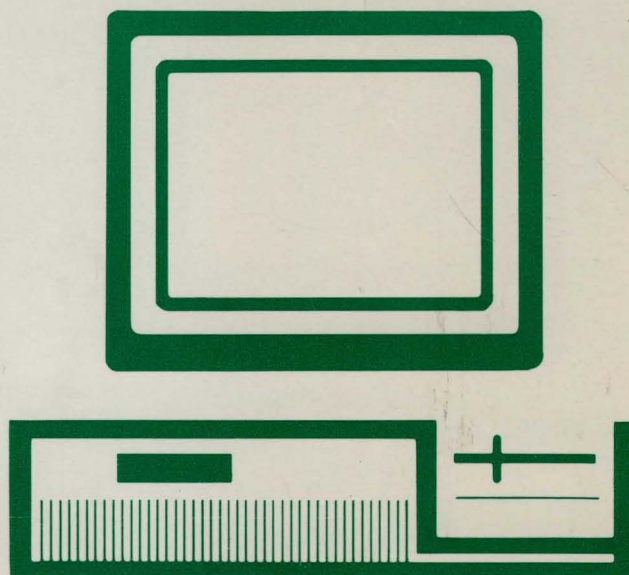


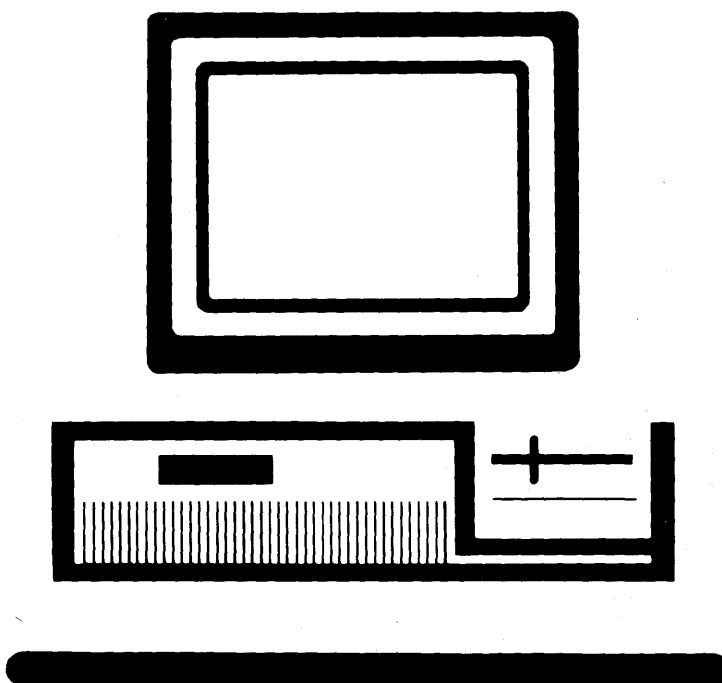
IBM 3270 Workstation Program

Programming Guide



84X0390
SA23-0343-0

Programming Guide



First Edition (April 1987)

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

Publications are not stocked at the address given below. Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, address comments to IBM Corporation, Department 95H/998, 11400 Burnet Road, Austin, TX 78758. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Preface

This manual describes how to use the services provided by the Application Program Interface (API) for the IBM 3270 Workstation Program (also referred to as the *workstation program*).

This book consists of five parts:

- The chapters in Part 1 introduce the Application Program Interface (API) and the two types of services you can use:
 - Application program services that most application programs will use. Described also are some supervisor services that directly support the application program services.
 - Those supervisor services that allow application programs to run together under the multitasking capabilities of the workstation program.
- The chapters in Part 2 tell you how to invoke the application program services. A sample block of code is provided for each service, so that you can see how it is used in context.
- The chapters in Part 3 describe, and tell you how to invoke, the supervisor services. A sample block of code is provided for each service, so that you can see how it is coded in context.
- The chapters in Part 4 contain representative sample programs using most of the services described in Parts 2 and 3.
- Part 5 consists of appendixes with specialized information. In particular, Appendix A provides information on scan codes and shift states for all supported keyboards. Appendix A also contains ASCII/ASCII-mnemonic values common to all languages and the additional values specific to U.S. English.

You will also want to use Appendix H, "Return Codes."

Enhancements

The *3270 Workstation Program Programming Guide* contains revisions to the *3270 PC Control Program Programming Guide*, and incorporates the following new material:

- *Non-3270 PC Hardware.* IBM Personal Computer AT® and XT system units, without the keyboard adapter and 3270 PC display adapter cards installed, are supported in this release. IBM Personal Computer AT and XT keyboard foldouts can be found at the back of this book.
- *ASCII keystroke API support.* The Keyboard Service API allows applications to send and receive keys in ASCII or ASCII mnemonics. The 3270 Emulation Services API allows you to receive keystrokes in ASCII. Chapters 5 and 9 contain more information on ASCII keystroke API support.
- *Keystroke API Support for READ.* Keystroke API support for READ now allows you to receive keys with a NOWAIT option which prevents you from being suspended while waiting for input on your queue.
- *Outbound Data Stream Preprocessor Option.* ODSP allows the preprocessing of a 3270 outbound data stream which can reduce the amount of data traffic flowing through a network. See Appendix I for more information on ODSP.
- *SPIF Utility Enhancement.* The SPIF utility has been enhanced to allow you to run an application that installs an interrupt handler which changes to its own stack and then enables interrupts. This may cause unpredictable results on systems with an XMA card installed unless you use the SPIF utility to update the INDIBM2.SIF file first. See the *IBM 3270 Workstation Program User's Guide and Reference* for more information about updating INDIBM2.SIF.

Prerequisites for Your Using the API

The API is written for application and system programmers who are responsible for the design and implementation of assembler-language programs for the IBM 3270 Personal Computer.

To use the API, you must have available the following software:

- DOS 3.2
- The IBM 3270 Workstation Program
- The IBM Macro Assembler or an equivalent assembler written for the Intel 8088 architecture.

Prerequisite knowledge needed to be able to use the information in this manual includes:

- Proficiency in the use of the IBM Personal Computer Macro Assembler language
- Knowledge of the steps required to assemble, link, and run a macro assembler program on the IBM 3270 Personal Computer
- Familiarity with the DOS function calls that can be used in a macro assembler program
- Familiarity with the IBM 3270 data stream.

Related Publications

The following books are related to the 3270 Workstation Program and its prerequisite hardware and software:

- *Guide to Operations*

The *Guide to Operations* shipped with your system unit contains information about your work station hardware. It tells you how to set up, use, and diagnose problems with the hardware.

- *3270 Personal Computer Hardware Introduction and Preinstallation Planning*¹

This book contains information to help evaluate and plan for the 3270 PC hardware requirements at your site. For example, it lists the physical dimensions and electrical requirements for all 3270 PC hardware models.

- The following items are shipped with the workstation program diskettes:

- *3270 Workstation Program User's Guide and Reference*

This book contains information about setting up and using the workstation program.

- *3270 Workstation Program Problem Determination Guide and Reference*

¹ Contact your local IBM sales representative for information on how to obtain copies of these books.

This book explains the procedures, messages, and return codes that will help you solve software problems.

– 3270 Workstation Program Keyboard Quick Reference Cards

These cards are keyboard-specific synopses of information from the *User's Guide and Reference*. You can use the one that relates to your keyboard for quick reference. There are three cards in the workstation program package:

- 3270 PC keyboard Quick Reference
- Enhanced PC keyboard Quick Reference
- AT and XT keyboard Quick Reference

– 3270 Workstation Program Keyboard Templates

The keyboard templates provided in the package assist you in using the workstation program functions on your particular keyboard. There are three templates in the package:

- Enhanced PC keyboard template
- AT keyboard template
- XT keyboard template

– Online tutorial diskette (*Helper*)

This diskette contains introductory information and practice exercises to help in learning to use the 3270 Workstation Program.

- *3270 PC High Level Language Application Programming Interface (HLLAPI)*²

The diskette and book that comes in this package make it possible for you to write application programs in BASIC, Pascal, or COBOL languages to use the API functions provided with the 3270 Workstation Program.

- *The IBM Programmer's Guide to the Server-Requester Programming Interface for the IBM Personal Computer and the IBM 3270 PC*²

This book explains how to write PC applications that request services from an application at an IBM System/370 type host system. In this relationship, the PC application is called the *requester* and the host application is called the *server*. This book also contains the return codes that are generated at the work station if problems occur in transmitting requests or replies.

² Contact your local IBM sales representative for information on how to obtain copies of these books.

For information on IBM Personal Computer DOS, refer to the DOS manuals that were shipped with the version of DOS you are using.

For information on IBM Personal Computer assembler language, use this manual:

- *IBM Personal Computer Language Series: Macro Assembler*³

Provides a reference for experienced assembler language programmers who use the IBM Personal Computer Macro assembler. Specific information is provided on how to use the Macro assembler, cross-reference facilities, pseudo-operations, and machine instructions. (Includes diskette.)

For information on the IBM 3270 data stream, use this manual:

- *IBM 3270 Information Display System: Data Stream Programmer's Reference*³

Provides information for programmers who need to know what is involved in using the 3270 data stream to produce panels or information at displays and printers.

³ Contact your local IBM sales representative for information on how to obtain copies of these books.

Contents

Part 1. Introduction to the API

Chapter 1. Functions the API Provides	1-1
Overview of API Services	1-2
Terms You Need to Know	1-3
Examples of Using the API	1-4
Simplifying Setup and Control of Multiple Host Sessions	1-4
Using the Work Station Control Functions of the IBM 3270 Personal Computer	1-5
Enhancing Interaction between the Operator and a Host	1-5
Extending the Workstation Program through the Use of System Extensions	1-5
The Application Program Services	1-5
Session Information Services	1-6
Keyboard Services	1-6
Window Management Services	1-6
Host Interactive Services	1-6
Presentation Space Services	1-6
3270 Keystroke Emulation Services	1-7
Copy Services	1-7
Translate Service	1-7
Operator Information Area Services	1-7
Multiple DOS Support Services	1-7
The Supervisor Services	1-7
Supervisory Object Services	1-8
Request Services	1-8
Task State Modifier Services	1-8
Semaphore Management Services	1-8
Logical Timer Management Services	1-8
Fixed-Length Queue Management Services	1-8
Interrupt Handler Management Services	1-9
Environment Manager Services	1-9
Using the Application Program Interface	1-9
 Chapter 2. Programming Considerations	 2-1
Introduction	2-2
System Information Files	2-2
Program Information Files	2-3
Creating and Modifying Program Information Files (PIFs)	2-4
Restrictions on Running under the Workstation Program	2-5
Guidelines for Running under Multi-DOS	2-6
How Multi-DOS Affects Application Program Performance	2-7
Using the Interrupt X'10' Function	2-8
Tips on Writing Applications to Run in Multi-DOS	2-9
When Personal Computer Sessions Will Be Suspended	2-10
Non-3270 PC Hardware Restrictions	2-11
Determining the Type of PC Your Application Program Is Running On	2-13
Determining the Level of the Control Program or the Workstation Program That Is Loaded	2-13

Part 2. Application Program Services

Conventions Used in the API Service Descriptions

Chapter 3. Coding Supervisor Services	3-1
Introduction	3-2
Obtaining the Gate Name for the Services Your Application Program Will Use	3-2
Obtaining the Results of Services You Have Requested Asynchronously	3-3
Creating Fixed-Length Queue Entries	3-3
Obtaining Data from a Fixed-Length Queue	3-3
Deleting Fixed-Length Queues	3-4
Requesting the Supervisor Services	3-4
Supervisor Service X'81': Name Resolution	3-5
Supervisor Service X'83': Get Request Completion	3-7
Supervisor Service X'04': Create Fixed-Length Queue Entry	3-9
Supervisor Service X'13': Dequeue Data	3-12
Supervisor Service X'06': Delete Entry	3-15
 Chapter 4. Coding Session Information Service Requests	4-1
Introduction	4-2
Requesting the Session Information Services	4-3
Return Codes for the Session Information Services	4-4
Session Information Service X'01': Query Session ID	4-5
Session Information Service X'02': Query Session Parameters	4-10
Session Information Service X'04': Detach Session ID	4-14
Session Information Service X'05': Attach Session ID	4-17
Session Information Service X'06': Query Windows in Environment ..	4-20
Session Information Service X'07': Query Environment of Window ..	4-23
Session Information Service X'08': Query PC Session Program Information File (PIF) Information	4-26
Session Information Service X'0A': Query Base Window	4-30
Session Information Service X'0B': Query Session Cursor	4-33
 Chapter 5. Coding Keyboard Service Requests	5-1
Introduction	5-2
Scan Code/Shift States	5-3
ASCII/ASCII Mnemonics	5-7
Keyboard Services	5-7
Requesting the Keyboard Services	5-8
Return Codes for the Keyboard Services	5-8
Keyboard Service X'01': Connect to Keyboard	5-9
Keyboard Service X'02': Disconnect from Keyboard	5-13
Keyboard Service X'03': Read Input	5-16
Keyboard Service X'04': Write Keystroke	5-22
Keyboard Service X'05': Disable Input	5-30
Keyboard Service X'06': Enable Input	5-33
Keyboard Service X'07': Post Status Code	5-36
 Chapter 6. Coding Window Management Service Requests	6-1
Introduction	6-2

Requesting the Window Management Services	6-5
Return Codes for the Window Management Services	6-6
Window Management Service X'01': Connect to Work Station Control	6-7
Window Management Service X'02': Disconnect from Work Station Control	6-11
Window Management Service X'03': Add Window	6-14
Window Management Service X'04': Change Window Position on Screen	6-17
Window Management Service X'05': Change Window Size	6-21
Window Management Service X'06': Change Window Color	6-25
Window Management Service X'07': Change Window Position on Presentation Space	6-29
Window Management Service X'08': Change Hidden State	6-33
Window Management Service X'09': Change Enlarge State	6-36
Window Management Service X'0A': Change Screen Background ...	6-38
Window Management Service X'0B': Query Window Position on Screen	6-41
Window Management Service X'0C': Query Window Size	6-44
Window Management Service X'0D': Query Window Colors	6-47
Window Management Service X'0E': Query Window Position on Presentation Space	6-51
Window Management Service X'0F': Query Hidden State	6-54
Window Management Service X'10': Query Enlarge State	6-57
Window Management Service X'11': Query Screen Background Color	6-60
Window Management Service X'12': Query Window Names	6-63
Window Management Service X'13': Clear Screen	6-66
Window Management Service X'14': Select Active Window	6-69
Window Management Service X'15': Redraw Screen	6-72
Window Management Service X'16': Redraw Window	6-75
Window Management Service X'17': Delete Window	6-78
Window Management Service X'18': Query Active Window	6-81
Window Management Service X'19': Query Active Screen	6-84
Window Management Service X'1A': Query Window Attributes	6-87
Window Management Service X'1B': Change Window Attributes ...	6-92
Window Management Service X'1C': Select Active Screen	6-98
 Chapter 7. Coding Host Interactive Service Requests	7-1
Introduction	7-2
Requesting the Host Interactive Services	7-2
Return Codes for the Host Interactive Services	7-3
Host Interactive Service X'01': Connect to Host Session	7-4
Host Interactive Service X'02': Disconnect from Host Session	7-11
Host Interactive Service X'03': Read Structured Field	7-15
Host Interactive Service X'04': Write Structured Field	7-20
Host Interactive Service X'05': Define Buffer	7-25
 Chapter 8. Coding Presentation Space Service Requests	8-1
Introduction	8-2
Requesting the Presentation Space Services	8-2
Return Codes for the Presentation Space Services	8-2
Presentation Space Service X'01': Define Presentation Space	8-4
Presentation Space Service X'02': Delete Presentation Space	8-11
Presentation Space Service X'03': Display Presentation Space	8-14

Presentation Space Service X'04': Set Cursor Position	8-17
Presentation Space Service X'05': Switch Presentation Space	8-21
Chapter 9. Coding 3270 Keystroke Emulation Service Requests .	9-1
Introduction	9-2
Field Attribute Definition for 3270 Keystroke Emulation	9-2
Presentation Space Format for 3270 Keystroke Emulation	9-4
Requesting the 3270 Keystroke Emulation Services	9-5
Return Codes for the 3270 Keystroke Emulation Services	9-5
3270 Keystroke Emulation Service X'01': Connect for 3270 Keystroke Emulation	9-7
3270 Keystroke Emulation Service X'02': Disconnect for 3270 Keystroke Emulation	9-10
3270 Keystroke Emulation Service: Read Attention Identifier (AID) Key	9-13
Chapter 10. Coding Copy Service Requests	10-1
Introduction	10-2
Requesting the Copy Services	10-3
Return Codes for the Copy Services	10-4
Copy Service X'01': Copy String	10-5
Copy Service X'02': Copy Block	10-12
Copy Service X'03': Connect for Copy to PC Session	10-19
Copy Service X'04': Disconnect for Copy to PC Session	10-22
Chapter 11. Coding Translate Service Requests	11-1
Introduction	11-2
Requesting the Translate Service	11-2
Return Codes for the Translate Service	11-2
Translate Service X'01': Translate Data	11-4
Chapter 12. Coding Operator Information Area Service Requests	12-1
Introduction	12-2
Requesting the Operator Information Area Services	12-3
Return Codes for the Operator Information Area Services	12-3
Operator Information Area Service X'01': Read Operator Information Area Image	12-4
Operator Information Area Service X'02': Read Operator Information Area Group	12-7
Chapter 13. Coding Multi-DOS Support Service Requests	13-1
Introduction	13-2
Requesting the Multi-DOS Support Services	13-2
Return Codes for the Multi-DOS Support Services	13-3
Multi-DOS Support Service: Query Environment Size	13-4
Multi-DOS Support Service: Asynchronous DOS Function Requests	13-7
Multi-DOS Support Service X'01': Get Storage	13-12
Multi-DOS Support Service X'02': Free Storage	13-15
Multi-DOS Support Service X'03': Set Storage Allocation	13-18
Part 3. Supervisor Services	
Conventions Used in the API Service Descriptions	

Chapter 14. Supervisor Services	14-1
Introduction	14-2
Supervisory Object Creation and Deletion	14-2
Tasks	14-2
Components	14-3
Semaphores	14-4
Fixed-Length Queues	14-4
Gates	14-5
User Exit Tables	14-5
The Supervisor Call Instruction (SVC) Table	14-5
Creating Objects with Names	14-5
Supervisory Object Services Your Application Program Can Use	14-6
Task Requests	14-6
Use of Wait States	14-7
Sending a Request to Another Task	14-8
Receiving a Request from Another Task	14-8
Replying to a Request from Another Task	14-9
Obtaining Request Completion from Another Task	14-9
Task Request Services Your Application Program Can Use	14-9
Task State Modifiers	14-10
Dispatch Cycles	14-10
Task Dispatching Procedure	14-10
Task Dispatch Activity	14-11
Task Dispatcher States	14-11
Task State Modifier Services Your Application Program Can Use	14-12
Semaphore Management	14-13
Considerations for Using Code Serialization Semaphores	14-13
Restrictions on the Use of Semaphores	14-13
Semaphore Management Services Your Application Program Can Use	14-14
Logical Timer Management	14-14
Logical Timer Management Services Your Application Program Can Use	14-15
Fixed-Length Queue Management	14-15
Fixed-Length Queue Management Services Your Application Program Can Use	14-15
Interrupt Handler Management	14-15
Hardware Interrupt Handlers	14-16
Software Interrupt Handlers	14-17
Interrupt Handler Management Services Your Application Program Can Use	14-19
 Chapter 15. Coding Supervisory Object Services	 15-1
Introduction	15-2
Requesting the Supervisory Object Services	15-3
Return Codes for the Supervisory Object Services	15-3
Supervisory Object Service X'92': Create Task Entry	15-4
Supervisory Object Service X'93': Create Component Entry	15-8
Supervisory Object Service X'94': Create Semaphore Entry	15-11
Supervisory Object Service X'04': Create Fixed-Length Queue Entry	15-14
Supervisory Object Service X'9A': Create Gate Entry	15-17
Supervisory Object Service X'97': Create User Exit Table Entry	15-21

Supervisory Object Service X'0E': Install User Exit Table Entries . .	15-24
Supervisory Object Service X'81': Name Resolution	15-27
Supervisory Object Service X'01': ID Resolution	15-30
Supervisory Object Service X'06': Delete Entry	15-32
 Chapter 16. Coding Request Services	16-1
Introduction	16-2
Requesting the Request Services	16-2
Return Codes for the Request Services	16-2
Request Service X'09': Make a Request	16-3
Request Service X'96': Get a Request	16-8
Request Service X'82': Reply to a Request	16-11
Request Service X'83': Get Request Completion	16-14
Request Service X'12': Send a Signal to a Task	16-17
 Chapter 17. Coding Task State Modifier Services	17-1
Introduction	17-2
Requesting the Task State Modifier Services	17-2
Return Codes for the Task State Modifier Services	17-2
Task State Modifier Service X'9C': Query Active Task	17-3
Task State Modifier Service X'02': Set Task "Ready"	17-4
Task State Modifier Service X'05': Set Task "Unready"	17-7
Task State Modifier Service X'08': Set Task "Preemptable"	17-10
Task State Modifier Service X'07': Set Task "Nonpreemptable"	17-12
Task State Modifier Service X'03': Change Task's Priority	17-14
Task State Modifier Service X'95': Return to Dispatcher	17-16
 Chapter 18. Coding Semaphore Management Services	18-1
Introduction	18-2
Requesting the Semaphore Management Services	18-2
Return Codes for the Semaphore Management Services	18-2
Semaphore Management Service X'0D': Claim a Semaphore	18-3
Semaphore Management Service X'0A': Release a Semaphore	18-6
Semaphore Management Service X'0B': Query a Semaphore	18-8
 Chapter 19. Coding Logical Timer Management Services	19-1
Introduction	19-2
Requesting the Logical Timer Management Services	19-2
Return Codes for the Logical Timer Management Services	19-2
Logical Timer Management Service X'85': Get Logical Timer	19-3
Logical Timer Management Service X'84': Set Logical Timer	19-5
Logical Timer Management Service X'8A': Release Logical Timer . .	19-8
 Chapter 20. Coding Fixed-Length Queue Management Services	20-1
Introduction	20-2
Requesting the Fixed-Length Queue Management Services	20-2
Return Codes for the Fixed-Length Queue Management Services . .	20-2
Fixed-Length Queue Management Service X'0C': Enqueue Data	20-3
Fixed-Length Queue Management Service X'13': Dequeue Data	20-5
Fixed-Length Queue Management Service X'0F': Purge Queue Data . .	20-8
 Chapter 21. Coding Interrupt Handler Management Services	21-1
Introduction	21-2

