R) (in native mode on MPE XL)

All parameters are integer variables except I$, which is a string variable.

NOTE Because BASIC cannot read a string of more than 255 characters, and because the length of a screen will exceed 255 characters, you can use the *maxinbuflen* parameter to prevent the BASIC string variable from overflowing. *offset* allows you to choose the portion of the screen to read. Multiple calls to READSTREAM with different *offset* values can give BASIC programs the entire screen.

SPL Calling Sequence

```
READSTREAM (TERMINALID, OFFSET, MAXINBUFLEN, INBUF,
ACTINBUFLEN, RESULT)
```

All parameters are integer data items except *INBUF*, which is a byte array.

Pascal Calling Sequence

```
READSTREAM (TERMINALID, OFFSET, MAXINBUFLEN, INBUF,
ACTINBUFLEN, RESULT);
```

All parameters are short integers except for *INBUF*, which is a packed array of char.

C/XL Calling Sequence

```
READSTREAM (&TERMINALID, &OFFSET, &MAXINBUFLEN, INBUF,
&ACTINBUFLEN, &RESULT);
```

All parameters are of type short, except *INBUF*, which is a character array (a pointer to a char).

Pascal Program Excerpts

Following are excerpts from a Pascal program that calls SNA IMF intrinsics. For examples of complete Pascal programs in non-transparent and transparent modes, see Appendix F , “Sample Programs.”

```
{************************* Global Declarations *************************}
type
   shortint      = -32768..32767;       { global type, two bytes (half word) }
.
.
.
var
   terminalid    : shortint;          { value returned by OPEN3270 intrinsic }
   result        : shortint;
.
.
.
procedure READSTREAM;  intrinsic;
.
.
.
{************************* Local Declarations *************************}
var
   offset        : shortint;   { All READSTREAM variables except }
   maxinbuflen   : shortint;   { terminalid and result are local. }
   inbuf         : packed array[1..1920] of char;
   actinbuflen   : shortint;
.
.
.
{************** Variable Initialization and Intrinsic Call **************}
offset := 0;
maxinbuflen := 100;
inbuf := ' ';

READSTREAM (terminalid, offset, maxinbuflen, inbuf, actinbuflen, result);
```

RECV3270

RECV3270 allows your program to receive the screen after the host has modified it.

Syntax

```
                I          I
RECV3270        (terminalid, result)
```

Parameters

terminalid (input)

Integer identifying the terminal. The *terminalid* is returned in a call to the OPEN3270 intrinsic.

result (output)

The following values can be generated by the REVC3270 intrinsic:

0 = Successful completion.

1 = Device not open.

22 = BASIC calling sequence error has occurred.

23 = Keyboard enable timeout has occurred.

24 = Response timeout has occurred.

25 = Intrinsic call made while in split stack mode.

26 = Intrinsic call made with the parameter value out of bounds.

29 = Called intrinsic with a request already outstanding. (No-wait I/O only)

91 = LU.T1 bind received.

92 = Unbind received.

93 = LU.T3 bind received.

94 = HOLD PRINT timer has expired (10 minutes).

95 = Host application requests PA key entry.

301 = Illegal DB register.

```
302 = Invalid session.
305 = Parameter bounds violation.
306 = Invalid flag parameter.
308 = Session is inactive.
315 = Internal error.
319 = LU-SSCP message pending.
325 = Request pending.
332 = Privilege mode required.
336 = Link shutdown occurred.
337 = Protocol shutdown requested.
338 = Quiesce shutdown requested.
340 = No stack space.
351 = Link failure occurred.
352 = Transport internal error shutdown.
353 = Hierarchical shutdown.
```

Description

The RECV3270 intrinsic performs 3 functions:

1. It allows your program to suspend processing and wait for new data, and then it informs your program when data has arrived from the host.
2. It tells SNA IMF whether your program has received the host data since the last time the host updated the screen.
3. It allows a program emulating a printer (LU.T1 or LU.T3) to control host transmissions so that data in the internal screen image is not overwritten before your program has a chance to process it.

With standard (wait) MPE I/O, the RECV3270 intrinsic suspends your program until data arrives from the host or until the *timeout* interval specified in the OPEN3270 intrinsic has expired.

If your program calls the RECV3270 intrinsic without having set the receive timeout, and the host does not send any data, your program will be suspended indefinitely. Therefore, you should always set the receive timer portion of the *timeout* parameter of the OPEN3270 intrinsic to ensure that your program can regain control after an unresolved RECV3270 call.

If your program is emulating an IBM 3278 display station (LU.T2) with standard I/O, a call to RECV3270 suspends processing until a

transmission arrives from the host. Then, RECV3270 updates a counter telling SNA IMF that your program has seen the most recent internal screen image.

SNA IMF/XL and SNA IMF/V handle LU.T2 emulation differently. When a new transmission arrives from the host, SNA IMF/XL does not update the internal screen image unless your program has called RECV3270 and is waiting for data from the host. If your program has not called RECV3270, SNA IMF/XL waits for a RECV3270 call before it updates the internal screen image. SNA IMF/V updates the internal screen image whether or not your program has called RECV3270.

NOTE With no-wait I/O, you must follow each call to RECV3270 with a call to IOWAIT, IOWAIT3270, IODONTWAIT, or IODONTWAIT3270.

If SNA IMF/V receives host data and updates the internal screen image before your program calls RECV3270, it can tell from the counter that your program has not read the internal screen image since the last host transmission. If your program then calls, for example, READFIELD (or any intrinsic that accesses the internal screen image), SNA IMF/V will return a result code of 9, indicating that the host has updated the internal screen image since the last time your program received it.

Result code 9 tells your program that the field it just attempted to read belongs to a different screen image from the data it read previously. Whenever your program receives result code 9, it should call RECV3270. SNA IMF/V then updates the counter and allows your program to access the most recent screen image.

SNA IMF/XL does not return a result code of 9, so you must check the *result* parameter after a call to RECV3270 to determine whether the host has sent new data. If *result = 0*, the host has sent new data, and the internal screen image has been updated. If *result = 24*, the receive timer has expired, and the internal screen image has not been updated.

If your program is emulating a printer (LU.T1 or LU.T3), the RECV3270 intrinsic can prevent the IBM host from sending new data before your program has a chance to process the current screen image.

Ordinarily, SNA IMF sends a positive response to the IBM host after it receives a transmission, whether or not your program has called RECV3270. However, during printer emulation, SNA IMF delays sending its positive response until your program processes the new data and calls RECV3270 again.

During printer emulation (with standard I/O), the first call to RECV3270 suspends your program until data has arrived for it, then it informs your program that the host has updated the internal screen image. Your program can then call other SNA IMF intrinsics to check the screen, locate field attributes, read data, and so on, while the host is still waiting for a positive response. When your program has finished

processing the current screen image, it must call RECV3270 again. This RECV3270 call tells SNA IMF that your program is finished processing and is ready for more data. SNA IMF then sends a positive response to the IBM host, and the host can continue transmitting.

After your program calls the OPEN3270 intrinsic, it must immediately call the RECV3270 intrinsic. This first RECV3270 call receives the SNA IMF banner screen from the OPEN3270 intrinsic. This first internal screen image can be discarded, and you can continue with your processing.

In most cases, you should follow a call to TRAN3270 with a call to RECV3270 to receive any data that the host sends in response to your transmission. However, if you know that the host will not respond to your transmission, you can call RESET3270 to reenable the keyboard. (Successful calls to TRAN3270 disable the keyboard.)

The host might update the screen in more than one transmission. To ensure that you always receive a complete screen from the host, you might need to write your program to issue a variable number of RECV3270 intrinsic calls.

If the host system always sends a keyboard enable in the last transmission, you can set the keyboard enable timer in your call to the OPEN3270 intrinsic. Then you need to call RECV3270 only once to receive a complete screen image. The RECV3270 intrinsic will not complete until the keyboard enable has been received or the timer has run out.

NOTE

Be aware of any changes in your host system software. For example, TSO always sends the keyboard enable timer in the last transmission, but other applications might not. Also be aware of changes in protocol. If you migrate from BSC to SDLC, more host transmissions are involved in starting your SNA session. You may need to modify the logon routine in your program to issue more RECV3270 intrinsic calls when an SNA session is initiated.

The RECV3270 intrinsic can be called in either transparent or non-transparent mode.

With no-wait I/O, your program can call RECV3270 for several different devices. It can then call IOWAIT or IOWAIT3270, and it will suspend processing until data has arrived for one of the devices.

After calling RECV3270 with no-wait I/O, your program can call IODONTWAIT or IODONTWAIT3270, instead of IOWAIT or IOWAIT3270, and it will not be suspended. Each time your program calls IODONTWAIT or IODONTWAIT3270, SNA IMF checks to see if data has arrived for a device and informs your program. If no data has arrived, your program continues processing, and it must call IODONTWAIT or IODONTWAIT3270 again later to see if data has arrived.

With no-wait I/O, every call to RECV3270 must be followed by a call to IOWAIT, IOWAIT3270, IODONTWAIT, or IODONTWAIT3270. Result codes that would be returned through the RECV3270 intrinsic with standard wait I/O are returned through the IOWAIT, IOWAIT3270, IODONTWAIT, and IODONTWAIT3270 intrinsics with no-wait I/O.

For more information on using the ECV3270 intrinsic with no-wait I/O, see Chapter 4 , "Intrinsics Used with No-Wait I/O."

COBOL Calling Sequence

CALL CRECV3270 USING TERMINALID RESULT. (on MPE V and in compatibility mode on MPE XL)

CALL INTRINSIC "RECV3270" USING TERMINALID RESULT. (in native mode on MPE XL)

Both parameters are numeric data items.

FORTRAN Calling Sequence

```
CALL RECV3270 (TERMINALID, RESULT)
```

Both parameters are integer variables.

BASIC Calling Sequence

CALL BRECV3270(T, R) (on MPE V and in compatibility mode on MPE XL)

CALL RECV3270(T, R) (in native mode on MPE XL)

Both parameters are integer variables.

SPL Calling Sequence

```
RECV3270 (TERMINALID, RESULT)
```

Both parameters are integer variables.

Pascal Calling Sequence

```
RECV3270 (TERMINALID, RESULT);
```

Both parameters are short integer variables.

C/XL Calling Sequence

```
RECV3270 (&TERMINALID, &RESULT);
```

Both parameters are of type short.

Pascal Program Excerpts

Following are excerpts from a Pascal program that calls SNA IMF intrinsics. For examples of complete Pascal programs in non-transparent and transparent modes, see Appendix F , “Sample Programs.”

```
{************************* Global Declarations *************************}
type
   shortint      = -32768..32767;       { global type, two bytes (half word) }
.
.
.
var
   terminalid    : shortint;          { value returned by OPEN3270 intrinsic }
   result        : shortint;
.
.
.
procedure RECV3270;  intrinsic;
.
.
.
{************************* Intrinsic Call *************************}
RECV3270 (terminalid, result);
```

RESET3270

RESET3270 emulates pressing the **[RESET]** key on an IBM 3278 display station keyboard.

Syntax

```
                I        I
RESET3270       (terminalid, result)
```

Parameters

terminalid (input)

Integer identifying the terminal. The *terminalid* is returned in a call to the OPEN3270 intrinsic.

result (output)

The following values can be generated by the RESET3270 intrinsic:

0 = Successful completion.

1 = Device not open.

22 = BASIC calling sequence error has occurred.

25 = Intrinsic call made while in split stack mode.

26 = Intrinsic call made with the parameter value out of bounds.

29 = Called intrinsic with a request already outstanding. (No-wait I/O only)

30 = Internal error occurred in IMF intrinsic.

53 = Invalid intrinsic called for data stream mode device.

Description

Using the RESET3270 intrinsic is equivalent to pressing the **[RESET]** key on an IBM 3278 keyboard. The RESET3270 intrinsic unlocks the physical keyboard so you can issue intrinsics like WRITEFIELD and STREAM3270 that write to the internal screen image.

If input is inhibited, and you try to write to the internal screen image with an intrinsic like WRITEFIELD or STREAM3270, the logical keyboard

that exists in memory locks. Then, the only way you can release it is by calling the RESET3270 intrinsic. This intrinsic sets the input inhibit bit to zero, which is equivalent to receiving a keyboard enable command from the host.

The TRAN3270 intrinsic disables input when it executes successfully. It also disables input if it returns error codes 14, 16, or 30. Usually, the RECV3270 intrinsic is the first intrinsic called after a call to TRAN3270, and your program receives a keyboard enable in the transmission from the host. However, if you are not expecting the host to send data in response to your TRAN3270 call, or if your call to TRAN3270 did not execute successfully, you should call RESET3270 to enable input.

Use the RESET3270 intrinsic in non-transparent mode.

COBOL Calling Sequence

CALL "CRESET3270" USING TERMINALID RESULT. (on MPE V and in compatibility mode on MPE XL)

CALL INTRINSIC "RESET3270" USING TERMINALID RESULT. (in native mode on MPE XL)

Both parameters are numeric data items.

FORTRAN Calling Sequence

CALL RESET3270 (TERMINALID,RESULT)

Both parameters are integer variables.

BASIC Calling Sequence

CALL BRESET3270 (T, R) (on MPE V and in compatibility mode on MPE XL)

CALL RESET3270 (T, R) (in native mode on MPE XL)

Both parameters are integer variables.

SPL Calling Sequence

RESET3270 (TERMINALID, RESULT)

Both parameters are integer variables.

Pascal Calling Sequence

RESET3270 (TERMINALID, RESULT);

Both parameters are short integer variables.

C/XL Calling Sequence

RESET3270 (&TERMINALID, &RESULT);

Both parameters are of type short.

Pascal Program Excerpts

Following are excerpts from a Pascal program that calls SNA IMF intrinsics. For examples of complete Pascal programs in non-transparent and transparent modes, see Appendix F , “Sample Programs.”

```
{************************* Global Declarations *************************}
type
   shortint      = -32768..32767;       { global type, two bytes (half word) }
.
.
.
var
   terminalid    : shortint;          { value returned by OPEN3270 intrinsic }
   result        : shortint;
.
.
.
procedure RESET3270;  intrinsic;
.
.
.
{************************* Intrinsic Call *************************}
RESET3270 (terminalid, result);
```

SCREENATTR

SCREENATTR returns information about the attributes of the internal screen image.

Syntax

```
                I            I            I            I
SCREENATTR     (terminalid, printformat, startprint, soundalarm,

                I            I            I            I
 keyboardlock, numfields, screenstatus, cursorrow,

                I            I
 cursorcolumn, result)
```

Parameters

terminalid (input)

Integer identifying the terminal. The *terminalid* is returned in a call to the OPEN3270 intrinsic.

printformat (output)

Integer specifying the line length for the printer. Possible values are as follows:

0 = the new line (NL), end of message (EM), and carriage return (CR) printer orders in the data fields determine line length. The default is a 132-character print line if you do not specify any printer format.

1 = specifies a 40-character print line.

2 = specifies a 64-character print line.

3 = specifies an 80-character print line.

startprint (output)

Integer indicating whether the printer is to print the contents of its buffer. This parameter is ignored if the host orders printing to start on a device that is not declared as a printer.

1 = The device is to print the contents of its buffer.

0 = The host program has not ordered the device to start printing.

soundalarm (output)

Integer indicating whether the device is to send an audible beep after it receives data.

1 = The device will send an audible beep.

0 = The host has not requested that the device send a beep.

keyboardlock (output)

Integer indicating whether input is disabled.

1 = Input is disabled (keyboard is locked). You cannot call TRAN3270 or any intrinsics that write to the internal screen image, unless the RESET3270 intrinsic has been called since the last receipt of data from the host.

0 = Input is not disabled (keyboard is unlocked).

numfields (output)

Integer indicating the number of protected and unprotected fields in the internal screen image. If *numfields = 0*, the internal screen image is unformatted, and you should specify field number 0 in your calls to the READFIELD and WRITEFIELD intrinsics.

screenstatus (output)

Integer indicating whether the internal screen image has changed since the last call to SCREENATTR.

0 = No change to either data or attribute characters.

1 = Data, attribute characters, or both have been changed.

The *screenstatus* parameter keeps you informed about changes in the internal screen image from one RECV3270 call to the next. You should call SCREENATTR and check the *screenstatus* parameter immediately after calling RECV3270. If *screenstatus = 1*, the internal screen image has been changed.

If the host sends a write command and a Write Control Character (WCC) without any data (for example, if the host transmission is just to enable the keyboard), the internal screen image will be unchanged, and *screenstatus* will return 1.

The only time the host can change the screen without sending data is when the WCC instructs the control unit to reset the Modified Data Tags (MDTs). Resetting the MDTs changes the field attributes, which in turn changes the screen. In this case, *screenstatus* will return 1.

The *screenstatus* parameter is initialized to 0 with each call to SCREENATTR.

If any SNA IMF/V intrinsic call fails with *result = 9*, this means that the host has sent a new screen image since the last RECV3270 request. The *screenstatus* parameter will be set to zero, which is meaningless in this case. You must call RECV3270 to make the new internal screen image accessible to SNA IMF intrinsics. Then call SCREENATTR and check the *screenstatus* parameter for changes to the internal screen image.

cursorrow (output)

Integer indicating the row where the cursor is located:

0 through 11 (480-character screen)

0 through 23 (1920-character screen)

0 through 42 (3440-character screen)

cursorcolumn (output)

Integer indicating the column where the cursor is located:

0 through 39 (480-character screen)

0 through 79 (1920-character screen)

0 through 79 (3440-character screen)

result (output)

The following values can be generated by the SCREENATTR intrinsic:

0 = Successful completion.

1 = Device not open.

9 = Host modified screen since last receive request. (MPE V only)

22 = BASIC calling sequence error has occurred.

25 = Intrinsic call made while in split stack mode.

26 = Intrinsic call made with the parameter value out of bounds.

29 = Called intrinsic with a request already outstanding. (No-wait I/O only)

30 = Internal error occurred in IMF intrinsic.

53 = Invalid intrinsic called for data stream mode device.

Description

The SCREENATTR intrinsic returns information about the current contents of the internal screen image. SCREENATTR returns the following information:

- The last Write Control Character (WCC) received from the host. Each write command includes a WCC byte, which specifies 4 things:
  1. The number of lines per printed page (the *printformat* parameter).
  2. Whether the host has requested that a device start printing the contents of its buffer. (the *startprint* parameter).
  3. Whether the device is to send a beep after it receives data. (the *soundalarm* parameter).
  4. Whether input is to be enabled after the device receives data. (the *keyboardlock* parameter).
- The number of fields defined in the internal screen image (the *numfields* parameter). This parameter returns 0 for an unformatted screen.
- The *screenstatus* parameter, which indicates host changes to the internal screen image.
- The current screen address of the cursor, which is returned in the *cursorrow* and *cursorcolumn* parameters.

You should call the SCREENATTR intrinsic after every call to the RECV3270 intrinsic. Call the SCREENATTR intrinsic in non-transparent mode.

COBOL Calling Sequence

CALL "CSCREENATTR" USING TERMINALID PRINTFORMAT STARTPRINT SOUNDALARM KEYBOARDLOCK NUMFIELDS SCREENSTATUS CURSORROW CURSORCOLUMN RESULT. (on MPE V and in compatibility mode on MPE XL)

```
CALL INTRINSIC "SCREENATTR" USING TERMINALID PRINTFORMAT
STARTPRINT SOUNDALARM KEYBOARDLOCK NUMFIELDS SCREENSTATUS
CURSORROW CURSORCOLUMN RESULT.
```
(in native mode on MPE XL)

All parameters are numeric data items.

FORTRAN Calling Sequence

```
CALL SCREENATTR (TERMINALID, PRINTFORMAT, STARTPRINT,
SOUNDALARM, KEYBOARDLOCK, NUMFIELDS, SCREENSTATUS,
CURSORROW, CURSORCOLUMN, RESULT)
```

All parameters are integer variables.

BASIC Calling Sequence

```
CALL BSCREENATTR (T, P, P1, A, K, N9, S9, R9, C9, R)
```
(on MPE V and in compatibility mode on MPE XL)

```
CALL SCREENATTR (T, P, P1, A, K, N9, S9, R9, C9, R)
```
(in native mode on MPE XL)

All parameters are integer variables.

SPL Calling Sequence

```
SCREENATTR (TERMINALID, PRINTFORMAT, STARTPRINT,
SOUNDALARM, KEYBOARDLOCK, NUMFIELDS, SCREENSTATUS,
CURSORROW, CURSORCOLUMN, RESULT)
```

All parameters are integer variables.

Pascal Calling Sequence

```
SCREENATTR (TERMINALID, PRINTFORMAT, STARTPRINT,
SOUNDALARM, KEYBOARDLOCK, NUMFIELDS, SCREENSTATUS,
CURSORROW, CURSORCOLUMN, RESULT);
```

All parameters are short integer variables.

C/XL Calling Sequence

```
SCREENATTR (&TERMINALID, &PRINTFORMAT, &STARTPRINT,
&SOUNDALARM, &KEYBOARDLOCK, &NUMFIELDS, &SCREENSTATUS,
&CURSORROW, &CURSORCOLUMN, &RESULT);
```

All parameters are of type short.

Pascal Program Excerpts

Following are excerpts from a Pascal program that calls SNA IMF intrinsics. For examples of complete Pascal programs in non-transparent and transparent modes, see Appendix F, "Sample Programs."

```
{************************* Global Declarations *************************}
type
   shortint      = -32768..32767;       { global type, two bytes (half word) }
.
.
.
var
   terminalid    : shortint;          { value returned by OPEN3270 intrinsic }
   result        : shortint;
   cursorrow     : shortint;
   cursorcolumn  : shortint;
.
.
.
procedure SCREENATTR;  intrinsic;
.
.
.
{************************* Local Declarations *************************}
var
   printformat   : shortint;           { All SCREENATTR variables }
   startprint    : shortint;           { except terminalid, result, }
   soundalarm    : shortint;           { cursorrow, and cursorcolumn }
   keyboardlock  : shortint;           { are local. }
   numfields     : shortint;
   screenstatus  : shortint;
.
.
.
{************************* Intrinsic Call *************************}
SCREENATTR (terminalid, printformat, startprint, soundalarm, keyboardlock,
            numfields, screenstatus, cursorrow, cursorcolumn, result);
```

STREAM3270

STREAM3270 emulates typing a series of keystrokes on an IBM 3278 display station keyboard.

```
                I           I           I           CA
STREAM3270      (terminalid, cursorrow, cursorcolumn, outbuf,

                I           I           I
 outbuflen, numprocessed, result)
```

Parameters

terminalid (input)

Integer identifying the terminal. The *terminalid* is returned in a call to the OPEN3270 intrinsic.

cursorrow (input and then output)

Integer indicating the row where the cursor: is located

0 through 11 (480-character screen)

0 through 23 (1920-character screen)

0 through 42 (3440-character screen)

When passed (input), the cursor is placed at the position you specify. When returned (output), *cursorrow* is set to the final location of the cursor after data has been entered. (The cursor moves as data is entered, as if the data were being typed at a keyboard.)

cursorcolumn (input and then output)

Integer indicating the column where the cursor is located:

0 through 39 (480-character screen)

0 through 79 (1920-character screen)

0 through 79 (3440-character screen)

When passed (input), the cursor is placed at the position you specify. When returned (output), *cursorcolumn* is set to the final location of the cursor after data has been entered. (The cursor moves as data is entered, as if the data were being typed at a keyboard.)

outbuf (input)

Character array containing simulated keyboard input. The allowable character codes are as follows:

23 through 176 (octal)

241 through 377 (octal)

13 through 7E (hexadecimal)

A1 through FF (hexadecimal)

An illegal character code causes the remainder of *outbuf* to be ignored. Most character codes are data that will be entered into a field. Remaining character codes manipulate the internal screen image or its state. If you attempt to enter data into a protected field, the *result* parameter returns 10, and input is not disabled.

You can use the end-of-stream character (octal 23) to terminate the *outbuf* string before *outbuflen* characters are put into the internal screen image. The *numprocessed* parameter indicates the number of characters in *outbuf* that were successfully processed. The end-of-stream character is not included in the count returned in *numprocessed*.

Except for the end-of-stream character, each character code is equivalent to a key on a 3278 keyboard. Table 3-2 lists the hexadecimal and octal values allowed in the *outbuf* string, and it gives the equivalent 3278 key for each value.

Table 3-3 STREAM3270 Character Codes

Hexadecimal Code	Octal Code	Equivalent 3280 Key
14	24	**[ERASE INPUT]**
15	25	**[ERASE EOF]**
16	26	Tab key
17	27	Back tab key
18	30	Back space
19	31	Cursor (right)
1A	32	Cursor (up)
1B	33	Cursor (down)
1C	34	**[DUP]**
1D	35	**[HOME]**
1E	36	**[Field Mark]**
1F	37	**[RESET]** (use as first character of *outbuf* only)
20–7E	38–176	ASCII graphic characters
7F–A0	177–240	invalid codes

outbuflen (input)

Integer specifying the length of the *outbuf* string, in characters.

numprocessed (output)

Integer indicating the number of successfully processed characters in *outbuf*. The count does not include the end-of-stream character (octal 23).

result (output)

The following values can be generated by the STREAM3270 intrinsic:

0 = Successful completion.

1 = Device not open.

9 = Host modified screen since last receive request. (MPE V only)

10 = Attempt made to update a protected field.

12 = Invalid character in field or data stream.

14 = Attempt made to update a field or transmit from an LU.T3 printer.

16 = Invalid cursor address was specified.

17 = Attempt made to write to a field where input is inhibited.

22 = BASIC calling sequence error has occurred.

25 = Intrinsic call made while in split stack mode.

26 = Intrinsic call made with the parameter value out of bounds.

29 = Called intrinsic with a request already outstanding. (No-wait I/O only)

30 = Internal error occurred in IMF intrinsic.

38 = Cannot start OUTBUF on an attribute byte.

53 = Invalid intrinsic called for data stream mode device.

Description

The STREAM3270 intrinsic accepts an array of characters (the *outbuf* parameter), interprets it as a series of keystrokes, and applies the keystrokes to the internal screen image. STREAM3270 lets you enter data into any unprotected field, erase data, and enable the keyboard.

You can use either STREAM3270 or WRITEFIELD to enter data into your internal screen image. The WRITEFIELD intrinsic allows you to access a field by its number and write only to that field. The STREAM3270 intrinsic allows you to move across the internal screen image and enter data into more than one field with a single intrinsic call.

STREAM3270 does not cause data to be sent to the host; it just modifies the internal screen image. Use TRAN3270 to send the internal screen image to the host.

You can move the cursor in two ways: (1) by setting the initial cursor location in the *cursorrow* and *cursorcolumn* parameters, and (2) by using the octal codes in the *outbuf* parameter to move the cursor within your data stream. Because it simulates keyboard input, STREAM3270 moves the cursor as it enters data into the internal screen image. The *cursorrow* and *cursorcolumn* parameters return the final position of the cursor.

STREAM3270 does not permit you to write over an attribute character or to update a protected field. You will receive result code 38 if you attempt to start STREAM3270 on an attribute character. If STREAM3270 finds an attribute character in *outbuf* while it is writing into a screen, it automatically skips over the character. If the next character in *outbuf* is a data character, STREAM3270 writes the character in the next field (assuming it is unprotected). If the field is protected, your program receives result code 10.

Call the STREAM3270 intrinsic in non-transparent mode.

COBOL Calling Sequence

```
CALL "CSTREAM3270" USING TERMINALID CURSORROW CURSORCOLUMN
OUTBUF OUTBUFLEN NUMPROCESSED RESULT.
```
(on MPE V and in compatibility mode on MPE XL)

```
CALL INTRINSIC "STREAM3270" USING TERMINALID CURSORROW
CURSORCOLUMN OUTBUF OUTBUFLEN NUMPROCESSED RESULT.
```
(in native mode on MPE XL)

All parameters are numeric data items except *OUTBUF*, which is an alphanumeric data item.

FORTRAN Calling Sequence

```
CALL STREAM3270 (TERMINALID, CURSORROW, CURSORCOLUMN,
OUTBUF, OUTBUFLEN, NUMPROCESSED, RESULT)
```

All parameters are integer variables except for *OUTBUF*, which is a character array.

BASIC Calling Sequence

```
CALL BSTREAM3270 (T, R9, C9, O$, L9, N5, R)
```
(on MPE V and in compatibility mode on MPE XL)

```
CALL STREAM3270 (T, R9, C9, O$, L9, N5, R)
```
(in native mode on MPE XL)

All parameters are integer variables except for O$, which is a string variable.

SPL Calling Sequence

```
STREAM3270 (TERMINALID, CURSORROW, CURSORCOLUMN, OUTBUF,
OUTBUFLEN, NUMPROCESSED, RESULT)
```

All parameters are integer variables except *OUTBUF*, which is a byte array.

Pascal Calling Sequence

```
STREAM3270 (TERMINALID, CURSORROW, CURSORCOLUMN, OUTBUF,
OUTBUFLEN, NUMPROCESSED, RESULT);
```

All parameters are short integer variables except *OUTBUF*, which is a packed array of char.

C/XL Calling Sequence

```
STREAM3270 (&TERMINALID, &CURSORROW, &CURSORCOLUMN, OUTBUF,
&OUTBUFLEN, &NUMPROCESSED, &RESULT);
```

All parameters are of type short, except *OUTBUF*, which is an array of characters (a pointer to a char).

Pascal Program Excerpts

Following are excerpts from a Pascal program that calls SNA IMF intrinsics. For examples of complete Pascal programs in non-transparent and transparent modes, see Appendix F , “Sample Programs.”

```
{************************* Global Declarations *************************}
type
   shortint      = -32768..32767;       { global type, two bytes (half word) }
.
.
.
var
   terminalid    : shortint;          { value returned by OPEN3270 intrinsic }
   result        : shortint;
   cursorrow     : shortint;
   cursorcolumn  : shortint;
.
.
.
procedure STREAM3270;  intrinsic;
.
.
.
{************************* Local Declarations *************************}
var
   outbuf        : packed array[1..MAXBUFSIZE] of char;   { The variables }
   outbuflen     : shortint;             { terminalid, result, cursorrow, }
   numprocessed  : shortint;                 { and cursorcolumn are global. }
.
.
.
{************** Variable Initialization and Intrinsic Call **************}
outbuflen := 51;
outbuf := 'logon applid(tso) data(sales/sales) logmode(imf2k) ';
                      { Assume that cursorrow and cursorcolumn }
                      { are set outside of this local procedure. }

STREAM3270 (terminalid, cursorrow, cursorcolumn, outbuf, outbuflen,
            numprocessed, result);
```

TRAN3270

TRAN3270 emulates pressing one of the transmit keys (listed below) on a 3278 display station (LU.T2) or a 3287 printer (LU.T1).

Syntax

```
    I         I        I       I           I         I
TRAN3270 (terminalid, aid, cursorrow, cursorcolumn, result)
```

Parameters

terminalid (input)

Integer identifying the terminal. The *terminalid* is returned in a call to the OPEN3270 intrinsic.

aid (input)

Integer specifying the Attention Identifier (AID) that will be sent to the host. Values are provided for LU.T2 terminals and LU.T1 printers. LU.T3 printers do not transmit keys or data to the host. Two LU.T1 printer keys, **[ENABLE]** and **[HOLD]**, cannot be used unless bit 12 of the OPEN3270 *flags* parameter is set to 1. The value -1 can be specified *aid* values are as follows:

Terminal keys (LU.T2 values)

-1 = Inhibit AID and cursor position transmission (transparent mode only)

-3 = Send a positive response, if one has not already been sent (transparent mode only)

-2 = **[ATTN]**	55 = **[PF7]**	67 = **[PF15]**	46 = **[PF23]**
39 = **[ENTER]**	56 = **[PF8]**	68 = **[PF16]**	60 = **[PF24]**
49 = **[PF1]**	57 = **[PF9]**	69 = **[PF17]**	37 = **[PA1]**
50 = **[PF2]**	58 = **[PF10]**	70 = **[PF18]**	62 = **[PA2]**
51 = **[PF3]**	35 = **[PF11]**	71 = **[PF19]**	44 = **[PA3]**
52 = **[PF4]**	64 = **[PF12]**	72 = **[PF20]**	95 = **[CLEAR]**
53 = **[PF5]**	65 = **[PF13]**	73 = **[PF21]**	48 = **[SYS REQ]**
54 = **[PF6]**	66 = **[PF14]**	91 = **[PF22]**	

Printer keys (LU.T1 values)

10 = **[HOLD]**

-11 = **[PA1]**

-12 = **[PA2]**

-13 = **[CANCEL]**

-14 = **[ENABLE]**

cursorrow (input)

Integer specifying the row where the cursor is located (for LU.T2):

0 through 11 (480-character screen)

0 through 23 (1920-character screen)

0 through 42 (3440-character screen)

Specify a value of -1 for transparent mode. This value does not send a cursor address. *cursorcolumn* must also be set to -1 to prevent sending the cursor address.

cursorcolumn (input)

Integer indicating the column where the cursor is located (for LU.T2):

0 through 39 (480-character screen)

0 through 79 (1920-character screen)

0 through 79 (3440-character screen)

Specify a value of -1 for transparent mode. This value does not send a cursor address. *cursorrow* must also be set to -1 to prevent sending the cursor address.

result (output)

The following values can be generated by the TRAN3270 intrinsic:

0 = Successful completion.

1 = Device not open.

9 = Host modified screen since last receive request. (MPE V only)

14 = Attempt made to update a field or transmit from an LU.T3 printer.

15 = Invalid AID parameter was sent.

16 = Invalid cursor address was specified.

17 = Attempt made to write to a field where input is inhibited.

22 = BASIC calling sequence error has occurred.

24 = Response timeout has occurred.

25 = Intrinsic call made while in split stack mode.

26 = Intrinsic call made with the parameter value out of bounds.

29 = Called intrinsic with a request already outstanding. (No-wait I/O only)

30 = Internal error occurred in IMF intrinsic.

43 = Transparent mode not requested for LU.T1 emulation.

51 = Called TRAN3270 without calling WRITESTREAM first.

80 = Your session is in receive state and cannot send data.

82 = Your screen is in 'unowned' state so it cannot send data.

301 = Illegal DB register.

302 = Invalid session.

305 = Parameter bounds violation.

308 = Session is inactive.

310 = Bad PI in RH.

311 = Bad BCI in RH.

312 = Bad ECI in RH.

313 = Bad EDI in RH.

314 = Reserved bits in RH must be set to zero.

315 = Internal error.

316 = Invalid RU size.

317 = NAU is inactive.

319 = LU-SSCP message pending.

323 = Negative LU-SSCP response.

325 = Request pending.

```
330 = Illegal call.
332 = Privilege mode required.
336 = Link shutdown occurred.
337 = Protocol shutdown requested.
338 = Quiesce shutdown requested.
340 = No stack space.
351 = Link failure occurred.
352 = Transport internal error shutdown.
353 = Hierarchical shutdown.
400 = Expedited response pending.
401 = Data traffic inactive.
402 = SDT request not received.
403 = Invalid session control protocol.
404 = RQR request pending.
405 = STSN request not pending.
407 = Unsupported CRV request/response.
408 = Unsupported session control request.
```

Description

Using the TRAN3270 intrinsic is equivalent to pressing one of the 3278 terminal or 3287 printer keys, which causes changed screen data to be sent to the host when it polls the devices. TRAN3270 emulates these keys:

- **[ENTER]** key
- Program Function keys (**[PF1]** through **[PF24]**)
- Program Attention keys (**[PA1]**, **[PA2]**, **[PA3]**)
- **[HOLD PRINT]** key
- **[ENABLE PRINT**] key
- **[CANCEL]** key
- **[CLEAR]** key
- **[SYS REQ]** key

The TRAN3270 also allows you to specify a cursor address along with the AID to be sent to the host.

If the TRAN3270 intrinsic executes successfully, or if it returns a result code of 14, 16, or 30, it automatically disables the keyboard, preventing your program from calling any intrinsics that write to the internal screen image.

After calling the TRAN3270 intrinsic, the RECV3270 intrinsic is usually the first SNA IMF intrinsic you call. Normally, the host reenables the keyboard when it sends data in response to your transmission. If you are not expecting the host to respond to your transmission, or if the TRAN3270 intrinsic fails with result code 14, 16, or 30, you can reenable input by calling the RESET3270 intrinsic.

For SNA IMF/V, if the host modifies your internal screen image just before or just after you call TRAN3270, new host data might arrive before the TRAN3270 intrinsic completes execution. You will receive a result code 9, indicating that the host has modified your internal screen image, and your data will not be sent. Your program must then call the RECV3270 intrinsic, which will complete immediately.

In transparent mode, SNA IMF delays sending a positive response to a host transmission until your program calls either RECV3270 or TRAN3270. If your program has no data to send or receive, you can call TRAN3270 with an *aid* value of -3 to send a positive response to the host.

In transparent mode, you can inhibit sending the AID key or the cursor address as follows:

- To inhibit transmission of both the AID key and the cursor address, set *aid* to -1. The *cursorrow* and *cursorcolumn* parameters are not used.
- To allow transmission of the AID key and inhibit transmission of the cursor address, set *aid* to a legitimate value and set both *cursorrow* and *cursorcolumn* to -1.

During LU.T2 sessions in transparent mode, if you are not suppressing the cursor, and if you are sending a **[CLEAR]** or **PA** key, call WRITESTREAM with the *outbuflen* parameter set to 0 before calling TRAN3270.

In transparent mode, SNA IMF will not check the AID key to determine whether it is a **[CLEAR]** or **PA** key.

When you specify the **[CLEAR]** key, the internal screen image is cleared (filled with null characters) before the **[CLEAR]** AID is sent to the host. When you send the **[CLEAR]** and **PA** keys in non-transparent mode, only the AID is sent to the host; no data is sent.

Call the TRAN3270 intrinsic in transparent or non-transparent mode.

COBOL Calling Sequence

```
CALL “CTRAN3270” USING TERMINALID AID CURSORROW
CURSORCOLUMN RESULT.
```
(on MPE V and in compatibility mode on MPE XL)

```
CALL INTRINSIC "TRAN3270" USING TERMINALID AID CURSORROW
CURSORCOLUMN RESULT.
```
(on MPE V and in compatibility mode on MPE XL)

All parameters are numeric data items.

FORTRAN Calling Sequence

```
CALL TRAN3270 (TERMINALID, AID, CURSORROW, CURSORCOLUMN,
RESULT)
```

All parameters are integer variables.

BASIC Calling Sequence

```
CALL BTRAN3270 (T, A9, R9, C9, R)
```
(on MPE V and in compatibility mode on MPE XL)

```
CALL TRAN3270 (T, A9, R9, C9, R)
```
(in native mode on MPE XL)

All parameters are integer variables.

SPL Calling Sequence

```
TRAN3270 (TERMINALID, AID, CURSORROW, CURSORCOLUMN, RESULT)
```

All parameters are integer variables.

Pascal Calling Sequence

```
TRAN3270 (TERMINALID, AID, CURSORROW, CURSORCOLUMN,
RESULT);
```
All parameters are short integer variables.

C/XL Calling Sequence

```
TRAN3270 (&TERMINALID, &AID, &CURSORROW, &CURSORCOLUMN,
&RESULT);
```

All parameters are of type short.

Pascal Program Excerpts

Following are excerpts from a Pascal program that calls SNA IMF intrinsics. For examples of complete Pascal programs in non-transparent and transparent modes, see Appendix F , “Sample Programs.”

```
************************* Global Declarations *************************}
type
   shortint      = -32768..32767;       { global type, two bytes (half word) }
.
.
.
var
   terminalid    : shortint;          { value returned by OPEN3270 intrinsic }
   result        : shortint;
   cursorrow     : shortint;
   cursorcolumn  : shortint;
.
.
.
procedure TRAN3270;  intrinsic;
.
.
.
{************************* Local Declarations *************************}
var
   aid    : shortint;      { All other TRAN3270 variables (terminalid, }
                           { result, cursorrow, and cursorcolum) }
                           { are declared as global. }
.
.
.
{************** Variable Initialization and Intrinsic Call **************}
aid := 48;                 { Assume that cursorrow and }
                           { cursorcolumn are set outside of this }
                           { local procedure. }

TRAN3270 (terminalid, aid, cursorrow, cursorcolumn, result);
```

VERS3270

VERS3270 identifies the software version number of SNA IMF that is running on the HP 3000.

Syntax

```
        CA
VERS3270  (version)
```

Parameters

version (output)

14-character array identifying the SNA IMF version number. The format is as follows:

(delta character)DCSIMFV.uu.ff

(delta character) is a space.

V is a letter that indicates the version.

uu is a number that indicates the update level.

ff is a number that indicates the fix level.

Description

The VERS3270 intrinsic returns the version number of the SNA IMF intrinsics. Although intrinsics are shared by both IMF/3000 and SNA IMF, internal checks ensure that the correct version of the product is used.

COBOL Calling Sequence

CALL "CVERS3270" USING VERSION. (on MPE V and in compatibility mode on MPE XL)

CALL INTRINSIC "VERS3270" USING VERSION. (in native mode on MPE XL)

VERSION is an alphanumeric data item.

FORTRAN Calling Sequence

CALL VERS3270 (VERSION)

VERSION is a character array.

BASIC Calling Sequence

CALL BVERS3270 (V$) (on MPE V and in compatibility mode on MPE XL)

CALL VERS3270 (V$) (in native mode on MPE XL)

V$ is a string variable.

SPL Calling Sequence

VERS3270 (VERSION)

VERSION is a byte array.

Pascal Calling Sequence

VERS3270 (VERSION);

VERSION is a packed array of char.

C/XL Calling Sequence

VERS3270 (VERSION);

VERSION is an array of characters (a pointer to a char).

Pascal Program Excerpts

Following are excerpts from a Pascal program that calls SNA IMF intrinsics. For examples of complete Pascal programs in non-transparent and transparent modes, see Appendix F, "Sample Programs."

```
{************************* Global Declarations *************************}
procedure VERS3270; intrinsic;
.
.
.
{************************* Local Declarations **************************}
var
   version  : packed array[1..14] of char;
.
.
.
{************** Variable Initialization and Intrinsic Call **************}
version := ' ';
VERS3270 (version);
```

WRITEFIELD

WRITEFIELD writes data from your program on the HP 3000 into an unprotected field of the internal screen image.

Syntax

```
                I           I       I      CA          I          I
WRITEFIELD     (terminalid, fieldnum, offset, outbuf, outbuflen, result)
```

Parameters

terminalid (input)

Integer identifying the terminal. The *terminalid* is returned in a call to the OPEN3270 intrinsic.

fieldnum (input)

Integer specifying the relative number of the field into which data will be inserted. Field number 1 is the field following the first attribute character in the internal screen image. If the screen is unformatted (the SCREENATTR intrinsic has returned *numfields = 0*), you should specify 0 in the *fieldnum* parameter.

offset (input)

Integer specifying the character offset at which writing begins within the field. The first character position in the field is zero. The *offset* parameter must be zero when the DBCS option to the OPEN3270 intrinsic is enabled.

outbuf (input

Character array containing the data to replace the previous contents of the field. Only uppercase and lowercase alphabetics, numerics, ASCII character codes 34 and 36 (octal), and certain special characters are allowed.

The uppercase and lowercase alphabetics, numerics, ASCII character codes 34 and 36 (octal), and the special characters correspond to the following ASCII character codes:

34, 36, and 40 through 377 (octal)

1C, 1E and 20 through FF (hexadecimal).

Entering 34 (octal) is equivalent to pressing the **[DUP]**

key on an IBM 3278 display station keyboard. Entering 36 (octal) is equivalent to pressing the **[FIELD MARK]** key on a 3278 keyboard.

Several characters are represented differently in ASCII and in EBCDIC. In Table 3-4, the first column shows the ASCII character representions used by HP terminals and printers; the second column shows the EBCDIC character representations used by IBM terminals and printers.

Table 3-4 Different Displayed Characters, HP and IBM

ASCII Character	EBCDIC Character
[(left bracket)	¢ (cent sign)
] (right bracket)	! (exclamation)
! (exclamation)	| (or sign)
^ (caret)	¬ (not sign)

In a numeric field, you can enter only numeric characters (0–9), the period (.), the minus sign (-), and the DUP character (octal 32). The attribute character indicates whether a field is numeric.

Asian users: When the DBCS option to the OPEN3270 intrinsic is enabled, you may enter Shift-Out (hex “0E”) and Shift-In (hex “0F”) control characters as valid data. Data between SO and SI control characters is transmitted to the host without any range checking.

outbuflen (input)

Integer indicating the length, in characters, of the string to be written into the field. *outbuflen* must be less than or equal to the *maxfieldlen* parameter returned by the FIELDATTR intrinsic.

If *outbuflen* is larger than the *maxfieldlen* value returned by FIELDATTR, or if *offset* extends beyond the end of the field, then no data is written to the internal screen image, and the *result* parameter returns 13.

result (output)

The following values can be generated by the WRITEFIELD intrinsic:

0 = Successful completion.

1 = Device not open.

9 = Host modified screen since last receive request. (MPE V only)

10 = Attempt made to update a protected field.

11 = Non-existent field number specified.

12 = Invalid character in field or data stream.

13 = Field length specified for WRITEFIELD is too long.

14 = Attempt made to update a field or transmit from an LU.T3 printer.

17 = Attempt made to write to a field where input is inhibited.

21 = Field offset specified is out of range.

22 = BASIC calling sequence error has occurred.

25 = Intrinsic call made while in split stack mode.

26 = Intrinsic call made with the parameter value out of bounds.

29 = Called intrinsic with a request already outstanding. (No-wait I/O only)

30 = Internal error occurred in IMF intrinsic.

53 = Invalid intrinsic called for data stream mode device.

97 = An attempt was made to write SO (hex "0E") control character to a field where only 8-bit data is allowed.

98 = An attempt was made to write SO (hex "0E") control character to a field where only 16-bit data is allowed.

100 = A bad (non-zero) offset was specified in the intrinsic call.

Description

The WRITEFIELD intrinsic allows you to replace the contents of an unprotected field with new data or to put data into a field that was previously empty. If WRITEFIELD is successful, the Modified Data Tag (MDT) bit in the attribute character for that field is turned on.

If the screen is unformatted (the SCREENATTR intrinsic has returned *numfields = 0*), you can write to the screen by calling WRITEFIELD with the *fieldnum* parameter set to zero.

You may use either WRITEFIELD or STREAM3270 to enter data into your internal screen image. The WRITEFIELD intrinsic allows you to access a field by its number and write only to that field. STREAM3270 allows you to move across the internal screen image and enter data into more than one field with a single intrinsic call.

Call the WRITEFIELD intrinsic in non-transparent mode.

COBOL Calling Sequence

```
CALL "CWRITEFIELD" USING TERMINALID FIELDNUM OFFSET OUTBUF
OUTBUFLEN RESULT.
```

(on MPE V and in compatibility mode on MPE XL)

```
CALL INTRINSIC "WRITEFIELD" USING TERMINALID FIELDNUM
OFFSET OUTBUF OUTBUFLEN RESULT.
```
(in native mode on MPE XL)

All parameters are numeric data items except *OUTBUF*, which is an alphanumeric data item.

FORTRAN Calling Sequence

```
CALL WRITEFIELD (TERMINALID, FIELDNUM, OFFSET, OUTBUF,
OUTBUFLEN, RESULT)
```

All parameters are integer variables except *OUTBUF*, which is a character array.

BASIC Calling Sequence

```
CALL BWRITEFIELD (T, N, O1, O$, L9, R)
```
(on MPE V and in compatibility mode on MPE XL)

```
CALL WRITEFIELD (T, N, O1, O$, L9, R)
```
(in native mode on MPE XL)

All parameters are integer variables except O$, which is a string variable.

NOTE

When WRITEFIELD is called from BASIC, the actual number of characters moved is no greater than the logical length of the string variable passed in O$. BASIC cannot write a string of more than 255 characters. Because a field length may exceed 255 characters, *offset* gives you a way to choose the portion of the field you want to write. By making multiple calls to WRITEFIELD with different offsets, your BASIC programs can accommodate a field that exceeds 255 characters.

SPL Calling Sequence

```
WRITEFIELD (TERMINALID, FIELDNUM, OFFSET, OUTBUF,
OUTBUFLEN, RESULT)
```

All parameters are integer variables except *OUTBUF*, which is a byte array.

Pascal Calling Sequence

```
WRITEFIELD (TERMINALID, FIELDNUM, OFFSET, OUTBUF,
OUTBUFLEN, RESULT);
```

All parameters are short integer variables except *OUTBUF*, which is a packed array of char.

C/XL Calling Sequence

```
WRITEFIELD (&TERMINALID, &FIELDNUM, &OFFSET, OUTBUF,
&OUTBUFLEN, &RESULT);
```

All parameters are of type short, except *OUTBUF*, which is an array of characters (a pointer to a char).

Pascal Program Excerpts

Following are excerpts from a Pascal program that calls SNA IMF intrinsics. For examples of complete Pascal programs in non-transparent and transparent modes, see Appendix F , “Sample Programs.”

```
{************************* Global Declarations *************************}
type
   shortint      = -32768..32767;       { global type, two bytes (half word) }
.
.
.
var
   terminalid    : shortint;          { value returned by OPEN3270 intrinsic }
   result        : shortint;
.
.
.
procedure WRITEFIELD;  intrinsic;
.
.
.
{************************* Local Declarations *************************}
var
   fieldnum    : shortint;         { All WRITEFIELD variables except }
   offset      : shortint;         { terminalid and result are local. }
   outbuf      : packed array[1..MAXINBUFLEN] of char;
                                   { The value of MAXINBUFLEN is returned }
                                   { by the FIELDATTR intrinsic. }
   outbuflen   : shortint;
.
.
.
{************** Variable Initialization and Intrinsic Call **************}
offset := 0;
fieldnum := 0;                     { Write to unformatted screen. }
outbuf := 'logon applid(tso) data(sales/sales) logmode(imf2k) ';
outbuflen := 51;

WRITEFIELD (terminalid, fieldnum, offset, outbuf, outbuflen, result);
```

WRITESTREAM

WRITESTREAM creates the data stream that your program on the HP 3000 sends to the host (LU.T2 emulation only).

Syntax

```
                I          I          I         CA       I
WRITESTREAM     (terminalid, offset, outbuflen, outbuf, result)
```

Parameters

terminalid (input)

Integer identifying the terminal. The *terminalid* is returned in a call to the OPEN3270 intrinsic.

offset (input)

Integer specifying the offset, in characters, into the internal screen image. The data transfer starts at the location you specify in *offset*. The first character has an offset value of zero. The *offset* parameter must be zero when the DBCS option to the OPEN3270 intrinsic is enabled.

outbuflen (input)

Integer specifying the length of the character string *outbuf*, in characters. The maximum values for *outbuflen* are as follows:

540 bytes for 480-character screens

2160 bytes for 1920-character screens

3870 bytes for 3440-character screens

outbuf (input)

Character array that contains the data stream to be written to the internal screen image.

result (output)

The following values can be generated by the WRITESTREAM intrinsic:

0 = Successful completion.

1 = Device not open.

9 = Host modified screen since last receive request. (MPE V only)

21 = Field offset specified is out of range.

22 = BASIC calling sequence error has occurred.

25 = Intrinsic call made while in split stack mode.

26 = Intrinsic call made with the parameter value out of bounds.

29 = Called intrinsic with a request already outstanding. (No-wait I/O only)

44 = WRITESTREAM called during LU.T1 session.

52 = Data stream is too long.

54 = Device not opened in transparent mode.

100 = A bad (non-zero) offset was specified in the intrinsic call.

Description

The WRITESTREAM intrinsic can be used only during LU.T2 emulation and only in transparent mode. To open a device in transparent mode, call the OPEN3270 intrinsic with bit 14 of the *flags* parameter set to 1.

To send data to the host, call WRITESTREAM to create the data stream, then call TRAN3270 to transmit it to the host. WRITESTREAM accepts an array of characters destined for the host and puts these characters into your internal screen image. You must supply the terminal identifier, the array of characters, and the number of characters in the data stream.

The maximum length of the *outbuf* string (and the maximum value for the *outbuflen* parameter) is as follows:

540 bytes for 480-character screens

2160 bytes for 1920-character screens

3870 bytes for 3440-character screens

After your application receives the initial screen from the host (for example, the CICS or TSO welcome screen), it is communicating over the LU-SSCP session. At this point, data you write to the internal screen image with the WRITESTREAM intrinsic will be appended to the end of the data received from the host. When you call TRAN3270 to transmit the internal screen image to the host, data is read starting from the first position after the last data received from the host. Therefore, you must set the *offset* parameter of the WRITESTREAM intrinsic to the length of the last data stream received from the host. (The *offset* in your WRITESTREAM call should equal the *actinbuflen* from your last call to READSTREAM). An *offset* of zero would place the

data at the beginning of the internal screen image, before the position where reading begins, and an empty RU would be sent to the host.

After the BIND, when the LU-LU session has been established, the internal screen image is read from the beginning (position 0). When you call TRAN3270, the entire screen image, beginning at position 0, is transmitted to the host. Therefore, while your application is communicating over the LU-LU session, you should set the *offset* parameter of the WRITESTREAM intrinsic to 0.

If you make multiple calls to WRITESTREAM before you call TRAN3270, each write must begin where the last one left off. In other words, the *offset* of the second WRITESTREAM call should be the *outbuflen* value from the first call, the *offset* of the third call should be the sum of the *outbuflen* values from the previous two calls, and so on.

When sending data to the host, your WRITESTREAM buffer needs to contain only the data to be transferred. SNA IMF divides this data into frames and adds the SDLC and SNA protocol headers. The AID key and cursor address, which you specify in your call to TRAN3270, are inserted at the front of the first frame. You can suppress insertion of the AID key and cursor address in transparent mode. See the description of TRAN3270, earlier in this chapter, for more information.

If cursor suppression is not used, and you are sending the **[CLEAR]** key or a **PA** key during an LU.T2 session, call WRITESTREAM with an *outbuf* of length zero before calling TRAN3270. In transparent mode, SNA IMF will not check the AID key to determine whether it is the **[CLEAR]** key or a **PA** key. In non-transparent mode, no data is sent with the **[CLEAR]** or **PA** keys.

Call WRITESTREAM only in transparent mode. See Chapter 2 , “Using SNA IMF Intrinsics,” for information about transparent mode.

COBOL Calling Sequence

```
CALL “CWRITESTREAM” USING TERMINALID OFFSET OUTBUFLEN
OUTBUF RESULT.
```
(on MPE V and in compatibility mode on MPE XL)

```
CALL INTRINSIC “WRITESTREAM” USING TERMINALID OFFSET
OUTBUFLEN OUTBUF RESULT.
```
(in native mode on MPE XL)

All parameters are numeric data items except *OUTBUF*, which is an alphanumeric data item.

FORTRAN Calling Sequence

```
CALL WRITESTREAM (TERMINALID, OFFSET, OUTBUFLEN, OUTBUF,
RESULT)
```

All parameters are integer variables except *OUTBUF*, which is a character array.

BASIC Calling Sequence

`CALL BWRITESTREAM (T, O1, L9, O$, R)` (on MPE V and in compatibility mode on MPE XL)

`CALL WRITESTREAM (T, O1, L9, O$, R)` (in native mode on MPE XL)

All parameters are integer variables except `O$`, which is a string variable.

SPL Calling Sequence

```
WRITESTREAM (TERMINALID, OFFSET, OUTBUFLEN, OUTBUF, RESULT)
```

All parameters are integer data items except *OUTBUF*, which is a byte array.

Pascal Calling Sequence

```
WRITESTREAM (TERMINALID, OFFSET, OUTBUFLEN, OUTBUF,
RESULT);
```

All parameters are short integer variables except for *OUTBUF*, which is a packed array of char.

C/XL Calling Sequence

```
WRITESTREAM (&TERMINALID, &OFFSET, &OUTBUFLEN, OUTBUF,
&RESULT);
```

All parameters are of type short, except *OUTBUF*, which is an array of characters (a pointer to a char).