Requesting the Interrupt Handler Management Services	21-2
Return Codes for the Interrupt Handler Management Services . . .	21-3
Interrupt Handler Management Service X'86': Install a Hardware Interrupt Handler	21-4
Interrupt Handler Management Service X'87': Install an Interrupt Handler	21-7
Interrupt Handler Management Service X'88': Query Interrupt Vector Contents	21-10
Interrupt Handler Management Service X'89': Remove an Interrupt Handler	21-12
 Chapter 22. Environments and the Environment Manager	22-1
Introduction	22-2
Environments	22-2
Stoppable Environments	22-2
Nonstoppable Environments	22-3
Environment Access Restrictions	22-3
Environment Management Services Your Program or System Extension Can Use to Control Environments	22-4
Resource Managers	22-4
Environment Management Services Your System Extension Can Use to Control Resource Management	22-6
 Chapter 23. Coding Environment Manager Services	23-1
Introduction	23-2
Requesting the Environment Manager Services	23-3
Return Codes for the Environment Manager Services	23-3
Environment Manager Service X'10': Identify Resource Manager . . .	23-4
Environment Manager Service X'8E': Add Resource	23-8
Environment Manager Service X'8B': Delete Resource	23-12
Environment Manager Service X'8C': Query Resource	23-15
Environment Manager Service X'90': Suspend/Resume Environment	23-17
Environment Manager Service X'99': Stop/Reset Environment	23-23
Environment Manager Service X'11': Query Task's Environment ID .	23-34
Environment Manager Service X'8D': Query Environment Characteristics	23-36
 Chapter 24. Coding System Extensions	24-1
Introduction	24-2
How to Create a System Extension	24-3
Resident Code	24-3
Fixed Data	24-4
Initialization Code	24-4
How to Tell the Workstation Program about Your System Extension	24-5
Customization Procedure	24-5
Creating and Modifying System Information Files (SIFs)	24-9
How to Determine the Numbers to Use for Your System Information File	24-10
How a System Extension Is Loaded	24-13
System Extension Messages and Return Codes	24-15
System Extension Return Codes	24-15
The System Extension Message Service	24-16

Identifying Error Return Codes with the System Extension	
Message Service	24-16
Requesting Error Messages with the System Extension Message	
Service	24-17
Requesting Informational Messages with the System Extension	
Message Service	24-17
Coding the System Extension Message Service to Identify Return	
Codes	24-18
Coding the System Extension Message Service to Request Error	
Messages	24-20
Coding the System Extension Message Service to Request	
Informational Messages	24-23
Managing Resources	24-26
Design Considerations for System Extensions and the XMA Card ...	24-26
Components	24-27
Tasks	24-27
Fixed-Length Queues	24-27
General Notes	24-27

Part 4. Sample Programs

Chapter 25. Sample Program 1	25-1
Chapter 26. Sample Program 2	26-1
Chapter 27. Sample Program 3	27-1
Chapter 28. Sample Program 4	28-1
Chapter 29. Sample Program 5	29-1

Part 5. Appendixes

Appendix A. Scan-Code/Shift-State and ASCII/ASCII-Mnemonic

Values	A-1
Introduction	A-2
Scan-Code/Shift-State Values	A-2
Scan Code	A-2
Shift State	A-4
ASCII/ASCII Mnemonics	A-4
Default Scan Codes for the IBM 3270 PC Keyboard (PC Mode)	A-5
Default Scan Codes for the IBM 3270 PC Keyboard (MFI Mode)	A-9
Default Scan Codes for the IBM Enhanced PC Keyboard (PC Mode)	A-13
Default Scan Codes for the IBM Enhanced PC Keyboard (MFI Mode)	A-16
Default Scan Codes for the PC XT Keyboard (PC Mode)	A-19
Default Scan Codes for the IBM PC XT Keyboard (MFI Mode)	A-22
Default Scan Codes for the IBM Personal Computer AT Keyboard	
(PC Mode)	A-25
Default Scan Codes for the IBM Personal Computer AT Keyboard	
(MFI Mode)	A-28
ASCII Characters Common to All Countries	A-31
ASCII Mnemonics Common to All Countries	A-34

Additional ASCII Characters Used by U.S. English	A-37
Appendix B. Destination/Origin Structured Fields	B-1
Introduction	B-3
The 3270 Outbound Data Stream	B-3
The 3270 Inbound Data Stream	B-3
Verifying That the IBM 3270 Personal Computer Interface Is	
Operational	B-4
The Read Partition Query Structured Field	B-4
The Query Reply Structured Field	B-4
Query Reply	B-6
Input Control	B-6
PC Application Program and Display Interaction	B-7
Exception Handling	B-8
Structured Fields	B-9
Destination/Origin	B-9
Exception Condition	B-10
X'D0' Structured Fields for Sending Data from the Host to the 3270	
Personal Computer	B-11
The Open X'D0' Structured Field	B-12
The Insert and Insert Data X'D0' Structured Fields	B-15
The Close X'D0' Structured Field	B-18
X'D0' Structured Fields for Sending Data from Personal Computer to	
Host	B-20
The Open X'D0' Structured Field	B-21
The Set Cursor and Get X'D0' Structured Field	B-24
The Close X'D0' Structured Field	B-28
Appendix C. Using Command Procedures for Save and Restore	
and for File Transfer	C-1
Introduction	C-2
Command Procedures for Save and Restore	C-2
Creating an AUTOEXEC.Bat File	C-3
Programmed Command Procedures	C-4
File Transfer Command Procedures	C-6
Appendix D. Technical Notes	D-1
Introduction	D-2
3270 Limitations	D-2
3270 Data Stream Functions	D-3
Interface Codes	D-3
Attributes	D-6
Commands	D-8
Write Control Character	D-9
3270 Data Stream Orders	D-11
Outbound 3270 Data Stream Structured Fields	D-12
Inbound Structured Fields	D-21
Transmission of Buffer Addresses	D-31
Changes or Limitations to the Personal Computer Session	D-34
Non-3270 PC Hardware Restrictions	D-34
Personal Computer Physical Cursor	D-35
Personal Computer Print Spooling	D-35
Control Unit Communication Session Termination	D-36

IBM 3270 Personal Computer Failure	D-36
Color Limitations	D-36
Notes on All-Points-Addressable Graphics	D-37
Using the Full-Screen APA Mode	D-37
Changing the Cursor Size or Position	D-38
Personal Computer Session Screen Size	D-38
 Appendix E. Problem Determination Procedures and Debugging	
Information	E-1
Introduction	E-2
System Error Problem Determination Procedures	E-2
Dump Data Utilities	E-3
Preparing Formatted Dump Diskettes	E-3
Taking the Dump	E-4
Using the Display Utility	E-5
Using the Trace Command	E-8
Workstation Program Loading Procedure	E-9
Debugging a PC Application Program	E-10
Debugging a System Extension	E-10
Control Blocks You Can Use during Debugging	E-10
The Supervisor's Data Area	E-11
The Dispatcher's Data Area	E-11
The Task Control Block	E-12
The Supervisor Name Table	E-14
 Appendix F. Presentation Space Considerations	
Introduction	F-1
Introduction	F-2
Attributes	F-2
Field Attributes	F-3
Extended Field Attributes and Character Attributes	F-4
Presentation Space Character Tables	F-6
Host and Notepad Session Character Codes	F-6
Personal Computer Session Character Codes	F-7
Presentation Space Sizes	F-7
Distributed Function Terminal (DFT) Host Presentation Space Sizes	F-7
Control Unit Terminal (CUT) Host Presentation Space Size	F-8
Notepad Presentation Space Size	F-8
Personal Computer Presentation Space Size	F-8
 Appendix G. Calling Save, Restore, Send, and Receive from Your	
Application Program	G-1
Introduction	G-2
The DOS SETBLOCK Function Call	G-2
The DOS EXEC Function Call	G-3
The Environment String	G-3
The Command Line	G-4
The File Control Blocks	G-4
 Appendix H. Return Codes	
Introduction	H-1
Introduction	H-2
Function ID X'12': System Services Return Codes	H-3
Function ID X'13': Environment Manager Services Return Codes ..	H-11
Function ID X'22' or X'23': DOS Subsystem Services Return Codes ..	H-16

Function IDs X'24' or X'25': System Loader Return Codes	H-22
Function ID X'30': DFT Operations Return Codes	H-26
Function ID X'32': Host Interactive Services Return Codes	H-32
Function ID X'46': CUT Return Codes	H-33
Function ID X'51': Notepad Operations Return Codes	H-34
Function ID X'62': Keyboard Services Return Codes	H-35
Function ID X'63': Window Management Services Return Codes ...	H-38
Function ID X'64': Copy Services Return Codes	H-41
Function ID X'67': Draw Service Return Codes	H-43
Function ID X'69': Presentation Space Services Return Codes	H-44
Function ID X'6B': Session Information Services Return Codes	H-47
Function ID X'6C': Translate Services Return Codes	H-49
Function ID X'6D': OIA Services Return Codes	H-50
Function ID X'6E': 3270 Keystroke Emulation Services Return Codes	H-51
Function ID X'6F': Keystroke Definition Return Codes	H-52
Function ID X'72': Error Handler Return Codes	H-53
Function ID X'7F': Dump Task Return Codes	H-54
Function ID X'81': Enhanced Connectivity Router Return Codes ...	H-55
Function IDs X'Dx through Fx': User System Extension Return Codes	H-56
Return Code Error Steps	H-57

Appendix I. Outbound Data Stream Preprocessor (ODSP) Option	I-1
Introduction	I-2
Customizing for ODSP	I-2
Initializing ODSP	I-2
Using ODSP	I-3
Entry Parameters	I-4
Return Parameters	I-4
ODSP Restrictions and Recommendations	I-5
Sample Program for Outbound Data Stream Preprocessing	I-5
Index	X-1

Figures

1-1.	Overview of the Application Program Interface	1-2
A-1.	Default Scan Codes for IBM 3270 PC Keyboard (PC Mode) ..	A-5
A-2.	Default Scan Codes for IBM 3270 PC Keyboard (MFI Mode) ..	A-9
A-3.	Default Scan Codes for IBM Enhanced PC Keyboard (PC Mode)	A-13
A-4.	Default Scan Codes for IBM Enhanced PC Keyboard (MFI Mode)	A-16
A-5.	Default Scan Codes for IBM PC XT Keyboard (PC Mode) ..	A-19
A-6.	Default Scan Codes for IBM PC XT Keyboard (MFI Mode) ..	A-22
A-7.	Default Scan Codes for IBM Personal Computer AT Keyboard (PC Mode)	A-25
A-8.	Default Scan Codes for IBM Personal Computer Keyboard (MFI Mode)	A-28
A-9.	Valid ASCII Characters Common to All Countries	A-31
A-10.	Valid ASCII Mnemonics Common to All Countries	A-34
A-11.	Additional ASCII Characters Used by U.S. English	A-37
B-1.	Read Partition Query Structured Field Format	B-4
B-2.	Query Reply Structured Field Format	B-5
D-1.	United States EBCDIC I/O Interface Code	D-3
D-2.	EBCDIC Control Character I/O Codes	D-5
D-3.	Field Attribute Byte Bit Positions	D-6
D-4.	Field Attribute Character Bit Assignments	D-6
D-5.	The Structure of an Attribute Pair	D-7
D-6.	Attribute Type X'41' - Extended Highlighting	D-7
D-7.	Attribute Type X'42' - Color	D-7
D-8.	Attribute Type X'43' - Character Set Selection	D-7
D-9.	3270 Data Stream Commands	D-8
D-10.	Non-SNA Channel Commands	D-8
D-11.	Write Control Character Reset Actions	D-10
D-12.	3270 Data Stream Orders	D-11
D-13.	Set Reply Mode Structured Field Format	D-13
D-14.	Erase/Reset Structured Field Format	D-14
D-15.	Outbound 3270DS Structured Field Format	D-14
D-16.	Read Partition Structured Field Format	D-15
D-17.	Inbound 3270 Data Stream	D-17
D-18.	Short Read Format	D-18
D-19.	Read Modified and Read Modified All Format	D-18
D-20.	Read Buffer Format in Field Reply Mode	D-19
D-21.	Read Buffer Format in Extended Field and Character Mode ..	D-20
D-22.	Usable Area Query Reply Structured Field Format	D-22
D-23.	Character Sets Query Reply Structured Field Format	D-23
D-24.	Character Set Descriptors	D-24
D-25.	Reply Modes Query Reply Structured Field Format	D-24
D-26.	DDM Query Reply Structured Field Format	D-25
D-27.	Auxiliary Device Query Reply Structured Field Format	D-25
D-28.	Document Interchange Architecture Query Reply Structured Field Format	D-26
D-29.	Direct Access Self-Defining Parameter	D-26
D-30.	Color Query Reply Structured Field Format	D-27

D-31.	Highlight Query Reply Structured Field Format	D-28
D-32.	Implicit Partition Query Reply Structured Field Format	D-29
D-33.	Self-Defining Parameters	D-30
D-34.	Hexadecimal Representations	D-33
E-1.	Frequently Used Trace Buffers	E-7
E-2.	Dispatcher Data Address Offsets	E-12
E-3.	Name Table Address Offsets	E-14
F-1.	Field Attribute Bit Positions	F-4
F-2.	Field Attribute Bit Assignment	F-4
F-3.	Extended Field Attribute and Character Attribute Bit Positions	F-5
F-4.	Extended Field Attribute and Character Attribute Bit Assignment	F-5
F-5.	Host and Notepad Presentation Space Character Table	F-6
F-6.	Personal Computer Presentation Space Character Table	F-7
F-7.	DFT Host Presentation Space Sizes	F-8
.	IBM 3270 Personal Computer U.S. English Keyboard	FO-1
.	IBM 3270 Enhanced Personal Computer U.S. English Keyboard, PC Mode	FO-3
.	IBM 3270 Enhanced Personal Computer U.S. English Keyboard, MFI Mode	FO-5
.	IBM Personal Computer AT U.S. English Keyboard	FO-7
.	IBM Personal Computer XT U.S. English Keyboard	FO-9

Part 1. Introduction to the API

The chapters in Part 1 introduce the Application Program Interface (API) and the two types of services you can use:

- Application program services that most application programs will use. Described also are some supervisor services that directly support the application program services.
- Those supervisor services that allow application programs to run together under the multitasking capabilities of the workstation program.

The chapters in this part are:

- Chapter 1, “Functions the API Provides,” which contains an overview of the API services and how you can use them.
- Chapter 2, “Programming Considerations,” which introduces system information files, describes program information files, and provides tips and guidelines for coding programs.

Chapter 1. Functions the API Provides

Overview of API Services	1-2
Terms You Need to Know	1-3
Examples of Using the API	1-4
Simplifying Setup and Control of Multiple Host Sessions	1-4
Using the Work Station Control Functions of the IBM 3270 Personal Computer	1-5
Enhancing Interaction between the Operator and a Host	1-5
Extending the Workstation Program through the Use of System Extensions	1-5
The Application Program Services	1-5
Session Information Services	1-6
Keyboard Services	1-6
Window Management Services	1-6
Host Interactive Services	1-6
Presentation Space Services	1-6
3270 Keystroke Emulation Services	1-7
Copy Services	1-7
Translate Service	1-7
Operator Information Area Services	1-7
Multiple DOS Support Services	1-7
The Supervisor Services	1-7
Supervisory Object Services	1-8
Request Services	1-8
Task State Modifier Services	1-8
Semaphore Management Services	1-8
Logical Timer Management Services	1-8
Fixed-Length Queue Management Services	1-8
Interrupt Handler Management Services	1-9
Environment Manager Services	1-9
Using the Application Program Interface	1-9

Overview of API Services

The Application Program Interface (API) is just what the name implies: an interface between an application program and the IBM 3270 Workstation Program. Your application program requests services from the workstation program using the API. The kinds of services that your application can request are grouped into two categories:

- Application program services
- Supervisor services.

The application program services are services that most application programs will use. The supervisor services are services that provide support for applications that use the multitasking capabilities of the IBM workstation program.

Service requests to the workstation program are generated by your application program. The supervisor processes the requests or routes the request to the appropriate API service.

Figure 1-1 illustrates the flow of a request from an application program.

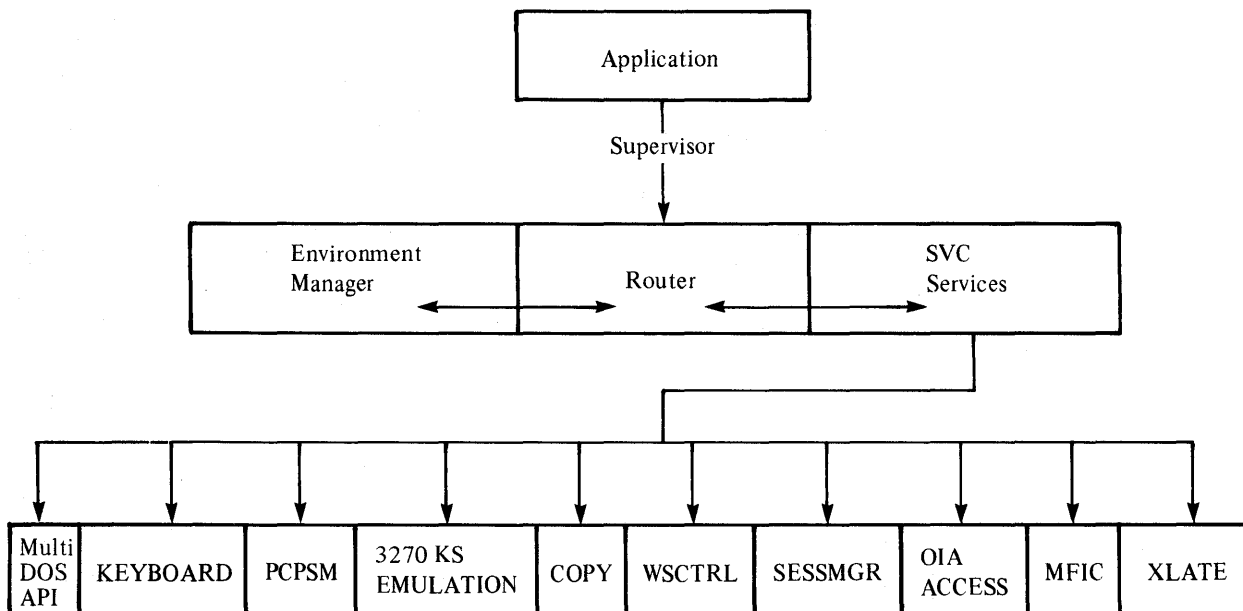


Figure 1-1. Overview of the Application Program Interface

Terms You Need to Know

ASCII	American National Standard Code for Information Interchange. ASCII/ASCII mnemonics can now be sent or received on the Read Input and Write Keystroke API.
EGA	Enhanced Graphics Adapter. For more information, see the <i>Technical Reference Options and Adapters</i> manual.
environment	A contiguous area of storage and a collection of system resources that are managed by an operating system to allow a program or a system extension to run. A program or a system extension is said to “run in an environment.”
stoppable environment	A type of environment that is used for running DOS or personal computer application programs. Stoppable environments can be used for any program that can be removed from the system without causing other programs to fail. That is, programs running in stoppable environments must not offer services to programs running in other environments.
nonstoppable environment	A type of environment used to run system extensions. These system extensions offer services to programs running in other environments and need to be in the system at all times.
Non-3270 PC Hardware	IBM Personal Computer AT® and/or XT system units without the keyboard adapter and 3270 PC display adapter cards installed.
ODSP	Outbound Data Stream Preprocessor. ODSP allows the preprocessing of a 3270 data stream, which can reduce the amount of traffic flowing through a network.
presentation space	An area of storage that represents a logical display. All IBM 3270 host sessions, IBM personal computer sessions, and notepad sessions have a presentation space. The data contained in a presentation space, or a portion of that data, is displayed on the screen when that session’s window is visible on the screen.
session	A connection between your work station and a host computer, personal computer, or notepad.

Examples of Using the API

system extension

Code that runs in a nonstopable environment. It is loaded as part of the workstation program and starts running automatically when the workstation program is IPLed. A system extension may offer services that other programs can use.

window

The portion of the screen through which you view a session's presentation space. A window can be the same size as your full IBM 3270 Personal Computer screen or as small as one character.

XMA card

Expanded memory adapter card. The XMA card is a hardware option card that provides up to 2Mb of additional storage for as many as 6 PC sessions.

Examples of Using the API

The Application Program Interface (API) allows an assembler-language application program in the personal computer session to use a powerful set of services from the workstation program.

Using these services, the application program can:

- Simplify setup and control of multiple host sessions
- Use the work station control functions of the IBM 3270 Personal Computer
- Enhance interaction between the operator and a host
- Extend the workstation program through the use of system extensions.

Simplifying Setup and Control of Multiple Host Sessions

For example, your application program can display a list of screen profiles to the work station operator. When the operator chooses one of the screen profiles, the program can send the necessary logon commands to each of the host sessions defined in the profile. In this way, you can set up the work station for use and eliminate the need for the operator to remember the various logon procedures.

Using the Work Station Control Functions of the IBM 3270 Personal Computer

Your application program can size and move windows, change the foreground and background colors of windows, jump to other windows, enlarge and hide windows, or do any of the other functions that are available in work station control mode.

For example, your application program can translate a single key typed by the operator into a series of work station control commands to set up the IBM 3270 Personal Computer for data entry into a particular window on the screen.

Enhancing Interaction between the Operator and a Host

For example, your application can log onto four host sessions and bring up a different application on each host. Your application can present the work station operator with a menu of functions to perform and then transform the operator's choice into a command to the appropriate host application.

Extending the Workstation Program through the Use of System Extensions

You can write a system extension that is loaded with the workstation program when the system is IPLed. A system extension can perform services for other application programs that you write, and act as a resource manager to allocate and deallocate resources to those application programs.

The Application Program Services

The API provides the following kinds of application program services:

- Session information services
- Keyboard services
- Window management services
- Host interactive services
- Presentation space services
- 3270 keystroke emulation services
- Copy services
- Translate service

The Application Program Services

- Operator information area services
- Multiple DOS support services.

Session Information Services

The session information services allow your application program to query the workstation program to find out what sessions are currently defined, attach and detach from these sessions, and query what the characteristics of these sessions are.

Keyboard Services

The keyboard services allow your application program to read and write keystroke data from a specified session, to disable and enable operator input from the keyboard of a specified session, and to notify the workstation program of the status of your application program's keystroke processing.