Pascal Program Excerpts

Following are excerpts from a Pascal program that calls SNA IMF intrinsics. For examples of complete Pascal programs in non-transparent and transparent modes, see Appendix F , “Sample Programs.”

```
{************************* Global Declarations *************************}
const
   MAXBUFLEN = 2160;                    { maximum buffer length for }
                                        { 1920-character screens }
                                        { in transparent mode }
.
.
.
type
   shortint      = -32768..32767;       { global type, two bytes (half word) }

var
   terminalid    : shortint;          { value returned by OPEN3270 intrinsic }
   result         : shortint;
.
.
.
procedure WRITESTREAM;  intrinsic;
.
.
.
{************************* Local Declarations *************************}
var
   offset      : shortint;         { All WRITESTREAM variables except }
   outbuflen   : shortint;         { terminalid and result are local. }
   outbuf      : packed array[1..MAXBUFLEN] of char;
.
.
.
{************** Variable Initialization and Intrinsic Call **************}
offset := 0;
outbuflen := 51;
outbuf := 'logon applid(tso) data(sales/sales) logmode(imf2k) ';
               WRITESTREAM (terminalid, offset, outbuflen, outbuf, result);
```

Summary of Intrinsic Calls

This section lists the calls to the SNA IMF intrinsics used with standard MPE I/O. It does not include `ABORT3270`, `IODONTWAIT`, `IODONTWAIT3270`, `IOWAIT`, and `IOWAIT3270`, which are intrinsics used only with no-wait I/O. Intrinsic calls are listed for each supported language: COBOL, FORTRAN, BASIC, SPL, Pascal, and C/XL.

NOTE

On MPE XL, your application programs that call SNA IMF intrinsics may be in either native mode or compatibility mode. However, if your applications are written in BASIC or COBOL, the SNA IMF intrinsic calls in native mode are different from those in compatibility mode. See Chapter 2 , “Using SNA IMF Intrinsics,” for more information on native mode and compatibility mode.

SNA IMF standard I/O intrinsics are patterned after and consistent with MPE V and MPE XL intrinsics. The following facts apply to MPE and SNA IMF standard I/O intrinsics:

- You can call intrinsics from COBOL, COBOL II, BASIC, FORTRAN, SPL, Pascal, and C/XL programs.
- All intrinsic parameters are passed by reference.
- The parameter lists are fixed; all parameters are required.
- Intrinsics do not return condition codes.
- The intrinsic calling sequences are essentially the same for all languages.

The next few sections in this chapter list the SNA IMF intrinsic calls for COBOL, BASIC, FORTRAN, SPL, Pascal, and C/XL.

Intrinsic Calls in COBOL

This section lists the SNA IMF intrinsic calls for COBOL and COBOL II.

Use COMPUTATIONAL and SYNCHRONIZED in COBOL picture clauses to guarantee the alignment of parameters on word boundaries.

Use LEVELS 01 and 77 to describe parameters passed in COBOL calls to guarantee the alignment of parameters on word boundaries.

The syntax of numeric, alphanumeric, and numeric table data items in COBOL is listed in Table 3-5.

Table 3-5 COBOL Data Types

Data Type	Syntax
Numeric (1 Word)	PICTURE S9(4) COMPUTATIONAL SYNCHRONIZED
Alphanumeric	PICTURE X(n)
Numeric Table	PICTURE S9(4) COMPUTATIONAL SYNCHRONIZED OCCURS

Table 3-6 lists the COBOL and COBOL II intrinsic calls used on MPE V and in compatibility mode on MPE XL. The MSGBUF, SNALNKINFO, INBUF, LOCATION, OUTBUF, VERSION, and PFN parameters are alphanumeric (character) data items. The TIMEOUT , and OFFSETLIST parameters are computational synchronized numeric tables. All other parameters are computational synchronized numerics.

Table 3-6 COBOL Intrinsic Calls, MPE V and CM

Intrinsic	COBOL Calling Sequence
ACQUIRE3270 (for SNA IMF/V)	CALL “CACQUIRE3270” USING SNALNKINFO DEVICENUM LDEV ENHANCE PRIORITY BLANKS FORMAT FLAGS RESULT.
ACQUIRE3270 (for SNA IMF/XL in compatibility mode)	CALL “CACQUIRE3270” USING SNALNKINFO DEVICENUM LDEV ENHANCE PRIORITY BLANKS FORMAT FLAGS OPTIONS PFN RESULT.
ATTRLIST	CALL “CATTRLIST” USING TERMINALID OFFSET SUBSCREENSIZE MAXLISTLEN FIELDNUM OFFSETLIST ACTLISTLEN RESULT.
CLOSE3270	CALL “CCLOSE3270” USING TERMINALID RESULT.
ERR3270	CALL “CERR3270” USING ERRORCODE MSGBUF MSGLEN RESULT.
EXTFIELDATTR	CALL “CEXTFIELDATTR” USING TERMINALID FIELDNUM FIELDROW FIELDCOLUMN PROTECTEDATTR NUMERICATTR DISPLAYATTR MDT DBCSATTR CURRENTFIELDLEN MAXFIELDLEN RESULT.

Table 3-6 COBOL Intrinsic Calls, MPE V and CM

Intrinsic	COBOL Calling Sequence
FIELDATTR	CALL "CFIELDATTR" USING TERMINALID FIELDNUM FIELDROW FIELDCOLUMN PROTECTEDATTR NUMERICATTR DISPLAYATTR MDT CURRENTFIELDLEN MAXFIELDLEN RESULT.
OPEN3270	CALL "COPEN3270" USING DEVICENUM SNALNKINFO FLAGS TERMINALID DEVTYPE FFINDEX SCREENSIZE TIMEOUT RESULT.
PRINT3270	CALL "CPRINT3270" USING TERMINALID FILEID ACTION LOCATION PRIORITY RESULT.
READFIELD	CALL "CREADFIELD" USING TERMINALID FIELDNUM OFFSET MAXINBUFLEN INBUF ACTINBUFLEN RESULT.
READSCREEN	CALL "CREADSCREEN" USING TERMINALID OFFSET MAXINBUFLEN INBUF ACTINBUFLEN RESULT.
READSTREAM	CALL "CREADSTREAM" USING TERMINALID OFFSET MAXINBUFLEN INBUF ACTINBUFLEN RESULT.
RECV3270	CALL "CRECV3270" USING TERMINALID RESULT.
RESET3270	CALL "CRESET3270" USING TERMINALID RESULT.
SCREENATTR	CALL "CSCREENATTR" USING TERMINALID PRINTFORMAT STARTPRINT SOUNDALARM KEYBOARDLOCK NUMFIELDS SCREENSTATUS CURSORROW CURSORCOLUMN RESULT.
STREAM3270	CALL "CSTREAM3270" USING TERMINALID CURSORROW CURSORCOLUMN OUTBUF OUTBUFLEN NUMPROCESSED RESULT.
TRAN3270	CALL "CTRAN3270" USING TERMINALID AID CURSORROW CURSORCOLUMN RESULT.
VERS3270	CALL "CVERS3270" USING VERSION.
WRITEFIELD	CALL "CWRITEFIELD" USING TERMINALID FIELDNUM OFFSET OUTBUF OUTBUFLEN RESULT.
WRITESTREAM	CALL "CWRITESTREAM" USING TERMINALID OFFSET OUTBUFLEN OUTBUF RESULT.

Table 3-7 lists the COBOL and COBOL II intrinsic calls used on MPE XL in native mode. The MSGBUF, SNALNKINFO, INBUF, LOCATION, OUTBUF, VERSION, and PFN parameters are alphanumeric (character) data items. The TIMEOUT and OFFSETLIST parameters are computational synchronized numeric tables. All other parameters are computational synchronized numerics.

Table 3-7 COBOL Intrinsic Calls, MPE XL Native Mode

Intrinsic	COBOL Calling Sequence
ACQUIRE3270	CALL INTRINSIC "ACQUIRE3270" USING SNALNKINFO DEVICENUM LDEV ENHANCE PRIORITY BLANKS FORMAT FLAGS OPTIONS PFN RESULT.
ATTRLIST	CALL INTRINSIC "ATTRLIST" USING TERMINALID OFFSET SUBSCREENSIZE MAXLISTLEN FIELDNUM OFFSETLIST ACTLISTLEN RESULT.
CLOSE3270	CALL INTRINSIC "CLOSE3270" USING TERMINALID RESULT.
ERR3270	CALL INTRINSIC "ERR3270" USING ERRORCODE MSGBUF MSGLEN RESULT.
EXTFIELDATTR	CALL INTRINSIC "EXTFIELDATTR" USING TERMINALID FIELDNUM FIELDROW FIELDCOLUMN PROTECTEDATTR NUMERICATTR DISPLAYATTR MDT DBCSATTR CURRENTFIELDLEN MAXFIELDLEN RESULT.
FIELDATTR	CALL INTRINSIC "FIELDATTR" USING TERMINALID FIELDNUM FIELDROW FIELDCOLUMN PROTECTEDATTR NUMERICATTR DISPLAYATTR MDT CURRENTFIELDLEN MAXFIELDLEN RESULT.
OPEN3270	CALL INTRINSIC "OPEN3270" USING DEVICENUM SNALNKINFO FLAGS TERMINALID DEVTYPE FFINDEX SCREENSIZE TIMEOUT RESULT.
PRINT3270	CALL INTRINSIC "PRINT3270" USING TERMINALID FILEID ACTION LOCATION PRIORITY RESULT.
READFIELD	CALL INTRINSIC "READFIELD" USING TERMINALID FIELDNUM OFFSET MAXINBUFLEN INBUF ACTINBUFLEN RESULT.
READSCREEN	CALL INTRINSIC "READSCREEN" USING TERMINALID OFFSET MAXINBUFLEN INBUF ACTINBUFLEN RESULT.
READSTREAM	CALL INTRINSIC "READSTREAM" USING TERMINALID OFFSET MAXINBUFLEN INBUF ACTINBUFLEN RESULT.
RECV3270	CALL INTRINSIC "RECV3270" USING TERMINALID RESULT.
RESET3270	CALL INTRINSIC "RESET3270" USING TERMINALID RESULT.
SCREENATTR	CALL INTRINSIC "SCREENATTR" USING TERMINALID PRINTFORMAT STARTPRINT SOUNDALARM KEYBOARDLOCK NUMFIELDS SCREENSTATUS CURSORROW CURSORCOLUMN RESULT.
STREAM3270	CALL INTRINSIC "STREAM3270" USING TERMINALID CURSORROW CURSORCOLUMN OUTBUF OUTBUFLEN NUMPROCESSED RESULT.
TRAN3270	CALL INTRINSIC "TRAN3270" USING TERMINALID AID CURSORROW CURSORCOLUMN RESULT.

Table 3-7 COBOL Intrinsic Calls, MPE XL Native Mode

Intrinsic	COBOL Calling Sequence
VERS3270	CALL "VERS3270" USING VERSION.
WRITEFIELD	CALL INTRINSIC "WRITEFIELD" USING TERMINALID FIELDNUM OFFSET OUTBUF OUTBUFLEN RESULT.
WRITESTREAM	CALL INTRINSIC "WRITESTREAM" USING TERMINALID OFFSET OUTBUFLEN OUTBUF RESULT.

Intrinsic Calls in FORTRAN

Table 3-8 lists the SNA IMF intrinsic calls for FORTRAN. The MSGBUF, SNALNKINFO, INBUF, LOCATION, OUTBUF, VERSION, and PFN parameters are character arrays. The TIMEOUT and OFFSETLIST parameters are integer arrays. All other parameters are one-word integer variables.

Table 3-8 FORTRAN Intrinsic Calls

Intrinsic	COBOL Calling Sequence
ACQUIRE3270 (for SNA IMF/V)	CALL ACQUIRE3270 (SNALNKINFO, DEVICENUM, LDEV, ENHANCE, PRIORITY, BLANKS, FORMAT, FLAGS, RESULT)
ACQUIRE3270 (for SNA IMF/XL	CALL ACQUIRE3270 (SNALNKINFO, DEVICENUM, LDEV, ENHANCE, PRIORITY, BLANKS, FORMAT, FLAGS, OPTIONS, PFN, RESULT)
ATTRLIST	CALL ATTRLIST (TERMINALID, OFFSET, SUBSCREENSIZE, MAXLISTLEN, FIELDNUM, OFFSETLIST, ACTLISTLEN, RESULT)
CLOSE3270	CALL CLOSE3270 (TERMINALID, RESULT)
ERR3270	CALL ERR3270 (ERRORCODE, MSGBUF, MSGLEN, RESULT)
EXTFIELDATTR	CALL EXTFIELDATTR (TERMINALID, FIELDNUM, FIELDROW, FIELDCOLUMN, PROTECTEDATTR, NUMERICATTR, DISPLAYATTR, MDT, DBCSATTR, CURRENTFIELDLEN, MAXFIELDLEN, RESULT)
FIELDATTR	CALL FIELDATTR (TERMINALID, FIELDNUM, FIELDROW, FIELDCOLUMN, PROTECTEDATTR, NUMERICATTR, DISPLAYATTR, MDT, CURRENTFIELDLEN, MAXFIELDLEN, RESULT)
OPEN3270	CALL OPEN3270 (DEVICENUM, SNALNKINFO, FLAGS, TERMINALID, DEVTYPE, FFINDEX, SCREENSIZE, TIMEOUT, RESULT)
PRINT3270	CALL PRINT3270 (TERMINALID, FILEID, ACTION, LOCATION, PRIORITY, RESULT)
READFIELD	CALL READFIELD (TERMINALID, FIELDNUM, OFFSET, MAXINBUFLEN, INBUF, ACTINBUFLEN, RESULT)

Table 3-8 FORTRAN Intrinsic Calls

Intrinsic	COBOL Calling Sequence
READSCREEN	CALL READSCREEN (TERMINALID, OFFSET, MAXINBUFLEN, INBUF, ACTINBUFLEN, RESULT)
READSTREAM	CALL READSTREAM (TERMINALID, OFFSET, MAXINBUFLEN, INBUF, ACTINBUFLEN, RESULT)
RECV3270	CALL RECV3270 (TERMINALID, RESULT).
RESET3270	CALL RESET3270 (TERMINALID, RESULT)
SCREENATTR	CALL SCREENATTR (TERMINALID, PRINTFORMAT, STARTPRINT, SOUNDALARM, KEYBOARDLOCK, NUMFIELDS, SCREENSTATUS, CURSORROW, CURSORCOLUMN, RESULT)
STREAM3270	CALL STREAM3270 (TERMINALID, CURSORROW, CURSORCOLUMN, OUTBUF, OUTBUFLEN, NUMPROCESSED, RESULT)
TRAN3270	CALL TRAN3270 (TERMINALID, AID, CURSORROW, CURSORCOLUMN, RESULT)
VERS3270	CALL VERS3270 (VERSION)
WRITEFIELD	CALL WRITEFIELD (TERMINALID, FIELDNUM, OFFSET, OUTBUF, OUTBUFLEN, RESULT)
WRITESTREAM	CALL WRITESTREAM (TERMINALID, OFFSET, OUTBUFLEN, OUTBUF, RESULT

Intrinsic Calls in BASIC

Table 3-9 lists the BASIC intrinsic calls used on MPE V and in compatibility mode on MPE XL. The M$ (MSGBUF), C$ (SNALNKINFO), I$ (INBUF), L$ (LOCATION), O$ (OUTBUF), V$ (VERSION), and P$ (PFN) parameters are character arrays. The T2(*) (TIMEOUT) and O2(*) (OFFSETLIST) parameters are integer arrays. All other parameters are one-word integer variables.

Table 3-9 BASIC Intrinsic Calls, MPE V and CM

Intrinsic	COBOL Calling Sequence
ACQUIRE3270 (for SNA IMF/V)	CALL BACQUIRE3270 (C$, D1, L0, E1, P2, B1, F3, F, R)
ACQUIRE3270 (for SNA IMF/XL in compatibility mode)	CALL BACQUIRE3270 (C$, D1, L0, E1, P2, B1, F3, F, O4, P$, R)
ATTRLIST	CALL BATTRLIST (T, 01, L1, L2, N, 02(*), L3, R)
CLOSE3270	CALL BCLOSE3270 (T, R)
ERR3270	CALL BERR3270 (E, M$, L4, R)
EXTFIELDATTR	CALL BEXTFIELDATTR (T, N, R0, C0, A1, A2, A3, A4, A5, L5, L6, R)
FIELDATTR	CALL BFIELDATTR (T, N, R0, C0, A1, A2, A3, A4, L5, L6, R)
OPEN3270	CALL BOPEN3270 (D1, C$, F, T, D2, F1, S, T2(*), R)
PRINT3270	CALL BPRINT3270 (T, F2, O3, L$, P2, R)
READFIELD	CALL BREADFIELD (T, N, O1, L7, I$, L8, R)
READSCREEN	CALL BREADSCREEN (T, O1, L7, I$, L8, R)
READSTREAM	CALL BREADSTREAM (T, O1, L7, I$, L8, R)
RECV3270	CALL BRECV3270 (T, R)
RESET3270	CALL BRESET3270 (T, R)
SCREENATTR	CALL BSCREENATTR (T, P, P1, A, K, N9, S9, R9, C9, R)
STREAM3270	CALL BSTREAM3270 (T, R9, C9, O$, L9, N5, R)
TRAN3270	CALL BTRAN3270 (T, A9, R9, C9, R)
VERS3270	CALL BVERS3270 (V$)
WRITEFIELD	CALL BWRITEFIELD (T, N, O1, O$, L9, R)
WRITESTREAM	CALL BWRITESTREAM (T, O1, L9, O$, R)

Table 3-10 lists the BASIC intrinsic calls used on MPE XL in native mode. The M$ (MSGBUF), C$ (SNALNKINFO), I$ (INBUF), L$ (LOCATION), O$ (OUTBUF), V$ (VERSION), and P$ (PFN) parameters are character arrays. The T2(*) (TIMEOUT) and O2(*) (OFFSETLIST) parameters are integer arrays. All other parameters are one-word integer variables.

Table 3-10 BASIC Intrinsic Calls, MPE XL Native Mode

Intrinsic	COBOL Calling Sequence
ACQUIRE3270	CALL ACQUIRE3270 (C$, D1, L0, E1, P2, B1, F3, F, O4, P$, R)
ATTRLIST	CALL ATTRLIST (T, O1, L1, L2, N, O2(*), L3, R)
CLOSE3270	CALL CLOSE3270 (T, R)
ERR3270	CALL ERR3270 (E, M$, L4, R)
EXTFIELDATTR	CALL EXTFIELDATTR (T, N, R0, C0, A1, A2, A3, A4, A5, L5, L6, R)
FIELDATTR	CALL FIELDATTR (T, N, R0, C0, A1, A2, A3, A4, L5, L6, R)
OPEN3270	CALL OPEN3270 (D1, C$, F, T, D2, F1, S, T2(*), R)
PRINT3270	CALL PRINT3270 (T, F2, O3, L$, P2, R)
READFIELD	CALL READFIELD (T, N, O1, L7, I$, L8, R)
READSCREEN	CALL READSCREEN (T, O1, L7, I$, L8, R)
READSTREAM	CALL READSTREAM (T, O1, L7, I$, L8, R)
RECV3270	CALL RECV3270 (T, R)
RESET3270	CALL RESET3270 (T, R)
SCREENATTR	CALL SCREENATTR (T, P, P1, A, K, N9, S9, R9, C9, R)
STREAM3270	CALL STREAM3270 (T, R9, C9, O$, L9, N5, R)
TRAN3270	CALL TRAN3270 (T, A9, R9, C9, R)
VERS3270	CALL VERS3270 (V$)
WRITEFIELD	CALL WRITEFIELD (T, N, O1, O$, L9, R)
WRITESTREAM	CALL WRITESTREAM (T, O1, L9, O$, R)

Table 3-11 lists the variables used in BASIC intrinsic calls and the parameters that correspond to the variables.

Table 3-11 Basic Variables

Variable	Parameter	Variable	Parameter
A	*SOUNDALARM*	L5	*CURRENTFIELD*
A1	*PROTECTEDATTR*	L6	*MAXFIELDLEN*
A2	*NUMERICATTR*	L7	*MAXINBUFLEN*
A3	*DISPLAYATTR*	L8	*ACTINBUFLEN*
A4	*MDT*	L9	*OUTBUFLEN*
A5	*DBCSATTR*	L$	*LOCATION*
A9	*AID*	M$	*MSGBUF*
B1	*BLANKS*	N	*FIELDNUM*
C0	*FIELDCOLUMN*	N5	*NUMPROCESSED*
C9	*CURSORCOLUMN*	N9	*NUMFIELDS*
C$	*SNALINKINFO*	01	*OFFSET*
D1	*DEVICENUM*	02(*)	*OFFSETLIST*
D2	*DEVTYPE*		
E	*ERRORCODE*	04	*OPTIONS*
E1	*ENHANCE*	0$	*OUTBUF*
F	*FLAGS*	P	*PRINTFORMAT*
F1	*FFINDEX*	P1	*STARTPRINT*
F2	*FILEID*	P2	*PRIORITY*
F3	*FORMAT*	P$	*PFN*
I$	*INBUF*	R	*RESULT*
K	*KEYBOARDLOCK*	R0	*FIELDROW*
L0	*LDEV*	R9	*CURSORROW*
L1	*SUBSCREENSIZE*	S	*SCREENSIZE*
L2	*MAXLISTLEN*	S9	*SCREENSTATUS*
L3	*ACTLISTLEN*	T	*TERMINALID*
L4	*MSGLEN*	T2(*)	*TIMEOUT*
03	*ACTION*	V$	*VERSION*

Intrinsic Calls in SPL

Table 3-12 lists the SNA IMF intrinsic calls for SPL. The MSGBUF, SNALNKINFO, INBUF, LOCATION, OUTBUF, VERSION, and PFN parameters are byte arrays. The TIMEOUT and OFFSETLIST parameters are integer arrays. All other parameters are one-word integer variables.

Table 3-12 SPL Intrinsic Calls

Intrinsic	SPL Calling Sequence
ACQUIRE3270 (for SNA IMF/V)	ACQUIRE3270 (SNALNKINFO, DEVICENUM, LDEV, ENHANCE, PRIORITY, BLANKS, FORMAT, FLAGS, RESULT)
ACQUIRE3270 (for SNA IMF/XL)	ACQUIRE3270 (SNALNKINFO, DEVICENUM, LDEV, ENHANCE, PRIORITY, BLANKS, FORMAT, FLAGS, OPTIONS, PFN, RESULT)
ATTRLIST	ATTRLIST (TERMINALID, OFFSET, SUBSCREENSIZE, MAXLISTLEN, FIELDNUM, OFFSETLIST, ACTLISTLEN, RESULT)
CLOSE3270	CLOSE3270 (TERMINALID, RESULT)
ERR3270	ERR3270 (ERRORCODE, MSGBUF, MSGLEN, RESULT)
EXTFIELDATTR	EXTFIELDATTR (TERMINALID, FIELDNUM, FIELDROW, FIELDCOLUMN, PROTECTEDATTR, NUMERICATTR, DISPLAYATTR, MDT, DBCSATTR, CURRENTFIELDLEN, MAXFIELDLEN, RESULT)
FIELDATTR	FIELDATTR (TERMINALID, FIELDNUM, FIELDROW, FIELDCOLUMN, PROTECTEDATTR, NUMERICATTR, DISPLAYATTR, MDT, CURRENTFIELDLEN, MAXFIELDLEN, RESULT)
OPEN3270	OPEN3270 (DEVICENUM, SNALNKINFO, FLAGS, TERMINALID, DEVTYPE, FFINDEX, SCREENSIZE, TIMEOUT, RESULT)
PRINT3270	PRINT3270 (TERMINALID, FILEID, ACTION, LOCATION, PRIORITY, RESULT)
READFIELD	READFIELD (TERMINALID, FIELDNUM, OFFSET, MAXINBUFLEN, INBUF, ACTINBUFLEN, RESULT)
READSCREEN	READSCREEN (TERMINALID, OFFSET, MAXINBUFLEN, INBUF, ACTINBUFLEN, RESULT)
READSTREAM	READSTREAM (TERMINALID, OFFSET, MAXINBUFLEN, INBUF, ACTINBUFLEN, RESULT)
RECV3270	RECV3270 (TERMINALID, RESULT)
RESET3270	RESET3270 (TERMINALID, RESULT)
SCREENATTR	SCREENATTR (TERMINALID, PRINTFORMAT, STARTPRINT, SOUNDALARM, KEYBOARDLOCK, NUMFIELDS, SCREENSTATUS, CURSORROW, CURSORCOLUMN, RESULT)

Table 3-12 SPL Intrinsic Calls

Intrinsic	SPL Calling Sequence
STREAM3270	STREAM3270 (TERMINALID, CURSORROW, CURSORCOLUMN, OUTBUF, OUTBUFLEN, NUMPROCESSED, RESULT)
TRAN3270	TRAN3270 (TERMINALID, AID, CURSORROW, CURSORCOLUMN, RESULT)
VERS3270	VERS3270 (VERSION)
WRITEFIELD	WRITEFIELD (TERMINALID, FIELDNUM, OFFSET, OUTBUF, OUTBUFLEN, RESULT)
WRITESTREAM	WRITESTREAM (TERMINALID, OFFSET, OUTBUFLEN, OUTBUF, RESULT)

Intrinsic Calls in Pascal

Table 3-13 lists the SNA IMF intrinsic calls for Pascal. The MSGBUF, SNALNKINFO, INBUF, LOCATION, OUTBUF, VERSION, and PFN parameters are character arrays. The TIMEOUT and OFFSETLIST parameters are integer arrays. All other parameters are one-word integer variables.

Table 3-13 Pascal Intrinsic Calls

Intrinsic	Pascal Calling Sequence
ACQUIRE3270 (for SNA IMF/V)	ACQUIRE3270 (SNALNKINFO, DEVICENUM, LDEV, ENHANCE, PRIORITY, BLANKS, FORMAT, FLAGS, RESULT);
ACQUIRE3270 (for SNA IMF/XL)	ACQUIRE3270 (SNALNKINFO, DEVICENUM, LDEV, ENHANCE, PRIORITY, BLANKS, FORMAT, FLAGS, OPTIONS, PFN, RESULT);
ATTRLIST	ATTRLIST (TERMINALID, OFFSET, SUBSCREENSIZE, MAXLISTLEN, FIELDNUM, OFFSETLIST, ACTLISTLEN, RESULT);
CLOSE3270	CLOSE3270 (TERMINALID, RESULT);
ERR3270	ERR3270 (ERRORCODE, MSGBUF, MSGLEN, RESULT);
EXTFIELDATTR	EXTFIELDATTR (TERMINALID, FIELDNUM, FIELDROW, FIELDCOLUMN, PROTECTEDATTR, NUMERICATTR, DISPLAYATTR, MDT, DBCSATTR, CURRENTFIELDLEN, MAXFIELDLEN, RESULT);
FIELDATTR	FIELDATTR (TERMINALID, FIELDNUM, FIELDROW, FIELDCOLUMN, PROTECTEDATTR, NUMERICATTR, DISPLAYATTR, MDT, CURRENTFIELDLEN, MAXFIELDLEN, RESULT);
OPEN3270	OPEN3270 (DEVICENUM, SNALNKINFO, FLAGS, TERMINALID, DEVTYPE, FFINDEX, SCREENSIZE, TIMEOUT, RESULT);

Table 3-13 Pascal Intrinsic Calls

Intrinsic	Pascal Calling Sequence
PRINT3270	PRINT3270 (TERMINALID, FILEID, ACTION, LOCATION, PRIORITY, RESULT);
READFIELD	READFIELD (TERMINALID, FIELDNUM, OFFSET, MAXINBUFLEN, INBUF, ACTINBUFLEN, RESULT);
READSCREEN	READSCREEN (TERMINALID, OFFSET, MAXINBUFLEN, INBUF, ACTINBUFLEN, RESULT);
READSTREAM	READSTREAM (TERMINALID, OFFSET, MAXINBUFLEN, INBUF, ACTINBUFLEN, RESULT);
RECV3270	RECV3270 (TERMINALID, RESULT);
RESET3270	RESET3270 (TERMINALID, RESULT);
SCREENATTR	SCREENATTR (TERMINALID, PRINTFORMAT, STARTPRINT, SOUNDALARM, KEYBOARDLOCK, NUMFIELDS, SCREENSTATUS, CURSORROW, CURSORCOLUMN, RESULT);
STREAM3270	STREAM3270 (TERMINALID, CURSORROW, CURSORCOLUMN, OUTBUF, OUTBUFLEN, NUMPROCESSED, RESULT);
TRAN3270	TRAN3270 (TERMINALID, AID, CURSORROW, CURSORCOLUMN, RESULT);
VERS3270	VERS3270 (VERSION);
WRITEFIELD	WRITEFIELD (TERMINALID, FIELDNUM, OFFSET, OUTBUF, OUTBUFLEN, RESULT);
WRITESTREAM	WRITESTREAM (TERMINALID, OFFSET, OUTBUFLEN, OUTBUF, RESULT);

Intrinsic Calls in C/XL

Table 3-14 lists the SNA IMF intrinsic calls for C/XL. The MSGBUF, SNALNKINFO, INBUF, LOCATION, OUTBUF, VERSION, and PFN parameters are character arrays (pointers to char). The TIMEOUT and OFFSETLIST parameters are arrays of (pointers to) short integers. All other parameters are one-word integer variables (type short).