Window Management Services

The window management services allow your application program to use the functions of the work station control session of the IBM 3270 Personal Computer. Using these services, your application program can determine the current size, position, or color of a window, and change them if desired. You can jump to specified windows, enlarge or hide windows, or change to a different screen profile. You can add a window, delete a window, or clear the entire screen.

Host Interactive Services

The host interactive services allow communication between a personal computer application program and a host application program using the destination/origin structured field protocol. The host interactive services also allow a personal computer application program to be notified when a host presentation space or operator information area is updated.

Presentation Space Services

The presentation space services allow your application program to create and delete personal computer presentation spaces, to display those personal computer presentation spaces, and to control the position of the cursor in those personal computer presentation spaces.

3270 Keystroke Emulation Services

The 3270 keystroke emulation services enable you to type into a personal computer presentation space as if it were a host presentation space.

Copy Services

The copy services allow your application program to copy data into a personal computer window, as well as copy data from one area of a personal computer window into another area within the same personal computer window. The copy services also allow copying of data from and to host and notepad sessions.

Translate Service

Data that is displayed in host and notepad presentation spaces is represented by numbers called *host/notepad character codes*. Data that is displayed in personal computer presentation spaces is represented by ASCII codes. The translate service allows your application program to translate the data in a buffer from one type of data representation to the other.

Note: You cannot translate graphic characters or programmed symbol set characters.

Operator Information Area Services

The operator information area services allow your application program to determine the current status of a session as shown on the operator information area (OIA).

The contents of the OIA can be determined by reading:

- An image of the OIA
- A bit string that represents a group of related OIA values.

Multiple DOS Support Services

The multiple DOS support services allow your application program to query the size in paragraphs of a specified environment, and to request DOS INT 21H function calls asynchronously.

The Supervisor Services

The API provides the following kinds of supervisor services:

- Supervisory object services
- Request services
- Task state modifier services

The Supervisor Services

- Semaphore management services
- Logical timer management services
- Fixed-length queue management services
- Interrupt handler management services
- Environment manager services.

Supervisory Object Services

The supervisory object services allow your application program to create gates and user exit tables, and create and delete tasks, components, semaphores, and fixed-length queues. The supervisory object services also allow your application program to obtain the numeric ID of a supervisory object by specifying its alphanumeric name, or obtain the alphanumeric name of the supervisory object by specifying its numeric ID.

Request Services

The request services allow tasks and components in your application program to request services of other tasks or components and respond to requests from other tasks.

Task State Modifier Services

The task state modifier services allow your application program to change the dispatch state or priority of a task.

Semaphore Management Services

The semaphore management services allow your application program to control the access to resources and the execution of nonreentrant code.

Logical Timer Management Services

The logical timer management services allow your application program to control time-dependent events through the use of logical timers.

Fixed-Length Queue Management Services

The fixed-length queue management services allow your application program to pass data to other tasks or components, and to receive data from other tasks or components, using the fixed-length queue as a “pipeline” for the data.

Interrupt Handler Management Services

The interrupt handler management services allow environments to share the interrupt vector table on a cooperative basis. On hardware interrupts, a device handler in any environment can receive control.

Environment Manager Services

The environment manager services allow a system extension to act as a resource manager to control the allocation and deallocation of resources to application programs. An application program has the ability to control its own environment using the environment manager services.

Using the Application Program Interface

To use the API, your program must store the required values in the system registers. Services that need more information than can be contained in the system registers use a data area called a *parameter list* to contain the additional information. System registers ES and DI must point to the segment and offset addresses of the parameter list. To request an API service, your application must issue an INT '7A' instruction to signal the workstation program that it has a request to process.

Chapter 2. Programming Considerations

Introduction	2-2
System Information Files	2-2
Program Information Files	2-3
Creating and Modifying Program Information Files (PIFs)	2-4
Restrictions on Running under the Workstation Program	2-5
Guidelines for Running under Multi-DOS	2-6
How Multi-DOS Affects Application Program Performance	2-7
Using the Interrupt X'10' Function	2-8
Tips on Writing Applications to Run in Multi-DOS	2-9
When Personal Computer Sessions Will Be Suspended	2-10
Non-3270 PC Hardware Restrictions	2-11
Determining the Type of PC Your Application Program Is Running On	2-13
Determining the Level of the Control Program or the Workstation Program That Is Loaded	2-13

Introduction

Many application programs written for the IBM PC assume that the PC is dedicated to running a single application program at a time. Without the workstation program, this assumption is valid, since DOS provides a single environment to run programs and does not provide multitasking facilities. Some of these application programs take advantage of facilities available in the PC that are not supported by DOS, or bypass the DOS facilities to run more efficiently.

The Multi-DOS feature of the workstation program provides a set of environments in which personal computer application programs can run, and a set of supervisor services that allow you to write programs that take advantage of this multitasking capability. To preserve the integrity of application programs running at the same time, the workstation program uses program information files (PIFs) and system information files (SIFs) to keep track of the application programs and system extensions that are running in the system.

System Information Files

System information files are used by the workstation program to allocate system resources for system extensions as well as for PC applications. A system extension is a module that you code. It is loaded with, and runs as part of, the workstation program. System extensions must observe all the rules for well-behaved programs that are described in this chapter. To include a system extension in your system, you must answer some questions about that system extension during customization, and you must create a system information file for that system extension. Chapter 24, "Coding System Extensions," describes system extensions and discusses things you need to know to create system information files.

There are several different system information files, which serve different purposes. They are:

- **INDIBM1.SIF** – A special system information file which must be on the IPL disk when you initialize the workstation program. It is created during customization. It tells the workstation program how many system resources to allocate for all the workstation program's system extensions.
- **INDIBM2.SIF** – A special system information file which must be on the IPL disk when you initialize the workstation program. It is shipped with the workstation program diskettes. It tells the workstation program how many system resources to allocate for use by programs running in the PC sessions. If you have configured for Multi-DOS, the workstation program will allocate this many system resources for each PC session. This SIF may be tuned by the user to increase system resources if the system runs short.

- Individual SIFs – Every user system extension must have a system information file to tell the workstation program how many system resources to allocate for its use. See the *IBM 3270 Workstation Program User's Guide and Reference* for more information on system information files.

The rest of this chapter concentrates on program information files.

Program Information Files

Program information files are used by workstation programs configured for Multi-DOS to control the execution of personal computer application programs. The workstation program needs to know whether or not the application program observes the rules for well-behaved programs so that the program does not interfere with other application programs that may be running at the same time.

To obtain optimum performance, any application that is written to use the API services should be well-behaved. If you have customized your system to include the Multi-DOS option, you should create a program information file to tell the workstation program that your application is well-behaved.

Note: If you do not have a PIF, your application will be considered ill-behaved and may suspend when it is not active.

There are different program information files which serve different purposes. They are:

- 3270PC.PIF – This “consolidated PIF” must be on the IPL disk when you initialize the workstation program. It is copied to the IPL disk as part of the customization procedure. It contains program information for many different programs, including the 3270 PC utilities. It is read at IPL time. It is not in the same format as an individual PIF, described below. Program information may be added to the consolidated PIF by using the INDSPIF utility. Refer to the *IBM 3270 Workstation Program User's Guide and Reference* for more information about the consolidated PIF.
- Individual PIFs – Every application COM or EXE file may have a corresponding PIF. An individual PIF may be created by either the INDSPIF utility or the TopView “Create Program Information” utility (if you use TopView). The individual PIFs used by the 3270 workstation program are compatible with the individual PIFs used by TopView. However, TopView PIFs have a different format than 3270 PC PIFs and require that you identify the specific interrupts your program will take over, even if your program takes them over using DOS and BIOS. The workstation program, however, requires you to identify the interrupts your program takes over only if you do not use DOS and BIOS. This means that, if you use your TopView PIF, the workstation program will interpret your program as poorly behaved even if it is well-behaved. See the *IBM 3270 Workstation Program User's Guide and Reference* for more information about PIF files.

Creating and Modifying PIFs

All of these PIFs and SIFs may be created, read, and modified using the INDSPIF utility, except TV.PIF.

When a PC application is run, the workstation program first searches to see if there was a record for it in the consolidated PIF (3270PC.PIF). Failing to find it there, it will look for an individual PIF. If that fails, the workstation program will assume that the answer is "yes" to all PIF questions and that vectors to be swapped are 00-FF.

Since the consolidated PIF and all SIFs are read at IPL time, any changes in these files will not be reflected until the next IPL of the workstation program. A change in an individual PIF will take effect when that program is next loaded.

Creating and Modifying Program Information Files (PIFs)

You use the INDSPIF utility to create and modify PIFs. The INDSPIF utility is provided on your workstation program diskettes.

To use the INDSPIF utility, follow these steps:

1. Determine the name of the EXE or COM file you will use.

Note: If you use a BAT file to run an application, you must create a PIF for the EXE or COM file, not for the BAT file.

2. Decide whether the module is a system extension or a personal computer application program.
 - a. For system extensions, determine how many of each type of control block are needed for the system extension to run.
 - b. For application programs, determine which options apply to the application.
3. At the DOS prompt, enter the INDSPIF command by typing INDSPIF and then pressing the Enter key.
4. On the home panel, you may enter the module name or the path name.
 - a. To create a system information file, press PF2.
 - b. To read an existing system information file, press PF3.
 - c. To create a program information file, press PF4.
 - d. To read an existing program information file, press PF5.
 - e. To read existing program information from the consolidated PIF (3270PC.PIF), press PF6.
 - f. To delete existing program information from the consolidated PIF (3270PC.PIF), press PF7.

Restrictions on Running under the Workstation Program

5. Depending on your choice, you will see either the PIF or the SIF panel.
6. Complete all items on the panel. When you are done, press PF3 to save the PIF or SIF on diskette, or press PF4 to save the program information into the consolidated PIF.
7. You may then press either Home, to return to the Home panel, or Esc, to quit the INDSPIF utility; or you may change any information on the panel and save it again.

Restrictions on Running under the Workstation Program

A number of situations should be avoided if the application is to run on the 3270 PC in a Multi-DOS environment. In general, anything that will cause a contention for nonshareable resources should be avoided. The most common example is reading or writing to fixed memory addresses. An application should only read or write to addresses within its own address space. If an application needs to interact with the hardware, it should do so by issuing BIOS function calls, DOS function calls, or 3270 PC API function calls.

The following are not supported and will cause system failures:

- Programming the Intel 8259 Interrupt Controller chip.
- Taking over interrupts X'50' through X'57' or X'7A'.
- Disabling interrupts for an extended period of time.
- Disabling Direct Memory Access (DMA).
- Jumping to hard-coded addresses in the BIOS. All BIOS calls should be made through the interrupt mechanism.
- Running two workstation program applications on the 8087 or 80287 math co-processor.
- Running an application that installs an interrupt handler that changes its own stack and then enables interrupts. This will produce unpredictable results on systems with an XMA card installed unless you revise the INDIBM2.SIF file. See the *IBM 3270 Workstation Program User's Guide* for more details about updating INDIBM2.SIF.
- Using application hardware interrupt handlers that modify the registers and then CALLFAR the CHAINON address.
- For Uni-DOS on non-3270 PC hardware, an application is assumed to be ill-behaved. The application will be suspended when:
 - It is not the top or active window
 - The WSCtrl key is pressed.

Guidelines for Running under Multi-DOS

Programs running with Multi-DOS should observe the following rules for optimum performance:

- The program should not write to storage reserved by BASIC. See the *Technical Reference* manual for the IBM PC/XT for these locations.

Note: Since BASIC itself writes to these locations, programs written in BASIC violate this rule.

- The program should not write to the PC's interrupt vector table. The DOS function calls or the supervisor interrupt handler management services should be used to set interrupt vectors.
- The program should not write to the display refresh buffer. The BIOS or DOS facilities should be used to write to the screen.
- The program should not reprogram the PC's timer. On non-3270 PC hardware, reprogramming the PC's timer will cause host communication failure.
- The program should not reprogram the PC's speaker.
- The program should not communicate directly with the keyboard. The BIOS or DOS facilities should be used to read data from the keyboard.
- The program should not wait in an idle loop until keys are pressed.
- The program should not directly poll the Asynchronous Communications Adapter (ACA). Instead, you should write an interrupt handler to communicate with the ACA.
- The program should not use graphics modes.
- The program should not read from, or write to, the BIOS data areas.
- For non-3270 PC hardware (XT only), if an application is intercepting keystrokes from a PC session that is running an ill-behaved application which takes over interrupt vector X'9', the first PC session will not receive any keystrokes. It will work only if you are running a well-behaved application.
- ANSI.SYS is not supported under Multi-DOS.
- Virtual Device Interface (VDI) is not supported under Multi-DOS.

If your application program does not follow the rules listed above, the Multi-DOS management portion of the workstation program may take special action when the program is running.

How Multi-DOS Affects Application Program Performance

Following is a list of the special actions the workstation program may take if your application program does not observe some of the rules described:

- If your application program issues a function call to DOS, this request is serialized. You may not be able to jump to another PC window until that request is completed. You can jump to a host or notepad window, however, as long as that window follows the PC window in an alphabetic sequence of short names. For instance, if you are issuing DOS function calls in a PC window with a short name of A, then the host or notepad window you want to jump to should have a short name of B.
- If the program writes to storage reserved by BASIC, the Multi-DOS manager saves an image of this storage before it runs the program. Each time the program is suspended or put into a wait state, the Multi-DOS manager saves the current contents of the storage and swaps them with the original image of storage. Before the program is allowed to run again, the Multi-DOS manager again swaps the original image of storage with the image that was saved when the program was suspended.
- If the program writes to the PC's interrupt vector table, the Multi-DOS manager saves an image of these vectors before it runs the program. Each time the program is suspended or put into a wait state, the Multi-DOS manager saves the current contents of the vectors and swaps them with the original contents of the vectors. Before the program is allowed to run again, the Multi-DOS manager again swaps the original contents of the vectors with the contents that were saved when the program was suspended.

Note: The workstation program fails if any program writes directly to interrupt vector X'7A'.

- If the program writes to the display refresh buffer on 3270 PC hardware, the Multi-DOS manager suspends the program any time you jump to another PC session. For non-3270 PC hardware, the Multi-DOS manager suspends the program when it is not in the active window.
- If the program reprograms the PC's timer, the Multi-DOS manager suspends the program any time it is not in the active window. Jumping to another PC window causes the timer to be reset to its default value.

Note: If you are on non-3270 PC hardware, you should not reprogram the timer.

- If the program reprograms the PC's interrupt controller, the system fails.
- If the program communicates directly with the keyboard (that is, it takes over interrupt X'09' and reads keyboard data directly), the

Using the Interrupt X'10' Function

Multi-DOS manager suspends the program any time it is not in the active window.

- If the program uses graphics modes, the Multi-DOS manager on 3270 PC hardware suspends the program any time you jump to another PC session. For non-3270 PC hardware, the Multi-DOS manager suspends the program when it is not in the active window. Only one PC session at a time may use graphics.

Using the Interrupt X'10' Function

On the PC, there are three ways of doing text output to the display screen. You can use DOS function calls, use BIOS function calls, or write directly to the video refresh buffer. The last method is frequently the only method acceptable where performance is a consideration. Unfortunately, using this method causes your application to be suspended when it is not the active window, in order to support Multi-DOS applications. If you want to run only on the workstation program, there is an alternative. You can do your video output through the API. This restricts your application to the 3270 PC.

There is a way to achieve high-performance screen output on a normal PC or on a 3270 PC with Multi-DOS without incurring performance degradation. This also works under TopView.

To get the address of your presentation space:

1. Load ES:DI with the assumed address of the video buffer (B000:0000).
2. Load AH with X'FE'.
3. Issue an interrupt X'10'.
4. The address of your presentation space will be returned in ES:DI. You should do all your display output to the address returned.

To display your data:

1. Load ES:DI with the address of the first character in the buffer that has been modified since the last display request.
2. Load CX with the number of sequential characters or attribute bytes that have been modified.
3. Load AH with X'FF'.
4. Issue an interrupt X'10'.
5. Data at ES:DI for CX bytes will be displayed in your PC window.

These two functions allow an application to run under TopView and 3270 Workstation Program, as well as on a PC running DOS.

Note: Version 3.0 of the control program and the workstation program do not support TopView.

Tips on Writing Applications to Run in Multi-DOS

Multi-DOS provides you with some powerful tools. For example, an application may download data from a CICS system, format it, and send it in the form of keystrokes to a customer's spreadsheet program.

There are, however, a number of restrictions. These restrictions are required because of the nature of existing PC software, software that was written to run in a single-tasking environment.

You should, for example, be aware that your application can be suspended when it is not the top window. This happens if your PIF has a "yes" for any question, if you have no PIF, or if another PC environment has a PIF that indicates that it swaps vectors in the range X'00' through X'7F'.

This can cause problems if, for example, you lock WSCTRL, jump to another PC session, jump back, and release the WSCTRL lock. If the other session swaps vectors in the range X'00' through X'7F', your program is suspended before it can jump back and release the lock. Since you are holding the lock, no keyboard activity can occur. Since you are suspended, you cannot release the lock. Since you are not the active session, you cannot be resumed. This situation, known as *circular waiting*, is a classic deadlock. It effectively requires the user to power off and re-IPL the system in order to regain control.

There are two ways to solve this problem. The first is to use the Query PC Session PIF Information service to determine whether you will be suspended. If you will, then avoid the above sequence of calls. The second is to claim a code serialization semaphore, issue the calls with a wait type of "no wait," and then release the semaphore. Since you will not be suspended while holding the semaphore, you will be able to issue the calls, release the semaphore, and survive. The calls will occur asynchronously.

You must be extremely careful with code serialization semaphores, however. If you are holding a code serialization semaphore across any segment of code that could cause a wait condition (for example, claim semaphore, get keyboard input, do something, release semaphore), you could also create a deadlock. If, while you are holding the semaphore, the user presses the JUMP key, this could happen:

- The system tries to suspend you.
- Since you hold a code serialization semaphore, the system waits until you release it to suspend you.
- You are waiting on keyboard input.
- The user cannot enter data into your session, because you are not the active session.

The result is as in the previous example, a circular-wait condition. The system is deadlocked, requiring a power off/on to restart.

Tips on Writing Applications to Run in Multi-DOS

If you are writing programs that create tasks and then exit back to DOS, make sure that you use a Terminate But Stay Resident call. If you do not, then the space in which your task is running will be overlaid by the next application to run.

When Personal Computer Sessions Will Be Suspended

The following grid indicates when a personal computer session is suspended. The foreground task is never suspended. Those boxes marked with an "X" indicate when the background session(s) is suspended.

The foreground session is:	The background session is:		
	Well-behaved	Moderately well-behaved	Poorly behaved
Well-behaved		X	X
Moderately well-behaved		X	X
Poorly behaved	X	X	X

Definitions:

Foreground: If the active window (that is, the one with double borders) is a PC window, then it is the foreground session. If a non-PC window (host or notepad) is active, then the foreground session depends on the type of hardware you are using:

- 3270 PC hardware: The PC window that was most recently active is considered the foreground session.
- Non-3270 PC hardware: No PC window is considered to be the foreground session. All PC windows are background sessions and will be suspended if they either write directly to the screen or are poorly behaved.

Background: Any window that is not the foreground window.

Well-behaved: A personal computer session running a program that has a PIF in which every answer was "no"; that is, the personal computer application does nothing bad. This might also be a personal computer session that is running COMMAND.COM (for example, doing a DIR).

Moderately well-behaved: A personal computer session running a program that has a PIF in which any answer was "yes," but for which the space for "vectors swapped" did not include vectors in the range X'00' through X'7F'.

Poorly behaved: A personal computer session running a program that has no PIF, or for which the space for "vectors swapped" did include vectors in the range X'00' through X'7F'.

Notes:

- 1. If any program terminates and stays resident, then the workstation program treats that personal computer session as though the program were still running (that is, the program's PIF remains in effect when determining whether to suspend the session). For example, suppose a program takes over an interrupt in the range of X'80' through X'FF', then terminates and stays resident. If another program is loaded, that program is still suspended when it is not in the foreground window. When the session is rebooted, the session is considered well-behaved again.*
- 2. If a program loads and runs another program using DOS function X'4B', the program that was loaded inherits the PIF characteristics of the loading program (for example, menu programs).*

Non-3270 PC Hardware Restrictions

The following restrictions apply to non-3270 PC hardware (XT and AT). Failure to follow these guidelines on the use of non-3270 PC hardware could result in system failure.

- An application is assumed to be ill-behaved when running in Uni-DOS on non-3270 PC hardware.
- An ill-behaved application will be suspended when:
 - It is not the top or active window
 - The WSCTRL key is pressed
 - It uses the Input Control API to connect to the WSCTRL keyboard. (Note that this will affect all applications that will attempt to send Jump, ChgSc, and Enlarge keystrokes, as well as any other keystrokes that perform functions while in WSCTRL.)
 - On IPL if your PC session is not in the top window. Any program you start in your PC window will not be completed until you make it the active session (for example, your AUTOEXEC.BAT file).
- Ill-behaved applications will run only in an active session. If an ill-behaved application exits and stays resident in the active session, then any other application you run in that session will be seen as ill-behaved and will never run in the background.
- On non-3270 PC hardware, ill-behaved applications will be displayed full-screen when they are made active, even if they are sized. Even if you sized your windows using the API, they may be forced to full-screen and appear enlarged when active under the following conditions:
 - Your application uses graphics mode
 - Your application uses 40-column mode
 - Your application writes directly to the screen

Non-3270 PC Hardware Restrictions

- Your application runs in Uni-DOS.
- When an ill-behaved application uses the work station control API, the redraw screen and redraw window functions will not affect what is seen on the screen. Any changes made while the application is connected to the API will not be seen on the screen until the application disconnects from the API.
- When using work station control API, the WS Ctrl OIA will not be displayed under the following conditions:
 - Your application uses graphics mode
 - Your application uses 40-column mode
 - Your application writes directly to the screen
 - Your application runs in Uni-DOS.
- If you are running an ill-behaved application in a PC session, the shift state of that session may not remain as you originally set it after jumping to other windows and back again. For instance, if you have set Caps Lock on in this PC session and jump to another window, your session may be in lowercase mode when you jump back to that window. Also, applications that write directly to the display adapter registers may not be restored properly after jumping to another window and back again.
- Input Control API is not supported for sessions running ill-behaved applications that read port 60 directly.
- Do not run PC applications that reprogram the timer. This could cause host communication failure.
- An ill-behaved application that receives keystrokes from any other session will not work, because the application will be suspended when it is not the active session, so the keystrokes that it would normally be receiving will be queued up until the application becomes active. An ill-behaved application that sends keystrokes will work as long as its session remains active.
- Use the BIOS INT 10, “Set Color Palette” call, to change the palette colors. Otherwise, the colors will be changed in all sessions.
- If you load an alternate character generator into one session, then all your sessions will use this alternate character set.