Table 3-14 C/XL Intrinsic Calls

Intrinsic	C/XL Calling Sequence
ACQUIRE3270	ACQUIRE3270 (SNALNKINFO, &DEVICENUM, &LDEV, &ENHANCE, &PRIORITY, &BLANKS, &FORMAT, &FLAGS, &OPTIONS, PFN, &RESULT);
ATTRLIST	ATTRLIST (&TERMINALID, &OFFSET, &SUBSCREENSIZE, &MAXLISTLEN, &FIELDNUM, OFFSETLIST, &ACTLISTLEN, &RESULT);

Table 3-14 C/XL Intrinsic Calls

Intrinsic	C/XL Calling Sequence
CLOSE3270	CLOSE3270 (&TERMINALID, &RESULT);
ERR3270	ERR3270 (&ERRORCODE, MSGBUF, &MSGLEN, &RESULT);
EXTFIELDATTR	EXTFIELDATTR (&TERMINALID, &FIELDNUM, &FIELDROW, &FIELDCOLUMN, &PROTECTEDATTR, &NUMERICATTR, &DISPLAYATTR, &MDT, &DBCSATTR, &CURRENTFIELDLEN, &MAXFIELDLEN, &RESULT);
FIELDATTR	FIELDATTR (&TERMINALID, &FIELDNUM, &FIELDROW, &FIELDCOLUMN, &PROTECTEDATTR, &NUMERICATTR, &DISPLAYATTR, &MDT, &CURRENTFIELDLEN, &MAXFIELDLEN, &RESULT);
OPEN3270	OPEN3270 (&DEVICENUM, SNALNKINFO, &FLAGS, &TERMINALID, &DEVTYPE, &FFINDEX, &SCREENSIZE, TIMEOUT, &RESULT);
PRINT3270	PRINT3270 (&TERMINALID, &FILEID, &ACTION, LOCATION, &PRIORITY, &RESULT);
READFIELD	READFIELD (&TERMINALID, &FIELDNUM, &OFFSET, &MAXINBUFLEN, INBUF, &ACTINBUFLEN, &RESULT);
READSCREEN	READSCREEN (&TERMINALID, &OFFSET, &MAXINBUFLEN, INBUF, &ACTINBUFLEN, &RESULT);
READSTREAM	READSTREAM (&TERMINALID, &OFFSET, &MAXINBUFLEN, INBUF, &ACTINBUFLEN, &RESULT);
RECV3270	RECV3270 (&TERMINALID, &RESULT);
RESET3270	RESET3270 (&TERMINALID, &RESULT);
SCREENATTR	SCREENATTR (&TERMINALID, &PRINTFORMAT, &STARTPRINT, &SOUNDALARM, &KEYBOARDLOCK, &NUMFIELDS, &SCREENSTATUS, &CURSORROW, &CURSORCOLUMN, &RESULT);
STREAM3270	STREAM3270 (&TERMINALID, &CURSORROW, &CURSORCOLUMN, OUTBUF, &OUTBUFLEN, &NUMPROCESSED, &RESULT);
TRAN3270	TRAN3270 (&TERMINALID, &AID, &CURSORROW, &CURSORCOLUMN, &RESULT);
VERS3270	VERS3270 (VERSION);
WRITEFIELD	WRITEFIELD (&TERMINALID, &FIELDNUM, &OFFSET, OUTBUF, &OUTBUFLEN, &RESULT);
WRITESTREAM	WRITESTREAM (&TERMINALID, &OFFSET, &OUTBUFLEN, OUTBUF, &RESULT);

4 Intrinsics Used with No-Wait I/O

No-wait I/O, or unblocked I/O, allows a program to issue I/O requests and continue processing without waiting until the I/O has completed. No-wait I/O overlaps the processing of multiple I/O requests, overlaps I/O with CPU processing, and responds to the completion of the first of several outstanding I/O requests.

All of the SNA IMF standard (wait) MPE I/O intrinsics can be used with no-wait I/O. The following SNA IMF intrinsics can be used only with no-wait I/O:

ABORT3270	
IODONTWAIT	(SNA IMF/V only)
IODONTWAIT3270	(SNA IMF/XL only)
IOWAIT	(SNA IMF/V only)
IOWAIT3270	(SNA IMF/XL only)

The rules for using these intrinsics are consistent with the rules for using MPE V and MPE XL intrinsics. In fact, two of the SNA IMF intrinsics (IOWAIT and IODONTWAIT) are MPE V and MPE XL intrinsics. Their use as SNA IMF intrinsics differs from their use as MPE intrinsics in two ways:

1. The *cstation* parameter of the IOWAIT and IODONTWAIT intrinsics is interpreted as the *result* parameter of the RECV3270 and TRAN3270 intrinsics.
2. The *tcount* and *target* parameters of the IOWAIT and IODONTWAIT intrinsics are meaningless in SNA IMF. They are optional in calls to IOWAIT and IODONTWAIT, but the *tcount* parameter must be passed in calls to the SNA IMF/XL intrinsics IOWAIT3270 and IODONTWAIT3270.

The MPE intrinsics IOWAIT and IODONTWAIT can be called only from SNA IMF/V. IOWAIT and IODONTWAIT cannot be called directly from SNA IMF/XL, so you must use the SNA IMF/XL intrinsics IOWAIT3270 and IODONTWAIT3270, which in turn call the MPE intrinsics IOWAIT and IODONTWAIT.

The intrinsic descriptions in this chapter are listed in alphabetical order. Each description includes parameter definitions, a list of possible return codes that the intrinsic can generate, and the intrinsic calling sequence. Abbreviations above the parameters in the syntax descriptions indicate the data types of the parameters. The abbreviations are as follows:

I	Integer passed by reference
IV	Integer passed by value

L	Logical
LA	Logical Array

See Appendix A , “Intrinsic Result Codes,” for a complete list of all result codes returned in the *result* parameter of SNA IMF intrinsics.

The descriptions of IOWAIT, IOWAIT3270, IODONTWAIT, and IODONTWAIT3270 on the following pages show only the SPL calling sequences. For a description of how to use these intrinsics in FORTRAN, Pascal, COBOL II, or C, see the *FORTRAN/3000 Reference Manual* (MPE V), the *FORTRAN 77/XL Reference Manual* (MPE XL), the *Pascal/3000 Reference Manual* (MPE V), the *Pascal Reference Manual* (MPE XL), the *COBOL II/3000 Reference Manual* (MPE V), the *COBOL II Reference Manual* (MPE XL), or the *C/XL Reference Manual* (MPE XL).

No-wait MPE I/O requires MPE privileged mode (PM) capability. However, you do not need PM capability to write an SNA IMF application that uses no-wait I/O, because the SNA IMF intrinsics themselves have PM capability. When your application calls the SNA IMF OPEN3270 intrinsic with bit 15 of the *flags* parameter set to 1 (the no-wait I/O option), your program can call any of the SNA IMF intrinsics with no-wait I/O, whether or not you have PM capability.

If you do not have PM capability, SNA IMF intrinsics are the only intrinsics you can call with no-wait I/O. If you call, for example, the MPE FREAD intrinsic, your program will suspend processing until the I/O request has finished, even though you specified no-wait I/O in your OPEN3270 intrinsic call.

You need PM capability to call the MPE V and MPE XL intrinsic FOPEN from a program that uses no-wait I/0.

WARNING

The normal checks and limitations that apply to the standard users in MPE V and MPE XL are bypassed in privileged mode. It is possible for a privileged mode program to destroy file integrity, including the MPE V and MPE XL operating system software. Hewlett-Packard will investigate and attempt to resolve problems resulting from the use of privileged mode. This service, which is not provided under the standard Service Contract, is available on a time and materials billing basis. However, Hewlett-Packard will not support, correct, or attend to any modification to the MPE V and MPE XL operating system software.

If your program will use no-wait MPE I/O, it must be written in COBOL II, FORTRAN, Pascal, SPL, or C. The IOWAIT, IOWAIT3270, IODONTWAIT, and IODONTWAIT3270 intrinsics return a functional value, pass parameters by value, allow a variable number of parameters, and set condition codes. Neither COBOL nor BASIC has these capabilities.

NOTE

In the following discussion, "IOWAIT" is used to mean IOWAIT for SNA IMF/V and IOWAIT3270 for SNA IMF/XL. "IODONTWAIT" is used to mean IODONTWAIT for SNA IMF/V and IODONTWAIT3270 for SNA IMF/XL. Use the intrinsics that are appropriate to your software.

When to Use No-Wait I/O

You may want to use no-wait I/O with SNA IMF to do the following:

- Overlap processing on the host by using multiple SNA IMF devices.
- Respond to host or local (HP 3000) terminal interrupts asynchronously.

You can overlap processing on the host during execution of one program on the HP 3000 by using multiple HP devices. With no-wait I/O, the program can issue a `RECV3270` request for each device in use and then wait for the first host response with `IOWAIT`.

You may want some of your applications to respond either to a host interrupt or to a terminal local to the HP 3000. For example, when Pass Thru, which uses SNA IMF intrinsics, issues a read to the HP terminal, it follows the read with a call to `RECV3270` and then a call to `IOWAIT`. `IOWAIT` delays Pass Thru processing until either the `RECV3270` or the terminal read completes.

When No-Wait I/O is Unnecessary

It is possible to achieve some overlap of host and HP 3000 processing without using no-wait I/O. With standard blocked I/O, you can issue TRAN3270 to start host processing and then do some local (HP 3000) processing before calling RECV3270. If the host completes processing and sends new data to your device, this data is stored in the internal screen image to wait for your RECV3270 call. This method of I/O processing may be acceptable to you if you do not have to worry about the host modifying your internal screen image again before you call the RECV3270 intrinsic.

How to Use No-Wait I/O

To use no-wait I/O with SNA IMF, set bit 15 of the OPEN3270 *flags* parameter to 1. Call either IOWAIT or IODONTWAIT after each TRAN3270 or RECV3270 call to determine whether the requested I/O operation is finished. IOWAIT suspends your program until the I/O operation is complete. IODONTWAIT checks to see if I/O is complete but does not wait for its completion.

Use IOWAIT or IODONTWAIT in addition to RECV3270 and TRAN3270, not instead of them.

As with standard wait I/O, a TRAN3270 request issued under no-wait I/O completes immediately; that is, it does not wait for a response from the host. After a TRAN3270 call, issue either IOWAIT or IODONTWAIT to verify that the TRAN3270 request actually completed the I/O.

A RECV3270 request completes as soon as data is received from the host. If the host sends data before RECV3270 is called, the data is stored in the internal screen image. No further SNA IMF internal I/O processing of this data is necessary; as soon as you call RECV3270 and IOWAIT or, they complete.

With no-wait I/O, some result codes normally returned to TRAN3270 and RECV3270 are returned in the *cstation* parameter of the IOWAIT or IODONTWAIT intrinsic instead. These result codes are listed in the intrinsic descriptions, later in this chapter.

TRAN3270 and RECV3270 do not initiate I/O if any value other than zero is returned in the *result* parameter. If your program receives a code other than zero, do not call IOWAIT or IODONTWAIT. Resolve the error and reissue your call to TRAN3270 or RECV3270.

ABORT3270

ABORT3270 aborts an outstanding no-wait RECV3270 or TRAN3270 request.

Syntax

```
                I         I
ABORT3270       (terminalid, result)
```

Parameters

terminalid (input)

Integer identifying the terminal. The *terminalid* is returned in a call to the OPEN3270 intrinsic.

result (output)

The following values can be generated by the ABORT3270 intrinsic:

0 = Successful completion.

1 = Device not open.

22 = BASIC calling sequence error has occurred.

25 = Intrinsic call made while in split stack mode.

26 = Intrinsic call made with the parameter value out of bounds.

Description

The ABORT3270 intrinsic aborts an outstanding no-wait RECV3270 or TRAN3270 request. Do not call ABORT3270 from SNA IMF programs that use standard MPE V and MPE XL I/O; call it only from programs that use no-wait I/O.

If no I/O requests are outstanding, the *result* parameter will return zero.

COBOL II Calling Sequence

CALL "CABORT3270" USING TERMINALID RESULT. (on MPE V and in compatibility mode on MPE XL)

CALL INTRINSIC "ABORT3270" USING TERMINALID RESULT. (in native mode on MPE XL)

Both parameters are numeric data items.

FORTRAN Calling Sequence

```
CALL ABORT3270 (TERMINALID, RESULT)
```

Both parameters are integer variables.

SPL Calling Sequence

```
ABORT3270 (TERMINALID, RESULT)
```

Both parameters are integer variables.

Pascal Calling Sequence

```
ABORT3270 (TERMINALID, RESULT);
```

Both parameters are short integer variables.

C/XL Calling Sequence

```
ABORT3270 (&TERMINALID, &RESULT);
```

Both parameters are of type short.

IODONTWAIT

For SNA IMF/V only. **IODONTWAIT** provides the status of a previous I/O operation.

Syntax

```
    IV      LA       I        L
IODONTWAIT  (filenum, target, tcount, cstation)
```

Parameters

filenum (input)

Integer identifier specifying the file number or SNA IMF *terminalid* for which an I/O request is pending. If you specify zero, IODONTWAIT checks to see if any no-wait I/O request in your program has completed. *filenum* is a required parameter.

target (input)

Logical array. The *target* parameter has no meaning for SNA IMF. If you omit this parameter, be sure to retain the comma.

tcount (output)

Integer. The *tcount* parameter has no meaning for SNA IMF. If you omit this parameter, be sure to retain the comma.

cstation (output)

Integer that can contain selected SNA IMF *result* codes. The codes returned in *cstation* could, with wait I/O, be returned through the *result* parameter of either the RECV3270 or the TRAN3270 intrinsic. Several other result codes may be returned through IODONTWAIT, instead of either RECV3270 or TRAN3270, if the condition causing them arises during I/O.

The following codes can be returned through the *cstation* parameter of the IODONTWAIT intrinsic:

0 = Successful completion.

23 = Keyboard enable timeout has occurred.

24 = Response timeout has occurred.

91 = LU.T1 bind received.

92 = Unbind received.

93 = LU.T3 bind received.

94 = HOLDPRINT timer has expired (10 minutes).

95 = Host application requests PA key entry.

301 = Illegal DB register.

303 = Invalid SNA catalog file.

305 = Parameter bounds violation.

307 = Session is active.

308 = Session is inactive.

310 = Bad PI in RH.

311 = Bad BCI in RH.

312 = Bad ECI in RH.

313 = Bad EDI in RH.

314 = Reserved bits in RH must be set to zero.

315 = Internal Error.

316 = Invalid RU size.

319 = LU-SSCP message pending.

320 = RU buffer too small.

323 = Negative LU-SSCP response.

333 = Invalid InfoWanted parameter.

336 = Link shutdown occurred.

337 = Protocol shutdown requested.

338 = Quiesce shutdown requested.

340 = No stack space.

348 = Invalid data offset.

351 = Link Failure occurred.

352 = Transport Internal Error Shutdown.

353 = Hierarchical Shutdown.

400 = Expedited response pending.

401 = Data traffic inactive.

402 = SDT request not received.

403 = Invalid session control protocol.

```
404 = RQR request pending.
405 = STSN request not pending.
407 = Unsupported CRV request/response.
408 = Unsupported session control request.
```

Description

Use IODONTWAIT with no-wait MPE V I/O. IODONTWAIT either tells your program that a previous I/O operation has completed, or it returns before completion. IODONTWAIT is also an MPE V and MPE XL intrinsic. See the *MPE V Intrinsics Reference Manual* or the *MPE XL Intrinsics Reference Manual* for more information.

If you open an SNA IMF/V device, specifying no-wait I/O in your call to the OPEN3270 intrinsic, you must follow every RECV3270 and TRAN3270 request with a call to either IODONTWAIT or IOWAIT, before you can call any other intrinsic. You can delay the call to IODONTWAIT or IOWAIT as long as necessary to allow effective I/O and processing overlap.

The IODONTWAIT intrinsic acts the same as IOWAIT with one exception: if your program calls IODONTWAIT, and no I/O has completed, control is returned to your program. (Condition code CCE is returned, and a zero is returned in the *cstation* parameter.) If your program calls IOWAIT, and no I/O has completed, your program is suspended until some I/O completes.

You can call IODONTWAIT from programs written in SPL, COBOL II, Pascal, and FORTRAN.

SPL Procedure Declaration

```
INTEGER PROCEDURE IODONTWAIT (FILENUM, TARGET, TCOUNT,
CSTATION);

INTEGER FILENUM,TCOUNT;

LOGICAL CSTATION;

LOGICAL ARRAY TARGET;

OPTION VARIABLE;
```

IODONTWAIT has a functional return, as shown below in the calling sequence. FNUM returns an integer that represents the *terminalid* for which the I/O operation completed. If no I/O completed, zero is returned in FNUM.

SPL Calling Sequence

```
FNUM:=IODONTWAIT (FILENUM, TARGET, TCOUNT, CSTATION);
```

Condition Codes

CCE	Request granted. If the functional return is non-zero, then I/O completion occurred with no errors. If the return is zero, then no I/O has completed.
CCG	An end of file was encountered.
CCL	Request denied. Normal I/O completion did not occur because there were no I/O requests pending, a parameter error occurred, or an abnormal I/O completion occurred.

IODONTWAIT3270

For SNA IMF/XL only. **IODONTWAIT3270** provides the status of a previous I/O operation.

Syntax

```
    IV      LA       I        L
IODONTWAIT3270  (filenum, target, tcount, cstation)
```

Parameters

filenum (input)

Integer identifier specifying the file number or SNA IMF *terminalid* for which an I/O request is pending. If you specify zero, IODONTWAIT3270 checks to see if any no-wait I/O request in your program has completed. *filenum* is a required parameter.

target (input)

Logical array. The *target* parameter has no meaning for SNA IMF. If you omit this parameter, be sure to retain the comma.

tcount (output)

Integer. Do not omit this parameter. It has no meaning for SNA IMF, but you must include it in the call, because IODONTWAIT3270 must pass it on to IODONTWAIT.

cstation (output)

Integer that can contain selected SNA IMF *result* codes. The codes returned in *cstation* could, with wait I/O, be returned through the *result* parameter of either the RECV3270 or the TRAN3270 intrinsic. Several other result codes may be returned through IODONTWAIT3270, instead of either RECV3270 or TRAN3270, if the condition causing them arises during I/O.

The following codes can be returned through the *cstation* parameter of the IODONTWAIT3270 intrinsic:

0 = Successful completion.

23 = Keyboard enable timeout has occurred.

24 = Response timeout has occurred.

```
91 = LU.T1 bind received.
92 = Unbind received.
93 = LU.T3 bind received.
94 = HOLDPRINT timer has expired (10 minutes).
95 = Host application requests PA key entry.
301 = Illegal DB register.
303 = Invalid SNA catalog file.
305 = Parameter bounds violation.
307 = Session is active.
308 = Session is inactive.
310 = Bad PI in RH.
311 = Bad BCI in RH.
312 = Bad ECI in RH.
313 = Bad EDI in RH.
314 = Reserved bits in RH must be set to zero.
315 = Internal Error.
316 = Invalid RU size.
319 = LU-SSCP message pending.
320 = RU buffer too small.
323 = Negative LU-SSCP response.
333 = Invalid InfoWanted parameter.
336 = Link shutdown occurred.
337 = Protocol shutdown requested.
338 = Quiesce shutdown requested.
340 = No stack space.
348 = Invalid data offset.
351 = Link Failure occurred.
352 = Transport Internal Error Shutdown.
353 = Hierarchical Shutdown.
400 = Expedited response pending.
401 = Data traffic inactive.
402 = SDT request not received.
```

```
403 = Invalid session control protocol.
404 = RQR request pending.
405 = STSN request not pending.
407 = Unsupported CRV request/response.
408 = Unsupported session control request.
```

Description

Use IODONTWAIT3270 with no-wait MPE XL I/O. IODONTWAIT3270 either tells your program that a previous I/O operation has completed, or it returns before completion. IODONTWAIT3270 calls IODONTWAIT, which is an MPE V and MPE XL intrinsic. See the *MPE V Intrinsics Reference Manual* or the *MPE XL Intrinsics Reference Manual* for more information.

If you open an SNA IMF/XL device, specifying no-wait I/O in your call to the OPEN3270 intrinsic, you must follow every RECV3270 and TRAN3270 request with a call to either IODONTWAIT3270 or IOWAIT3270, before you can call any other intrinsic. You can delay the call to IODONTWAIT3270 or IOWAIT3270 as long as necessary to allow effective I/O and processing overlap.

The IODONTWAIT3270 intrinsic acts the same as IOWAIT3270 with one exception: if your program calls IODONTWAIT3270, and no I/O has completed, control is returned to your program. (Condition code CCE is returned, and a zero is returned in the *cstation* parameter.) If your program calls IOWAIT3270, and no I/O has completed, your program is suspended until some I/O completes.

You can call IODONTWAIT3270 from programs written in SPL, COBOL II, Pascal, C, and FORTRAN.

SPL Procedure Declaration

```
INTEGER PROCEDURE IODONTWAIT3270 (FILENUM, TARGET, TCOUNT,
CSTATION);

INTEGER FILENUM,TCOUNT;

LOGICAL CSTATION;

LOGICAL ARRAY TARGET;

OPTION VARIABLE;
```

IODONTWAIT3270 has a functional return, as shown below in the calling sequence. FNUM returns an integer that represents the *terminalid* for which the I/O operation completed. If no I/O completed, zero is returned in FNUM.

SPL Calling Sequence

```
FNUM:=IODONTWAIT3270 (FILENUM, TARGET, TCOUNT, CSTATION);
```

Condition Codes

CCE	Request granted. If the functional return is non-zero, then I/O completion occurred with no errors. If the return is zero, then no I/O has completed.
CCG	An end of file was encountered.
CCL	Request denied. Normal I/O completion did not occur because there were no I/O requests pending, a parameter error occurred, or an abnormal I/O completion occurred.

IOWAIT

For SNA IMF/V only. **IOWAIT** waits for the completion of a previous no-wait I/O request.

Syntax

```
                IV      LA      I         L
IOWAIT          (filenum, target, tcount, cstation)
```

Parameters

filenum (input)

Integer identifier specifying the file number or SNA IMF *terminalid* for which an I/O request is pending. If you specify zero, IOWAIT checks to see if any no-wait I/O request in your program has completed. *filenum* is a required parameter.

target (input)

Logical array. The *target* parameter has no meaning for SNA IMF. If you omit this parameter, be sure to retain the comma.

tcount (output)

Integer. The *tcount* parameter has no meaning for SNA IMF. If you omit this parameter, be sure to retain the comma.

cstation (output)

Integer that can contain selected SNA IMF *result* codes. The codes returned in *cstation* could, with wait I/O, be returned through the *result* parameter of either the RECV3270 or the TRAN3270 intrinsic. Several other result codes may be returned through IOWAIT, instead of either RECV3270 or TRAN3270, if the condition causing them arises during I/O.

The following codes can be returned through the *cstation* parameter of the IOWAIT intrinsic:

0 = Successful completion.

23 = Keyboard enable timeout has occurred.

24 = Response timeout has occurred.

91 = LU.T1 bind received.

```
92 = Unbind received.
93 = LU.T3 bind received.
94 = HOLDPRINT timer has expired (10 minutes).
95 = Host application requests PA key entry.
301 = Illegal DB register.
303 = Invalid SNA catalog file.
305 = Parameter bounds violation.
307 = Session is active.
308 = Session is inactive.
315 = Internal Error.
319 = LU-SSCP message pending.
320 = RU buffer too small.
336 = Link shutdown occurred.
337 = Protocol shutdown requested.
338 = Quiesce shutdown requested.
340 = No stack space.
348 = Invalid data offset.
351 = Link Failure occurred.
352 = Transport Internal Error Shutdown.
353 = Hierarchical Shutdown.
```

Description

Use IOWAIT with no-wait MPE V I/O. IOWAIT suspends processing until an outstanding I/O request completes. IOWAIT is also an MPE V and MPE XL intrinsic. See the *MPE V Intrinsics Reference Manual* or the *MPE XL Intrinsics Reference Manual* for more information.

If you open an SNA IMF/V device, specifying no-wait I/O in your call to the OPEN3270 intrinsic, you must follow every RECV3270 and TRAN3270 request with a call to either IODONTWAIT or IOWAIT, before you can call any other intrinsic. You can delay the call to IODONTWAIT or IOWAIT as long as necessary to allow effective I/O and processing overlap.

The IOWAIT intrinsic acts the same as the IODONTWAIT intrinsic with one exception: if your program calls IOWAIT, and no I/O has completed, your program is suspended until some I/O completes; if your program calls IODONTWAIT, and no I/O has completed, control is returned to your program.

You can call IOWAIT from programs written in SPL, COBOL II, Pascal, or FORTRAN.

SPL Procedure Declaration

```
INTEGER PROCEDURE IOWAIT (FILENUM, TARGET, TCOUNT,
CSTATION);

INTEGER FILENUM,TCOUNT;

LOGICAL CSTATION;

LOGICAL ARRAY TARGET;

OPTION VARIABLE;
```

IOWAIT has a functional return, as shown below in the calling sequence. The IOWAIT intrinsic returns an integer that represents the *terminalid* for which the I/O completion occurred. If no completion occurred, zero is returned in FNUM.

SPL Calling Sequence

```
FNUM:=IOWAIT (FILENUM, TARGET, TCOUNT, CSTATION);
```

Condition Codes

CCE	Request granted. If the functional return is non-zero, then I/O completion occurred with no errors. If the return is zero, then no I/O has completed.
CCG	An end of file was encountered.
CCL	Request denied. Normal I/O completion did not occur because there were no I/O requests pending, a parameter error occurred, or an abnormal I/O completion occurred.

IOWAIT3270

For SNA IMF/XL only. **IOWAIT3270** waits for the completion of a previous no-wait I/O request.

Syntax

```
                IV      LA      I         L
IOWAIT3270     (filenum, target, tcount, cstation)
```

Parameters

filenum (input)

Integer identifier specifying the file number or SNA IMF *terminalid* for which an I/O request is pending. If you specify zero, IOWAIT3270 checks to see if any no-wait I/O request in your program has completed. filenum is a required parameter.

target (input)

Logical array. The *target* parameter has no meaning for SNA IMF. If you omit this parameter, be sure to retain the comma.

tcount (output)

Integer. Do not omit this parameter. It has no meaning for SNA IMF, but you must include it in the call, because IOWAIT3270 must pass it on to IOWAIT.

cstation (output)

Integer that can contain selected SNA IMF *result* codes. The codes returned in *cstation* could, with wait I/O, be returned through the *result* parameter of either the RECV3270 or the TRAN3270 intrinsic. Several other result codes may be returned through IOWAIT3270, instead of either RECV3270 or TRAN3270, if the condition causing them arises during I/O.

The following codes can be returned through the *cstation* parameter of the IOWAIT3270 intrinsic:

0 = Successful completion.

23 = Keyboard enable timeout has occurred.

24 = Response timeout has occurred.

91 = LU.T1 bind received.

```
92 = Unbind received.
93 = LU.T3 bind received.
94 = HOLDPRINT timer has expired (10 minutes).
95 = Host application requests PA key entry.
301 = Illegal DB register.
303 = Invalid SNA catalog file.
305 = Parameter bounds violation.
307 = Session is active.
308 = Session is inactive.
315 = Internal Error.
319 = LU-SSCP message pending.
320 = RU buffer too small.
336 = Link shutdown occurred.
337 = Protocol shutdown requested.
338 = Quiesce shutdown requested.
340 = No stack space.
348 = Invalid data offset.
351 = Link Failure occurred.
352 = Transport Internal Error Shutdown.
353 = Hierarchical Shutdown.
```

Description

Use IOWAIT3270 with no-wait MPE XL I/O. suspends processing until an outstanding I/O request completes. IOWAIT3270 calls IOWAIT, which is an MPE V and MPE XL intrinsic. See the *MPE V Intrinsics Reference Manual* or the *MPE XL Intrinsics Reference Manual* for more information.

If you open an SNA IMF/XL device, specifying no-wait I/O in your call to the OPEN3270 intrinsic, you must follow every RECV3270 and TRAN3270 request with a call to either IODONTWAIT3270 or IOWAIT3270, before you can call any other intrinsic. You can delay the call to IODONTWAIT3270 or IOWAIT3270 as long as necessary to allow effective I/O and processing overlap.

The IOWAIT3270 intrinsic acts the same as the IODONTWAIT3270 intrinsic with one exception: if IOWAIT3270 is called and no I/O has completed, the process that issued the call is suspended until some I/O

completes; if your program calls IODONTWAIT3270, and no I/O has completed, control is returned to your program.

You can call IOWAIT3270 from programs written in SPL, COBOL II, FORTRAN, C, or Pascal.

SPL Procedure Declaration

```
INTEGER PROCEDURE IOWAIT3270 (FILENUM, TARGET, TCOUNT,
CSTATION);

INTEGER FILENUM,TCOUNT;

LOGICAL CSTATION;

LOGICAL ARRAY TARGET;

OPTION VARIABLE;
```

IOWAIT3270 has a functional return, as shown below in the calling sequence. The IOWAIT3270 intrinsic returns an integer that represents the *terminalid* for which the I/O completion occurred. If no completion occurred, zero is returned in FNUM.

SPL Calling Sequence

```
FNUM:=IOWAIT3270 (FILENUM, TARGET, TCOUNT, CSTATION);
```

Condition Codes

CCE	Request granted. If the functional return is non-zero, then I/O completion occurred with no errors. If the return is zero, then no I/O has completed.
CCG	An end of file was encountered.
CCL	Request denied. Normal I/O completion did not occur because there were no I/O requests pending, a parameter error occurred, or an abnormal I/O completion occurred.

A Intrinsic Result Codes

This Appendix lists all the result codes that can be returned to SNA IMF intrinsics. Result codes are returned in the *result* parameter of most intrinsics; however, with no-wait MPE I/O, some result codes may be returned in the *cstation* parameter of the IOWAIT, IOWAIT3270, IODONTWAIT, and IODONTWAIT3270 intrinsics.

This Appendix lists the result codes by number. Each result code is followed by the message associated with it, the probable cause of the result code, and the actions you should take to resolve any errors.

Messages generated by Pass Thru can appear on Pass Thru terminals and printers, and on the system console. For more information about Pass Thru messages, see the *Using SNA IMF Pass Thru* manual.

NOTE

Although this Appendix focuses on intrinsic result codes, you may encounter other messages while using SNA IMF. Refer to the *SNA Link Services Reference Manual* or *Using the Node Management Services Utilities* for generic messages produced by the NMMGR, NMMAINT, and NMDUMP utilities.

If you are using the HP SNA Server/Access products on a LAN, some of the *ACQUIRE3270* parameters are different from those described in Chapter 3 , “Intrinsics Used with Standard MPE I/O,” of this manual. Therefore, a few of the messages in this Appendix relate only to the use of SNA IMF with the HP SNA Server/Access products. Consult the *HP SNA Server/Access User's Guide* for more specific information.

Less than 0

MESSAGE: The message that appears is an MPE V message.

CAUSE: This is an MPE V message, not an SNA IMF message.

ACTION: Look up the absolute value of the result code under “GENMESSAGE” in the *MPE V Intrinsics Reference Manual*.

0

MESSAGE: Successful completion.

CAUSE: The intrinsic completed successfully.

ACTION: None.

1

MESSAGE: Device not open.

CAUSE: An intrinsic other than OPEN3270 was called without first having opened the device.

ACTION: Call the OPEN3270 intrinsic to open the device before attempting to use it.

2

MESSAGE: Could not access IMF configuration file. (MPE V only)

CAUSE: The OPEN3270 intrinsic could not FOPEN the specified IMF configuration file.

ACTION: Check that the IMF configuration file name is syntactically correct. Make sure the IMF configuration file exists as an old, permanent file.

2 **MESSAGE: Invalid devicenum parameter specified. (MPE XL only)**

CAUSE: This error is returned to the OPEN3270 intrinsic, indicating that an inappropriate *devicenum* parameter value was specified.

ACTION: This error is returned to the OPEN3270 intrinsic, indicating that an inappropriate *devicenum* parameter value was specified.

9 **MESSAGE: Host modified screen since last receive request. (MPE V only)**

CAUSE: The host has modified the screen since the last call to RECV3270.

ACTION: Because any attempt to change or send the screen at this point would be based on old information, issue a call to the RECV3270 intrinsic to clear the error and receive the latest screen. Call the READFIELD or READSCREEN intrinsic to give you the new contents of the screen.

10 **MESSAGE: Attempt made to update a protected field.**

CAUSE: You cannot change the contents of a protected field.

ACTION: Call the SCREENATTR or FIELDATTR intrinsic to find the unprotected fields of your screen and make the changes again.

11 **MESSAGE: Non-existent field number specified.**

CAUSE: The field number specified in a call to the FIELDATTR, READFIELD, or WRITEFIELD intrinsic does not exist in the screen.

ACTION: Call the SCREENATTR intrinsic to find the numbers of the fields.

12 **MESSAGE: Invalid character in field or data stream.**

CAUSE: At least one character in the field or data stream was outside the range of allowable characters.

ACTION: See the descriptions of the *outbuf* parameter in the WRITEFIELD and STREAM3270 intrinsics. Make sure that every character is within the range specified.

13 **MESSAGE: Field length specified for WRITEFIELD is too long.**

CAUSE: The field length specified in the call to the WRITEFIELD intrinsic is longer than the field length in the screen.

ACTION: Call the FIELDATTR intrinsic to find the current field length. Specify a field length less than or equal to the current field length.

14 **MESSAGE: Attempt made to update a field or transmit from an LU.T3 printer.**

CAUSE: You have opened an LU.T3 printer. You are not allowed to change the contents of any field or to transmit from an LU.T3 printer.

ACTION: Open an LU.T2 or LU.T1 session. Only LU.T1 sessions support printer key entry.

15 **MESSAGE: Invalid AID parameter was sent.**

CAUSE: The *aid* parameter specified in the TRAN3270 intrinsic is invalid.

ACTION: See the TRAN3270 intrinsic description for a list of valid *aid* codes. Note that the valid codes differ depending on the type of LU session.

16 **MESSAGE: Invalid cursor address was specified.**

CAUSE: The *cursorrow* or *cursorcolumn* parameter of the TRAN3270 or STREAM3270 intrinsic is invalid. The row address must be less than the number of rows on the screen. The column address must be less than the number of columns on the screen. (SNA IMF starts with 0 when counting rows and columns.)

ACTION: See the TRAN3270 and STREAM3270 intrinsic descriptions for valid cursor address values

17 **MESSAGE: Attempt made to write to a field where input is inhibited.**

CAUSE: You called the WRITEFIELD, STREAM3270, or TRAN3270 intrinsic while input was inhibited by the host.

ACTION: Either wait for the host to reenable input by issuing another call to the RECV3270 intrinsic or call the RESET3270 or STREAM3270 intrinsic to emulate pressing the **[RESET]** key.

21 **MESSAGE: Field offset specified is out of range.**

CAUSE: The specified offset value was outside the field length.

ACTION: Call the FIELDATTR intrinsic to find the length of the field. Be sure to specify an offset that is within the field length, and make sure that the offset for the WRITESTREAM and READSTREAM intrinsics is valid.

22 **MESSAGE: BASIC calling sequence error has occurred.**

CAUSE: An error was detected in the BASIC calling sequence.

ACTION: Make sure the parameters of the intrinsic call from BASIC are in the correct order.

23 **MESSAGE: Keyboard enable timeout has occurred.**

CAUSE: The host failed to enable the keyboard within the time limit specified in the *timeout* parameter of the OPEN3270 intrinsic.

ACTION: Use the RESET3270 intrinsic to reset the keyboard for key entry. This response may be host application specific.

24 **MESSAGE: Response timeout has occurred.**

CAUSE: The host is possibly waiting for data from the user application or failed to send data to a device within the transmit/receive time limit set in the *timeout* parameter of the OPEN3270 intrinsic.

ACTION: If this message was expected, no action is required; the host may be expecting input from the user application. If the message was not expected, either the timer value given in the OPEN3270 intrinsic was not long enough to allow the host enough time to send data, or communications with the host have been disrupted. If the timer value was not long enough, use a longer timer value. If communications with the host have been disrupted, call the CLOSE3270 intrinsic to close the session.

25 **MESSAGE: Intrinsic call made while in split stack mode.**

CAUSE: An intrinsic was called while in split stack mode.

ACTION: Make sure that DB is pointing to your own stack before you call SNA IMF intrinsics.

26 **MESSAGE: Intrinsic call made with the parameter value out of bounds.**

CAUSE: A parameter address was either less than DL or greater than S when you called this intrinsic. You also could have passed an out of range parameter. A byte address may have been used instead of a word address.

ACTION: Check the call parameters and make sure they are being passed by reference and not by value.

27 **MESSAGE: Could not open device. Insufficient virtual memory was available.**

CAUSE: There was insufficient virtual memory to allocate the extra data segment to contain the screen.

ACTION: Not enough virtual memory was available for system load, or too many processes are functioning at once. Have your system administrator configure more virtual memory.

28 **MESSAGE: Could not open device. Insufficient real memory was available.**

CAUSE: There was not enough room in the PCBX of your process to allocate a file control entry for the device.

ACTION: Allocate more space for process using the MPE PREP command parameter *DL = dlsize*.

29 **MESSAGE: Called intrinsic with a request already outstanding.**

CAUSE: You were in no-wait I/O and called the RECV3270 or TRAN3270 intrinsic while a previous request to one of these intrinsics was outstanding.

ACTION: Call the IOWAIT or IODONTWAIT intrinsic to complete the request before issuing any new SNA IMF intrinsic calls, or call ABORT3270 to abort the previous request.

30 **MESSAGE: Internal error occurred in IMF intrinsic.**

CAUSE: An internal software error has occurred in an SNA IMF intrinsic.

ACTION: A file named IMFDUMxx was probably created in your group and account or in the PUB group of the SYS account. Save this file for your HP representative. Note the circumstances and report them to your HP representative.

38 **MESSAGE: Cannot start OUTBUF on an attribute byte.**

CAUSE: The stream data you supplied positioned the cursor on top of an attribute byte. You may not write over an attribute character.

ACTION: Use the ATTRLIST intrinsic to locate the attribute bytes.

42 **MESSAGE: Specified MAXINBUFLEN parameter is too large.**

CAUSE: You specified a *maxinbuflen* value that extends beyond the end of the screen. *maxinbuflen* does not wrap to the beginning of the screen.

ACTION: Make sure values for the *offset* and *maxinbuflen* parameters are less than the screen size.

43 **MESSAGE: Transparent mode not requested for LU.T1 emulation.**

CAUSE: You specified a *devicenum* of -1 in your OPEN3270 intrinsic call without setting bit 14 of the OPEN3270 *flags* parameter to one.

ACTION: Set bit 14 of the OPEN3270 *flags* parameter to one and be sure to call only those intrinsics that are valid for transparent mode.

44 **MESSAGE: WRITESTREAM called during LU.T1 session.**

CAUSE: The WRITESTREAM intrinsic was called during an LU.T1 session.

ACTION: The WRITESTREAM intrinsic cannot be called during an LU.T1 session. The only data that may be sent to the PLU are the printer keys, which are sent using the TRAN3270 intrinsic.

49 **MESSAGE: Value specified for the INBUF parameter is too small to hold the entire data stream.**

CAUSE: The size of the data stream received from the host is larger than the size of the *inbuf* parameter. This result code applies to the READSTREAM intrinsic.

ACTION: Modify the *maxinbuflen* parameter of the READSTREAM intrinsic so the *inbuf* parameter can hold the entire data stream and issue a call to the READSTREAM intrinsic.

50 **MESSAGE: Called READSTREAM without calling RECV3270 first.**

CAUSE: The RECV3270 intrinsic must be issued to receive host data before the READSTREAM intrinsic can be called.

ACTION: Issue a call to the RECV3270 intrinsic to accept host data. Then call the READSTREAM intrinsic to obtain the data from the extra data segment. This result code applies only to transparent mode.

51 **MESSAGE: Called TRAN3270 without calling WRITESTREAM first.**

CAUSE: The buffer was not filled before trying to transmit.

ACTION: Put the data you want to send to the host in the extra data segment by calling the WRITESTREAM intrinsic before you call the TRAN3270 intrinsic. This result code applies only to transparent mode.

52 **MESSAGE: Data stream is too long.**

CAUSE: The size of the data stream is larger than the maximum allowable size.

ACTION: For 480-character screens, 540 bytes is the maximum size. For 1920-character screens, 2160 bytes is the maximum size. For 3440-character screens, 3870 bytes is the maximum size. If this error occurs when the RECV3270 intrinsic is called, only the first 540, 2160, or 3870 bytes of the data stream will be buffered in the extra data segment. This result code applies only to transparent mode.

53 **MESSAGE: Invalid intrinsic called for data stream mode device.**

CAUSE: An intrinsic was issued that cannot be called while in transparent mode.

ACTION: Do not use an intrinsic such as READFIELD or STREAM3270 in transparent mode. Because there is no internal screen image in transparent mode, do not use any intrinsic that reads from or writes to an internal screen image.

54 **MESSAGE: Device not opened in transparent mode.**

CAUSE: You may not use the data stream intrinsics READSTREAM or WRITESTREAM, because you did not specify transparent mode in the *flags* parameter of your call to the OPEN3270 intrinsic.

ACTION: Request transparent mode by setting bit 14 of the OPEN3270 *flags* parameter to one.

60 **MESSAGE: Invalid spool file priority. Value must be between 1 and 13 inclusive.**

CAUSE: A spool file priority other than a value from 1 through 13 was specified.

ACTION: Make sure the *priority* parameter of the PRINT3270 intrinsic is from 1 through 13. The *priority* parameter is passed on to the FOPEN intrinsic, which is described in the *MPE V Intrinsics Reference Manual* and the *MPE XL Intrinsics Reference Manual*.

61 **MESSAGE: Failed to open PRINT3270 spool file.**

CAUSE: Your attempt to open the spool file for the PRINT3270 intrinsic failed when the FOPEN intrinsic was called.

ACTION: The PRINT3270 intrinsic leaves the *fileid* parameter set to zero so that you may call the file system intrinsic FCHECK to determine the specific reason for FOPEN's failure. Use FCHECK to determine the cause of the error.

62 **MESSAGE: Failed to write to PRINT3270 spool file.**

CAUSE: The PRINT3270 intrinsic attempted to call the file system intrinsic FWRITE to write to the spool file identified by *fileid*; however, the call to FWRITE failed.

ACTION: Make sure the *fileid* value is correct. Insufficient disk space may also cause this error. Call the file system intrinsic FCHECK to determine the specific reason for FWRITE's failure.

63 **MESSAGE: The action parameter must be an integer from 0 through 4.**

CAUSE: An invalid value was specified for the *action* parameter.

ACTION: Make sure the value of the *action* parameter in the PRINT3270 intrinsic is an integer from 0 through 4.

64 **MESSAGE: Wrong file type for PRINT3270 output file.**

CAUSE: The FILE command you used to override the formal designator LOGIMF, which PRINT3270 uses, is incompatible with the requirements of PRINT3270 for a spooled output file.

ACTION: Exit the application and reissue the FILE command for LOGIMF. If you equate the LOGIMF file to another file, you must observe the following restrictions: the file must have a record size of 133 bytes and contain ASCII data. The equated file may not be a KSAM file.

65 **MESSAGE: Failed to close PRINT3270 output spooled file.**

CAUSE: The PRINT3270 intrinsic attempted to call the file system intrinsic FCLOSE to close the file specified by *fileid*; this attempt failed.

ACTION: Make sure you supplied the proper value for *fileid* in your call to the PRINT3270 intrinsic. If you used a file equation for the LOGIMF file, be sure you did not equate it to an existing file. Call the file system intrinsic FCHECK to determine the specific reason for FCLOSE's failure.

66 **MESSAGE: Failed to open CATIMF.PUB.SYS.**

CAUSE: The file CATIMF.PUB.SYS, which is the message catalog for SNA IMF, could not be opened.

ACTION: Make sure the CATIMF.PUB.SYS file is present.

67 **MESSAGE: GENMESSAGE failed to extract message. (MPE V only)**

CAUSE: The MPE intrinsic GENMESSAGE did not execute properly.

ACTION: Check to see that CATIMF.PUB.SYS and GENMESSAGE are properly installed.

67 **MESSAGE: CATREAD failed to extract message. (MPE XL only)**

CAUSE: The MPE intrinsic CATREAD did not execute properly.

ACTION: Check to see that CATIMF.PUB.SYS and CATREAD are properly installed.

68 **MESSAGE: Out of stack space. Increase your maxdata size.**

CAUSE: There was insufficient stack space to print the screen.

ACTION: Increase the maximum size of the data area using the MPE PREP command parameter MAXDATA = *segsize*.

69 **MESSAGE: Unable to find snaclassname.**

CAUSE: You entered the node name and the separator #, but you did not enter anything after the separator.

ACTION: Enter at least one alphabetic character after the separator # to identify the snaclassname.

70 **MESSAGE: Invalid LDEV specified or LDEV is already in use.**

CAUSE: You specified an incorrect value for the *ldev*, or the *ldev* is being used by another process.

ACTION: Verify that you used the correct value for *ldev*. Also, determine whether another process is using the same *ldev*.

71 **MESSAGE: The ENHANCE parameter must be an integer between 0 and 3.**

CAUSE: An invalid value was specified for the *enhance* parameter.

ACTION: Make sure the *enhance* parameter value is an integer from 0 through 3.

72 **MESSAGE: The PRIORTY parameter must be an integer between 1 and 13.**

CAUSE: An invalid value was specified for the *priority* parameter.

ACTION: Make sure the *priority* parameter value is an integer from 1 through 13.

73 **MESSAGE: The BLANKS parameter must be either 0 or 1.**

CAUSE: An invalid value was specified for the *blanks* parameter.

ACTION: Make sure the *blanks* parameter value is either 0 or 1.

74 **MESSAGE: The FORMAT parameter must be an integer between 1 and 4.**

CAUSE: An invalid value was specified for the *format* parameter.

ACTION: Make sure the *format* parameter value is an integer from 1 through 4.

75 **MESSAGE: Invalid flags parameter.**

CAUSE: An invalid value was specified for the *flags* parameter of the ACQUIRE3270 intrinsic.

ACTION: Determine the correct value for the *flags* parameter by checking the description of the ACQUIRE3270 intrinsic in Chapter 3 , "Intrinsics Used with Standard MPE I/O," of this manual.

76 **MESSAGE: TTSSON.PUB.SYS is missing.**

CAUSE: Your HP 3000 could not find the Pass Thru program file, TTSSON.PUB.SYS.

ACTION: Make sure that TTSSON.PUB.SYS is installed. Also, make sure that this file was loaded into the correct group and account.

77 **MESSAGE: An HP LAN nodename must begin with an alphabetic character.**

CAUSE: You began an HP LAN node name with either a numeric or a special character.

ACTION: Modify your node name so that it begins with an alphabetic character. See the *HP SNA Server/Access User's Guide* for more information.

78 **MESSAGE: The SNA Server must be installed properly to use this feature.**

CAUSE: You entered a node name to either route printer output to another node on the LAN or to acquire other terminals and printers on the LAN, but the SNA Server cannot provide service because it is not properly installed.

ACTION: Check with your HP representative to make sure the SNA Server is installed properly. See the *HP SNA Server/Access User's Guide* for more information.

79 **MESSAGE: Fewer than three characters were specified for snanodename#snaclassname.**

CAUSE: You did not enter enough characters to specify the *snanodename#snaclassname*.

ACTION: Enter at least one alphabetic character for the *snanodename*, enter the pound sign (#) for the separator, and enter at least one alphabetic character for the *snaclassname*.

80 **MESSAGE: Your session is in receive state and cannot send data.**

CAUSE: A screen in receive mode cannot send data.

ACTION: The host controls who can send data at any particular time. Right now, you can only receive data. Check the host application.

82 **MESSAGE: Your screen is in the "unowned" state so it cannot send data.**

CAUSE: Your screen is in the unowned state.

ACTION: Use the TRAN3270 intrinsic to enter the **[SYS REQ]** key. Then log on to your system.

83 **MESSAGE: Unable to find HP LAN nodename.**

CAUSE: You entered a colon (:) in the *netinfo* parameter of the ACQUIRE3270 intrinsic, but you did not enter a node name after the colon.

ACTION: Enter a node name after the colon. It may be from one through eight alphanumeric characters and must begin with an alphabetic character. See the *HP SNA Server/Access User's Guide* for more information.

84 **MESSAGE: An HP LAN nodename must contain from one through eight alphanumeric characters.**

CAUSE: Your node name was more than eight characters in length.

ACTION: Modify your node name so that it is from one through eight characters in length. See the *HP SNA Server/Access User's Guide* for more information.

85 **MESSAGE: Unable to find end of user and account string (]).**

CAUSE: You forgot to enter the right bracket after your user and account logon in the *netinfo* parameter of the ACQUIRE3270 intrinsic ([user.account]).

ACTION: Modify your *netinfo* parameter so that your user.account logon is fully bracketed ([user.account]). See the *HP SNA Server/Access User's Guide* for more information.

86 **MESSAGE: An SNA nodename must begin with an alphabetic character.**

CAUSE: You began your node name with either a numeric or a special character.

ACTION: Modify your node name so that it begins with an alphabetic character.

87 **MESSAGE: An SNA classname must begin with an alphabetic character.**

CAUSE: You began your class name with either a numeric or a special character.

ACTION: Modify your class name so that it begins with an alphabetic character.

88 **MESSAGE: A user.account is required for 3287 printer emulation over a LAN.**

CAUSE: You did not enter a user and account after your node name in the *netinfo* parameter of the ACQUIRE3270 intrinsic.

ACTION: Modify your *netinfo* parameter so that you enter the needed logon for printer emulation as part of your *ldev* specification. Be sure that your user and account is enclosed in brackets. See the *HP SNA Server/Access User's Guide* for more information.

89 **MESSAGE: Fewer than three characters were specified for user.account.**

CAUSE: You did not enter enough characters to specify the user and account.

ACTION: Enter at least one alphabetic character for the user, enter the period for the separator, and enter at least one alphabetic character for the account. See the *HP SNA Server/Access User's Guide* for more information.

91 **MESSAGE: LU.T1 bind received.**

CAUSE: Bit 12 of the OPEN3270 *flags* parameter was set to one to allow either LU.T1 or LU.T3 emulation. The secondary LU (the HP 3000) has received a BIND from the primary LU (the IBM host) to begin an SNA LU.T1 session. This message also implies that a transparent mode extra data segment (XDS) has been established for LU.T1 emulation.

ACTION: Issue a call to the RECV3270 intrinsic to begin LU.T1 printer emulation.

92 **MESSAGE: Unbind received.**

CAUSE: Bit 12 of the OPEN3270 *flags* parameter was set to one to allow either LU.T1 or LU.T3 emulation. The secondary LU (the HP 3000) has received an UNBIND from the primary LU (the IBM host).

ACTION: Issue a call to the RECV3270 intrinsic to receive the next LU.T1 or LU.T3 BIND from the host, or call the CLOSE3270 intrinsic to stop printer emulation.

93 **MESSAGE: LU.T3 bind received.**

CAUSE: Bit 12 of the OPEN3270 *flags* parameter was set to one to allow either LU.T1 or LU.T3 emulation. The secondary LU (the HP 3000) has

received a BIND from the primary LU (the IBM host) to begin an SNA LU.T3 session. This message also implies that a non-transparent (screen) mode extra data segment (XDS) has been established for LU.T3 emulation.

ACTION: Issue a call to the RECV3270 intrinsic to begin LU.T3 printer emulation.

94 **MESSAGE: HOLDPRINT timer has expired (10 minutes).**

CAUSE: The printer has notified the IBM host that intervention is required. The HOLDPRINT timer has expired and you must use the TRAN3270 intrinsic to emulate pressing the **[ENABLE PRINT]** printer key to reenable printing from the IBM host (LU.T1 only).

ACTION: When the LU.T1 device is ready to receive, use the TRAN3270 intrinsic to either send the **[ENABLE]** key, or precede the **[ENABLE]** key with the **[PA1]**, **[PA2]**, or **[CANCEL]** key.

95 **MESSAGE: Host application requests PA key entry.**

CAUSE: It is the printer's turn to send data.

ACTION: Use the TRAN3270 intrinsic to enter the **[PA1]** or **[PA2]** key for an LU.T1 printer.

97 **MESSAGE: An attempt was made to write SO (hex "0E") control character to a field where only 8-bit data is allowed. (MPE XL only)**

CAUSE: The field is defined by the host to contain 8-bit characters only. Writing the Shift-Out (SO) control character is not allowed.

ACTION: Use the EXTFIELDATTR intrinsic to find the *dbcsattr* value defined for this field. Check and modify the data.

98 **MESSAGE: An attempt was made to write SO (hex "0E") control character to a field where only 16-bit data is allowed. (MPE XL only)**

CAUSE: The field is defined by the host to contain 16-bit characters only. Writing the Shift-Out (SO) control character is not allowed.

ACTION: Use the EXTFIELDATTR intrinsic to find the *dbcsattr* value defined for this field. Check and modify the data.

99 **MESSAGE: An invalid intrinsic was called while the DBCS option was not set. (MPE XL only)**

CAUSE: You may not use the EXTFIELDATTR intrinsic because you did not specify the DBCS option in the *flags* parameter of your call to the OPEN3270 intrinsic.

ACTION: Specify the DBCS option by setting bit 10 of the OPEN3270 *flags* parameter to one.

100 **MESSAGE: A bad (non-zero) offset was specified in the intrinsic call. (MPE XL only)**

CAUSE: When the DBCS option is set, the *offset* parameter must be zero. The affected intrinsics are ATTRLIST, READFIELD, WRITEFIELD, READSCREEN, READSTREAM, and WRITESTREAM.

ACTION: Specify the *offset* to be zero.

101 **MESSAGE: An error is detected in NLS (Native Language Support). (MPE XL only)**

CAUSE: NLS procedures are used to perform the translation between EBCDIC and ASCII. An error is returned by the NLS routines.

ACTION: Use NLS utilities to verify that the proper conversion tables and NLS are installed correctly.

251 **MESSAGE: Could not open NMCONFIG.PUB.SYS. (MPE XL only)**

CAUSE: The configuration file could not be opened because either it doesn't exist or something is wrong with the file.

ACTION: Check with your system manager to ensure the configuration file has the name NMCONFIG and resides in the PUB group of the SYS account.

252 **MESSAGE: Could not read from NMCONFIG.PUB.SYS. (MPE XL only)**

CAUSE: An unexpected error occurred while attempting to read from the configuration file.