Determining the Type of PC Your Application Program Is Running On

An application may perform the following test to determine whether it is executing on a 3270 PC.

1. Perform a function call (INT X'10') to the BIOS Video Routine with the following parameters:
 - AH register = X'30'
 - AL register = X'00'
 - CX register = X'00'
 - DX register = X'00'
2. If the CX and DX registers are still zero (0) on return, the machine is not a 3270 PC or a non-3270 PC with the 3270 Workstation Program loaded. If the CX and/or DX registers are not zero, the following test should be made to determine whether the machine is a 3270 PC or a non-3270 PC with the 3270 Workstation Program loaded:
 - a. Read the byte at CX:DX + 2.
 - b. If the value of this byte is X'FF', the machine is a PC without the 3270 Workstation Program loaded. If the value of this byte is not X'FF', the machine is a 3270 PC or a non-3270 PC with the 3270 Workstation Program loaded.

Determining the Level of the Control Program or the Workstation Program That Is Loaded

If the machine is a 3270 PC or a non-3270 PC, an application may determine whether the 3270 PC Control Program or the 3270 Workstation Program is loaded in memory, and if so what level of the program is currently resident, by performing the following tests:

If CX:DX is not zero and not X'C040:0220', read the two-byte location at address CX:DX + 8.

1. If the contents of the two-byte location at address CX:DX + 8 are X'0000', the following test should be performed to determine whether a pre-Version 2.0 level of the control program is resident:
 - a. Read the BIOS high memory limit at address X'0413'.
 - b. Read the 8 bytes of data located 36 bytes beyond the BIOS high memory word obtained above.

In rare instances, BIOS isolates bad high memory locations by placing the high address limit below these locations. If the control program or the workstation program is not loaded in memory when

Determining the Program Level

this test is performed, the application may inadvertently address these bad memory locations, which will result in a parity error. This is a nonrecoverable condition.

- c. If the data at this memory location is X'2322272031AFA210', then the control program or workstation program is resident in memory.
- d. If the control program is resident, the application can determine whether Version 1.0/1.1 or Version 1.2/1.21/1.22 is resident as follows:
 - 1) Read the 8 bytes of data located 16 bytes beyond the BIOS high memory limit.
 - 2) If the data found at this location is X'353636392D303032', the resident control program is Version 1.2 or 1.21 or 1.22. Otherwise, the resident control program is Version 1.0 or 1.1.
2. If the contents of the two-byte location at address CX:DX + 8 are not zero (Control Program Version 2.0 and later or the workstation program), it contains the segment address (at offset zero) where the application will find a two-byte field containing the identifier of the control program or workstation program in hexadecimal notation (0200 for Control Program 2.0, 0210 for Control Program 2.1, 0300 for Control Program 3.0, and 0400 for Workstation Program 1.0). Immediately following the identifier is a single byte indicating the specific control program or workstation program installed:
 - a. X'00' - Standard Control Program or Workstation Program
 - b. X'01' to X'FF' - Reserved

Following the identifier is a 27-byte field containing an ASCII text string that identifies the type of control program or workstation program installed (for example, "IBM 3270 PC CONTROL PROGRAM").

Part 2. Application Program Services

This part contains information about application program services provided by the Application Programming Interface.

- Chapter 3, “Coding Supervisor Services,” describes the supervisor services that your application program needs to use the rest of the services described in this part of the manual.
- Chapter 4, “Coding Session Information Service Requests,” describes the session information services that your application program can use.
- Chapter 5, “Coding Keyboard Service Requests,” describes the keyboard services that your application program can use.
- Chapter 6, “Coding Window Management Service Requests,” describes the window management services that your application program can use.
- Chapter 7, “Coding Host Interactive Service Requests,” describes the host interactive services that your application program can use.
- Chapter 8, “Coding Presentation Space Service Requests,” describes the presentation space services that your application program can use.
- Chapter 9, “Coding 3270 Keystroke Emulation Service Requests,” describes the keystroke emulator services that your application program can use.
- Chapter 10, “Coding Copy Service Requests,” describes the copy services that your application program can use.
- Chapter 11, “Coding Translate Service Requests,” describes the translate service that your application program can use.
- Chapter 12, “Coding Operator Information Area Service Requests,” describes the operator information area services that your application program can use.
- Chapter 13, “Coding Multi-DOS Support Service Requests,” describes the multiple DOS support services that your application program can use to query the size in paragraphs of a specified environment, and to request DOS INT 21H function calls asynchronously.

Conventions Used in the API Service Descriptions

The following conventions are used in the descriptions of the API services:

- Hexadecimal numbers are represented in the notation X'nn' for byte values and X'nnnn' for word values.
- Offsets into data structures used by the API services are given as decimal numbers.
- Bits within a byte are numbered with the high-order (leftmost) bit as bit 0 and the low-order (rightmost) bit as bit 7, as follows:

0	1	2	3	4	5	6	7

This order of bit numbering follows the IBM 360/370 convention and is the reverse of the Intel 8088 bit-numbering convention.

Chapter 3. Coding Supervisor Services

Introduction	3-2
Obtaining the Gate Name for the Services Your Application Program Will Use	3-2
Obtaining the Results of Services You Have Requested Asynchronously	3-3
Creating Fixed-Length Queue Entries	3-3
Obtaining Data from a Fixed-Length Queue	3-3
Deleting Fixed-Length Queues	3-4
Requesting the Supervisor Services	3-4
Supervisor Service X'81': Name Resolution	3-5
Supervisor Service X'83': Get Request Completion	3-7
Supervisor Service X'04': Create Fixed-Length Queue Entry	3-9
Supervisor Service X'13': Dequeue Data	3-12
Supervisor Service X'06': Delete Entry	3-15

Introduction

This chapter describes how to code requests for supervisor services provided by the API that are needed for the rest of the application program services described in this part of the manual. These supervisor services are a small subset of the supervisor services that are described in Part 3.

You use the services in this chapter to obtain the gate name for the services your application program will use, to obtain the results of services that you have requested asynchronously, and to create, obtain data from, and delete fixed-length queues.

Obtaining the Gate Name for the Services Your Application Program Will Use

A gate is a grouping of services (or requests, as is the case with the multiple DOS support services) that perform a common function. Each gate is assigned a name when the gate is created. The workstation program provides the following groups of services/requests, or gates:

Services or Requests	Gate Name
Session information services	SESSMGR
Keyboard services	KEYBOARD
Window management services	WSCTRL
Host interactive services	MFIC
Presentation space services	PCPSM
3270 keystroke emulation services	3270EML
Copy services	COPY
Translate service	XLATE
Operator information area services	OIAM
Multiple DOS support services	
Query environment size	INDJQRY
Asynchronous DOS function	INDJASY
Get storage	MEMORY

Each group of services is identified to the workstation program by a 16-bit number, called the *gate ID*. Before your application program can use any of the services in a particular gate, you must obtain the gate ID that the workstation program assigns to the gate. You do this by requesting the Name Resolution service, specifying an alphanumeric gate name on the request. The workstation program returns the gate ID to your application program. You should save the gate ID in a variable, because you must provide it as input when you request any of the services in the gate.

Obtaining the Results of Services You Have Requested Asynchronously

Most of the application program interface services described in this part of the manual are processed synchronously by the workstation program. That is, when control is returned to your application program, the registers and the parameter list contain the values assigned to them on request completion. However, you can specify asynchronous processing of the following services:

- Keyboard service X'04': Write Keystroke
- Host interactive service X'01': Connect to Host Session
- Host interactive service X'02': Disconnect from Host Session
- Host interactive service X'03': Read Structured Field
- Host interactive service X'04': Write Structured Field
- Host interactive service X'05': Define Buffer

When you specify asynchronous processing of these requests, control can be returned to your application program before the workstation program has completed the request. You must use the Get Request Completion service to obtain the values in the parameter list when the request is completed.

You can also use the Get Request Completion service for request processing of task or components. Request processing is described in Part 3.

Creating Fixed-Length Queue Entries

For some of the application program interfaces, your application program must create a fixed-length queue.

You create a fixed-length queue by requesting the Create Fixed-Length Queue Entry service. The workstation program uses fixed-length queues to notify your application program about events that have occurred and that affect the operation of your program, and also to pass keystrokes that were typed on the keyboard to your application program for processing.

The space for the fixed-length queue must reside in your application program's program space. The first 10 bytes of the queue are reserved for use by the workstation program.

You can write code that uses fixed-length queues to pass data between programs running in the 3270 Personal Computer. The services that you use to do this are described in Part 3.

Obtaining Data from a Fixed-Length Queue

As described above, the workstation program uses fixed-length queues to notify your application program about events that have occurred and that affect the operation of your program. You obtain the data on a fixed-length queue by requesting the Dequeue Data service. The Dequeue Data service returns the specified number of bytes of information to your application program.

Introduction

You can write code that uses fixed-length queues to pass data between programs running in the 3270 Personal Computer. The services that you use to do this are described in Part 3.

Deleting Fixed-Length Queues

When your application program no longer needs a fixed-length queue, it must tell the workstation program that the entry it has created for the fixed-length queue is no longer needed. You tell the workstation program to delete its entry for a fixed-length queue by requesting the Delete Entry service.

The Delete Entry service can also be used to tell the workstation program to delete its entries for other supervisory objects. Information on these other supervisory objects is in Part 3.

Note that the workstation program will not allow you to delete objects on which requests are still pending. For example, if a task has done a dequeue with a "wait for data" option set, the queue it is waiting on cannot be deleted until that request has been satisfied (that is, data is enqueued to that queue and returned to the dequeuing task).

The supervisor services that you will need to code requests for the rest of the services in this part of the manual are:

- **Name Resolution:** Use this service to resolve the application interface gate name to its numeric gate ID.
- **Get Request Completion:** Use this service to obtain the results of services requested asynchronously.
- **Create Fixed-Length Queue Entry:** Use this service to create an entry in the SVC table for a fixed-length queue.
- **Dequeue Data:** Use this service to dequeue data from the specified fixed-length queue.
- **Delete Entry:** Use this service to delete the entry in the SVC table for the specified fixed-length queue.

Requesting the Supervisor Services

To request any of the supervisor services, load the registers and the parameter list with the proper values, and use the INT 7AH instruction to signal the workstation program that it has a request to process.

Supervisor Service X'81': Name Resolution

Use this service to resolve the application interface gate name to its numeric gate ID. You can also use this service for name resolution of other supervisory objects. Refer to Chapter 15, "Coding Supervisory Object Services," for information about the additional uses of the Name Resolution service.

Register Values

On Request

AH = X'81'
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

BH = X'07'
CH = X'12' or X'13'
CL = Return code
DX = Gate ID

The contents of registers AX, BL, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The ES register contains the segment address of the parameter list.
- The DI register contains the offset address of the parameter list.

Completion Registers:

- The DX register contains the resolved name, which is the numeric representation of the alphanumeric ASCII gate name.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0 - 7	8 bytes	Gate name	Unchanged

Name Resolution

Parameter Definitions

Request Parameters:

- The gate name must be ASCII characters, and must be padded to the right with blanks if it is less than eight characters long. The gate name to use for a group of services is described in the introduction to each chapter in this part of the manual.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion of the request
X'0F'	Invalid environment access
X'2E'	Name not found

Coding Example

```
;
; PARAMETER LIST FOR NAME RESOLUTION
;
SERVNAME DB 'KEYBOARD'
.
.
.

;
; INITIALIZE REGISTERS FOR NAME RESOLUTION
;
MOV AH,81H ; AH = X'81'
MOV DI,SEG SERVNAME ; SEGMENT ADDRESS OF
; THE PARAMETER LIST
MOV ES,DI ; ES = SEGMENT ADDRESS OF
; THE PARAMETER LIST
MOV DI,OFFSET SERVNAME ; DI = OFFSET ADDRESS OF
; THE PARAMETER LIST

;
; SIGNAL WORKSTATION PROGRAM FOR NAME RESOLUTION SERVICE
;
INT 7AH
.
.
.
```

Supervisor Service X'83': Get Request Completion

Use this service to obtain the results of the following services when they are requested with asynchronous processing specified:

- Keyboard service X'04': Write Keystroke
- Host interactive service X'01': Connect to Host Session
- Host interactive service X'02': Disconnect from Host Session
- Host interactive service X'03': Read Structured Field
- Host interactive service X'04': Write Structured Field
- Host interactive service X'05': Define Buffer

Register Values

On Request

AH = X'83'
BL = X'00' or X'40'

On Completion

AX = Request ID
BL = X'00' or X'40'
CH = X'12'
CL = Return code
ES = Segment address of the parameter list
DI = Offset address of the parameter list

The contents of registers BH and DX are unpredictable.

Set the BL register to:

- X'00' if you want to check whether results are available (asynchronous processing)
- X'40' if you want to wait until results are available (synchronous processing).

Register Definitions

Completion Registers:

- The AX register contains the request ID of the service whose results were obtained. This is the same value that was returned in the AX register of a previously requested service. You can determine which previously requested service the results are for by matching the request IDs.
- The ES and DI registers are set to the segment and offset addresses of the service's parameter list, which contains the results of the service.

Get Request Completion

Parameter List Format

See the description of the requested service for the format of the parameter list.

Return Codes

Code	Meaning
X'00'	Successful completion of the request.
X'09'	No results are available.

Additional return codes pertaining to the requested service may appear. Refer to the description of the requested service for a listing of the possible return codes.

Coding Example

```
.  
. .  
;  
; INITIALIZE REGISTERS FOR GET REQUEST COMPLETION  
;  
    MOV    AH,83H  
    MOV    BL,40H           ; WAIT TYPE = COMPLETION QUEUE  
  
; SIGNAL WORKSTATION PROGRAM FOR GET REQUEST COMPLETION SERVICE  
INT     7AH  
. .  
.
```

Supervisor Service X'04': Create Fixed-Length Queue Entry

Use this service to create an entry in the SVC table for a fixed-length queue.

Register Values

On Request

AH = X'04'
BL = 0 = no name / 1 = name
CX = Queue length
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12' or X'13'
CL = Return code
DX = Queue ID

The contents of registers AX, BX, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The BL register indicates whether the queue has a name associated with it.

Possible values for the BL register are :

0 = the queue has no name
1 = the queue's name is in the parameter list

- The CX register contains the number of bytes your application program has reserved for the fixed-length queue. The queue must be greater than 10 bytes long, because the first 10 bytes of the queue are reserved for use by the workstation program.
- The ES register contains the segment address of the parameter list.
- The DI register contains the offset address of the parameter list.

Completion Registers:

- The DX register contains the ID of the fixed-length queue.

Create Fixed-Length Queue Entry

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 word	Offset address of the queue	Unchanged
2	1 word	Segment address of the queue	Unchanged
4 – 11	8 bytes	Queue name	Unchanged

Parameter Definitions

Request Parameters:

- The queue name is an optional parameter and is needed only if the BL register is set to 1 on request. The queue name can be a maximum of eight ASCII characters and should be padded to the right with blanks if necessary.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion of the request.
X'01'	Queue name already exists.
X'02'	SVC table full.
X'03'	Name table full.
X'41'	Invalid queue length.

Usage Notes

- The fixed-length queue resides in the requester's environment.

Coding Example

```
;
; DEFINE PARAMETER LIST FOR CREATE QUEUE
;
CQQOFFS  DW    0
CQSEGM   DW    0
CQQNAME  DB    8 DUP(' ')

      .
      .
      .

;
; INITIALIZE FIRST 2 ENTRIES OF PARAMETER LIST
;
      MOV    CQQOFFS,OFFSET Q ; OFFSET OF QUEUE
      MOV    CQSEGM,SEG Q     ; SEGMENT OF QUEUE

;
; THE USER HAS A QUEUE NAME
;
      MOV    BL,01H           ; INDICATE A QNAME IS SPECIFIED
      CLD                    ; BEGIN MOVING QNAME TO THE PARAM LIST
      MOV    CX,4             ; QNAME IS FOUR WORDS LONG
      MOV    SI,OFFSET QNAME  ; SOURCE OFFSET OF QUEUE
      MOV    DI,OFFSET CQQNAME; DESTINATION OFFSET IS CQQNAME
      REP    MOVSW            ; MOVE QNAME TO PARAMETER LIST

;
; INITIALIZE REGISTERS FOR CREATE QUEUE
;
      MOV    AH,04H
      MOV    CX,50             ; CX = NUMBER OF BYTES FOR QUEUE
      MOV    DI,SEG CQQOFFS    ; ADDRESSABILITY TO
      MOV    ES,DI             ; PARAMETER LIST
      MOV    DI,OFFSET CQQOFFS; USING ES:DI

;
; SIGNAL WORKSTATION PROGRAM FOR CREATE QUEUE SERVICE
;
      INT    7AH
      .
      .
      .
```


Supervisor Service X'13': Dequeue Data

Use this service to dequeue data from the specified fixed-length queue.

Register Values

On Request

AH = X'13'
BL = Wait type
CX = Number of bytes
DX = Fixed-length queue ID
ES = Segment address of data
DI = Offset address of data

On Completion

CH = X'12' or X'13'
CL = Return code
DX = Number of bytes

The contents of registers
AX, BX, ES, and DI are
unpredictable.

Register Definitions

Request Registers:

- The BL register specifies the type of wait state your application program will go into until the request is completed. The type of wait is specified through a bit mask. When more than one type of wait is specified, the wait state ends when any one of the conditions is satisfied. The bits in the wait type are as follows:

0	1	2	3	4	5	6	7
Request queue	Comp queue	Comp signal	Sema-phore	Timer	Signal	Data	Reserved

- If bit 0 is set to 1, your application program will wait until a request queue element is in its request queue. If an RQE is already in its request queue, the application stays dispatchable.
- If bit 1 is set to 1, your application program will wait until a request queue element is in its completion queue. If an RQE is already in its completion queue, the application stays dispatchable.
- If bit 2 is set to 1, your application program will wait until it receives a 'completion' signal.
- If bit 3 is set to 1, your application program will wait until it receives a 'got semaphore' signal.

- If bit 4 is set to 1, your application program will wait until it receives a ‘timer tick’ signal.
- If bit 5 is set to 1, your application program will wait until it receives a ‘generic’ signal.
- If bit 6 is set to 1, your application program will wait until it receives a ‘data available’ signal.
- Bit 7 is reserved and must be set to 0.

Note: A wait type of “no wait” is specified by setting the wait type to X‘00’.

- The CX register contains the number of bytes to be dequeued from the specified fixed-length queue. When you are dequeuing information that the workstation program has placed on a fixed-length queue, this number should be X‘04’.
- The DX register contains the ID of the fixed-length queue.
- The ES and DI registers point to the beginning of a data area provided by your application to contain the dequeued data.

Completion Registers:

- The BL register indicates the type of wait condition that was satisfied to return control to the requesting task. The return type is specified through a bit mask. The bits in the return type have the same meaning as the bits in the wait type.
- The DX register contains the number of bytes remaining on the fixed-length queue.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X‘12’ or X‘13’ (found in the CH register). The error codes that can be received for this service are:

Code	Meaning
X‘00’	Successful completion of the request.
X‘05’	Invalid index specified.
X‘09’	The fixed-length queue is empty.
X‘13’	Number of bytes requested is too large.
X‘37’	Not your turn to dequeue.

Dequeue Data

Usage Notes

- Programs running in stoppable environments cannot dequeue data from fixed-length queues in other stoppable environments.
- If you want to get control back as soon as data appears on your queue, use a wait type of “wait for data available.”
- If two or more tasks request the Dequeue Data service for the same fixed-length queue, the supervisor processes the requests in first-in first-out (FIFO) order.
- If you use a wait type other than “wait for data available,” and another request for data from the queue was received before your request, you will receive a return code indicating that it is not your turn to dequeue.

Coding Example

```
;
; DATA AREA FOR DEQUEUE
;
DATAAREA DB 4 DUP(0)          ; DATA AREA TO RECEIVE 4 BYTES FROM THE
                                ; DEQUEUE

.
.
.
;
; INITIALIZE REGISTERS FOR DEQUEUE
;
MOV     AH,13H
MOV     BL,02H                ; WAIT UNTIL INFORMATION IS AVAILABLE
MOV     CX,0004H              ; DEQUEUE 4 BYTES
MOV     DX,QUEUEID            ; FIXED-LENGTH QUEUE ID IN DX
MOV     DI,SEG DQSESSID       ; SEGMENT ADDRESS OF DATA AREA IN ES
MOV     ES,DI
MOV     DI,OFFSET DQSESSID    ; OFFSET ADDRESS OF DATA AREA IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR DEQUEUE SERVICE
;
INT     7AH
.
.
.
```

Supervisor Service X'06': Delete Entry

Use this service to delete the entry in the SVC table representing the specified fixed-length queue.

Register Values

On Request

AH = X'06'
DX = Fixed-length queue ID

On Completion

CH = X'12' or X'13'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The DX register contains the ID of the supervisory object to be deleted from the SVC table.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion of the request.
X'05'	Invalid supervisory object ID.
X'0F'	Specified object is in an inaccessible environment.
X'30'	Cannot delete a task, fixed-length queue, or semaphore that has pending requests.
X'31'	Cannot delete a task that has timers.
X'3F'	Cannot delete a service entry in a gate.

Usage Notes

- If requests are outstanding for the fixed-length queue entry, then the entry will not be removed and an error indicator will be returned.
- An application program running in a stoppable environment can only delete entries in its own environment.
- Part 3 of this manual describes additional supervisory objects the Delete Entry service can delete that you can create in application programs.

Delete Entry

Coding Example

```
      .  
      .  
      .  
;   
; INITIALIZE REGISTERS FOR DELETE AN ENTRY REQUEST  
;  
      MOV    AH,06H  
      MOV    DX,QUE$ID      ; DX = FIXED LENGTH QUEUE ID  
;  
; SIGNAL WORKSTATION PROGRAM FOR DELETE AN ENTRY SERVICE  
;  
      INT    7AH  
      .  
      .  
      .
```

Chapter 4. Coding Session Information Service Requests

Introduction	4-2
Requesting the Session Information Services	4-3
Return Codes for the Session Information Services	4-4
Session Information Service X'01': Query Session ID	4-5
Session Information Service X'02': Query Session Parameters	4-10
Session Information Service X'04': Detach Session ID	4-14
Session Information Service X'05': Attach Session ID	4-17
Session Information Service X'06': Query Windows in Environment ..	4-20
Session Information Service X'07': Query Environment of Window ...	4-23
Session Information Service X'08': Query PC Session Program Information File (PIF) Information	4-26
Session Information Service X'0A': Query Base Window	4-30
Session Information Service X'0B': Query Session Cursor	4-33

Introduction

This chapter describes how to code requests for the session information services provided by the API.