ACTION: Check the configuration file with the VALIDATION option in the NMMGR program. If NMMGR does not reveal the problem, then note the circumstances and report them to your HP representative.

253 **MESSAGE: Could not close NMCONFIG.PUB.SYS. (MPE XL only)**

CAUSE: An error occurred while attempting to close the configuration file.

ACTION: Note the circumstances and report them to your HP representative.

254 **MESSAGE: Invalid SNAnode name. (MPE XL only)**

CAUSE: The node name passed in *snalnkinfo* to OPEN3270 or ACQUIRE3270 is not configured under SNANODE (for MPE V) or IMF (for MPE XL) in the file NMCONFIG.PUB.SYS.

ACTION: Add the node to the configuration file, or use another node that is currently configured.

255 **MESSAGE: Invalid security class name. (MPE XL only)**

CAUSE: The security class name passed in *snalnkinfo* to OPEN3270 or ACQUIRE3270 is not configured under SNANODE in the file NMCONFIG.PUB.SYS.

ACTION: Add the security class to the configuration file, or use another security class that is currently configured.

256 **MESSAGE: Security class not properly configured. (MPE XL only)**

CAUSE: No LUs were found in the configuration file for the specified security class.

ACTION: Add LUs to the security class, or use a security class that is properly configured.

257 **MESSAGE: Program not authorized to use this security class. (MPE XL only)**

CAUSE: Your program is not included in the PGMLIST for the security class specified.

ACTION: Add your program to the PGMLIST for the security class, or use another security class that is already configured properly for use with your program.

258 **MESSAGE: User is not authorized to use this security class. (MPE XL only)**

CAUSE: The user trying to run the program is not included in the USERLIST for the security class specified.

ACTION: Add the user's name to the USERLIST for the security class, or use a security class that is already configured for use by that user.

301 **MESSAGE: Illegal DB register. (MPE V only)**

CAUSE: The DB register was pointing to an extra data segment on a call to an SNA link intrinsic.

ACTION: Note the circumstances and report them to your HP representative.

302 **MESSAGE: Invalid session. A call was made to an SNA Transport intrinsic with a bad session number.**

CAUSE: You probably passed an invalid terminal identifier (*terminalid*) to an SNA IMF intrinsic.

ACTION: Be sure you used the terminal identifier returned by OPEN3270.

303 **MESSAGE: Invalid SNA catalog file. (MPE V only)**

CAUSE: The parameter was omitted on a call to an SNA link intrinsic.

ACTION: Note the circumstances and contact your HP representative.

303 **MESSAGE: Missing Completer Parameter. (MPE XL only)**

CAUSE: The SNA completer was activated with an invalid *Cstation*, *Xfercount*, or *TargetFlag* parameter.

ACTION: Note the circumstances and contact your HP representative.

304 **MESSAGE: Security violation. (MPE V only)**

CAUSE: Your user name or program is not authorized to use this LU class.

ACTION: Check the SNA configuration for your SNA node.

305 **MESSAGE: Parameter bounds violation.**

CAUSE: SNA IMF passed the SNA link product an out-of-bounds parameter.

ACTION: Note the circumstances and contact your HP representative.

306 **MESSAGE: Invalid flag parameter.**

CAUSE: SNA IMF passed an invalid *flag* parameter to an SNA link intrinsic.

ACTION: Note the circumstances and contact your HP representative.

307 **MESSAGE: Session is active.**

CAUSE: The NAU has been activated.

ACTION: None.

309 **MESSAGE: No available AFT entry.**

CAUSE: The SNA link product was unable to expand the PXFILE portion of your program's stack in order to set up an additional AFT entry to accommodate the SNA session you requested.

ACTION: Too many files have been opened by your program. Try again.

310 **MESSAGE: Bad PI in RH.**

CAUSE: Invalid Pacing Indicator in Request/Response Header.

ACTION: Note the circumstances and report them to your HP representative.

311 **MESSAGE: Bad BCI in RH.**

CAUSE: Invalid Begin Chain Indicator in Request/Response Header.

ACTION: Note the circumstances and report them to your HP representative.

312 **MESSAGE: Bad ECI in RH.**

CAUSE: Invalid End Chain Indicator in Request/Response Header.

ACTION: Note the circumstances and report them to your HP representative.

313 **MESSAGE: Bad EDI in RH.**

CAUSE: Invalid Enciphered Data Indicator in Request/Response Header.

ACTION: Note the circumstances and report them to your HP representative.

314 **MESSAGE: Reserved bits in RH must be set to zero.**

CAUSE: SNA IMF failed to initialize reserved bits in the Request/Response Header.

ACTION: Note the circumstances and report them to your HP representative.

315 **MESSAGE: Internal Error.**

CAUSE: The SNA link product detected an internal error in the SNA Transport subsystem.

ACTION: Note the circumstances and report them to your HP representative.

316 **MESSAGE: Invalid RU size.**

CAUSE: SNA IMF passed to an SNA link intrinsic an RU that exceeded the allowable size for this session.

ACTION: Note the circumstances and report them to your HP representative.

317 **MESSAGE: NAU is inactive.**

CAUSE: The NAU is not active.

ACTION: Ensure that the NAU has been activated on the host side.

318 **MESSAGE: Invalid Plabel.**

CAUSE: SNA IMF passed an invalid Plabel to the SNAcontrol intrinsic.

ACTION: Note the circumstances and report them to your HP representative.

319 **MESSAGE: LU-SSCP message pending.**

CAUSE: A message on the LU-SSCP session is pending and must be completed before proceeding further.

ACTION: Issue a call to RECV3270 before proceeding.

320 **MESSAGE: RU buffer too small.**

CAUSE: The RU size used by SNA IMF is too small to contain the data to be returned by the SNA link product.

ACTION: Note the circumstances and report them to your HP representative.

321 **MESSAGE: Invalid class name. (MPE V only)**

CAUSE: The *snaclassname* specified in the *snalnkinfo* parameter of the OPEN3270 intrinsic is invalid for this node.

ACTION: Ask the network manager to check the SNA configuration and provide you with a valid SNA LU class.

321 **MESSAGE: Invalid LU name. (MPE XL only)**

CAUSE: The *securityclass* specified in the *snalnkinfo* parameter of the OPEN3270 intrinsic is invalid for this node.

ACTION: Ask the network manager to check the SNA IMF configuration and provide you with a valid SNA IMF security class name.

322 **MESSAGE: Invalid session type.**

CAUSE: An invalid session type parameter was passed to the SNA link product.

ACTION: Note the circumstances and report them to your HP representative.

323 **MESSAGE: Negative LU-SSCP response.**

CAUSE: A negative response occurred on the LU-SSCP session.

ACTION: Make sure the host application that you are trying to access is operational. If it is, note the circumstances and contact your HP representative.

324 **MESSAGE: Invalid configuration access.**

CAUSE: The node manager (NMMGR) configuration file name (NMCONFIG) is incorrect, or the file is corrupt or missing.

ACTION: Check the node manager configuration file (NMCONFIG); it should not contain errors.

325 **MESSAGE: Request pending.**

CAUSE: A request is already pending.

ACTION: Note the circumstances and report them to your HP representative.

326 **MESSAGE: No available NAU.**

CAUSE: The SNA link product determined that there were no free NAUs in the specified SNA class name.

ACTION: Ask the system manager to check the SNA configuration for other valid *snaclassname* values you can use.

327 **MESSAGE: I/O pending.**

CAUSE: A previous I/O request has not been completed.

ACTION: Note the circumstances and report them to your HP representative.

328 **MESSAGE: Invalid function code.**

CAUSE: SNA IMF passed the SNA link product an invalid function code parameter.

ACTION: Note the circumstances and report them to your HP representative.

329 **MESSAGE: Inactive node or invalid node name.**

CAUSE: The SNA link product detected either an inactive SNA node or an invalid SNA nodename.

ACTION: Check the SNA nodename you requested. Either it is incorrect or you must start the node by using the SNACONTROL START command.

330 **MESSAGE: Illegal call.**

CAUSE: SNA IMF and the SNA link product are not synchronized with respect to the type of session established.

ACTION: Note the circumstances and report them to your HP representative.

331 **MESSAGE: Invalid error code.**

CAUSE: SNA IMF passed SNAErrMsg an invalid error code parameter.

ACTION: Note the circumstances and report them to your HP representative.

332 **MESSAGE: Privilege mode required.**

CAUSE: An SNA link intrinsic was called which required Privilege Mode (PM) capability.

ACTION: Note the circumstances and report them to your HP representative.

333 **MESSAGE: Invalid InfoWanted parameter.**

CAUSE: SNA IMF passed SNASessInfo an invalid InfoWanted parameter.

ACTION: Note the circumstances and report them to your HP representative.

335 **MESSAGE: No request pending.**

CAUSE: There are no requests pending.

ACTION: None.

336 **MESSAGE: Link shutdown occurred.**

CAUSE: The SNA node has been shutdown.

ACTION: Call the CLOSE3270 intrinsic.

337 **MESSAGE: Protocol shutdown requested.**

CAUSE: The SNA link product is processing a protocol shutdown. The SNA node is being shutdown.

ACTION: Finish current processing, then call the CLOSE3270 intrinsic.

338 **MESSAGE: Quiesce shutdown requested.**

CAUSE: The SNA link product has requested line quiesce. The SNA node is being restrained from starting new jobs. As the current jobs finish, the SNA system gradually winds down until jobs are no longer running.

ACTION: Call the CLOSE3270 intrinsic.

339 **MESSAGE: Invalid parameters. (MPE V only)**

CAUSE: The message catalog CATSNA.PUB.SYS is invalid.

ACTION: Inform your system manager

339 **MESSAGE: CATSNA.PUB.SYS access error. (MPE XL only)**

CAUSE: The SNA Transport message catalog file (CATSNA.PUB.SYS) is missing, invalid, or locked by another process.

ACTION: Inform your system manager.

340 **MESSAGE: Out of stack space.**

CAUSE: The SNA link product could not process a request made by the SNA IMF Pass Thru process due to insufficient stack space.

ACTION: Try running or preparing your program again, but specify a larger stack size in the *stack* parameter of the PREP or RUN command.

348 **MESSAGE: Invalid data offset.**

CAUSE: SNA IMF passed the SNA link product an invalid data offset parameter. It was probably negative or longer than the length of the message.

ACTION: Note the circumstances and report them to your HP representative.

351 **MESSAGE: Link Failure occurred.**

CAUSE: The SNA link product detected a link failure. The link for the SNA node has gone down.

ACTION: Call the CLOSE3270 intrinsic.

352 **MESSAGE: Transport Internal Error Shutdown.**

CAUSE: The SNA link product is processing an internal shutdown. The SNA nodes are being shutdown.

ACTION: Call the CLOSE3270 intrinsic.

353 **MESSAGE: Hierarchical Shutdown. (MPE V only)**

CAUSE: The SNA link product is processing a hierarchical shutdown (the host has deactivated your session). The LU is being shutdown.

ACTION: Call the CLOSE3270 intrinsic.

355 **MESSAGE: Bad Maxinfo length parameter.**

CAUSE: SNA IMF passed the SNA link product an invalid *MaxInfoLength* parameter.

ACTION: Note the circumstances and report them to your HP representative.

358 **MESSAGE: Error msg truncated because buffer was too small. (MPE XL only)**

CAUSE: A buffer too small to hold the error message from the SNA Transport message catalog file was passed in a call to the HPSNAErrmsg or SNAErrmsg intrinsic. The message was truncated to fit into the buffer that was passed.

ACTION: Note the circumstances and report them to your HP representative.

400 **MESSAGE: Expedited response pending.**

CAUSE: An expedited response is pending and must be completed before proceeding.

ACTION: Note the circumstances and report them to your HP representative.

401 **MESSAGE: Data traffic inactive.**

CAUSE: The host has not sent the necessary command sequence to activate your session.

ACTION: Check with your system administrator or IBM operator to make sure the NAUs on the HP 3000 have been properly activated, then try again.

402 **MESSAGE: SDT request not received.**

CAUSE: The host has not sent the necessary command sequence to activate your session.

ACTION: Check with your system administrator or IBM operator to make sure the NAUs on the HP 3000 have been properly activated, then try again.

403 **MESSAGE: Invalid session control protocol.**

CAUSE: An internal error has occurred in the SNA link subsystem, or an illegal request was received from the host.

ACTION: Note the circumstances and report them to your HP representative.

404 **MESSAGE: RQR request pending.**

CAUSE: SNA IMF has requested recovery for the session.

ACTION: Note the circumstances and contact your HP representative.

405 **MESSAGE: STSN request not pending.**

CAUSE: There is no STSN pending.

ACTION: Note the circumstances and report them to your HP representative.

407 **MESSAGE: Unsupported CRV request/response.**

CAUSE: The primary requested an unsupported function. The SNA link product does not support Cryptography Verification (CRV).

ACTION: Change the host application's BIND to turn off cryptography.

408 **MESSAGE: Unsupported session control request.**

CAUSE: An unsupported session control request was attempted.

ACTION: Note the circumstances and contact your HP representative.

409 **MESSAGE: Unsupported session control response.**

CAUSE: An unsupported session control response was detected by the SNA link product.

ACTION: Note the circumstances and contact your HP representative.

410 **MESSAGE: No such deactivation request (UNBIND, DACTLU, DACTPU) received.**

CAUSE: An SNA service is trying to send a response to a deactivation request (UNBIND, DACTLU, DACTPU), but no such deactivation request has been received.

ACTION: Note the circumstances and contact your HP representative.

411 **MESSAGE: No such activation request (BIND, ACTLU, ACTPU) received.**

CAUSE: An SNA service is trying to send a response to an activation request (BIND, ACTLU, ACTPU), but no such activation request has been received.

ACTION: Ensure that the NAU has been activated on the host.

412 **MESSAGE: CLEAR request not received. (MPE V only)**

CAUSE: The primary session control has not sent a Clear.

ACTION: Note the circumstances and contact your HP representative.

412 **MESSAGE: The target parameter passed to iowait is too small. (MPE XL only)**

CAUSE: Internal error in SNA Transport's IOWAIT intrinsic.

ACTION: Contact your HP representative.

413 **MESSAGE: An error occurred during a switch from CM to NM. (MPE XL only)**

CAUSE: When changing from compatibility mode (CM) to native mode (NM), an error occurred when an SNA compatibility mode intrinsic was called.

ACTION: Note the circumstances and contact your HP representative.

415 **MESSAGE: Couldn't send BIND on a Secondary LU. (MPE XL only)**

CAUSE: Internal SNA Transport error.

ACTION: Contact your HP representative.

416 **MESSAGE: A waited send intrinsic call was aborted by SNA Transport to avoid a deadlock. (MPE XL only)**

CAUSE: Internal SNA Transport error.

ACTION: Contact your HP representative.

B SNA Character String (SCS) Support

SNA IMF supports **SNA Character String** (SCS) control codes, allowing programs on the HP 3000 to emulate LU.T1 (SCS mode) printers. SNA IMF also supports the 3270 data stream, so HP programs can also emulate LU.T3 (3270 mode) printers. Only one mode can be active at a time.

If your program is emulating an LU.T1 printer, it must be in transparent mode. The data stream, with SCS codes, is passed untranslated into the internal screen image. Your program is responsible for interpreting the SCS codes.

The SCS data stream has fifty-six control codes. The IBM 3287 printer supports 18 of the SCS control codes, and HP printers support 17 of those 18 control codes. The Set Attribute (SA) SCS control code is the only code supported by the IBM 3287 that HP printers do not support. The SA control code is an extension of the SCS data stream and is considered an option by IBM. Also considered optional SCS extensions are the Structured Field and Attribute Processing (SFAP)

An LU.T1 session can support only a screen image of 3870 bytes and an RU size of 3584 bytes. If you exceed this RU limit, the screen image will be overwritten.

For a complete description of the 17 supported SCS codes, refer to the *IBM 3287 Printer Models 1 and 2 Component Description* (IBM P/N GA26-3153).

As with the IBM 3287 printer, you must physically align paper forms at the first line to be printed when you insert forms into the printer. This action will ensure correct feeding of the forms. Also, any change in format (SHF, SLD, SVF) must be followed by the appropriate synchronizing function (CR, NL, FF) in order to maintain format integrity.

Table B-1 lists the SCS codes supported by SNA IMF.

Table B-1 SCS Codes Supported by SNA IMF

SCS Codes	EBCDIC (hex)	SCS Name
CR	0D	Carriage Return
LF	25	Line Feed
NL	15	New Line
BS	16	Back Space
FF	0C	Form Feed
HT	05	Horizontal Tab
VT	0B	Vertical Tab
SHF	2BC1	Set Horizontal Format
SVF	2BC2	Set Vertical Format
SLD	2BC6	Set Line Density
VCS	04nn	Vertical Channel Select
TRN	35	Transparent
IRS	1E	Interchange Record Separator
BEL	2F	Bell
INP	24	Inhibit Presentation
ENP	14	Enable Presentation
GE	08nn	Graphics Escape

C 3270 Bit Assignment and Character Translation Tables

This Appendix contains the following tables related to 3270 data stream communication:

1. See Table C-1 for 3270 Write Control Character (WCC) bit assignment.
2. See Table C-2 for Attention ID (AID) codes generated by SNA IMF.
3. See Table C-3 for 3270 command codes for IBM control units.
4. See Table C-4 for bit assignments for the 3270 Field Attribute Character.
5. See Table C-5 3270 buffer control orders.

Information about ASCII and EBCDIC character translation tables and HP 3000 Native Language Support (NLS) is also provided at the end of this Appendix.

Table C-1 3270 Write Control Character Bis Assignment

Bits	Meaning
0	Value is determined by the contents of bits 2–7.
1 RESERVED	Always a 1.
2–3 Line length	Defines line length in printout as follows: 00 = 132-character line when orders are not present. The NL, EM, and CR orders in the data stream determine printed line length. 01 = Printed line is 40 characters. 10 = Printed line is 64 characters. 11 = Printed line is 80 characters.
4 Start printer	When set to 1, bit 4 starts printing at the completion of the write operation.
5 Alarm	When set to 1, bit 5 sounds an alarm at a selected output device as soon as an operation ends.
6 Keyboard enable	When set to 1, bit 6 reenables the keyboard of a selected device so that it will accept input.
7 MDT reset	When set to 1, bit 7 resets all modified data tag (MDT) bits in the data in the existing buffer of the selected device before writing any data or executing any orders.

See Table C-2 for Attention ID (AID) codes that are generated by SNA IMF.

Table C-2 Attention ID Codes Generated by SNA IMF

AID	Decimal Code	ASCII Code (Octal/Hex)	EVCDIC Code (Hexadecimal)	Graphic Character
ENTER	39	047/27	7D	‘ (Apostrophe)
PF 1	49	061/31	F1	1
PF 2	50	062/32	F2	2
PF 3	51	063/33	F3	3
PF 4	52	064/34	F4	4
PF 5	53	065/35	F5	5
PF 6	54	066/36	F6	6
PF 7	55	067/37	F7	7
PF 8	56	070/38	F8	8
PF 9	57	071/39	F9	9
PF 10	58	072/3A	7A	:
PF 11	35	043/23	7B	#
PF 12	64	100/40	7C	@
PF 13	65	101/41	C1	A
PF 14	66	102/42	C2	B
PF 15	67	103/43	C3	C
PF 16	68	104/44	C4	D
PF 17	69	105/45	C5	E
PF 18	70	106/46	C6	F
PF 19	71	107/47	C7	G
PF 20	72	110/48	C8	H
PF 21	73	111/49	C9	I
PF 22	91	133/5B	4A	[(ASCII) (cent sign) (EBCDIC)
PF 23	46	056/2E	4B	. (period)
PF 24	60	074/3C	4C	<
PA 1	37	045/25	6C	%
PA 2	62	076/3E	6E	>

Table C-2 Attention ID Codes Generated by SNA IMF

AID	Decimal Code	ASCII Code (Octal/Hex)	EVCDIC Code (Hexadecimal)	Graphic Character
PA 3	44	054/2C	6B	, (comma)
CLEAR	95	137/5F	6D	_ (underscore)
SYSTEM REQUEST	48	060/30	F0	0

See Table C-3 for the Command Codes for IBM Control Units.

Table C-3 Command Codes for IBM Control Units

Command	ASCII Code (Octal/Hex)	EBCDIC Code (Hexadecimal)	Graphic Character
Erase All Unprotected	77/3F	6F	?
Erase/Write	65/35	F5	5
Erase/Write Alternate	75/3D	7E	=
Read Buffer	62/32	F2	2
Read Modified	66/36	F6	6
Write	61/31	F1	1
Read Modified All	76/3E	6E	>

NOTE SNA IMF does not support the Write Structured Field command.

See Table C-4 for bit assignments for the 3270 Field Attribute Character.

Table C-4 3270 Field Attribute Character Bit Assignment

Bits	Field Description
0	Value is determined by the contents of bits 2–7.
1 RESERVED	Always a 1.
2 U/P	0 = Unprotected 1 = Protected
3 A/N	0 = Alphanumeric 1 = Numeric
4–5	Display indicator 00 = Normal display 01 = Normal display and light pen detectable 10 = Intensified display and light pen detectable 11 = Non-display and non-print
6 RESERVED	Set 0 0 by the host.
7 MDT	Modified data tag (MDT) indicates whether the field associated with this attribute character has been modified. 0 = Field has not been modified. 1 = Field has been modified either by the terminal operator or by the host program when it sent this attribute character to the terminal

NOTE When bits 2 and 3 are both on, an automatic skip occurs.

See Table C-5 for 3270 buffer control orders.

Table C-5 3270 Buffer Control Orders

Byte 1		Byte 2	Byte 3	Byte 4
Order	Order Code (EBCDIC Hex/ ASCII Hex)			
Start Field (SF)	1D/1D	Field attribute character		
Set Buffer Address (SBA)	11/11	Address (1st byte)	Address (2nd byte)	
Insert Cursor (IC)	13/13			
Program Tab (PT)	05/09			
Repeat to Address (RA)	3C/14	Address (1st byte)	Address (2nd byte)	Character to be repeated
Erase Unprotected to Address (EAU)	12/12	Address (1st byte)	Address (2nd byte)	

NOTE SNA IMF does not support the following 3270 buffer control orders:

- Start Field Extended
- Set Attribute
- Modify Field
- Graphic Escape

Character Sets

Every computer uses a standard or default character set. A character set is a collection of graphic and control characters. Each character is normally represented by a unique 7- or 8-bit code. The standard character set for a particular computer is used throughout that computer system. Terminals, printers, and communications controllers, as well as sort utilities, editors, compilers, and command interpreters, must agree on a standard code.

The standard code for the HP 3000 is the American Standard Code for Information Interchange (ASCII). IBM computers use the Extended Binary Coded Decimal Interchange Code (EBCDIC).

SNA IMF provides automatic translation between EBCDIC, the code used over the communications line to the host, and ASCII, the code used for the terminal screen and printer buffers on the HP 3000 side.

NLS system utilities are available that allow you to add languages to your system, or delete them, and to modify local formats. NLS tables are available for each of the foreign languages supported by the HP 3000. These NLS tables reside on the system. For more information about NLS, refer to the *Native Language Support Reference Manual*.

SNA IMF uses Native Language Support (NLS) translation tables to perform the ASCII-to-EBCDIC and EBCDIC-to-ASCII translations. NLS features allow the application programmer to create local language applications for end users. These features include architecture and peripheral support, as well as software facilities within the operating systems and subsystems. NLS addresses the internal functions of a program (for example, sorting) as well as the user interface (for example, message formats).

D Differences Between IMF/3000 and SNA IMF/V

If you are migrating from IMF/3000 to SNA IMF/V on an MPE V system, you may find the information in this Appendix helpful. IMF/3000 uses either BSC or SDLC protocol in a Node Type 1 environment. SNA IMF uses only SDLC protocol in a Node Type 2 environment.

NOTE If you are migrating from IMF/3000 to SNA IMF/V, the following IMF/3000 components are not required: INP download, IMF Monitor, Pseudo Driver, and IMF Manager.

Unlike IMF/3000, which is a complete subsystem that incorporates functions from the user level down to and including the INP, SNA IMF runs entirely on the user's stack. SNA IMF exists once for each user who invokes the SNA IMF intrinsics.

NOTE IMF/3000 is currently supported only on MPE V.

This Appendix compares and contrasts IMF/3000 with SNA IMF/V in these areas:

- Product structure
- Configuration file parameter
- Screen sizes
- Display screen ownership
- Intrinsics

Product Structure

Both IMF/3000 and SNA IMF perform the same basic function: they support interactive communication between an HP 3000 and an IBM host. Both products emulate the operation of a remote cluster controller with attached display stations and printers. However, the two products have different components that allow them to accomplish this common goal.

IMF/3000

IMF/3000 supports both Binary Synchronous Communications (BSC) and Synchronous Data Link Control (SDLC) line protocols. The SDLC protocol used by IMF/3000 supports a 3271 Model 11 Communications Controller, which uses a subset of SNA functions in a Node Type 1 environment. IMF/3000 includes functions from the user level all the way down to and including the INP. The major components of IMF/3000 are as follows:

- Pass Thru
- IMF Intrinsics
- Pseudo Driver
- IMF Monitor
- INP Driver

Pass Thru is an application program that uses IMF/3000 intrinsics to make HP terminals and printers attached to an HP 3000 appear to the IBM host as IBM 3278 display stations and IBM 3287 printers.

IMF/3000 intrinsics allow an application programmer to exchange screens of data with the host using 3270 screen images. Once the intrinsics obtain the necessary information from you, they use the Pseudo Driver to pass the data to the IMF Monitor for processing.

The Pseudo Driver allows IMF/3000 intrinsics and the IMF Manager programs, running on many different user stacks, to queue requests to the IMF Monitor for processing.

The IMF Monitor processes data that comes from either the Pseudo Driver or the INP Driver. Also, the IMF Monitor translates and formats 3270-type screens.

The INP Driver refers to the INP download file and the INP. The INP Driver carries on the actual conversation with the IBM host. The INP takes data from the IMF Monitor through CS (Communication Services) buffers, places the necessary Data Link/Path Control level information into the data, and sends the data to the host.

The IMF/3000 components are not discussed in detail in this manual. Refer to the *IMF/3000 User/Programmer Reference Manual* for more information about IMF/3000.

SNA IMF

SNA IMF uses HP terminals and printers to emulate IBM 3278 display stations and 3287 printers attached to an IBM remote control unit in an SNA Node Type 2 environment.

SNA IMF does not have an INP download, does not use Communication Services (although SNA Link/V does), does not have an IMF Monitor, and does not have a Pseudo Driver. Also, SNA IMF does not have an IMF Manager program because SNA IMF no longer controls the node. The node is managed by the physical unit (PU), which is controlled by the Node Management Services (NMS) software. In contrast, IMF/3000 starts and stops the node and selects the appropriate configuration file.

NOTE SNA IMF software includes the Pass Thru program and the IMF intrinsics. When both IMF/3000 and SNA IMF/V are running on the same HP 3000, they share the same Pass Thru program file (TTSSON.PUB.SYS) and IMF intrinsic SL segments.

Configuration File Parameter

The configuration file parameter (*snalnkinfo* for SNA IMF, or *confile* for IMF/3000) that you specify in the parameter list of the ACQUIRE3270 and OPEN3270 intrinsics has a different meaning depending on which IMF product you use. The parameter occupies the same position in the parameter list, and it is still a byte array, but its meaning differs between IMF/3000 and SNA IMF.

IMF/3000

For IMF/3000, the configuration file parameter represents the name of the configuration file used by the IMF/3000 user.

SNA IMF

For SNA IMF/V, the configuration file parameter is the name of an SNA node and an SNA LU class configured in the configuration file NMCONFIG.PUB.SYS. The node and class names identify the parts of the NMCONFIG configuration file that the SNA Link product uses for configuration.

Node Management Services, a part of the SNA Link/V product, creates, accesses, and manages the NMCONFIG configuration file.

For SNA IMF/XL, the configuration file parameter is the name of an SNA node and a security class listed in the NMCONFIG configuration file. Node Management Services manages the NMCONFIG configuration file. On MPE XL, Node Management Services is part of the Fundamental Operating System.

Screen Sizes

This section explains the differences between IMF/3000 screen sizes and SNA IMF screen sizes.

IMF/3000

Screen size is specified either by the IBM host or in the configuration file on the HP 3000. After Pass Thru is started, the screen size is assumed to be whatever was configured in the IMF/3000 configuration file.

SNA IMF

Like IMF/3000, SNA IMF allows the host to set the screen size. However, screen size cannot be specified on the HP 3000. As part of SNA protocol, the IBM host sends a `BIND` command at the beginning of a session. The `BIND` command, which is transparent to the user, establishes the screen size for SNA IMF. A screen must be 480, 1920, or 3440 characters; otherwise, SNA IMF rejects it.

Display Screen Ownership

When you use IMF/3000, you need not be aware of who owns the display screen at any particular time. With SNA IMF, however, knowing who owns the display screen at a particular time can help you determine the state of your session and which keys or commands you should enter.

IMF/3000

To use IMF/3000, you do not need to be aware of who owns the display screen.

SNA IMF

To use SNA IMF, you need to be aware of how screen ownership is handled.

A session is the logical connection between Network Addressable Units (NAUs). Three types of NAU sessions exist: SSCP-PU, SSCP-LU, and LU-LU. During session initiation, display screen ownership passes through three stages: (1) unowned; (2) owned by an SSCP-LU session; and (3) owned by the LU-LU session.

For example, if you are using the Time Sharing Option (TSO), you will receive four different types of screens before completing your logon to the host:

1. An unowned screen. After the first `RECV3270` intrinsic call, you receive an unowned screen that contains the SNA IMF banner. Send the **[SYS REQ]** key now. Sending the **[SYS REQ]** key affects screen ownership. Sending the **[SYS REQ]** key also clears the screen image and sets the cursor to the top of the screen.
2. A blank SSCP screen. This screen appears after the **[SYS REQ]** key is sent. The blank screen is produced internally by SNA IMF. Screen ownership switches to the SSCP-LU session. After receiving this screen, enter your logon to the host.
3. An LU-LU owned BIND screen. The host receives this screen after TSO determines that your LU will be permitted to start a TSO session.
4. An unowned screen. This screen is produced by SNA IMF whenever an UNBIND is received. TSO sends a BIND-UNBIND-BIND sequence when establishing a session. The unowned screen appears immediately after the SSCP-LU BIND screen and before your TSO session is created.

5. An LU-LU owned screen. This screen contains three items: logon messages from the host, an indication of whether logon was successful, and the READY message.

NOTE Sending the **[SYS REQ]** key while using a full screen-oriented application destroys the integrity of your screen image. Once this happens, the host application must recover the screen image.

Table D-1 lists the three states of display screen ownership, what each state means, and how you can change display screen ownership.

Table D-1 Display Screen Ownership

Display Screen Ownership	Display Screen Ownership	How to Change Screen Ownership
Unowned	You have either just opened a device for SNA IMF communication or just logged off the host.	Send the **[SYS REQ]** key to transfer screen ownership to the SSCP-LU session if you have just opened a device.
Owned by an SSCP-LU session	You have sent the [SYS REQ] key, which changes the screen ownership from unowned (the previous screen) to SSCP-LU ownership (the current screen). You can now log on to the system.	If there is no LU-LU session, send the **[SYS REQ]** key to make the screen unowned. If an LU-LU session exists, send the **[SYS REQ]** key to transfer screen ownership back to the LU-LU session.
Owned by an LU-LU session	An SSCP-LU session was the owner of the previous screen. Logon was completed successfully.	Either send the **[SYS REQ]** key to transfer screen ownership to the SSCP-LU session or log off to make the screen unowned.

The display screen is unowned when a device is first opened for SNA IMF communication.

The SSCP-LU session enables you to create an LU-LU session, which allows communication between two end users (application programs, devices, or people). Once you enter the logon message followed by the **[SYS REQ]** key, the newly created LU-LU session allows you to exchange data with the IBM host through 3270-type screen images.

When a terminal receives a BIND from the host, the terminal's screen is put into an LU-LU session. Receipt of an UNBIND from the host puts the screen into an unowned state.

Intrinsics

The meanings of some parameters of the OPEN3270 and ACQUIRE3270 intrinsics vary depending on the IMF product you use.

IMF/3000

The syntax of the IMF/3000 ACQUIRE3270 and OPEN3270 intrinsics is as follows:

```
ACQUIRE3270     (confile, devicenum, ldev, enhance, priority,
                blanks, format, flags, result)

OPEN3270        (devicenum, confile, flags, terminalid, devtype,
                ffindex, screensize, timeout, result)
```

The *devicenum* parameter specifies a specific device number on the IBM 3271 Cluster Controller that IMF/3000 emulates. This device number, which must be an integer from 0 through 31, must correspond to an equivalent specification in the IMF/3000 configuration file (*confile*).

The *confile* parameter indicates the name of the configuration file IMF/3000 uses.

SNA IMF

The syntax of the SNA IMF ACQUIRE3270 and OPEN3270 intrinsics is as follows:

```
ACQUIRE3270     (snalnkinfo, devicenum, ldev, enhance, priority,
                blanks, format, flags, result)

OPEN3270        (devicenum, snalnkinfo, flags, terminalid, devtype,
                ffindex, screensize, timeout, result)
```

The SNA IMF *snalnkinfo* parameter replaces the IMF/3000 *confile* parameter. SNA IMF does not have an IMF configuration file; instead, SNA IMF references an LU class and an SNA node name. You must configure both the LU class and the node name in the Node Management configuration file that was specified in the SNACONTROL START command used to bring up SNA IMF.

The SNA IMF *devicenum* parameter specifies a device type: -2 for LU.T2 terminal emulation, -1 for LU.T1 printer emulation, or -3 for LU.T3 printer emulation. The LU class (specified in the *snalnkinfo* parameter) is a group of available LUs that may be used to emulate a device. For example, you can configure an LU class for IBM 3278 display stations and an LU class for IBM 3287 printers.

E HP and IBM Differences in DBCS Implementation

This Appendix describes how HP and IBM differ in their implementations of Asian Double-Byte Character Sets (DBCS).

Differences in DBCS implementation that occur during Pass Thru sessions are not described in this appendix. For information on IBM and HP DBCS differences during Pass Thru, see Appendix G of *Using SNA IMF Pass Thru*.

This appendix contains the following sections:

- Control character mapping
- Undefined DBCS characters
- User-defined DBCS characters

Control Character Mapping

Double-byte printer control codes in the LU.T2 data stream will be displayed as two single-byte characters, each character based on eight of the 16 bits.

Undefined DBCS Characters

Some infrequently used HP DBCS characters are not translated to IBM DBCS equivalents because they do not exist in the IBM character set. Likewise, some IBM DBCS characters are not translated to HP DBCS characters, because they do not exist in the HP character set. The HP and IBM DBCS character sets differ slightly. Currently, the HP NLS conversion intrinsic will map an undefined character to either a user-configurable character (hexadecimal “40FE” to hexadecimal “FFFE”) or the default character (a double-byte blank). See the *Native Language Support Reference Manual* for more information.

Table E-1 lists the default mapping characters from the HP-IBM and IBM-HP tables for Japanese, Korean, and Traditional Chinese.

Table E-1 Default Mapping of Undefined Characters

Language	IBM-to-HP	HP-to-IBM
Japanese	HP-15 blank (hex “8140”)	DBCS blank (hex “4040”)
Korean	HP-15 blank (hex “A1A1”)	DBCS blank (hex “4040”)
Traditional Chinese	HP-15 blank (hex “F5D1”)	DBCS blank (hex “4040”)

User-Defined DBCS Characters

User-defined characters on the HP 3000 will be supported if the translation tables are correctly updated. Your HP 3000 system manager can modify the ASCII-to-EBCDIC translation tables by running the NLS utility program `LANGINST.PUB.SYS`. To make the changes, the system manager should run `LANGINST.PUB.SYS` and select option 3 (Modify Native Format). See Appendix A of the *Native Language Support Reference Manual* for detailed information.

User-defined characters on the Asian IBM host should be handled by the IBM system programmer.

F Sample Programs

These sample Pascal programs demonstrate how to declare, define, and call SNA IMF intrinsics in non-transparent mode and transparent mode.

NOTE These programs use the *options* and *pfn* parameters of the ACQUIRE3270 intrinsic, which are supported only for SNA IMF/XL. If you are running SNA IMF/V, your programs cannot use the *options* and parameters.

NOTE Calling sequences might differ from one system to another depending on how the host is configured. For example, some systems are configured so that, after receiving the initial unowned screen, the host immediately sends another screen without waiting for a system request from the user. These sample programs assume that the host waits for a system request from the user before sending the next screen.

Sample Program in Non-Transparent Mode

When this program is given an input *parm* value of zero, it emulates a display station (LU Type 2) powering on, logging onto the host, logging off, and powering off. When it is given a *parm* value greater than zero, it starts a Pass Thru session with a spooler file (LU Type 3).

```
program imf3270 (input, output, parm);


const
   SCREENSIZE    = 1920;
   LINESIZE      = 80;


{ constant for OPEN3270 }
   TERMINAL      = -2;


{ constants for action code of PRINT3270 }
   OPEN_FILE     = 0;
   PRINT_SCREEN  = 2;
   PRINT_BANNER  = 3;
   CLOSE_FILE    = 4;


{ constants for AID of TRAN3270 }
   SYS_REQ_KEY   = 48;
   ENTER_KEY     = 39;


{ constants for ACQUIRE3270 }
   LU_T3         = -3;
   SPOOLER_FILE  = 6;


type
   shortint      = -32768..32767;     { 16 bits = 2 bytes }
   string        = packed array[1..LINESIZE] of char;
   screen        = packed array[1..SCREENSIZE] of char;
{ type for OPEN3270: }
```

```
{ This type takes up 2 bytes.  It can be replaced by the shortint type. }
{ This type is defined for ease of assigning values to each }
{ of the different bit groups }
   flags_type = packed record
      filler     : 0..1023;  { ten bits }
      dbcs       : 0..1;     { one bit }
      unbind     : 0..1;     { one bit }
      LUT1_LUT3  : 0..1;     { one bit }
      int_trace  : 0..1;     { one bit }
      trans_mode : 0..1;     { one bit }
      IO_mode    : 0..1;     { one bit }
   end;                      { total of 16 bits = 2 bytes }

{ type for ACQUIRE3270: }
{ This type takes up two bytes.  It can be replaced by the shortint type. }
{ This type is defined for ease of assigning values to each }
{ of the different bit groups }
{ Note that the readTO field takes up nine bits, but the legal }
{ range of values is 10-255, which only takes up 8 bits. }
{ So, the range is set at 10-256 to take up nine bits, }
{ but a value of 256 or higher is not allowed. }

   options_type = packed record
      filler     : 0..1;     { one bit }
      int_trace  : 0..1;     { one bit }
      sPriority  : 1..13;    { four bits }
      readTO     : 10..256;  { nine bits.  256 is illegal value. }
      LJ2        : 0..1;     { one bit }
   end;                      { total of 16 bits = 2 bytes }
Global variables: }
{ These variables are global because they must be used in a number of }
{ procedures.  Some of them are used to actually pass parameters }
{ to other procedures.  Making them global simplifies the code }
{ and makes intuitive sense. }
```

```
var
   terminalid     : shortint;  { terminalid used for intrinsics }
   result         : shortint;  { result code of intrinsic calls }
   fileid         : shortint;  { ID of file printed to }
   action         : shortint;  { action code for PRINT3270 }
   priority       : shortint;  { priority given to spooler file }
   cursorrow      : shortint;  { current cursor row position }
   cursorcolumn   : shortint;  { current cursor column position }
   numfields      : shortint;  { number of fields in the current screen }
   error          : boolean;   { global error flag }
   location       : string;    { location string used by PRINT3270 }
   parm           : shortint;  { parameter specified in command line }
procedure VERS3270;     intrinsic;
procedure ERR3270;      intrinsic;
procedure ACQUIRE3270;  intrinsic;
procedure ATTRLIST;     intrinsic;
procedure CLOSE3270;    intrinsic;
procedure FIELDATTR;    intrinsic;
procedure OPEN3270;     intrinsic;
procedure PRINT3270;    intrinsic;
procedure READSCREEN;   intrinsic;
procedure RECV3270;     intrinsic;
procedure SCREENATTR;   intrinsic;
procedure TRAN3270;     intrinsic;
procedure WRITEFIELD;   intrinsic;
procedure READFIELD;    intrinsic;


{
 * The following procedure takes an errorcode, which is usually a result code
 * from another intrinsic call, and prints out the corresponding message.
 * The conversion is done by ERR3270.  ERR3270 takes an errorcode, fills
msgbuf
 * with the corresponding message, assigns msglen to the message length,
 * and sets result to the result code.
```

```
}

procedure print_message (errorcode: shortint);
var
   msgbuf  : packed array[1..144] of char;
   msglen  : shortint;

begin

{ Set flag if any fatal errors have occurred. }
{ errorcode=0 (and errorcode=9 on MPE V) are not errors. }

   if errorcode <<>> 0 then
      error := TRUE;
   msgbuf := ' '
   ERR3270 (errorcode, msgbuf, msglen, result);

{ Check whether the ERR3270 intrinsic generated any errors. }
   if result = 0 then
      writeln (msgbuf:msglen)
   else begin
      writeln ('INTERNAL ERROR in program:  ERR3270 result = ', result:2);
      error := TRUE;
   end  { if - else }
end;

{
 * The following procedure calls RECV3270 to receive and print a screen.
 * The variables terminalid, fileid, action, location, and priority are
 * global.  terminalid is set by OPEN3270.  fileid is set by the first
 * PRINT3270 call, which is the call that opens the print file.  The variables
 * action, location, and priority are set in procedure initialize.
 * Note that the priority variable isn't checked after the first call
 * to PRINT3270, which opened the print file.
```

```
}
procedure call_recv3270;
begin
   write (' RECV3270.........');
   RECV3270 (terminalid, result);
   print_message (result);
   write (' PRINT3270 print to file.....')
   PRINT3270 (terminalid, fileid, action, location, priority, result);
   print_message (result);
end;

 * The following procedure prints the internal screen image to the spooler
 * file before it calls TRAN3270.  The variables terminalid, fileid, action,
 * location, priority, cursorrow, and cursorcolumn are global.
 * terminalid is set by OPEN3270.  fileid is set by the first PRINT3270 call,
 * which is the call that opens the print file.  The variables
 * action, location, and priority are set in procedure initialize.
 * Note that priority is not checked after the first call to PRINT3270,
 * the call that opened the print file.
 * cursorrow and cursorcolumn are set by the previous call to SCREENATTR.
 * Usually, call_recv3270 follows shortly after a call to this procedure,
 * because RECV3270 is usually the first intrinsic called after TRAN3270.
}
procedure call_tran3270 (aid:shortint);
begin
   write (' PRINT3270 print to file.....');
   PRINT3270 (terminalid, fileid, action, location, priority, result);
   print_message (result);
   write (' TRAN3270..................');
   TRAN3270 (terminalid, aid, cursorrow, cursorcolumn, result);
   print_message (result);
end;
```

```
{
 * The following procedure writes a string to the specified field
 * in the internal screen.  The variable terminalid is global and
 * is set by OPEN3270.
}
procedure call_writefield (outbuf:string; outbuflen, fieldnum:shortint);
var
   offset : shortint;

begin
   offset := 0;
   write (' WRITEFIELD.............');
   WRITEFIELD (terminalid, fieldnum, offset, outbuf, outbuflen, result);
   print_message (result);
end;

{
 * The following procedure calls READSCREEN to read the internal screen
 * image and output it to standard output.  The variable terminalid
 * is global and is set by OPEN3270.
}
procedure call_readscreen;
var
   offset       : shortint;
   maxinbuflen  : shortint;
   actinbuflen  : shortint;
   inbuf        : screen;

begin
   offset := 0;
   maxinbuflen := SCREENSIZE;
   inbuf := ' ';
   write (' READSCREEN.............');
   READSCREEN (terminalid, offset, maxinbuflen, inbuf, actinbuflen, result);
```

```
   print_message (result);
   writeln ('                          the screen read is shown below:')
   writeln (inbuf:actinbuflen);
end;


 * The following procedure calls ATTRLIST to find all the attribute
characters
 * in the internal screen and their positions (thereby locating all the
fields).
 * ATTRLIST sets the variable actlistlen, which then indicates the number of
 * fields in the screen.  The procedure then calls SCREENATTR to find the
 * number of fields, the print format, the current cursor position, and other
 * screen information.  SCREENATTR returns the number of fields through
 * the numfields parameter, which is global and is used outside of
 * this procedure after SCREENATTR sets it.  SCREENATTR's numfields parameter
 * and ATTRLIST's actlistlen parameter should contain the same value,
 * since they both represent the number of fields in the screen.
 * This procedure also calls FIELDATTR once for each field, to get
 * information about each of the fields.  Finally, it calls the
call_readscreen
 * procedure to output the screen image.  The variable terminalid
 * is global and is set by OPEN3270.
}
procedure check_screen;
var
{ ATTRLIST parameters }
   offset        : shortint;
   subscreensize : shortint;
   maxlistlen    : shortint;
   offsetlist    : array[1..20] of shortint;
   actlistlen    : shortint;
   fieldnum      : shortint;

{ SCREENATTR parameters }
   printformat   : shortint;
```

```
   startprint    : shortint;
   soundalarm    : shortint;
   keyboardlock  : shortint;
   screenstatus  : shortint;

{ FIELDATTR parameters }
   protectedattr   : shortint;
   currentfieldlen : shortint;
   fieldrow        : shortint;
   fieldcolumn     : shortint;
   numericattr     : shortint;
   displayattr     : shortint;
   mdt             : shortint;
   maxfieldlen     : shortint;
begin
   writeln;
   writeln ('Checking screen.....');
   offset := 0;
   subscreensize := SCREENSIZE;
   maxlistlen := 20;

   write (' ATTRLIST.............');
   ATTRLIST (terminalid, offset, subscreensize, maxlistlen, fieldnum,
             offsetlist, actlistlen, result);
   print_message (result);

   if actlistlen = 0 then
      writeln ('                 no attribute characters');
   else
      for fieldnum := 1 to actlistlen do
         writeln ('              attribute character #', fieldnum:1,
                  ' position = ', offsetlist[fieldnum]:2);

   write (' SCREENATTR.............');
```

```
   SCREENATTR (terminalid, printformat, startprint, soundalarm,
               keyboardlock, numfields, screenstatus, cursorrow,
               cursorcolumn, result);
   print_message (result);
   writeln ('              printformat = ', printformat:2,
            '                numfields = ', numfields:2);
   writeln ('                cursorrow = ', cursorrow:2,
            '             cursorcolumn = ', cursorcolumn:2);

   for fieldnum := 1 to numfields do
   begin
     write (' FIELDATTR ', fieldnum:1, '.............');
     FIELDATTR (terminalid, fieldnum, fieldrow, fieldcolumn,
                protectedattr, numericattr, displayattr, mdt,
                currentfieldlen, maxfieldlen, result);
      print_message (result);
      writeln ('        protectedattr = ', protectedattr:2,
               '             fieldlen = ', currentfieldlen:2);
   end;

   call_readscreen;
end;

{
 * The following function uses READFIELD to check all the fields of
 * the current screen for the string "READY".  This function is used
 * to tell when the host is finished sending screens.  It calls SCREENATTR
 * to get the number of fields in the screen.  SCREENATTR returns this
 * value through the parameter numfields.  The variable terminalid is
 * global and is set by OPEN3270.
}
function ready : boolean;
var
   returnval      : boolean;
```

```
{ SCREENATTR parameters }
   printformat    : shortint;
   startprint     : shortint;
   soundalarm     : shortint;
   keyboardlock   : shortint;
   screenstatus   : shortint;

{ READFIELD parameters }
   fieldnum       : shortint;
   maxinbuflen    : shortint;
   actinbuflen    : shortint;
   inbuf          : string;
   offset         : shortint;

begin
   returnval := FALSE;

   write (' SCREENATTR.............');
   SCREENATTR (terminalid, printformat, startprint, soundalarm,
               keyboardlock, numfields, screenstatus, cursorrow,
               cursorcolumn, result);
   print_message (result);
   writeln ('               printformat = ', printformat:2,
            '                 numfields = ', numfields:2);
   writeln ('                 cursorrow = ', cursorrow:2,
            '              cursorcolumn = ', cursorcolumn:2);
offset := 0;
   maxinbuflen := LINESIZE;

   for fieldnum := 1 to numfields do
   begin
      inbuf := ' ';
      write (' READFIELD ', fieldnum:1, '..............');
```

```
      READFIELD (terminalid, fieldnum, offset, maxinbuflen, inbuf,
                 actinbuflen, result);
      print_message (result);
      writeln ('        inbuf = "', inbuf:actinbuflen, '"');

      if inbuf = 'READY' then
         returnval := TRUE
   end;

   ready := returnval;
end;

{
 * The following procedure calls OPEN3270, simulating powering on an
 * IBM display station.  OPEN3270 assigns a value to the global variable
 * terminalid.  This value is used to reference the device in all subsequent
 * SNA IMF intrinsic calls.  This procedure also uses PRINT3270 to
 * open a spooler file, and it prints the attribute characteristic banner
 * to the file.  Note that RECV3270 must be called after a call to OPEN3270
 * to receive the unowned screen.
}
procedure initialize;
var

{ VERS3270 parameter}
   version         : string;

{ OPEN3270 parameters }
   devicenum    : shortint;
   snalnkinfo   : string;
   flags        : flags_type;
   devtype      : shortint;
   ffindex      : shortint;   { not used by SNA IMF. Included }
                              { for backwards compatibility. }
```

```
   screensize   : shortint;
   timeout      : packed array[1..2] of shortint;

begin
   writeln (' VERS3270.............');
   VERS3270 (version);
   writeln ('                       version = ', version:14);
   devicenum := TERMINAL;
   snalnkinfo := 'IBMNODE#IMFCLASS ';  { non-alphanumeric character }
                                       { marks end of string }
   with flags do
   begin
      filler := 0;
      dbcs := 0;       { disable double byte character set option }
      unbind := 0;     { disable unbind option }
      LU1_LU3 := 0;    { not applicable to terminal emulation. Set to 0. }
      int_trace := 0;  { internal tracing off }
      trans_mode := 0; { transparent mode off }
      IO_mode := 0;    { standard I/O mode }
   end;
   ffindex := 0;
   timeout[1] := 30;
   timeout[2] := 30;

   writeln ('Now opening LU.T2 session.....');
   write (' OPEN3270.....................');
   OPEN3270 (devicenum, snalnkinfo, flags, terminalid, devtype,
             ffindex, screensize, timeout, result);
   print_message (result);

   action := OPEN_FILE;   { set action to open spooler file }
   location := ' ';
   priority := 6;
   write (' PRINT3270 open..............');
```

```
   PRINT3270 (terminalid, fileid, action, location, priority, result);
   print_message (result);

   action := PRINT_BANNER;  { set action to print attribute character banner }
   write (' PRINT3270 print banner.......');
   PRINT3270 (terminalid, fileid, action, location, priority, result);
   print_message (result);
action := PRINT_SCREEN;  { set action to print internal screen image }
                            { for the rest of the PRINT3270s }
   writeln;
   writeln ('Receiving unowned screen.....');
   call_recv3270;
   check_screen;

   error := FALSE;
end;

{
 * The following procedure makes the major procedure calls.
 * It does the following:
 *    1. Sends a system request to the host and receives the LU-SSCP screen.
 *    2. Logs onto the host (starting an LU-LU session).  Note that
 *       after receiving the logon message, the host may send more than one
 *       screen before it is ready to receive again.
 *    3. Logs off the host.
 * Note that a call_recv3270 follows shortly after each call_tran3270.
}
procedure process;
var
   outbuf     : string;
   outbuflen  : shortint;
   fieldnum   : shortint;

begin
```

```
{ Transmit System Request Key and receive LU-SSCP screen. }
   writeln;
   writeln ('Transmitting System Request Key and receiving LU-SSCP screen');
   call_tran3270 (SYS_REQ_KEY);
   call_recv3270;
   check_screen;

{ Write logon message to LU-SSCP screen. }
   writeln;
   writeln ('Writing "logon applid...." to field 0');
   outbuf := 'logon applid(tso) data(sales/sales) logmode(imf2k)';
   outbuflen := 50;
   fieldnum := 0;
   call_writefield (outbuf, outbuflen, fieldnum);

   call_readscreen;

{ Transmit ENTER key to send logon message to host. }
   writeln;
   writeln ('Transmitting ENTER key.....');
   call_tran3270 (ENTER_KEY);



{ The host may send more than one screen here, so we have to make sure }
{ the host is finished sending before we try to send.  Function ready }
{ checks to see if the host is finished by searching for the string "READY" }
{ in the screens received from the host.  We could just as easily have }
{ waited for a RECV3270 to time out to tell us when the host was finished }
{ sending and ready to receive, but searching for "READY" is more efficient.
}

   repeat
      writeln;
      writeln ('Receiving screen and checking for READY or error.....');
```

```
      call_recv3270;
   until error or ready;

   check_screen;

{ Write logoff to unprotected field. }
   outbuf := 'logoff';
   outbuflen := 6;

   fieldnum := numfields - 1;  { Write to second-to-last field }
                               { because it's the unprotected field. }
   if fieldnum << 0 then fieldnum := 0;
   writeln;
   writeln ('Writing "logoff" to field ', fieldnum:2, '.....');
   call_writefield (outbuf, outbuflen, fieldnum);

{ Transmit ENTER key to send logoff message to host. }
   writeln;
   writeln ('Transmitting ENTER key.....');
   call_tran3270 (ENTER_KEY);
   call_recv3270;
end;

{
 * The following procedure does some cleanup.  It closes the spooler file,
 * and it calls CLOSE3270, which simulates powering off the device.
}
procedure terminate;
begin
   action := CLOSE_FILE;
   write (' PRINT3270 close.........');
   PRINT3270 (terminalid, fileid, action, location, priority, result);
   print_message (result);
```

```
   write (' CLOSE3270.....');
   CLOSE3270 (terminalid, result);
   print_message (result);
end;

{
 * The following procedure calls ACQUIRE3270, starting a Pass Thru session
 * on an LU.T3 printer.  The printer must be free from MPE control when
 * ACQUIRE3270 is called, or an error will occur.
}
procedure call_acquire;
var
   snalnkinfo  : string;
   devicenum   : shortint;
   ldev        : shortint;
   enhance     : shortint;
   priority    : shortint;
   blanks      : shortint;
   format      : shortint;
   flags       : shortint;
   options     : options_type;
   pfn         : string;

begin
   snalnkinfo := 'IBMNODE#IMFCLASS ';  { non-alphanumeric character }
                                       { marks end of string }
   devicenum := LU_T3;
   ldev := SPOOLER_FILE;
   enhance := 0;
   priority := 6;
   blanks := 0;    { convert leading blanks to nulls }
   format := 2;    { print screen as it appears on terminal }
   flags := 2;     { continue execution after Pass Thru is activated }
```

```
   with options do
   begin
      filler := 0;
      int_trace := 1;   { internal trace on }
      sPriority := 7;   { spooler file priority = 7 }
      readTO := 15;     { terminal timeout, not used in this case }
      LJ2 := 1;         { LaserJet II option is on }
   end;

   pfn := 'example ';   { non-alphanumeric character marks end of string }

   writeln ('Now starting Pass Thru session with spooler file.....');
   write (' ACQUIRE3270........');
   ACQUIRE3270 (snalnkinfo, devicenum, ldev, enhance, priority,
                blanks, format, flags, options, pfn, result);
   print_message (result);

end;




begin  { main }
   if parm >> 0 then
      call_acquire
   else begin
      initialize;
      process;
      terminate;
   end;
end.
```