The session information services allow your application program to query the workstation program to find out what sessions are currently defined, and what the characteristics of those sessions are. The session information services are:

- **Query Session ID Service:** Use this service to obtain the ID of the session you specify. You can specify a particular session by its short or long name, or ask for the IDs of all sessions of a particular session type. The session types that are supported on the 3270 Personal Computer are:

- The work station control session
- Distributed function terminal (DFT) host session
- Central unit terminal (CUT) host session
- Notepad session
- Personal computer session

A session ID is required as input for most of the remaining API services. Your application program must request the Query Session ID service for all sessions it will be referencing in API service requests. Typically, obtaining the necessary session IDs is included in the initialization portion of an application program.

- **Query Session Parameters Service:** Use this service to obtain the session characteristics of a particular session. The characteristics obtained by this service are:
 - The session type
 - Whether the session has base or extended attribute support (host session only)
 - Whether the session supports programmed symbols (host session only)
 - The number of rows and columns in the session's presentation space
 - The segment and offset address of the session's presentation space
- **Attach and Detach Session ID Services:** Use these services to attach to and detach from a session. These services should be used by system extensions that provide some service to a session. Attaching to the session assures that the session will not be deleted until you detach from it.
- **Query Windows in Environment Service:** Use this service to obtain a list of the windows that are defined within a specified environment.

- **Query Environment of Window Service:** Use this service to obtain the environment ID of a specified window.
- **Query PC Session PIF Information:** Use this service to obtain a flag that indicates the answers to questions about the Program Information File. This is for the application program running in the specified personal computer session.
- **Query Base Window Service:** Use this service to obtain the session ID and short name of the base window of a particular environment. The base window is the window that was defined at configuration time for the specified environment. You can use this service to obtain the session ID for the session your application program is currently running in.
- **Query Session Cursor Service:** Use this service to obtain the cursor type and the row and column addresses of the specified session's cursor on the session's presentation space. The possible cursor types are as follows:
 - Underscore cursor (blinking or not blinking)

An underscore cursor appears as: _
 - Box cursor (blinking or not blinking)

A box cursor appears as: ■
 - Inhibited cursor

An inhibited cursor is not displayed. When the cursor position changes, the text in the window is not moved to keep the cursor inside the window borders.
 - Inhibited cursor with autoscroll

An inhibited cursor with autoscroll is not displayed. When the cursor position changes, the text in the window is moved to keep the cursor inside the window borders.

Requesting the Session Information Services

To request any of the session information services, load the registers and the parameter list with the proper values, and use the INT 7AH instruction to signal the workstation program that it has a request to process.

Note: Before your application can request the session information services, it must request the Name Resolution service, using 'SESSMGR' as the gate name in the parameter list. (Remember that the gate name must be padded to the right with blanks if it is less than eight characters.)

Return Codes for the Session Information Services

Each session information service has two return codes associated with it, a system return code and a session management return code. Both types of return codes are 2-byte values made up of a function ID and an error number. The function ID indicates the portion of the workstation program in which the error occurred. The error number indicates the specific type of error that has occurred. An error number of X'00' always indicates a successful acceptance or completion of the request.

- System return codes:

After your application has requested a session information service, the CH and CL registers contain a return code generated by the request processing portion of the workstation program. The function ID is in the CH register, and the error number is in the CL register. System return codes use a function ID of X'12'. The error codes that can appear are:

Code	Meaning
X'00'	Request accepted.
X'05'	Invalid index specified.
X'07'	Invalid reply specified.
X'08'	Invalid wait type specified.
X'0B'	RQE pool depleted.
X'0F'	Invalid environment access.
X'34'	Invalid gate entry.

These system return codes apply to all session information services.

- Session information services return codes:

After a requested session information service is completed, bytes 0 and 1 of the parameter list contain a return code generated by the session management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Session information return codes use a function ID of X'6B'. The error numbers that can appear are specific to the service that was requested and are included in the descriptions of each service.

See Appendix H, "Return Codes," for more information.

Session Information Service X'01': Query Session ID

Use this service to obtain the session ID of the session you specify. You can specify a session by its short or long name, or ask for the IDs of all sessions of a particular type.

Register Values

On Request

AH = X'09'
AL = X'01'
BH = X'80'
BL = X'20'
CX = X'0000'
DX = Resolved value for SESSMGR
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'6B')
2	1 byte	Option code	Unchanged
3	1 byte	Data code	Unchanged
4	1 word	Offset address of name array	Unchanged
6	1 word	Segment address of name array	Unchanged
8 – 15	8 bytes	Session long name	Unchanged

Parameter Definitions

Request Parameters:

- **To obtain the session ID of a session whose short name you specify:**
 - The option code must be X'01'.
 - The data code must be the 1-character (A – Z) ASCII short name of the session.

The session long name is ignored.

Query Session ID

- **To obtain the session ID of a session whose long name you specify:**
 - The option code must be X'01'.
 - The data code must be X'00'.
 - Bytes 8 through 15 must contain the long name of the session, padded to the right with blanks if it is less than eight characters long.
- **To obtain the session ID for all sessions of a specific session type:**
 - The option code must be X'00'.
 - The data code must be:
 - X'01' for a work station control session.
 - X'02' for a DFT host session.
 - X'03' for a CUT host session.
 - X'04' for a notepad session.
 - X'05' for a personal computer session.

The session long name is ignored.

Name Array Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Name array length	Unchanged
1	1 byte	Reserved	Number of matching sessions
2 *	1 byte	Reserved	Short name of session 1
3 *	1 byte	Reserved	Type of session 1
4 *	1 byte	Reserved	Session ID of session 1
5 *	1 byte	Reserved	Reserved
6 – 13 *	8 bytes	Reserved	Long name of session 1
⋮			
and so on for all possible matching sessions.			
* The format of the name array offsets 2 through 13 must be repeated for as many possible sessions as can match the Query Session ID service request.			

Name Array Parameter Definitions

Request Parameters:

- The name array length is the number of bytes in the name array. The name array must be at least 14 bytes long and no greater than 170 bytes long. In addition, if you are coding this service to obtain the session ID for all sessions of a specific type, the name array must be large enough for all the possible matching sessions that can be returned for the session type.

Completion Parameters:

- The number of matching sessions contains the number of sessions that matched the request.
- The session short name is the 1-character uppercase ASCII alphabetic name of the session (A through Z).
- The session type is one of the following:
 - X'01' for a work station control session.
 - X'02' for a DFT host session.
 - X'03' for a CUT host session.
 - X'04' for a notepad session.
 - X'05' for a personal computer session.
- The session ID is the ID that the workstation program uses to identify the session. You use the session ID to specify this session in any following API service requests.
- The session long name is the 8-character ASCII name assigned to the session when it was configured. The session long name is padded to the right with blanks if necessary.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Session Information Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the session management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Session information services return codes use a function ID of X'6B'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'03'	The specified long name is invalid.
X'09'	The session type is invalid.
X'0B'	The specified short name is invalid.
X'0C'	Byte 0 of the parameter list not zero on request.
X'0D'	Invalid option code.
X'11'	No session has been configured for the specified session type.
X'12'	The name array length is invalid.
X'13'	The specified short name is not an uppercase ASCII alphabetic character.

See Appendix H, "Return Codes," for more information.

Coding Example

```

;
; DEFINE PARAMETER LIST FOR QUERY SESSION ID
;
QDRCODE DB 0 ; RETURN CODE
QFXNID DB 0 ; FUNCTION ID
QDOPT DB 0 ; OPTION BYTE
QDDATA DB 0 ; DATA BYTE
QDAOFF DW 0 ; NAMES ARRAY OFFSET
QDASEG DW 0 ; NAMES ARRAY SEGMENT ADDRESS
QDLNAM DB 8 DUP(' ') ; SESSION LONG NAME

.
.
.
MOV AX,SEG QDRCODE ; ADDRESSABILITY TO
MOV ES,AX ; PARAMETER LIST
MOV DI,OFFSET QDRCODE ; USING ES:DI
;
; INITIALIZE PARAMETER LIST FOR QUERY SESSION ID
;
MOV AL,01H ; OBTAIN THE SESSION ID OF
MOV QDOPT,AL ; A LONG NAME SPECIFIED
MOV AL,00H ;
MOV QDDATA,AL ; DATA BYTE
MOV QDAOFF,OFFSET ARRAYNAME ; ARRAY OFFSET
MOV QDASEG,SEG ARRAYNAME ; ARRAY SEGMENT
MOV QDRCODE,00H ; RETURN CODE = 0 ON REQUEST
MOV QFXNID,00H ; FUNCTION ID = 0 ON REQUEST
;
; THERE IS A LONG SESSION NAME
;
CLD ; BEGIN MOVING NAME
PUSH DS ; INTO PARAMETER LIST
MOV CX,4 ; NAME IS FOUR WORDS LONG
MOV SI,OFFSET LONGNAME ; SOURCE OFFSET IN SI
MOV AX,SEG LONGNAME ;
MOV DS,AX ; SOURCE SEGMENT IN DS
MOV DI,OFFSET QDLNAM ; DESTINATION OFFSET IN DI
REP MOVSW ; MOVE SESSION NAME TO
POP DS ; TO PARAMETER LIST
;
; SET UP REGISTERS FOR QUERY SESSION ID
;
MOV AH,09H
MOV AL,01H
MOV BH,80H
MOV BL,20H
MOV CX,00H
MOV DX,SESSMGR ; RESOLVED VALUE FOR 'SESSMGR'
MOV DI,OFFSET QDRCODE ; OFFSET ADDRESS OF
; PARAMETER LIST
;
; SIGNAL WORKSTATION PROGRAM FOR QUERY SESSION ID SERVICE
;
INT 7AH
.
.
.

```

Session Information Service X'02': Query Session Parameters

Use this service to obtain the session characteristics of the session you specify.

Register Values

On Request

AH = X'09'
AL = X'02'
BH = X'80'
BL = X'20'
CX = X'0000'
DX = Resolved value for SESSMGR
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers
AX, BX, DX, ES, and DI
are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'6B')
2	1 byte	Session ID	Unchanged
3	1 byte	Reserved	Reserved
4	1 byte	Reserved	Session type
5	1 byte	Reserved	Session characteristics
6	1 byte	Reserved	Rows
7	1 byte	Reserved	Columns
8	1 word	Reserved	Offset address of presentation space
10	1 word	Reserved	Segment address of presentation space

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session whose characteristics you are requesting.

Completion Parameters:

- The session type byte is as follows:
 - X'01' for a work station control session.
 - X'02' for a DFT host session.
 - X'03' for a CUT host session.
 - X'04' for a notepad session.
 - X'05' for a personal computer session.
- The bits in the session characteristics byte are as follows:

0	1	2 – 7
EAB	PSS	Reserved

- If bit 0 (EAB) = 0, the session has base attributes.
- If bit 0 (EAB) = 1, the session has extended attributes.
- If bit 1 (PSS) = 0, the session does not support programmed symbols.
- If bit 1 (PSS) = 1, the session supports programmed symbols.
- “Rows” is the hexadecimal number of rows in the session’s presentation space.
- “Columns” is the hexadecimal number of columns in the session’s presentation space.
- The offset and segment addresses of the presentation space point to the session’s presentation space. See Appendix F for a discussion of presentation space considerations.

Query Session Parameters

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Session Information Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the session management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Session information services return codes use a function ID of X'6B'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Specified session ID is invalid.
X'06'	Specified session ID not in use.
X'0C'	Byte 0 of the parameter list is not zero on request.

See Appendix H, "Return Codes," for more information.

Usage Notes

- The session ID required as input for this service can be obtained in the following ways:
 - By requesting the Query Session ID service.
 - By requesting the Query Base Window service
 - Or, if you defined a presentation space with the Define Presentation Space service, the session ID would be returned

Coding Example

```

;
; PARAMETER LIST FOR QUERY SESSION PARAMETERS SERVICE
;
QPRETNCD DB 0 ; RETURN CODE
QPFYNID DB 0 ; FUNCTION NUMBER
QPSESSID DB 0 ; SESSION ID
QPRESERV DB 0 ; RESERVED
QPSESTYP DB 0 ; SESSION TYPE
QPSESCHR DB 0 ; SESSION CHARACTERISTICS
QPROWS DB 0 ; NUMBER OF ROWS
QPCOLS DB 0 ; NUMBER OF COLUMNS
QPPSOFF DW 0 ; OFFSET OF PRESENTATION SPACE
QPPSSEG DW 0 ; SEGMENT OF PRESENTATION SPACE
.
.
.
;
; INITIALIZE PARAMETER LIST FOR QUERY SESSION PARAMETERS SERVICE
;
MOV QPRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV QPFYNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID INTO THE LIST
MOV QPSESSID,AL
;
; INITIALIZE REGISTERS FOR QUERY SESSION PARAMETERS SERVICE
;
MOV AH,09H
MOV AL,02H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,SESSMGR ; RESOLVED VALUE FOR 'SESSMGR '
MOV DI,SEG QPRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET QPRETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR QUERY SESSION PARAMETERS SERVICE
;
INT 7AH
.
.
.

```

Detach Session ID

Session Information Service X'04': Detach Session ID

Use this service to detach from a currently defined session.

Register Values

On Request

AH = X'09'
AL = X'04'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for SESSMGR
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers
AX, BX, DX, ES, and DI
are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'6B')
2	1 byte	Session ID	Unchanged
3	1 byte	Reserved	Reserved

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session to detach from.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Session Information Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the session management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Session information services return codes use a function ID of X'6B'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Specified session ID is invalid.
X'06'	Specified session ID not in use.
X'0C'	Byte 0 of the parameter list is not zero on request.
X'14'	Cannot detach from the session now.

See Appendix H, "Return Codes," for more information.

Usage Notes

This service should be used only by a system extension that provides a service for some session. Attaching to a session ID by issuing an Attach Session ID service request guarantees that the session ID you are providing services for will not be deleted until you detach from it. However, it is possible for the fixed-length queue or presentation space associated with the session to be deleted. If a deletion of this type occurs before you issue a Detach Session ID request for the session, an appropriate error code will be issued when you request a service for the session. If this error occurs, you should request the Detach Session ID service for the session, to make the session ID available to some other system extension.

Detach Session ID

Coding Example

```
;
; PARAMETER LIST FOR DETACH SESSION ID
;
DTRETNCD DB 0 ; RETURN CODE
DTFXNID DB 0 ; FUNCTION ID
DTSESSID DB 0 ; SESSION ID
DTRSRVD DB 0
.
.
.
;
; INITIALIZE PARAMETER LIST FOR DETACH SESSION ID
;
MOV DTRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV DTFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID IN
MOV DTSESSID,AL ; PARAMETER LIST
;
; INITIALIZE REGISTERS FOR DETACH SESSION ID
;
MOV AH,09H
MOV AL,04H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,SESSMGR ; NAME RESOLUTION FOR SESSMGR
MOV DI,SEG DTRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET DTRETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR DETACH SESSION ID SERVICE
;
INT 7AH
.
.
.
```

Session Information Service X'05': Attach Session ID

Use this service to attach to a currently defined session.

Register Values

On Request

AH = X'09'
AL = X'05'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for SESSMGR
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers
AX, BX, DX, ES, and DI
are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'6B')
2	1 byte	Session ID	Unchanged
3	1 byte	Reserved	Reserved

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session to attach to.

Attach Session ID

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Session Information Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the session management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Session information services return codes use a function ID of X'6B'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Specified session ID is invalid.
X'05'	Attachment limit exceeded.
X'06'	Specified session ID not in use.
X'0C'	Byte 0 of the parameter list is not zero on request.

See Appendix H, "Return Codes," for more information.

Usage Notes

This service should only be used by a system extension that provides a service for some session. Attaching to a session ID guarantees that the session ID you are providing services for will not be deleted until you detach from it. However, it is possible for the fixed-length queue or presentation space associated with the session to be deleted. If a deletion of this type occurs before you issue a Detach Session ID request for the session, an appropriate error code will be issued when you request a service for the session. If this error occurs, you should request the Detach Session ID service for the session, to make the session ID available to some other system extension.

Coding Example

```

;
; PARAMETER LIST FOR ATTACH SESSION ID
;
ATRETNCD DB 0 ; RETURN CODE
ATFXNID DB 0 ; FUNCTION ID
ATSESSID DB 0 ; SESSION ID
ATRSRVD DB 0

.
.
.

;
; INITIALIZE PARAMETER LIST FOR ATTACH SESSION ID
;
MOV ATRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV ATFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID IN
MOV ATSESSID,AL ; PARAMETER LIST

;
; INITIALIZE REGISTERS FOR ATTACH SESSION ID
;
MOV AH,09H
MOV AL,05H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,SESSMGR ; NAME RESOLUTION FOR SESSMGR
MOV DI,SEG ATRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET ATRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR ATTACH SESSION ID SERVICE
;
INT 7AH
.
.
.

```


Query Windows in Environment

Session Information Service X'06': Query Windows in Environment

Use this service to obtain a list of the windows that are defined within a specified environment. The windows in the environment are listed by their short name. You can also use this service to obtain the ID of the currently active environment.

Register Values

On Request

AH = X'09'
AL = X'06'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for SESSMGR
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'6B')
2	1 byte	Environment ID or X'00'	Unchanged or environment ID
3	1 byte	Reserved	Number of windows
4 - 23	20 bytes	Reserved	Window short names

Parameter Definitions

Request Parameters:

- The environment ID is the ID of the environment whose window short names you wish to obtain. If you want to obtain the ID of the currently active environment, specify X'00' in this field in the parameter list on request.

Completion Parameters:

- If you specified X'00' in byte 2 of the parameter list on request, the ID of the currently active environment is returned.
- The number of windows is the number of windows defined to belong to the specified environment.
- The window short name is the 1-character uppercase ASCII alphabetic name of each window that belongs to the environment.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Session Information Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the session management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Session information services return codes use a function ID of X'6B'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'0A'	Invalid environment ID.
X'0C'	Byte 0 of the parameter list is not zero on request.

See Appendix H, "Return Codes," for more information.

Usage Notes

- You can use the Query Environment service to obtain the environment ID to use as input for this service.

Query Windows in Environment

Coding Example

```
;
; PARAMETER LIST FOR QUERY WINDOWS IN ENVIRONMENT
;
QQRETNCD DB 0 ; RETURN CODE
QQFXNID DB 0 ; FUNCTION ID
QQENVID DB 0 ; ENVIRONMENT ID
QQNUMWIN DB 0 ; NUMBER OF WINDOWS
QQWNAMS DB 20 DUP(?) ; WINDOW SHORT NAMES

.
.
.

;
; INITIALIZE PARAMETER LIST FOR QUERY WINDOWS IN ENVIRONMENT
;
MOV QQRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV QQFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,ENVID ; ENVIRONMENT ID
MOV QQENVID,AL ; IN PARAMETER LIST

;
; INITIALIZE REGISTERS FOR QUERY WINDOWS IN ENVIRONMENT
;
MOV AH,09H
MOV AL,06H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,SESSMGR ; NAME RESOLUTION FOR SESSMGR
MOV DI,SEG QQRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET QQRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR QUERY WINDOWS IN ENVIRONMENT SERVICE;
INT 7AH
.
.
.
```

Session Information Service X'07': Query Environment of Window

Use this service to obtain the environment ID of a specified window.

Register Values

On Request

AH = X'09'
AL = X'07'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for SESSMGR
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code
 The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'6B')
2	1 byte	Window short name	Unchanged
3	1 byte	Reserved	Environment ID

Parameter Definitions

Request Parameters:

- The window short name is the ASCII short name of the window whose environment ID you are requesting.

Completion Parameters:

- The environment ID is the ID of the environment that owns the specified window.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Session Information Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the session management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Session information services return codes use a function ID of X'6B'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'0B'	Specified short name is invalid.
X'0C'	Byte 0 of the parameter list not zero on request.
X'13'	Specified short name is not an uppercase ASCII alphabetic character.

See Appendix H, "Return Codes," for more information.

Usage Notes

- The window name is either:
 - The short window name selected at customization time (which may be found using the Query Base Window service), or
 - The window name specified on the Define Presentation Space service or returned by this service, if you allowed the system to select an available window name for you.

Coding Example

```
;
; PARAMETER LIST FOR QUERY ENVIRONMENT OF WINDOW
;
QIRETNCD DB 0 ; RETURN CODE
QIFXNID DB 0 ; FUNCTION ID
QIWINDOW DB 0 ; WINDOW SHORT NAME
QIENVID DB 0 ; ENVIRONMENT ID

.
.

;
; INITIALIZE PARAMETER LIST FOR QUERY ENVIRONMENT OF WINDOW
;
MOV QIRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV QIFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,'P' ; WINDOW SHORT NAME IN
MOV QIWINDOW,AL ; PARAMETER LIST

;
; INITIALIZE REGISTERS FOR QUERY ENVIRONMENT OF WINDOW
;
MOV AH,09H
MOV AL,07H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,SESSMGR ; NAME RESOLUTION FOR SESSMGR
MOV DI,SEG QIRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET QIRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR QUERY ENVIRONMENT OF WINDOW SERVICE;
;
INT 7AH
.
.
.
```

Session Information Service X'08': Query PC Session Program Information File (PIF) Information

Use this service to obtain a flag that represents the PIF for the application program running in the specified personal computer session.

Register Values

On Request

AH = X'09'
AL = X'08'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for SESSMGR
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'6B')
2	1 byte	Session ID	Unchanged
3	1 word	Reserved	PC session flags

Parameter Definitions

Request Parameters:

- The session ID is the ID of the PC session being queried.

Completion Parameters

- The bits in the PC session flags indicate the answers to questions in the PIF for the application running in the specified session. Chapter 2, "Programming Considerations" contains information on creating PIFs and how they are used.

- The format of the PC session flags is as follows (remember that bit 0 is the high-order, leftmost, bit in the word and bit 15 is the low-order, rightmost, bit in the word):
 - Bit 0 = 1 means that the DISPLAY question was answered “yes.”
 - Bit 1 = 1 means that the INTERRUPT VECTORS swapped include vectors in the range X'00' through X'7F'.
 - Bit 2 = 1 means that the INTERRUPT VECTORS swapped include vectors in the range X'80' through X'FF'.
 - Bit 3 = 1 means that the TIMER question was answered “yes.”
 - Bit 4 = 1 means that the KEYBOARD question was answered “yes.”
 - Bit 5 is reserved.
 - Bit 6 is reserved.
 - Bit 7 is reserved.
 - Bit 8 = 1 means that the 8087 question was answered “yes.”
 - Bit 9 is reserved.
 - Bit 10 = 1 means that the FOREGROUND question was answered “yes.”
 - Bit 11 is reserved.
 - Bit 12 is reserved.
 - Bit 13 is reserved.
 - Bit 14 = 1 means that the MEMORY question was answered “yes.”
 - Bit 15 is reserved.

For example, to test flag 1 to determine whether the session swaps interrupt vectors in the range X'00' through X'7F', you can code:

```

      .
      .
      .
MOV    AX,SFLFLG          ; LOAD REGISTER WITH FLAGS
TEST   AX,4000H           ; BINARY '01000000 00000000'
JE     FLAGSET            ; TEST SUCCEEDED, TAKE JUMP
      .
      .
      .
FLAGSET:                  ; FLAG 1 IS SET
      .
      .
      .
```


Query PC Session PIF Information

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Session Information Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the session management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Session information services return codes use a function ID of X'6B'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Specified session ID is invalid.
X'06'	Specified session ID is not in use.
X'0C'	Byte 0 of the parameter list is not zero on request.

See Appendix H, "Return Codes," for more information.

Usage Notes

- An application program can request this service for its own session or for any other PC session in the same environment or any other environment.
- The use of this service is helpful if, for example, an application program needs to know whether it will be suspended if it is not the foreground application. Using the information obtained by this service allows the application program to avoid situations such as jumping to another session, becoming suspended, and being unable either to jump back or to release a lock on the work station control session. A situation such as this will hang the workstation program.
- If you are running an application using this service in a system that is not configured for Multi-DOS, the flag byte returned will be 00.

Coding Example

```
;
; PARAMETER LIST FOR QUERY PC SESSION PIF INFORMATION
;
SFLRCVAL DB 0 ; RETURN CODE
SFLRCFNC DB 0 ; FUNCTION ID
SFLSID DB 0 ; SESSION ID
SFLFLG DW 0

.
.
.

;
; INITIALIZE PARAMETER LIST FOR QUERY PC SESSION PIF INFORMATION
;
MOV SFLRCVAL,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV SFLRCFNC,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID IN
MOV SFLSID,AL ; PARAMETER LIST

;
; INITIALIZE REGISTERS FOR QUERY PC SESSION PIF INFORMATION
;
MOV AH,09H
MOV AL,08H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,SESSMGR ; NAME RESOLUTION FOR SESSMGR
MOV DI,SEG SFLRCVAL ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET SFLRCVAL ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR QUERY PC SESSION PIF INFORMATION SERVICE
;
INT 7AH
.
.
.
```

Session Information Service X'0A': Query Base Window

Use this service to obtain the session ID and short name of the base window of the specified environment. A base window is any window that was defined at configuration time or created using the INDSPLIT or INDMERGE commands.

Register Values

On Request

AH = X'09'
AL = X'0A'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for SESSMGR
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'6B')
2	1 byte	Environment ID	Unchanged
3	1 byte	Reserved	Session ID
4	1 byte	Reserved	Window short name
5	1 byte	Reserved	Reserved

Parameter Definitions

Request Parameters:

- The environment ID is the ID of the environment whose base window identity you are requesting. If this parameter is zero, the environment ID defaults to the current environment.

Completion Parameters:

- The session ID is the ID of the session associated with the base window.
- The window short name is the one-character uppercase ASCII name of the base window.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Session Information Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the session management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Session information services return codes use a function ID of X'6B'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'0A'	Invalid environment ID.
X'0C'	Byte 0 of the parameter list is not zero on request.
X'0E'	Base window is not found.

See Appendix H, "Return Codes," for more information.

Usage Notes

- Use this service to obtain the session ID of the window you are currently working in, if that window was not created using the Define Presentation Space service.

Query Base Window

Coding Example

```
;
; PARAMETER LIST FOR QUERY BASE WINDOW
;
QSRETNCDB DB 0 ; RETURN CODE
QSFXNID DB 0 ; FUNCTION NUMBER
QSENVID DB 0 ; ENVIRONMENT ID
QSSESSID DB 0 ; SESSION ID
QSWINDOW DB 0 ; WINDOW SHORT NAME
QSRESERV DB 0 ; RESERVED
.
.
.
;
; INITIALIZE PARAMETER LIST FOR QUERY BASE WINDOW
;
MOV QSRETNCDB,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV QSFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV QSENVID,0 ; USE DEFAULT OF CURRENT ENVIRONMENT
;
; INITIALIZE REGISTERS FOR QUERY BASE WINDOW
;
MOV AH,09H
MOV AL,0AH
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,SESSMGR ; RESOLVED VALUE FOR 'SESSMGR '
MOV DI,SEG QSRETNCDB ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET QSRETNCDB ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR QUERY BASE WINDOW SERVICE
;
INT 7AH
.
.
.
```

Session Information Service X'0B': Query Session Cursor

Use this service to obtain the cursor type and the row and column addresses of the specified session's cursor.

Register Values

On Request

AH = X'09'
AL = X'0B'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for SESSMGR
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code
 The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'6B')
2	1 byte	Session ID	Unchanged
3	1 byte	Reserved	Cursor type
4	1 byte	Reserved	Row address
5	1 byte	Reserved	Column address

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session whose cursor information you are requesting.

Completion Parameters:

- The cursor type byte is as follows (where bit 0 is the high-order bit and bit 7 is the low-order bit):

0	Reserved
1	Reserved
2	Reserved
3	Inhibited cursor with autoscroll
4	Reserved
5	Inhibited cursor
6	Blinking cursor
7	Box cursor
- The row address is the address in the session's presentation space representing the cursor's row position. Row addresses start with zero.
- The column address is the address in the session's presentation space representing the cursor's column position. Column addresses start with zero.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Session Information Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the session management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Session information services return codes use a function ID of X'6B'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Specified session ID is invalid.
X'06'	Specified session ID not in use.
X'0C'	Byte 0 of the parameter list is not zero on request.

See Appendix H, "Return Codes," for more information.

Coding Example

```

;
; PARAMETER LIST FOR QUERY SESSION CURSOR
;
CRRETNCD DB 0 ; RETURN CODE
CRFXNID DB 0 ; FUNCTION ID
CRSESSID DB 0 ; SESSION ID
CRCURSOR DB 0 ; CURSOR TYPE
CRROWADD DB 0 ; ROW ADDRESS
CRCOLADD DB 0 ; COLUMN ADDRESS

.
.
.

;
; INITIALIZE PARAMETER LIST FOR QUERY SESSION CURSOR
;
MOV CRRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV CRFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID IN
MOV CRSESSID,AL ; PARAMETER LIST

;
; INITIALIZE REGISTERS FOR QUERY SESSION CURSOR
;
MOV AH,09H
MOV AL,0BH
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,SESSMGR ; NAME RESOLUTION FOR SESSMGR
MOV DI,SEG CRRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET CRRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR QUERY SESSION CURSOR SERVICE
;
INT 7AH
.
.
.

```

Chapter 5. Coding Keyboard Service Requests

Introduction	5-2
Scan Code/Shift States	5-3
Keytop Characteristics	5-4
Attention Identifier (AID) Keys	5-5
Work Station Control Keys	5-6
Special Keys	5-6
ASCII/ASCII Mnemonics	5-7
Keyboard Services	5-7
Requesting the Keyboard Services	5-8
Return Codes for the Keyboard Services	5-8
Keyboard Service X'01': Connect to Keyboard	5-9
Keyboard Service X'02': Disconnect from Keyboard	5-13
Keyboard Service X'03': Read Input	5-16
Keyboard Service X'04': Write Keystroke	5-22
Keyboard Service X'05': Disable Input	5-30
Keyboard Service X'06': Enable Input	5-33
Keyboard Service X'07': Post Status Code	5-36

Introduction

This chapter describes how to code requests for the keyboard services provided by the API.

The keyboard services allow your application program to read and write keystroke data from a specified session, to disable and enable operator input from the keyboard of a specified session, and to notify the workstation program of the status of your application program's keystroke processing.

A recommended way of performing keystroke processing is for an application program to create a separate task (using the Create Task Entry service) that processes keystrokes in a loop until the terminating keystroke is detected. The application program should use the Set Task Ready service to set the task to the ready state and cause it to run.

As an example, the keystroke processing task might consist of the following:

1. An initialization section that would perform the system functions needed to obtain information required by the task. Some functions that the initialization section may include are:
 - Creating an input queue and if necessary an event queue
 - Requesting any query services needed
 - Readyng the keystroke processing task.
2. A key processing section that should first connect to any sessions with which the task will interact; it then will loop while performing the Read Input and Write Keystroke functions as required as well as any other actions required to process keystrokes. This section could continue to loop until some terminating condition was detected, such as some predefined terminating keystroke.

Note that while this task is in the process of doing a Read Input function with a WAIT option, it will be suspended until something appears on its input queue. This means that no other processing can be done by this task until either a keystroke is pressed on the keyboard, or a 4-byte item is enqueued to the task's input queue by another task. The task may also do a Read Input function with a NOWAIT option. In this case, control will be returned if there is nothing on the queue and processing can be done without pressing another keystroke or receiving input from another task. In either case, the processing loop can detect the value returned in the parameter list as a keystroke, continue processing, and determine the need to terminate either by the keystroke value or by some other mechanism. It would then avoid doing another Read Input and set itself unready.

3. A cleanup section that will be run when the task is signaled in some way to end its processing. This section would disconnect from all services that it had connected to and delete any items it had created, such as the input queue. When its cleanup activities are completed, this section could signal some controlling task that it is done, and set itself unready. This prevents it from being dispatched again. The controlling task could delete the key processing task prior to returning to DOS.

Scan Code/Shift States

The Keyboard Services utilize scan codes/shift states to identify 3270 PC keyboard events. Each keytop on the keyboard is represented by a unique scan code byte. An additional shift state byte describes the condition of keyboard modifying keys (that is, Upshift, Caps Lock, Alt, and Ctrl).

The scan code values are listed in Appendix A and are shown on foldouts at the back of this book. These 3270 PC API values are *not* the same as those received when reading keystrokes via standard PC keyboards (for example, using the BIOS method). For information on the standard PC scan codes, refer to the *IBM Personal Computer Technical Reference* manual.

Regardless of what keyboard you have attached, the scan codes/shift states you either receive or send are the values for the 3270 PC keyboard. These 3270 PC API scan codes/shift states must be used in all cases involving the use of the keyboard service API. See Appendix A of this manual for a list of these scan code/shift state values.

Note that, internal to the workstation program, keystrokes are represented by four bytes as indicated by bytes 8 through 11 of the Read Input parameter list.

The 3270 PC keyboard has many more keytops than a standard PC keyboard. However, only those 3270 PC keytops that are found on the standard PC are available to an application using the standard PC methods of reading keystrokes. Alternatively, all 3270 PC keytops (except the workstation control keytops) can be made available to the connecting application via the Read Input service; the Connect to Keyboard service with the intercept option of "All Keystrokes" is used to enable this alternative.

The API scan codes/shift states sent by an application are processed using the Write Keystroke service; as a result, a receiving application using standard PC methods can expect the same results as if running on a standard PC.

The keyboard shift state is indicated by a 1-byte value that indicates which of the functions or characters printed on the keytop of a given position is being sent. The shift state byte is described in Appendix A. Note that PC sessions require the use of bits 2 and 3 of that byte to determine which of the two shift keys was depressed, while bit 7 alone is sufficient for all other sessions to recognize the upshifted condition.

Introduction


Keytop Characteristics

Four types of keytop characteristics are used on the 3270 PC keyboard:





1. "Make Only," where one scan code is sent for each depression of the key, no matter how long it is held down.
2. "Make/Break," where a scan code is sent when the key is pressed (make) and then a pair of scan codes is sent when the key is released (break). The first of the two break scan codes is X'F0', which indicates that a key is released. The second scan code is that of the key that was released. (This second scan code is the same as that sent when the key is pressed.)
3. "Typematic," where a single scan code is sent when the key is pressed and, if after a short time the key is not released, that same scan code is sent every 100 milliseconds until the key is released.
4. "Typematic Make/Break," which is the same as that described for "Typematic," except that upon release the breaking pair of scan codes is sent just as described for the "Make/Break" type of keys.

With the workstation program loaded, the characteristics of the keyboard are altered to match those expected by the session that is currently active. From a keystroking perspective, there are only two types of sessions: PC sessions (those that appear on a standard PC) and non-PC sessions (host, notepad, and WS Ctrl). The non-PC sessions are all coded to use the host style of keyboard characteristics, while the PC sessions expect to receive the standard PC style.

The keytop characteristics (with API scan codes in hex) are as follows:

- PC and non-PC sessions; for all cases, the following keytops are "Make Only." (Regardless of the active session, these keytop scan codes are always sent to the WS Ctrl session.)
 - WS Ctrl (scan code 04)
 - ChgSc/Jump (scan code 03)
 - Enlarge () (scan code 01)
- PC sessions; all keytops (except those in item 1) are "Typematic Make/Break" to match the characteristics of the PC's keyboard.
- Non-PC sessions;
 - The following keys are "Make/Break":
 - Upshift, left and right (scan codes 12 and 29)
 - Alt, left and right (scan codes 19 and 39)
 - Caps Lock (scan code 14)
 - Ctrl (scan code 09)

- The following keys are “Make Only”:

Key(s)	Scan Code(s)
PF1 through PF24	See page FO-1
PA1 through PA3	67, 6E and 6F
Help	05
Clear	06
WS Ctrl	04
Finish	0C
ChgSc/Jump	03
Erase EOF	0B
Print	83
Copy/Auto	0A
Enlarge ()	01
Reset	11
Enter	58
Insert ()	65
Delete ()	6D
Home ()	62
On the Numeric Keypad	
Esc	76
NumLk	77
./ScrLk	7E
Space	84
./Del	71
Enter/+	79

- All other keytops are “Typematic.”

Care should be taken when sending keystrokes between differing session types (that is, reading a PC keyboard and sending the results to a non-PC session, or vice versa), to filter the keystroke characteristics in such a way as to match what the destination session expects. The same care should be taken for the scan codes. For example, a host session does not expect the Enter key to be “Make/Break” and may treat the breaking scan code as a second Enter key. Similarly, a scan code for the Esc key has no meaning to a host session and, if sent, will terminate a Write Keystroke request with an X'10' error condition.

Attention Identifier (AID) Keys

Attention Identifier (AID) keys are those keys that, when pressed in a host session, cause immediate host interaction. The term *AID* applied to these keys is meaningful only during a host session. In a host session, AID keys are “Make Only” and references to AID keys do not take into account typematic or make/break characteristics.

Introduction

The Connect to Keyboard service with the intercept option of “AID keystrokes only” allows the API program to control host mainframe interactions without interfering with normal data entry keystrokes.

The keytops treated as AID keys during a host session are:

PF1 through PF24
Enter (↵)
Clear
SysRq
CrSel
Test
Attn
PA1 through PA3

Work Station Control Keys

Work station control keys are those keyboard keys that, when pressed, have their scan codes routed to the WS Ctrl session or the interceptor of the WS Ctrl session’s keystrokes. This routing occurs without regard for what session is active at the time. These keys have no meaning to any other session and will be rejected if their scan codes are sent to those sessions through the Write Keystroke service.

The keytops treated as work station control keys are:

WS Ctrl (in the upper and lower shift states, but not in the control shift or alternate shift states).
ChgSc
Jump
Enlarge (⇄)

Special Keys

There are some additional API scan codes used by the workstation program that do not appear in the scan code table, but do appear in the Read Input and Write Keystroke service requests:

- ‘F0’ is used to indicate that a make/break type key is being released and that the next scan code to follow represents the key being released.
- ‘7F’ is sent by the workstation program to notify sessions of the current shift state of the keyboard. The workstation program sends this scan code whenever the real keyboard is reattached to a session; this ensures that the session interprets the current shift state as that perceived by the keyboard operator. This scan code occurs whenever a session is jumped into or whenever keystrokes have been sent to a session other than directly from the keyboard. To the session receiving it, this scan code means “align the session shift state to match the shift state sent with this scan code.”

ASCII/ASCII Mnemonics

The Keyboard Services API supports an ASCII option on the Read Input and Write Keystroke API services to allow applications to send and receive keys in ASCII or ASCII mnemonics. The ASCII values that can be sent or received include:

- All standard ASCII characters representing keys that can be received from the keyboard
- ASCII mnemonics that represent HOST and PC keystrokes that do not have ASCII codes. All mnemonics are two bytes, three bytes, four bytes, or six bytes long. All mnemonics start with @.

Note: When intercepting keystrokes with ASCII, only the Make key is returned. Shift, break, and shift alignment keys are not returned using this option.

Appendix A contains a complete list of all ASCII values and their corresponding characters.

Keyboard Services

The keyboard services provided by the API are:

- **Connect to Keyboard Service:** Use this service to connect to a session for keyboard services.
- **Disconnect from Keyboard Service:** Use this service to disconnect from a session for keyboard services.
- **Read Input Service:** Use this service to read keystroke data from a session.
- **Write Keystroke Service:** Use this service to write keystroke data to a session.
- **Disable Input Service:** Use this service to disable operator input to the session.
- **Enable Input Service:** Use this service to reenale operator input to the session.
- **Post Status Code Service:** Use this service to notify the workstation program of the status of your application program's keystroke processing.

Introduction

Requesting the Keyboard Services

To request any of the keyboard services, load the registers and the parameter list with the proper values, and use the INT 7AH instruction to signal the workstation program that it has a request to process.

*Note: Before your application can request the keyboard services, it must request the Name Resolution service, using **KEYBOARD** as the gate name in the parameter list.*

Return Codes for the Keyboard Services

Each keyboard service has two return codes associated with it: a system return code and a keyboard management return code. Both types of return codes are 2-byte values made up of a function ID and an error number. The function ID indicates the portion of the workstation program in which the error occurred. The error number indicates the specific type of error that has occurred. An error number of X'00' always indicates a successful acceptance or completion of the request.

- **System Return Codes:**

After your application has requested a keyboard service, the CH and CL registers contain a return code generated by the request processing portion of the workstation program. The function ID is in the CH register, and the error number is in the CL register. System return codes use a function ID of X'12'. The error codes that can appear are:

Code	Meaning
X'00'	Request accepted.
X'05'	Invalid index specified.
X'07'	Invalid reply specified.
X'08'	Invalid wait type specified.
X'0B'	RQE pool depleted.
X'0F'	Invalid environment access.
X'34'	Invalid gate entry.

These system return codes apply to all keyboard services.

- **Keyboard Services Return Codes:**

After a requested keyboard service is completed, bytes 0 and 1 of the parameter list contain a return code generated by the keyboard management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Keyboard services return codes use a function ID of X'62'. The error numbers that can appear are specific to the service that was requested and are included in the descriptions of each service.

See Appendix H, "Return Codes," for more information.

Keyboard Service X'01': Connect to Keyboard

Use this service to connect to a session for keyboard services.

Register Values

On Request

AH = X'09'
AL = X'01'
BH = X'80'
BL = X'20'
CX = X'0000'
DX = Resolved value for KEYBOARD
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'62')
2	1 byte	Session ID	Unchanged
3	1 byte	Reserved	Reserved
4	1 word	Event queue ID or zero	Unchanged
6	1 word	Input queue ID or zero	Unchanged
8	1 byte	Intercept options	Unchanged
9	1 byte	Reserved	First connection indicator

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session you want to connect to.
- The event queue ID is the ID of a fixed-length queue that the workstation program uses to notify you when the program running in another session has been stopped. If you intend to interact with programs in other personal computer sessions, using this event queue is a way of finding out when any of those programs is stopped. Use the Create Queue service to create this fixed-length queue, and use the Dequeue Element service to obtain the event information. This parameter is optional and should be set to zero if not used.

Connect to Keyboard

The event that can be reported is as follows:

Offset	Length	Contents
0	1 byte	Session ID
1	1 byte	X'00' (Reserved)
2	1 byte	X'02' (code)
3	1 byte	X'62' (function ID)

This event indicates that the specified session has been disconnected.

- The input queue ID is the ID of a fixed-length queue used to receive intercepted keystrokes typed at a session. Use the Create Queue service to create this fixed-length queue. This parameter is optional and should be set to zero if not used. If this parameter is set to a nonzero value, then a valid intercept option must also be set. Keystrokes can be intercepted from the specified session by requesting the Read Input service.
- The intercept options specify which types of keystrokes are to be intercepted from the specified session. If this byte is nonzero, you must also supply an input queue ID. The bits in the intercept options byte are as follows:

0	1	2-7
AID keystrokes only	All keystrokes	Reserved

- Bit 0 set to 1 indicates that only keystrokes that would normally generate an AID in a host session are to be sent to your application program.
- Bit 1 set to 1 indicates that all keystrokes in the session are to be sent to your application program, including AID keys.
- Bits 2 through 7 are reserved and should be set to all zeros.

Completion Parameters:

- The first connection indicator is set to X'FF' if this is the first time a Connect to Keyboard request has been made to this session.
 - If connecting to a session defined at configuration time or by an INDSPLIT or INDMERGE command, the workstation program has already issued a Connect to Keyboard request for the session. Therefore, a value of X'FF' indicates an error condition.
 - If connecting to a session defined by your application program by a Define Presentation Space service request, the workstation program will not have issued a prior Connect to Keyboard request for that session. Therefore, a value of X'FF' is normal.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Keyboard Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the keyboard management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Keyboard services return codes use a function ID of X'62'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'01'	Invalid intercept option.
X'02'	Invalid session ID.
X'04'	Session busy; cannot connect at this time.
X'0C'	Byte 0 of the parameter list is not zero on request.
X'12'	Request failed; an autokey record operation is in progress.
X'14'	Request failed; an autokey playback operation is in progress.

See Appendix H, "Return Codes," for more information.

Usage Notes

- Your application program can be connected simultaneously for keystroke data to each of the host, notepad, work station control, and personal computer sessions that your 3270 Personal Computer is customized to support.
- The recommended size for input queues is 50 bytes.
- A maximum of two connections for keyboard services can be made to a session at any one time. If your session was defined at configuration time or by issuing an INDSPLIT or INDMERGE command, one connection has already been made to the session by the workstation program.
- The workstation program issues Connect to Keyboard service requests for each configured session to provide the capability for the session to receive and process keystrokes.

A session defined by your application program (by a Define Presentation service request) requires that a Connect to Keyboard with an All Keys Intercept option be issued in order to receive and process keystrokes. Note that this implies that the application must also provide the keystroking task for such sessions. A second Connect to Keyboard request can be issued relative to a Define Presentation Space session.