Sample Program in Transparent Mode

This program emulates a display station (LU Type 2) powering on, logging onto the host, logging off, and powering off. Note that, in transparent mode, although there is still an internal screen image, intrinsics that access the fields in the internal screen image (such as READFIELD and WRITEFIELD) are not allowed. The internal screen image is used to store the untranslated data stream received from or waiting to be sent to the host.

```
program imftran (input, output);


const
   SCREENSIZE    = 2160;  { in transparent mode, a 1920-character }
                          { screen has a buffer limit of 2160 bytes. }
   LINESIZE       = 80;
   EBCDICtoASCII  = 1;
   ASCIItoEBCDIC  = 2;


{ constant for OPEN3270 }
   TERMINAL       = -2;


{ constants for AID of TRAN3270 }
   SYS_REQ_KEY    = 48;
   ENTER_KEY      = 39;
   CLEAR_KEY      = 95;


type
   shortint       = -32768..32767;
   string        = packed array[1..LINESIZE] of char;
   screen        = packed array[1..SCREENSIZE] of char;
                 { type for OPEN3270: }
                 { This type takes up 2 bytes.  It can be replaced by the
                 shortint type. }
                 { This type is defined for ease of assigning values to each }
                 { of the different bit groups }
                    flags_type = packed record
```

```
      filler     : 0..1023;  { ten bits }
      dbcs       : 0..1;     { one bit }
      unbind     : 0..1;     { one bit }
      LUT1_LUT3  : 0..1;     { one bit }
      int_trace  : 0..1;     { one bit }
      trans_mode : 0..1;     { one bit }
      IO_mode    : 0..1;     { one bit }
   end;                      { total of 16 bits = 2 bytes }
{ Global variables: }
{ These variables are global because they must be used in a
number of }
{ procedures.  Some of them are used to actually pass
parameters }
{ to other procedures.  Making them global simplifies the
code }
{ and makes intuitive sense. }
var
   terminalid     : shortint;  { terminalid used for
intrinsics }
   result         : shortint;  { result code of intrinsic
calls }
   offset        : shortint;  { current offset in data stream
}
   numfields      : shortint;  { number of fields in the
current screen }
   error          : boolean;   { global error flag }
procedure VERS3270;     intrinsic;
procedure ERR3270;      intrinsic;
procedure CLOSE3270;    intrinsic;
procedure OPEN3270;     intrinsic;
procedure READSTREAM;   intrinsic;
procedure RECV3270;     intrinsic;
procedure TRAN3270;     intrinsic;
procedure WRITESTREAM;  intrinsic;
procedure CTRANSLATE;   intrinsic;   { MPE intrinsic that
provides }
                                     { EBCDIC/ASCII conversions
}
{
```

```
 * The following procedure takes an errorcode, which is usually a result code
 * from another intrinsic call, and prints out the corresponding message.
 * The conversion is done by ERR3270.  ERR3270 takes an errorcode, fills
msgbuf
 * with the corresponding message, assigns msglen to the message length,
 * and sets result to the result code.
}
procedure print_message (errorcode: shortint);
var
   msgbuf  : packed array[1..144] of char;
   msglen  : shortint;

begin

{ Set flag if any fatal errors have occurred. }
{ errorcode=0 (and errorcode=9, on MPE V) are not errors. }
   if errorcode <<>> 0 then
      error := TRUE;
   msgbuf := ' '
   ERR3270 (errorcode, msgbuf, msglen, result);

{ Check whether the ERR3270 intrinsic generated any errors. }
   if result = 0 then
      writeln (msgbuf:msglen)
   else begin
      writeln ('INTERNAL ERROR in program:  ERR3270 result = ', result:2);
      error := TRUE;
   end  { if - else }
end;
{
 * The following procedure calls RECV3270 to receive data
 * from the host and buffer it in the internal screen image.  The variable
 * terminalid is global and is set by OPEN3270.
}
```

```
procedure call_recv3270;
begin
   write (' RECV3270.........');
   RECV3270 (terminalid, result);
   print_message (result);
end;

{
 * The following procedure calls TRAN3270.  The variable terminalid
 * is global and is set by OPEN3270.
}

procedure call_tran3270 (aid : shortint);
var
   cursorrow    : shortint;
   cursorcolumn : shortint;

begin
   cursorrow := -1;
   cursorcolumn := -1;  { suppress sending cursor address in transparent mode
}
   write (' TRAN3270............');
   TRAN3270 (terminalid, aid, cursorrow, cursorcolumn, result);
   print_message (result);
end;
{
 * The following procedure calls READSTREAM to read the data stream
 * received from the host.  It also calls the MPE V intrinsic CTRANSLATE
 * to translate the data from EBCDIC to ASCII.  Note that a RECV3270 call
 * must be made prior to each call to READSTREAM.  RECV3270 stores data
 * from the host in the internal screen image, while READSTREAM
 * retrieves the data.  So, a call to RECV3270 and a call to READSTREAM
 * must be made for each RU received.  This procedure also sets the
```

```
 * offset variable to the length of the data stream.  This value is used as
 * the offset in the next call to WRITESTREAM.  The variable terminalid
 * is global and is set by OPEN3270.
}

procedure call_readstream (var offset: shortint);
var
   read_offset  : shortint;
   maxinbuflen  : shortint;
   inbuf        : screen;
   actinbuflen  : shortint;
begin
   read_offset := 0;
   maxinbuflen := SCREENSIZE;
   write (' READSTREAM.............');
   READSTREAM (terminalid, read_offset, maxinbuflen, inbuf,
               actinbuflen, result);
   print_message (result);
   writeln ('                the data sent is shown below:');

   CTRANSLATE (EBCDICtoASCII, inbuf, inbuf, actinbuflen, );
                { An optional parameter has been omitted, so the comma }
                { after the actinbuflen parameter is necessary. }
                { The same variable may be used for the input and output }
                { to the CTRANSLATE intrinsic. }
   writeln (inbuf:actinbuflen);
   offset := actinbuflen;  { set offset for the next call to WRITESTREAM }
end;
{
 * The following procedure uses CTRANSLATE to
 * translate the data from ASCII to EBCDIC.  Then, it calls WRITESTREAM
 * to put the data into the internal screen image.  The data will be sent
 * when TRAN3270 gets called from procedure call_tran3270.
 * When data is sent to the SSCP, the internal screen image is searched,
```

```
 * beginning at the initial cursor address (where the cursor was left
 * after the previous host message).  Therefore, during the LU-SSCP session,
 * the offset must be set to the length of the previous data stream received.
 * The offset value is set from the previous call to call_readstream.
 * The variable terminalid is global and is set by OPEN3270.
}

procedure call_writestream (outbuf:string; outbuflen, offset: shortint);
begin
   CTRANSLATE (ASCIItoEBCDIC, outbuf, outbuf, outbuflen,)
                { An optional parameter has been omitted, so the comma }
                { after the outbuflen parameter is necessary. }
                { The same variable may be used for the input and output }
                { to the CTRANSLATE intrinsic. }



   write (' WRITESTREAM.....');
   WRITESTREAM (terminalid, offset, outbuflen, outbuf, result);
   print_message (result);
end;
{
 * The following function calls READSTREAM to get the data received
 * from the host and CTRANSLATE to translate it.  READSTREAM reads the data
 * from the internal screen image and writes it to inbuf.  CTRANSLATE
 * reads data from inbuf, translates it, and writes the translated data
 * back into inbuf.  The function then searches the translated data for the
 * string "READY".  The variable terminalid is global and is set by OPEN3270.
}

function ready : boolean;
var
   read_offset     : shortint;
   maxinbuflen     : shortint;
```

```
   actinbuflen     : shortint;
   inbuf           : screen;
   inbuf_offset    : shortint;
   word            : packed array[1..5] of char;
   i               : shortint;
   done            : boolean;
begin
   read_offset := 0;
   maxinbuflen := SCREENSIZE;
   inbuf := ' ';
   write (' READSTREAM..........');
   READSTREAM (terminalid, read_offset, maxinbuflen, inbuf,
               actinbuflen, result);
   print_message (result);

   writeln ('               the data sent is shown below:');
   CTRANSLATE (EBCDICtoASCII, inbuf, inbuf, actinbuflen, );
                { An optional parameter has been omitted, so the comma }
                { after the actinbuflen parameter is necessary. }
                { The same variable may be used for the input and output }
                { to the CTRANSLATE intrinsic. }
   writeln (inbuf:actinbuflen);
   inbuf_offset := 0;
   done := FALSE;
   ready := FALSE;


@COMPUTERTEXTW =    while not done do
   begin
{ search for 'R' }
      while (inbuf_offset << actinbuflen) and (inbuf[inbuf_offset + 1] <<>>
'R') do
         inbuf_offset := inbuf_offset + 1;
```

```
      if inbuf_offset <<= (actinbuflen - 5) then
      begin


{ copy candidate string into word }
         for i := 1 to 5 do
            word[i] := inbuf[inbuf_offset + i];

{ check whether the string is "READY" }
         if word = 'READY' then
         begin
            ready := TRUE;
            done := TRUE;
         end;
inbuf_offset := inbuf_offset + 1;
      end
      else done := TRUE;
   end;  { while }
end;
{
 * The following procedure calls OPEN3270, simulating turning on an IBM
 * display station.  OPEN3270 assigns a value to the global variable
 * terminalid.  This value is used to reference the device in all subsequent
 * SNA IMF intrinsic calls.  Note that RECV3270 must be called after a call
 * to OPEN3270 to receive the unowned screen.
}
procedure initialize;
var
   version     : string;
   devicenum   : shortint;
   snalnkinfo  : string;
   flags       : flags_type;
   devtype     : shortint;
   ffindex     : shortint   { not used by SNA IMF. Included }
```

```
                              { for backwards compatibility. }
   screensize  : shortint;
   timeout     : packed array[1..2] of shortint;

begin
   writeln (' VERS3270.............')
   VERS3270 (version);
   writeln ('                      version = ', version:14);

   devicenum := TERMINAL;
   snalnkinfo := 'IBMNODE#IMFCLASS ';  { non-alphanumeric character }
                                       { marks end of string }

   with flags do
   begin
      filler := 0;
      dbcs := 0;       { disable double byte character set option }
      unbind := 0;     { disable unbind option }
      LUT1_LUT3 := 0;  { not applicable to terminal emulation. Set to 0 }
      int_trace := 0;  { internal tracing off }
      trans_mode := 1; { transparent mode on }
      IO_mode := 0;    { standard I/O mode }
   end;

   ffindex := 0;
   timeout[1] := 30;
   timeout[2] := 30;
   writeln ('Now opening LU.T2 session.....');
   write (' OPEN3270.............');
   OPEN3270 (devicenum, snalnkinfo, flags, terminalid, devtype,
             ffindex, screensize, timeout, result);
   print_message (result);
   writeln;
   writeln ('Receiving unowned screen.....');
```

```
   call_recv3270;
   call_readstream (offset);

   error := FALSE;
end;

@COMPUTERTEXTW = {
 * The following procedure makes the major procedure calls.
 * It does the following:
 *   1. Sends a system request to the host and receives the LU-SSCP screen.
 *   2. Logs onto the host, starting an LU-LU session.  Note that after
 *      receiving the logon message, the host may send more than one screen
 *      before it is ready to receive again.
 *   3. Clears the screen and logs off the host.
 * Note that a call_recv3270 follows shortly after each call_tran3270.
}
procedure process;
var
   offset          : shortint;
   total_offset    : shortint;
   outbuf          : string;
   outbuflen       : shortint;

begin
{ Transmit System Request Key and receive untranslated LU-SSCP screen. }
   writeln;
   writeln ('Transmitting System Request Key and receiving screen.....');
   call_tran3270 (SYS_REQ_KEY);
   call recv3270;
   call_readstream (offset);

{ Write logon data stream to internal screen image. }
   writeln;
   writeln ('Writing "logon applid....."');
```

```
   outbuf := 'logon applid(tso) data(sales/sales) logmode(imf2k)';
   outbuflen := 50;
   call_writestream (outbuf, outbuflen, offset);

{ Transmit ENTER key to send logon message to host. }
   writeln;
   writeln ('Transmitting ENTER key.....');
   call_tran3270 (ENTER_KEY);

   offset := 0;
@COMPUTERTEXTW = { The host may send more than one RU here, so we have to make
sure the }
{ host is finished sending before we try to send.  The function ready }
{ checks whether the host is finished sending by searching for the string }
{ "READY" in the data received from the host.  We could just as easily have }
{ waited for a RECV3270 to time out to tell us when the host was finished }
{ sending and ready to receive, but searching for "READY" is more efficient.
}

   repeat
      writeln;
      writeln ('Receiving screen and checking for "READY" or error.....');
      call_recv3270;
   until error or ready;

{ Transmit CLEAR key to host and receive cleared screen from host. }
   call_tran3270 (CLEAR_KEY);
   call_recv3270;
   call_readstream (offset);

{ Write logoff to internal screen image. }
   outbuf := 'logoff';
   outbuflen := 6;
   writeln;
```

```
   writeln ('Writing "logoff" to screen.....');
   call_writestream (outbuf, outbuflen, offset);

{ Transmit ENTER key to send logoff message to host. }
   writeln;
   writeln ('Transmitting ENTER key.....');
   call_tran3270 (ENTER_KEY);
end;
@COMPUTERTEXTW = {
 * The following procedure calls CLOSE3270,
 * which simulates turning off the device.
}

procedure terminate;
begin
   write (' CLOSE3270...........');
   CLOSE3270 (terminalid, result);
   print_message (result);
end;



begin  { main }
   initialize;
   process;
   terminate;
end.
```

G Migrating Applications from SNA IMF/V to SNA IMF/XL

Following are the changes you might have to make to your SNA IMF/V applications in order to migrate them to SNA IMF/XL:

1. If your program calls the ACQUIRE3270 intrinsic, you will have to add two parameters to your ACQUIRE3270 intrinsic calls, because the ACQUIRE3270 intrinsic has two more parameters for SNA IMF/XL than for SNA IMF/V.

   SNA IMF/V:

   ACQUIRE3270 (*snalnkinfo*, *devicenum*, *ldev*, *enhance*, *priority*, *blanks*, *format flags*, *result*)

   SNA IMF/XL:

   ACQUIRE3270 (*snalnkinfo*, *devicenum*, *ldev*, *enhance*, *priority*, *blanks*, *format flags*, *options*, *pfn*, *result*)

   The two added parameters are *options* (a 16-bit word that specifies trace, spooler priority, read timeout, and LaserJet II options) and *pfn* (a character array that specifies a spooler file name). Both are input parameters. See the description of the ACQUIRE3270 intrinsic, in Chapter 3 , “Intrinsics Used with Standard MPE I/O,” for more information.

   All SNA IMF intrinsic parameters are required, so you must declare, initialize, and pass the *options* and *pfn* parameters in your ACQUIRE3270 intrinsic call, even if your program does not use their values.

2. You might have to modify your program to call RECV3270 more times in order to receive a complete transmission from the host. This is because SNA IMF/XL receives less data in a single RECV3270 call than SNA IMF/V.

   One way to modify your program is to have it check the internal screen image after every call to RECV3270. If you know how many fields should be in the screen sent by the host, or you know what should be in the last field on the screen, your program can call RECV3270 in a loop that terminates when the last field of the screen has been received.

   Another way to modify your program is to specify a short *timeout* value in your OPEN3270 call, and call RECV3270 in a loop that terminates when the timeout interval expires. Your program will keep calling RECV3270 until it receives all the data from the host; then, the timeout interval will expire and your program will exit the loop.

3. If your program uses result code 9 as a signal to call the RECV3270 intrinsic, you must modify your program, because SNA IMF/XL never returns result code 9.

SNA IMF/V returns result code 9 to indicate that the internal screen image has changed since the last time your program looked at it. Result code 9 indicates that your program should call RECV3270 to receive the current screen image. SNA IMF/XL does not return result code 9, because it does not modify the internal screen image until your program calls RECV3270.

Your SNA IMF/XL program should call RECV3270 after calling OPEN3270 or TRAN3270, or whenever it expects data from the host. If new data has arrived from the host, RECV3270 will return result code 0. If no data has arrived, the receive timeout interval will expire, and RECV3270 will return result code 24.

If your SNA IMF/V application uses no-wait I/O, you must make the following changes in addition to the ones just mentioned:

4. Change all MPE IOWAIT and IODONTWAIT intrinsic calls to SNA IMF IOWAIT3270 and IODONTWAIT3270 intrinsic calls. Calling MPE IOWAIT and IODONTWAIT intrinsics from an SNA IMF/XL application can cause the system to hang.
5. If your SNA IMF/V application calls IOWAIT or IODONTWAIT with the f*ilenum* parameter set to zero, make sure when you change your intrinsic calls to IOWAIT3270 or IODONTWAIT3270 that your program checks the *fnum* value returned by these intrinsics. When you set the *filenum* parameter to zero, IOWAIT3270 and IODONTWAIT3270 indicate the completion of any I/O, even if it is only the opening of a message file. The *fnum* value tells your program which I/O operation has completed.

A

ACF/NCP Advanced Communications Function for the Network Control Program. An IBM program product that resides in the 37xx Communication Controller and supports single and multiple domains.

ACF/VTAM Advanced Communications Function for the Virtual Telecommunication Access Method. An IBM program on the host that provides single-domain SNA network capability, and optionally, multiple-domain capability.

B

baud A measure of signaling speed equal to the number of signal changes or events per second. Baud often refers to bits per second.

BIND A request to activate a session between two logical units. A BIND is an SNA request sent by the host to SNA IMF. A BIND request specifies the detailed protocol that SNA IMF accepts before initiating an LU-LU session.

blocked See **wait I/O**.

C

CICS Customer Information Control System. An IBM interactive subsystem that acts as an interface between terminal users and a host's application programs and databases. CICS also includes facilities for building, using, and maintaining databases.

cluster controller A programmable device that supports one or more terminals or printers. A cluster controller communicates with the host either through a local channel attachment or through a communications controller, modems, and phone lines. The HP 3000 emulates a cluster controller in an SNA network.

communications controller A type of front-end processor, such as an IBM 3705, 3720, or 3725, that communicates between the communications facilities and a host computer. IBM supports both programmable and non-programmable communication controllers. Hewlett-Packard's INP and PSI are kinds of communications controllers.

compatibility mode A processing mode on HP 3000 Series 900 computers that allows MPE V applications to run on MPE XL machines without changes or recompilation.

control unit (CU) A device that controls input and output for one or more devices such as printers or display stations.

D

data stream mode See **transparent mode**.

DBCS Double-Byte Character Set. A character set that includes one- and two-byte characters, used by many Asian countries.

E

end user A person at a terminal, a device, or a program running in a processor that interacts with the network to obtain a service provided by the network. An end user is the source and destination of application data flowing through the network.

extra data segment (XDS An MPE V extra data segment, which SNA IMF/V uses to maintain data structures related to the internal screen image, half sessions, and message units in an LU-LU session.

F

FMD Function Management Data Services. A generic term that describes both session presentation and session network services.

H

half session A component of an SNA network that provides FMD services and data flow and transmission control for one of the sessions of an NAU.

HDLC High-level Data Link Control. A bit-oriented data link protocol used in full-duplex communications. HDLC was developed by the International Standards Organization

host A central computer that provides services for other computers and terminals attached to it.

I

IMF/3000 An HP Interactive Mainframe Facility product (like SNA IMF) that runs only on MPE V. IMF/3000 supports both BSC and SDLC protocols. It emulates a Node Type 1 in an SNA network

IMS Information Management System (IMS). An IBM program product that provides data communication interfaces between application programs and terminals and between application programs and databases.

INP Intelligent Network Processor. A hardware portion of SNA Link/V that implements the Physical Control and Data Link Control layers of SNA.

internal screen image A data handling technique used by SNA IMF. When using non-transparent mode, the internal screen image holds a character-by-character image of an IBM 3278 screen. When using transparent mode, your application program can obtain the untranslated data stream from the internal screen image.

intrinsic A subprogram provided by Hewlett-Packard to perform common functions such

as opening files, opening communication lines, or sending and receiving data over a communication line.

L

link The physical or logical connection between two devices in a network.

Logical Unit (LU) A program or a set of programs within a node that provides access to an SNA network for an end user. A Logical Unit can support two types of sessions: LU-to-SSCP sessions and LU-LU sessions.

LU class Logical Unit class. A set of Logical Units on the HP 3000. For SNA IMF, an LU class may contain multiple LUs. For NRJE, each LU class may contain only one LU.

LU-LU session A connection between two LUs.

LU.T1 Logical Unit Type 1. A program or set of programs within a node that provides communication between application programs and data processing programs using interactive or batch data transfer. Character oriented printer devices, batch support (RJE), and the SNA Character String (SCS) are supported.

LU.T2 Logical Unit Type 2. A program or set of programs within a node that provides application program to 3270-type display station communication using interactive data transfer. IBM 3270 data stream capability and 3270-type display station support is provided.

LU.T3 Logical Unit Type 3. A program or set of programs within a node that provides application program to printer communication. IBM 3270 data stream capability is provided. Many devices support both LU.T1 and LU.T3 print requests.

M

MPE V Multiprogramming Executive V. An operating system for the Hewlett-Packard 3000 computer, Series 37 through 70. MPE V consists of programs that handle exchanges between HP terminals, printers, and executing programs and the internal HP 3000 Communications Services.

MPE XL Multiprogramming Executive XL. An operating system for the Hewlett-Packard 3000 computer, Series 930 and 950. MPE XL consists of programs that handle exchanges between HP terminals, printers, and executing programs.

MVS Multiple Virtual Storage. An IBM operating system that is an extension of OS/MVT. MVS also is called OS/VS2 Release 2.

MVT Multiprogramming with a Variable number of Tasks. An IBM operating system that supports multiple programming with a variable number of tasks on the System/360.

N

Native Language Support (NLS) An HP product that allows the HP 3000 to produce

localized application programs for end users without reprogramming for each language or country.

native mode The normal processing mode of the MPE XL operating system. Native mode applications are those that have been compiled in the native instruction set of the HP 3000 Series 900 computer.

NAU Network Addressable Unit. Either a program or a group of programs that represents the source and destination of data in a network. The three kinds of network addressable units are SSCP, LU, and PU.

NCP See **ACF/NCP**.

NLS See **Native Language Support**.

NM capability Node Management capability. An MPE user capability required for SNA IMF operator tasks. The node manager configures an SNA service on the HP 3000.

no-wait I/O A form of input/output handling in which the system overlaps multiple I/O requests, overlaps I/O requests with CPU, and responds to the completion of the first of several outstanding I/O requests. With no-wait I/O, your program does not suspend and continues processing while I/O completes. See **wait I/O**.

node A basic component of an SNA network, which consists of a set of hardware devices and associated software that are at the end of a data link. Specifically, nodes within an SNA network can be distributed or host processors, communications controllers, cluster controllers, or terminals.

Node Management Services (NMS) A set of utilities on the HP 3000 that handles link and node level startup and shutdown, logging, tracing, and diagnostic functions. On MPE V, NMS is part of the SNA Link/V product. On MPE XL, NMS is part of the Fundamental Operating System.

Node Type 1 A terminal or printer.

Node Type 2 cluster controller, such as the IBM 3274.

Node Type 4 A communications controller such as the IBM 3705, 3720, and 3725.

Node Type 5 A host processor with a System Services Control Point (SSCP).

non-transparent mode A mode of SNA IMF operation in which the internal screen image holds a character-by-character image of an IBM 3278 screen. This image is accessible by the SNA IMF intrinsics.

NRJE Network Remote Job Entry. A Hewlett-Packard product that uses IBM's SNA/SDLC protocol to emulate an IBM 8100 Distributed Processing Program RJE workstation. This emulation allows users on the HP 3000 to submit batch jobs, through an SNA network, to an IBM host or compatible mainframe for processing. The host can then send the output back to the HP 3000 for printing or storing on disk.

P

Pass Thru An SNA IMF application program that uses SNA IMF intrinsics to allow certain HP terminals attached to an HP 3000 to emulate, with some differences, IBM 3278 display stations and IBM 3287 printers connected to an IBM host.

physical unit (PU) A set of SNA components that provide services that manage and monitor the resources of a node. Each node, whether a control unit, terminal, controller, or processor, contains one physical unit.

PM (privileged mode) capability MPE capability that allows programs to operate with no-wait I/O. PM capability bypasses the normal checks and limitations that apply to standard users.

programmatic access mode A form of programmatic communication using SNA IMF that allows programs on the HP 3000 to access host programs such as CICS, IMS, and TSO through a set of high-level intrinsics. To the host, the HP 3000 program using SNA IMF intrinsics looks like either an IBM 3278 display station or an IBM 3287 printer.

PSI Programmable Serial Interface. A hardware portion of SNA/SDLC Link/XL and SNA/X.25 Link/XL that implements the Physical Control and Data Link Control layers of SNA

R

Request Unit (RU) A message unit that contains control information such as a request code or function management headers, end-user data, or both.

Response Unit (RU) A message unit that acknowledges a Request Unit. If positive, the Response Unit may contain additional information (such as session parameters in response to BIND SESSION), or if negative, contains sense data defining the exception condition.

S

SCS See **SNA Character String**.

SDLC Synchronous Data Link Control. A protocol for managing synchronous, code-transparent, serial-by-bit information transfer over a link connection. Transmission exchanges may be full-duplex or half-duplex over switched or nonswitched links. The configuration of the link connection may be point-to-point, multipoint, or loop.

security video An SNA IMF feature that emulates the non-display feature of an IBM 3278 display station. Security video allows you to type data into a non-display field without having the data appear on your screen.

session A logical connection between Network Addressable Units.

Shift-in (SI) The control code with the hexadecimal value of “0F” that is used in the data stream to signal the end of a sequence of Asian (DBCS) characters.

Shift-out (SO) The control code with the hexadecimal value of "0E" that is used in the data stream to signal the beginning of a sequence of Asian (DBCS) characters.

SNA Systems Network Architecture. A comprehensive specification for distributed data processing developed by IBM in 1974. SNA defines rules, procedures, and a layered protocol for communication and control within a network.

SNA Character String (SCS) A character string composed of EBCDIC control characters, optionally intermixed with end-user data, that is carried within a Request Unit (RU).

SNA IMF SNA Interactive Mainframe Facility. An HP product that emulates a remote IBM 3274 Cluster Controller with attached IBM 3278 display stations and IBM 3287 printers. SNA IMF provides interactive communication between an HP 3000 and an IBM host computer in an IBM SNA environment.

SNA Link/V Bundled software and hardware that provides a logical and physical connection between an HP 3000, with MPE V, and an SNA network. SNA Link/V consists of the following:

Node Management Services (NMS) — software

SNA Transport — software

INP and cable — hardware

SNA/SDLC Link/XL Bundled software and hardware that provides a logical and physical connection between an HP 3000, with MPE XL, and an SNA network. SNA/SDLC Link/XL consists of the following:

SNA Transport — software

PSI and cable — hardware

SNA NRJE See **NRJE**.

SNA Service A Hewlett-Packard software product that provides the user interface to an SNA network. An SNA Service implements the upper three SNA layers; Data Flow Control, Function Management Data Services, and NAU Services Manager. SNA IMF, SNA NRJE, and LU 6.2 API are examples of SNA services.

SNA Transport Software residing immediately below SNA IMF that provides the functions of the Path Control and Transmission Control layers and manages all the sessions with the host SSCP. SNA Transport is part of the SNA link product.

split order A condition that occurs when SCS control codes or parameters span RU boundaries within an RU chain.

SSCP Systems Services Control Point. A part of the SNA host node that helps to manage configurations, does problem solving, controls network operations, and provides other

session services for end users. An SSCP exists only in the host and is exercised by the host's communications access method.

standard MPE I/O See **wait I/O**.

T

Time Sharing Option (TSO) A mode of SNA IMF operation that allows your application program to obtain the untranslated data stream when transferring data between the host and the HP 3000.

U

unblocked I/O See **no-wait I/O**.

V

VM Virtual Machine. An operating system that allows an IBM mainframe to run simultaneously several different operating systems, including multiple copies of CMS. These different operating systems run under VM.

VTAM See **ACF/VTAM**.

W

wait I/O The standard method of handling input/output on MPE systems. A process issuing an I/O request is suspended until the I/O request completes. Unlike no-wait I/O, wait I/O does not allow overlapping of I/O requests. See **no-wait I/O**.

X

XDS See **extra data segment**.

Index