Connect to Keyboard

Coding Example

```
;
; PARAMETER LIST FOR CONNECT TO KEYBOARD
;
CKRETNCD DB 0 ; RETURN CODE
CKFXNID DB 0 ; FUNCTION NUMBER
CKSESSID DB 0 ; SESSION ID
CKRESRV1 DB 0 ; RESERVED
CKEVENTQ DW 0 ; EVENT QUEUE ID
CKKEYSTQ DW 0 ; INPUT QUEUE ID
CKOPTION DB 0 ; OPTION BYTE
CK1STCON DB 0 ; FIRST CONNECTION INDICATOR
.
.
.
;
; INITIALIZE PARAMETER LIST FOR CONNECT TO KEYBOARD
;
MOV CKRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV CKFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID INTO THE LIST
MOV CKSESSID,AL ;
MOV CKOPTION,01000000B ; OPTION = INTERCEPT ALL
MOV AX,KEYSTQ ; KEYSTROKE QUEUE ID INTO THE LIST
MOV CKKEYSTQ,AX ;
MOV AX,EVENTQ ; EVENT QUEUE ID INTO THE LIST
MOV CKEVENTQ,AX ;
;
; INITIALIZE REGISTERS FOR CONNECT TO KEYBOARD (
;
MOV AH,09H
MOV AL,01H
MOV BH,80H
MOV BL,20H
MOV CX,00H
MOV DX,KEYBOARD ; RESOLVED VALUE FOR 'KEYBOARD'
MOV DI, SEG CKRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET CKRETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR CONNECT TO KEYBOARD SERVICE
;
INT 7AH
.
.
.
```

Keyboard Service X'02': Disconnect from Keyboard

Use this service to disconnect from the session when you are finished using the keyboard services.

Register Values

On Request

AH = X'09'
AL = X'02'
BH = X'80'
BL = X'20'
CX = X'0000'
DX = Resolved value for KEYBOARD
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'62')
2	1 byte	Session ID	Unchanged
3	1 byte	Reserved	Reserved
4	1 word	Connector's task ID	Unchanged

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session you want to disconnect from. The session must have been previously connected to the keyboard through a Connect to Keyboard service request.
- The connector's task ID is needed only if the task that requested the Connect to Keyboard service for this session is different from the task requesting the Disconnect from Keyboard service. This parameter is optional and should be set to zero if not used.

Disconnect from Keyboard

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Keyboard Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the keyboard management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Keyboard services return codes use a function ID of X'62'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for keyboard services.
X'0C'	Byte 0 of the parameter list is not zero on request.

See Appendix H, "Return Codes," for more information.

Usage Notes

- This service also enables operator input to the session if it was previously disabled through a Disable Input service request.
- Before exiting, your application program must use the Disconnect from Keyboard service for each session that was connected for keystroking data. This service should be requested at all error exit points as well as during normal processing.

Coding Example

```
;
; PARAMETER LIST FOR DISCONNECT FROM KEYBOARD
;
DKRETNCD DB 0 ; RETURN CODE
DKFXNID DB 0 ; FUNCTION NUMBER
DKSESSID DB 0 ; SESSION ID
DKRESRV1 DB 0 ; RESERVED
DKTASKID DW 0 ; CONNECTOR'S TASK ID

.
.
.

;
; INITIALIZE PARAMETER LIST FOR DISCONNECT FROM KEYBOARD
;
MOV DKRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV DKFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID INTO THE LIST
MOV DKSESSID,AL ;
MOV AX,TASKID ; CONNECTOR'S TASK ID INTO THE LIST
MOV DKTASKID,AX ;

;
; INITIALIZE REGISTERS FOR DISCONNECT FROM KEYBOARD
;
MOV AH,09H
MOV AL,02H
MOV BH,80H
MOV BL,20H
MOV CX,00H
MOV DX,KEYBOARD ; RESOLVED VALUE FOR 'KEYBOARD'
MOV DI, SEG DKRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET DKRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR DISCONNECT FROM KEYBOARD SERVICE
;
INT 7AH
.
.
.
```


Keyboard Service X'03': Read Input

Use this service to read intercepted keystroke data destined for the session. An options byte is set to indicate whether the Read is done with scan code/shift states or with ASCII mnemonics. This service returns the scan code/shift state and/or the ASCII mnemonic for one keystroke with each request made.

Register Values

On Request

AH = X'09'
AL = X'03'
BH = X'80'
BL = X'20'
CX = X'0000'
DX = Resolved value for KEYBOARD
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'62')
2	1 byte	Session ID	Unchanged
3	1 byte	Reserved	Reserved
4	1 word	Connector's task ID	Unchanged
6	1 byte	Options byte	Unchanged

The bits in the options byte are as follows:

0	1	2	3	4	5	6	7
ASCII	0	SC/SS	0	Nowait	0	0	0

- Bit 0 set to 1 means to read the keystroke in ASCII format.
- Bit 2 set to 1 means to read the keystroke in scan code/shift state format.

Note: Choose either ASCII or scan code/shift state format. Do not select both (that is, do not set bits 0 and 2 equal to 1).

- Bit 4 set to 1 means to read the keystroke with the NOWAIT option. If a keystroke is not available on the queue when this is issued, a return code of X'09' will be placed in the parameter list and control returned to the caller. If the bit is not set, the calling test will be suspended until keystroke data appears on the task input queue.

The remainder of the parameter list depends on whether you have specified the scan code/shift state format or the ASCII/ASCII mnemonic format.

If you have selected the scan code/shift state format, the parameter list is as follows:

Offset	Length	Contents on Request (SC/SS)	Contents on Completion (SC/SS)
7	1 byte	Reserved	Reserved
8	1 byte	Reserved	Scan code of the key
9	1 byte	Reserved	Shift state of the key
10	1 byte	Reserved	X'01'
11	1 word	Reserved	X'00'

If you have selected the ASCII/ASCII mnemonic format, the parameter list is as follows:

Offset	Length	Contents on Request (ASCII)	Contents on Completion (ASCII)
7	1 byte	Reserved	Length of ASCII/ASCII mnemonic returned in bytes 8 - 13
8 - 13	6 bytes	Reserved	ASCII/ASCII mnemonic

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session from which keystroke data is intercepted.
- The connector's task ID is needed only if the task that requested the Connect to Keyboard service for this session is different from the task requesting the Read Input service. This parameter is optional and should be set to zero if not used.

Completion Parameters for Scan Code/Shift State Option:

Keystroke data is sent to a session in a format that uses four bytes of data to represent the key being sent. The first byte is called the *scan code*, and the second byte is called the *shift state*. The third and fourth bytes are particular to the device being used and, for a keyboard, should normally be X'0100'.

- The scan code represents a particular key position on the keyboard.
- The shift state indicates which of the possible characters or functions located at that key position is being sent. The possible shift states are the lower shift, the upper shift, the alt shift, and the ctrl shift.

Appendix A lists the scan codes for each key position and describes the format of the shift state byte. In addition, the foldout for the IBM 3270 PC keyboard (at the back of this book) shows the scan codes for each key position on the keyboard.

Completion Parameters for ASCII/ASCII mnemonic option:

- ASCII/ASCII mnemonics are from 1 to 6 bytes long. ASCII mnemonics start with @.
- The length of field is the length of the ASCII code or ASCII mnemonic.

Note: Any bytes left unused will remain unchanged.

See Appendix A for a list of all the ASCII/ASCII mnemonics that can be received.

Note: If the shift state of the key pressed does not contain a unique ASCII/ASCII mnemonic or if multiple shift states are active, ASCII mnemonics for the shift state will be prefixed. The valid shift prefix mnemonics are:

@A - Alt shift active

@S - Upshift active

@r - Ctrl shift active.

An example of this would be pressing alt-a. The ASCII mnemonic returned would be @Aa. If the Ctrl-shift-A were pressed, the ASCII mnemonic returned would be @rA.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Keyboard Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the keyboard management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Keyboard services return codes use a function ID of X'62'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'01'	Invalid intercept option (see Note).
X'02'	Invalid session ID.
X'04'	The session is not connected for keyboard services.
X'09'	Queue empty, no keystroke available.
X'0C'	Byte 0 of the parameter list is not zero on request.
X'10'	Invalid keystroke.

Note: An invalid intercept option means that the application is trying to do a Read Input when no Read options were specified on the connect.

See Appendix H, "Return Codes," for more information.

Read Input

Usage Notes

- To be able to use the Read Input service, you must do the following when you request the Connect to Keyboard service:
 - Specify which types of keystrokes are to be intercepted from the connected session
 - Provide the ID of a fixed-length queue to be used by the workstation program to store the intercepted keystroke data.
- The scan code and shift state returned by this service are not the same as those returned by BIOS or DOS read keys.
- When using the Read Input service with the scan code option for a PC session, you must keep in mind that each key will generate an 'X'F0' scan code if the key is breaking, followed by the scan code of the key. See "Special Scan Codes" in Appendix A.
- It is recommended that each Read Input request be followed by a Post Status Code request, particularly if keystrokes are rejected for any reason.
- Applications running on non-3270 PC XT hardware that send keystrokes to or receive keystrokes from the host session will not run on Uni-DOS.
- Applications using the keyboard services should be well-behaved. Additionally, if your application is using the API to another PC session, then that session must also be well-behaved.

Coding Example

```

;
; PARAMETER LIST FOR READ INPUT
;
RKRETNCD DB 0 ; RETURN CODE
RKFXNID DB 0 ; FUNCTION NUMBER
RKSESSID DB 0 ; SESSION ID
RKRESRV1 DB 0 ; RESERVED
RKTASKID DW 0 ; CONNECTOR'S TASK ID
          DB 20H ; MUST BE 20H FOR SCAN CODES
RKRESRV2 DB 0 ; RESERVED
RKSCANCD DB 0 ; SCAN CODE OF THE KEY
RKSHIFST DB 0 ; SHIFT STATE OF THE KEY
          DB 0 ; 01H ON RETURN
          DB 0 ; 00H ON RETURN
.
.
.
;
; INITIALIZE PARAMETER LIST FOR READ INPUT
;
      MOV     RKRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
      MOV     RKFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
      MOV     AL,SESSID ; SESSION ID INTO THE LIST
      MOV     RKSESSID,AL ;
      MOV     AX,TASKID ; CONNECTOR'S TASK ID INTO THE LIST
      MOV     RKTASKID,AX ;
;
; INITIALIZE REGISTERS FOR READ INPUT
;
      MOV     AH,09H
      MOV     AL,03H
      MOV     BH,80H
      MOV     BL,20H
      MOV     CX,00H
      MOV     DX,KEYBOARD ; RESOLVED VALUE FOR 'KEYBOARD'
      MOV     DI,SEG RKRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
      MOV     ES,DI ; IN ES
      MOV     DI,OFFSET RKRETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR READ INPUT SERVICE
;
      INT     7AH
.
.

```

Keyboard Service X'04': Write Keystroke

Use this service to send keystroke data to the session. An options byte is set to indicate whether the write is done with scan code/shift states or with ASCII/ASCII mnemonics. Appendix A contains the valid scan code/shift states or ASCII/ASCII mnemonics that can be sent.

Register Values

On Request

AH = X'09'
AL = X'04'
BH = X'80' or X'40' (see Note)
BL = X'20' or X'00' (see Note)
CX = X'0000'
DX = Resolved value for KEYBOARD
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

AX = Request ID
CH = X'12'
CL = Return code

The contents of registers BH, DX, ES, and DI are unpredictable.

Note: The combined contents of the BH and BL registers are either X'8020' or X'4000'.

- Request Register Values:

For asynchronous processing of the Write Keystroke service request, set the BH register to X'40' and the BL register to X'00'. When asynchronous processing is specified, you must request the Get Request Completion service to obtain the results of each Write Keystroke service request.

For synchronous processing of the Write Keystroke service request, set the BH register to X'80' and the BL register to X'20'.

- Completion Register Values:

If you specified asynchronous processing (the BH register = X'40' and the BL register = X'00' on request), the AX register contains a request ID that the workstation program assigned to the request. You use this request ID to match the results of the service obtained by the Get Request Completion service to a previously requested service.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'62')
2	1 byte	Session ID	Unchanged
3	1 byte	Reserved	Reserved
4	1 word	Connector's task ID	Unchanged
6	1 byte	Options byte	Unchanged

The bits in the options byte are as follows:

0	1	2	3	4	5	6	7
ASCII	0	SC/SS	List	0	0	0	0

- Bit 0 set to 1 means to write the keystroke or list of keystrokes in ASCII format.
- Bit 2 set to 1 means to write keystroke or list of keystrokes in scan code/shift state format.

Note: Choose either ASCII or scan code/shift state format. Do not select both (that is, do not set bits 0 and 2 equal to 1).

- Bit 3 is set as follows:
 - 1 – Write a list of keystrokes.
 - 2 – Write a single keystroke.

The remainder of the parameter list depends on which option you have specified and on whether you are writing a single keystroke or a list of keystrokes.

If you are sending a single keystroke with the scan code/shift state option, the parameter list must be formatted as follows:

Offset	Length	Contents on Request (SC/SS Single)	Contents on Completion (SC/SS Single)
7	1 byte	Reserved	Reserved
8	1 byte	Scan code of the key	Unchanged
9	1 byte	Shift state of the key	Unchanged

Write Keystroke

If you are sending a single keystroke with the ASCII option, the parameter list must be formatted as follows:

Offset	Length	Contents on Request (ASCII) Single	Contents on Completion (ASCII) Single
7	1 byte	Number of bytes of ASCII/ASCII mnemonics to send (in bytes 8 – 13)	Number of bytes of ASCII/ASCII mnemonics sent
8 – 13	6 bytes	ASCII mnemonic	Unchanged

If you are sending a list of keystrokes with the scan code/shift state option, the parameter list must be formatted as follows:

Offset	Length	Contents on Request (SC/SS List)	Contents on Completion (SC/SS List)
7	1 byte	Reserved	Number of keys sent
8	1 word	Offset address of list of keys	Unchanged
10	1 word	Segment address of list of keys	Unchanged

If you are sending a list of keystrokes with the ASCII option, the parameter list must be formatted as follows:

Offset	Length	Contents on Request (ASCII)	Contents on Completion (ASCII)
7	1 byte	Reserved	Number of bytes of ASCII/ASCII mnemonics in the list that were sent
8	1 word	Offset address of list of keys	Unchanged
10	1 word	Segment address of list of keys	Unchanged

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session to write the keystrokes to.
- The connector's task ID is needed only if the task that requested the Connect to Keyboard service for this session is different from the task requesting the Write Keystroke service. This parameter is optional and should be set to zero if not used.
- The format of a list of scan code/shift state keystrokes is as follows:

Offset	Length	Contents (SC/SS List)
0	1 word	$2 \times n$ (where n is the number of keys being sent to the session)
2	1 byte	Scan code of the first key being sent
3	1 byte	Shift state of the first key being sent
4	1 byte	Scan code of the second key being sent
5	1 byte	Shift state of the second key being sent
⋮		
$2n$	1 byte	Scan code of the n th key being sent
$2n + 1$	1 byte	Shift state of the n th key being sent

Note: $n = 255$ keys, which is the maximum value allowed.

The format of a list of ASCII keystrokes is as follows:

Offset	Length	Contents
0	1 word	N (where N is the number of bytes of ASCII/ASCII mnemonics being sent to the session)
$2 - (N + 2)$	N bytes	ASCII/ASCII mnemonic of the keys being sent

Note: $N = 255$ bytes, which is the maximum value allowed.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Keyboard Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the keyboard management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Keyboard services return codes use a function ID of X'62'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'01'	Invalid option byte value.
X'02'	Invalid session ID, or the list of scan code/shift state keystrokes is longer than 255 bytes, or the list of ASCII/ASCII mnemonics is longer than 255 bytes.
X'04'	The session is not connected for keyboard services.
X'0C'	Byte 0 of the parameter list is not zero on request.
X'10'	Processing stopped because of invalid scan code or input inhibited, or invalid ASCII mnemonic.
X'12'	Processing stopped because AID key was detected; successful completion.

See Appendix H, "Return Codes," for more information.

Usage Notes

- The maximum number of keystrokes that can be sent is 255.
- The maximum number of ASCII/ASCII mnemonic bytes in the list that can be sent is 255.
- AID keys must not be embedded within a list, although one may be the last key in a list. The processing of a keystroke list ends when an AID key is processed or when a keystroke is rejected for any reason. If a return code of X'10' or X'12' is received, it is good practice to check byte 7 of the parameter list on completion to determine how many of the keystrokes or how many bytes of ASCII/ASCII mnemonics were actually sent before processing was ended.
- If you specified asynchronous processing (BH = X'40' and BL = X'00' on request), you must use the Get Request Completion service to obtain the results in the parameter list when the Write Keystroke service is completed.
- Host sessions must be functional in order for keystroke data to be processed. That is, if your 3270 Personal Computer is customized to support a host session, but no port is available for that session on the control unit, any keystroke data sent to that host session will not be processed.
- If your 3270 Personal Computer is connected to more than one control unit through a coaxial switch, you can switch to an alternate control unit by toggling the coaxial switch and pressing the Ctrl key and the Clear key at the same time. This sequence of keystrokes enables you to log on to an alternate host system without having to re-IPL the workstation program.
- A successful completion return code from a Write Keystroke request to a host session does not mean that the host has processed the keystroke. It indicates only that the keystroke has been successfully sent to the host session.
- Applications running on non-3270 PC XT hardware that send keystrokes to or receive keystrokes from the host session will not run on Uni-DOS.
- Applications using the keyboard services should be well-behaved. Additionally, if your application is using the API to another PC session, then that session must also be well-behaved.

Write Keystroke

Coding Example

```
;
; PARAMETER LIST STRUCTURE FOR WRITE KEYSTROKE
;
WRKYPAR1  STRUC
WKRETNCD  DB  0                      ; RETURN CODE
WKFXNID   DB  0                      ; FUNCTION NUMBER
WKSESSID  DB  0                      ; SESSION ID
WKRESRV1  DB  0                      ; RESERVED
WKTASKID  DW  0                      ; CONNECTOR'S TASK ID
WKOPTION  DB  0                      ; OPTIONS BYTE
WKNUMKEY  DB  0                      ; KEYS SENT SUCCESSFULLY
WKSCNCOD  DB  0                      ; SCAN CODE OF THE KEY
WKSHFST   DB  0                      ; SHIFT STATE OF THE KEY
WRKYPAR1  ENDS

WRKYPARM  STRUC
DB  8 DUP(00)
WKLSTOFF  DW  0                      ; OFFSET OF LIST OF KEYSTROKES
WKLSTSEG  DW  0                      ; SEGMENT OF LIST OF KEYSTROKES
WRKYPARM  ENDS
;
; ALLOCATE STORAGE FOR THE PARAMETER LIST
;
WKPARLST  WRKYPARM <>

.
.
.

;
; SET UP THE PARAMETER LIST FOR WRITE KEYSTROKE
;
MOV  WKPARLST.WKRETNCD,0H ; RETURN CODE MUST BE 0 FOR THE CALL
MOV  WKPARLST.WKFXNID,0H ; FUNCTION ID MUST BE 0 FOR THE CALL
MOV  AL,SESSID           ; SESSION ID INTO THE LIST
MOV  WKPARLST.WKSESSID,AL ;
MOV  AX,TASKID           ; CONNECTOR'S TASK ID INTO THE LIST
MOV  WKPARLST.WKTASKID,AX ;

;
; IF SENDING ONE KEYSTROKE
;
MOV  AL,SCANCD           ; PUT THE SCAN CODE INTO THE
MOV  WKPARLST.WKSCNCOD,AL ; PARAMETER LIST
MOV  AL,SHIFST           ; PUT SHIFT STATE INTO THE
MOV  WKPARLST.WKSHFST,AL ; PARAMETER LIST
;
MOV  AL,20H              ; PUT THE OPTION BYTE FOR SENDING SCAN
MOV  WKPARLST.WKOPTION,AL ; CODES ONE CHARACTER IN THE PARM LIST
;
; IF SENDING A LIST OF KEYSTROKES
;
MOV  AX,OFFSET LIST      ; PUT THE OFFSET ADDRESS OF THE LIST
MOV  WKPARLST.WKLSTOFF,AX ; INTO THE PARAMETER LIST
MOV  AX,SEG LIST         ; PUT THE SEGMENT ADDRESS OF THE LIST
MOV  WKPARLST.WKLSTSEG,AX ; INTO THE PARAMETER LIST
;
MOV  AL,30H              ; PUT THE OPTION BYTE FOR SENDING A
MOV  WKPARLST.WKOPTION,AL ; LIST OF CHARS. IN THE PARM LIST
```

```
;
; SET UP THE REGISTERS FOR WRITE KEYSTROKE
;
      MOV  AH,09H
      MOV  AL,04H
      MOV  BH,40H
      MOV  BL,40H
      MOV  CX,00H
      MOV  DX,KEYBOARD      ; RESOLVED VALUE FOR 'KEYBOARD'
      MOV  DI, SEG WKPRLST   ; GET SEGMENT ADDRESS OF PARM LIST
      MOV  ES,DI             ; AND PUT IT IN ES
      MOV  DI, OFFSET WKPRLST ; SET DI TO OFFSET OF PARM LIST
;
; SIGNAL WORKSTATION PROGRAM FOR WRITE KEYSTROKE SERVICE
;
      INT    7AH
      .
      .
      .
```

Disable Input

Keyboard Service X'05': Disable Input

Use this service to disable operator input to the session.

Register Values

On Request

AH = X'09'
AL = X'05'
BH = X'80'
BL = X'20'
CX = X'0000'
DX = Resolved value for KEYBOARD
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers
AX, BX, DX, ES, and DI
are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'62')
2	1 byte	Session ID	Unchanged
3	1 byte	Reserved	Reserved
4	1 word	Connector's task ID	Unchanged

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session whose keyboard you want to disable.
- The connector's task ID is needed only if the task that requested the Connect to Keyboard service for this session is different from the task requesting the Disable Input service. This parameter is optional and should be set to zero if not used.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Keyboard Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the keyboard management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Keyboard services return codes use a function ID of X'62'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for keyboard services.
X'0C'	Byte 0 of the parameter list is not zero on request.

See Appendix H, "Return Codes," for more information.

Usage Notes

- Keystrokes typed on the keyboard of a session can become intermixed with the keystroke data that your application program is sending to that session. To prevent this, you can disable the processing of keystrokes typed on the keyboard of the session by using this service.
- Using the Disable Input service on the work station control session causes the work station control session to become active with operator inputs disabled.

Disable Input

Coding Example

```
;
; DEFINE PARAMETER LIST FOR DISABLE INPUT
;
DIRCODE    DB 0                      ; RETURN CODE
DIFXNID    DB 0                      ; FUNCTION CODE
DISESID    DB 0                      ; SESSION ID
DIRESRVD   DB 0                      ; RESERVED
DICONNID   DW 0                      ; CONNECTORS TASK ID

.
.
.

;
; INITIALIZE PARAMETER LIST FOR DISABLE INPUT
;
        MOV     AL,SESSID              ; KEYBOARD INPUT DISABLED FOR
        MOV     DISESID,AL             ; THIS SESSION
        MOV     DIRCODE,00H           ; RETURN CODE MUST=0 ON REQUEST
        MOV     DIFXNID,00H           ; FUNCTION ID MUST=0 ON REQUEST
                                           ; IF THERE IS A CONNECTORS ID
        MOV     AX,CONNID              ; THEN PUT IT IN THE
        MOV     DICONNID,AX           ; PARAMETER LIST

;
; INITIALIZE REGISTERS FOR DISABLE INPUT
;
        MOV     AH,09H
        MOV     AL,05H
        MOV     BH,80H
        MOV     BL,20H
        MOV     CX,00H
        MOV     DX,KEYBOARD           ; RESOLVED VALUE FOR 'KEYBOARD'
        MOV     DI,SEG DIRCODE        ; ADDRESSABILITY OF
        MOV     ES,DI                 ; PARAMETER LIST
        MOV     DI,OFFSET DIRCODE     ; USING ES:DI

;
; SIGNAL WORKSTATION PROGRAM FOR DISABLE INPUT SERVICE
;
        INT     7AH
.
.
.
```

Keyboard Service X'06': Enable Input

Use this service to reenable operator input to the session.

Register Values

On Request

AH = X'09'
AL = X'06'
BH = X'80'
BL = X'20'
CX = X'0000'
DX = Resolved value for KEYBOARD
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

 The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'62')
2	1 byte	Session ID	Unchanged
3	1 byte	Reserved	Reserved
4	1 word	Connector's task ID	Unchanged

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session whose keyboard you want to enable.
- The connector's task ID is needed only if the task that requested the Connect to Keyboard service for this session is different from the task requesting the Enable Input service. This parameter is optional and should be set to zero if not used.

Enable Input

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Keyboard Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the keyboard management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Keyboard services return codes use a function ID of X'62'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for keyboard requests.
X'0C'	Byte 0 of the parameter list is not zero on request.

See Appendix H, "Return Codes," for more information.

Usage Notes

- The Disconnect from Keyboard service also enables operator input to a session if it was previously disabled.
- Using the Enable Input service on the work station control session causes the work station control session to become inactive.

Coding Example

```

;
; DEFINE PARAMETER LIST FOR ENABLE INPUT
;
EIRCODE      DB 0                      ; RETURN CODE
EIFXNID      DB 0                      ; FUNCTION ID
EISESID      DB 0                      ; SESSID
EIRESRVD     DB 0                      ; RESERVED
EICONNID     DW 0                      ; CONNECTOR'S TASK ID

.
.
.

;
; INITIALIZE PARAMETER LIST FOR ENABLE INPUT
;
      MOV     AL,SESSID                ; KEYBOARD INPUT ENABLED FOR
      MOV     EISESID,AL              ; THIS SESSION
      MOV     EIRCODE,00H             ; RETURN CODE MUST=0 ON REQUEST
      MOV     EIFXNID,00H             ; FUNCTION ID MUST=0 ON REQUEST
                                      ; IF THERE IS A CONNECTOR'S ID
      MOV     AX,CONNID               ; THEN STORE THE ID
      MOV     EICONNID,AX             ; IN THE PARAMETER LIST

;
; INITIALIZE REGISTERS FOR ENABLE INPUT
;
      MOV     AH,09H
      MOV     AL,06H
      MOV     BX,80H
      MOV     BL,20H
      MOV     CX,00H
      MOV     DX,KEYBOARD             ; RESOLVED VALUE FOR 'KEYBOARD'
      MOV     DI,SEG EIRCODE          ; ADDRESSABILITY OF
      MOV     ES,DI                  ; PARAMETER LIST
      MOV     DI,OFFSET EIRCODE       ; USING ES:DI

;
; SIGNAL WORKSTATION PROGRAM FOR ENABLE INPUT SERVICE
;
      INT     7AH
.
.
.

```

Post Status Code

Keyboard Service X'07': Post Status Code

Use this service to notify the workstation program of the status of your application program's keystroke processing.

Register Values

On Request

AH = X'09'
AL = X'07'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for KEYBOARD
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'62')
2	1 byte	Session ID	Unchanged
3	1 byte	Reserved	Reserved
4	1 word	Connector's task ID	Unchanged
6	1 byte	Status code	Unchanged

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session from which keystroke data was intercepted.
- The connector's task ID is needed only if the task that requested the Connect to Keyboard service for this session is different from the task requesting the Post Status Code service. This parameter is optional and should be set to zero if not used.

- The status codes that can be sent to the workstation program are:

Code	Meaning
X'00'	The keystroke was accepted.
X'01'	The last keystroke sent was detected to be from an AID key.
X'02'	The keystroke was rejected or could not be processed. When the workstation program receives this status code, it signals the operator with a beep to indicate invalid input.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Keyboard Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the keyboard management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Keyboard services return codes use a function ID of X'62'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for keyboard services.
X'0C'	Byte 0 of the parameter list is not zero on request.

See Appendix H, "Return Codes," for more information.

Usage Notes

- Your application program should request the Post Status Code service after it has processed a keystroke that was received by the Read Input request and was not passed on to the original session for which the keystroke was intended.
- The primary usage of the Post Status Code service is to provide audible feedback to the user in case of rejected keystrokes.
- For applications providing full keystroke services for newly created sessions, the Post Status Code service must be issued after every Read Input function to provide full compatibility with the workstation program. The use of all three status codes listed above at the correct times allows for proper operation of the keyboard services and autokey.

Post Status Code

Coding Example

```
;
; PARAMETER LIST FOR POST STATUS CODE
;
PSRETNCD DB 0 ; RETURN CODE
PSFXNID DB 0 ; FUNCTION NUMBER
PSSESSID DB 0 ; SESSION ID
PSRESERV DB 0 ; RESERVED
PSTASKID DW 0 ; CONNECTOR'S TASK ID
PSRETCOD DB 0 ; RETURN CODE TO BE POSTED

.
.
.

;
; INITIALIZE PARAMETER LIST FOR POST STATUS CODE
;
MOV PSRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV PSFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID INTO THE LIST
MOV PSSESSID,AL
MOV AX,TASKID ; CONNECTOR'S TASK ID INTO THE LIST
MOV PSTASKID,AX
MOV AL,RETCODE ; RETURN CODE INTO THE LIST
MOV PSRETCOD,AL

;
; INITIALIZE REGISTERS FOR POST STATUS CODE
;
MOV AH,09H
MOV AL,07H
MOV BH,80H
MOV BL,20H
MOV CX,OFFH
MOV DX,KEYBOARD ; RESOLVED VALUE FOR 'KEYBOARD'
MOV DI, SEG PSRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET PSRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR POST STATUS CODE SERVICE
;
INT 7AH
.
.
.
```

Chapter 6. Coding Window Management Service Requests

Introduction	6-2
Requesting the Window Management Services	6-5
Return Codes for the Window Management Services	6-6
Window Management Service X'01': Connect to Work Station Control	6-7
Window Management Service X'02': Disconnect from Work Station Control	6-11
Window Management Service X'03': Add Window	6-14
Window Management Service X'04': Change Window Position on Screen	6-17
Window Management Service X'05': Change Window Size	6-21
Window Management Service X'06': Change Window Color	6-25
Window Management Service X'07': Change Window Position on Presentation Space	6-29
Window Management Service X'08': Change Hidden State	6-33
Window Management Service X'09': Change Enlarge State	6-36
Window Management Service X'0A': Change Screen Background ...	6-38
Window Management Service X'0B': Query Window Position on Screen	6-41
Window Management Service X'0C': Query Window Size	6-44
Window Management Service X'0D': Query Window Colors	6-47
Window Management Service X'0E': Query Window Position on Presentation Space	6-51
Window Management Service X'0F': Query Hidden State	6-54
Window Management Service X'10': Query Enlarge State	6-57
Window Management Service X'11': Query Screen Background Color	6-60
Window Management Service X'12': Query Window Names	6-63
Window Management Service X'13': Clear Screen	6-66
Window Management Service X'14': Select Active Window	6-69
Window Management Service X'15': Redraw Screen	6-72
Window Management Service X'16': Redraw Window	6-75
Window Management Service X'17': Delete Window	6-78
Window Management Service X'18': Query Active Window	6-81
Window Management Service X'19': Query Active Screen	6-84
Window Management Service X'1A': Query Window Attributes	6-87
Window Management Service X'1B': Change Window Attributes ...	6-92
Window Management Service X'1C': Select Active Screen	6-98

Introduction

This chapter describes how to code requests for the window management services provided by the API.

The window management services allow your application program to use the functions of the work station control session of the IBM 3270 Personal Computer. Using these services, your application program can determine the current size, position, or color of a window, and change them if desired. You can jump to specified windows, enlarge or hide windows, or change to a different screen profile. You can add a window, delete a window, or clear the entire screen.

Your application program must request the Connect to Work Station Control service before it can request any of the other window management services. After the Connect to Work Station Control service has been completed successfully, the keyboard is attached to the work station control session and the operator information area (OIA) is displayed on the bottom line of the screen, with the following symbols:

- WSCtrl
- viz x clock (X [] on non-3270 PC hardware)
- WINDOW = name
- SCREEN = number

Note: When using work station control API with non-3270 PC hardware, the WS Ctrl OIA will not be displayed on either a Uni-DOS or Multi-DOS system under the following conditions:

- Your application uses graphics mode
- Your application uses 40-column mode
- Your application writes directly to the screen.

When your application program is finished using the window management services, it must request the Disconnect from Work Station Control service. If an error occurs during the execution of an application program that has connected to the work station control session, the OIA remains visible and the keyboard remains attached to the work station control session. You can press the Quit key (Alt plus Reset) to force termination of the connection to the work station control session. The program that was connected to the work station control session will continue to run, but any window management service requests that it makes will fail with an error code indicating that the program is not connected to the work station control session.

Possible problems can occur by the simultaneous use of the keyboard service requests to connect to the work station control keyboard and the use of the window management service requests.

- If the keyboard for the work station control session is redirected to another session, the use of the Quit key to force termination of the connection to the work station control session for window management services will be lost.
- If keystrokes are sent to the work station control session while window management service requests are being made, those keystrokes will effectively be lost except for the Quit key, which will terminate the connection to the work station control session for window management services.

The window management services provided by the API are:

- **Connect to Work Station Control Service:** Use this service to connect to the work station control session, to be able to use the window management services.
- **Disconnect from Work Station Control Service:** Use this service to disconnect from the work station control session.
- **Add Window Service:** Use this service to add a window from screen profile 0 to the specified screen profile.
- **Change Window Position on Screen Service:** Use this service to change the position of a window on the specified screen profile. The window's new position is determined by placing the upper left corner of the window at the specified row and column numbers.
- **Change Window Size Service:** Use this service to change the size of a window on the specified screen profile. The window's new size is determined by the specified number of rows and columns.
- **Change Window Color Service:** Use this service to change the foreground and background colors of a window on the specified screen profile.
- **Change Window Position on Presentation Space Service:** Use this service to change the position of a window on the presentation space for the specified screen profile. The window's new position is determined by placing the upper left corner of the window at the specified row and column numbers.
- **Change Hidden State Service:** Use this service to toggle the "hidden" state of a window on the specified screen profile. A hidden window becomes not hidden, or a window that is not hidden becomes hidden.

- **Change Enlarge State Service:** Use this service to toggle the “enlarge” state of the display image. An enlarged display image becomes normal, or a normal display image becomes enlarged.
- **Change Screen Background Service:** Use this service to change the background color of the specified screen profile.
- **Query Window Position on Screen Service:** Use this service to obtain the position of a window on the specified screen profile. The window’s position is given by the row and column numbers of the upper left corner of the window.
- **Query Window Size Service:** Use this service to obtain the size of a window on the specified screen profile. The window’s size is given as the number of rows and columns in the window.
- **Query Window Colors Service:** Use this service to obtain the foreground and background colors of a window on the specified screen profile.
- **Query Window Position on Presentation Space Service:** Use this service to obtain the position of a window on the specified screen profile. The window’s position is given by the row and columns of the upper left corner of the window.
- **Query Hidden State Service:** Use this service to obtain the “hidden” state of a window on the specified profile. The hidden state tells whether the window is hidden or not hidden.
- **Query Enlarge State Service:** Use this service to obtain the “enlarge” state of the display image. The display image can be either enlarged or not enlarged. In an enlarged image, the active window is displayed on the entire screen.
- **Query Screen Background Color Service:** Use this service to obtain the background color of the specified screen profile.
- **Query Window Names Service:** Use this service to obtain the short names of all windows in the specified screen profile.
- **Clear Screen Service:** Use this service to delete all windows from the specified screen profile. Windows cannot be deleted from screen profile 0.
- **Select Active Window Service:** Use this service to select a window on the specified screen profile to become the active window.
- **Redraw Screen Service:** Use this service to redraw the specified screen profile, if it is the active screen.
- **Redraw Window Service:** Use this service to redraw a window on the specified screen profile, if it is the active screen.

- **Delete Window Service:** Use this service to delete a window from the specified screen profile. Windows cannot be deleted from screen profile 0.
- **Query Active Window Service:** Use this service to obtain the short name of the active window in the specified screen profile.
- **Query Active Screen Service:** Use this service to obtain the number of the active screen profile.
- **Query Window Attributes Service:** Use this service to obtain the following information about a window on the specified screen profile:
 - Number of rows and columns in the window
 - Row and column number of the upper left corner of the window on the screen
 - Window colors and border colors
 - Control flags
 - Row and column number of the upper left corner of the window on the presentation space.
- **Change Window Attributes Service:** Use this service to change the following information about a window on the specified screen profile:
 - Number of rows and columns in the window
 - Row and column number of the upper left corner of the window on the screen
 - Window colors and border colors
 - Control flags
 - Row and column number of the upper left corner of the window on the presentation space.
- **Select Active Screen Service:** Use this service to make active the specified screen profile.

Requesting the Window Management Services

To request any of the window management services, load the registers and the parameter list with the proper values, and use the INT 7AH instruction to signal the workstation program that it has a request to process.

Note: Before your application can request the window management services, it must request the Name Resolution service, using 'WSCTRL' as the gate name in the parameter list. (Remember that the gate name must be padded to the right with blanks if it is less than eight characters.)

Return Codes for the Window Management Services

Each window management service has two return codes associated with it: a system return code and a window management return code. Both types of return codes are two-byte values made up of a function ID and an error number. The function ID indicates the portion of the workstation program in which the error occurred. The error number indicates the specific type of error that has occurred. An error number of X'00' always indicates a successful acceptance or completion of the request.

- **System Return Codes:**

After your application has requested a window management service, the CH and CL registers contain a return code generated by the request processing portion of the workstation program. The function ID is in the CH register, and the error number is in the CL register. System return codes use a function ID of X'12'. The error codes that can appear are:

Code	Meaning
X'00'	Request accepted.
X'05'	Invalid index specified.
X'07'	Invalid reply specified.
X'08'	Invalid wait type specified.
X'0B'	RQE pool depleted.
X'0F'	Invalid environment access.
X'34'	Invalid gate entry.

These system return codes apply to all window management services.

- **Window Management Services Return Codes:**

After a requested window management service is completed, bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error numbers that can appear are specific to the service that was requested and are included in the descriptions of each service.

See Appendix H, "Return Codes," for more information.

Window Management Service X'01': Connect to Work Station Control

Use this service to connect to the work station control session, to be able to use the window management services.

Register Values

On Request

AH = X'09'
AL = X'01'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCTRL
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session requesting the use of the window management services.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

Connect to Work Station Control

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	A session is already connected for window management services.
X'0C'	Byte 0 of the parameter list is not zero on request.

See Appendix H, "Return Codes," for more information.

Usage Notes

- Only one session can be connected for window management services at a time.
- When the connection is completed, the following symbols will appear in the operator information area (OIA):
 - WSCTRL
 - viz X CLOCK (X [] on non-3270 PC hardware)
 - WINDOW=name
 - SCREEN=number
 - When using work station control API with non-3270 PC hardware, the WS Ctrl OIA will not be displayed on either a Uni-DOS or Multi-DOS system under the following conditions:
 - Your application uses graphics mode
 - Your application uses 40-column mode
 - Your application writes directly to the screen.
 - While your application program is using the window management services, the screen and windows are not redrawn. The screen is redrawn when your application requests the Disconnect from Work Station Control service. In order to update the display screen while connected to the work station control session, your application must request either the Redraw Screen service or the Redraw Window service.

- While a session is connected to the work station control session, the keyboard belongs to the work station control session. All keystrokes typed at the keyboard are rejected except the Quit key, unless your application program has issued a Connect to Keyboard service request to the work station control session to intercept keystrokes. In this case, the keystrokes will be sent to your application program instead of to the work station control session.
- The Quit key allows the user to disconnect from the work station control session at any time. The Quit key can be used to enable the keyboard in the event that an application program finishes without disconnecting from the work station control session. When the Quit key is pressed, the application program is disconnected from the work station control session. Keystrokes typed at the keyboard are directed either to the active window, or to the work station control session if no windows exist on the active screen.

Note: The Quit key is active once your application program issues a Connect to Work Station Control service request. If the Quit key is pressed while the application program is running, any subsequent requests to window management services (including the Disconnect from Work Station Control service) fail with a return code indicating that the session is not connected to the work station control session.

- If your application program hangs, because of a problem in the application, the usual recovery sequence of pressing the Ctrl-Alt-Del keys causes a re-IPL of the entire workstation program. To recover only the session in which the faulty application program is running, you must first disconnect from the work station control session by the Quit key. Once the Quit key is pressed, the Ctrl-Alt-Del keystroke sequence re-IPLs the session without disrupting the rest of the sessions that are running.

Connect to Work Station Control

Coding Example

```
;
; PARAMETER LIST FOR CONNECT TO WORK STATION CONTROL
;
CWRETNCD DB 0 ; RETURN CODE
CWFXNID DB 0 ; FUNCTION NUMBER
CWSESSID DB 0 ; SESSION ID
.
.
.

;
; INITIALIZE PARAMETER LIST FOR CONNECT TO WORK STATION CONTROL
;
MOV CWRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV CWFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSION ; SESSION ID INTO PARAMETER LIST
MOV CWSESSID,AL

;
; INITIALIZE REGISTERS FOR CONNECT TO WORK STATION CONTROL
;
MOV AH,09H
MOV AL,01H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR 'WSCTRL '
MOV DI, SEG CWRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET CWRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR CONNECT TO WORK STATION CONTROL SERVICE
;
INT 7AH
.
.
.
```

Window Management Service X'02': Disconnect from Work Station Control

Use this service to disconnect from the work station control session.

Register Values

On Request

AH = X'09'
AL = X'02'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCtrl
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the work station control session.

Disconnect from Work Station Control

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'04'	The session is not connected for window management services.
X'0C'	Byte 0 of the parameter list is not zero on request.

See Appendix H, "Return Codes," for more information.

Usage Notes

- When the disconnect is completed, the following occurs:
 - The screen is redrawn.
 - The work station control OIA is removed from the screen, and keystrokes typed on the keyboard are directed to the active window.
 - If there are no windows on the active screen:
 - On 3270 PC hardware, the work station control OIA remains on the screen,
 - On non-3270 PC hardware, the work station control OIA appears on the screen. The prompt "Valid Keys: WSCTRL List Print A-Z 0-9 Jump ChgSc Quit" will appear.

Coding Example

```
;
; PARAMETER LIST FOR DISCONNECT FROM WORK STATION CONTROL
;
DWRETNCD DB 0 ; RETURN CODE
DWFXNID DB 0 ; FUNCTION ID
DWSSESSIONID DB 0 ; SESSION ID
.
.
.
;
; INITIALIZE PARAMETER LIST FOR DISCONNECT FROM WORK STATION CONTROL
;
MOV DWRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV DWFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSIONID ; SESSION ID OBTAINED FROM REQUEST
MOV DWSSESSIONID,AL ; TO QUERY SESSION ID SERVICE
;
; INITIALIZE REGISTERS FOR DISCONNECT FROM WORK STATION CONTROL
;
MOV AH,09H
MOV AL,02H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR 'WSCTRL'
MOV DI,SEG DWRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET DWRETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR DISCONNECT FROM WORK STATION CONTROL
; SERVICE
;
INT 7AH
.
.
.
```

Add Window

Window Management Service X'03': Add Window

Use this service to add a window from screen profile 0 to the specified screen profile.

Register Values

On Request

AH = X'09'
AL = X'03'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCTRL
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Screen profile number	Unchanged
4	1 byte	Window short name	Unchanged

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the workstation control session.
- The screen profile number is the number (in ASCII) of the screen profile that you are adding the window to. Windows cannot be added to screen profile 0.
- The window short name is the 1-character ASCII name for the window being added. Window short names must be alphabetic characters (A through Z).

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'01'	The maximum number of windows has already been reached (no free WCBs).
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'05'	The window already exists on screen.
X'06'	Invalid screen ID.
X'09'	The window is not on screen 0.
X'0C'	Byte 0 of the parameter list is not zero on request.

See Appendix H, "Return Codes," for more information.

Usage Notes

- The added window becomes the active window.
- Windows cannot be added to screen profile 0.

Add Window

Coding Example

```
;
; PARAMETER LIST FOR ADD WINDOW
;
AWRETNCD DB 0 ; RETURN CODE
AWFXNID DB 0 ; FUNCTION ID
AWSESSID DB 0 ; SESSION ID
AWSCRPRO DB 0 ; SCREEN PROFILE NUMBER IN ASCII
AWWINDN DB 0 ; WINDOW SHORT NAME IN ASCII

.
.
.

;
; INITIALIZE PARAMETER LIST FOR ADD WINDOW
;
MOV AWRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV AWFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID
MOV AWSESSID,AL ; IN PARAMETER LIST
MOV AL,'1' ; SCREEN PROFILE NUMBER 1
MOV AWSCRPRO,AL ; IN PARAMETER LIST
MOV AL,'P' ; WINDOW SHORT NAME 'P'
MOV AWWINDN,AL ; IN PARAMETER LIST

;
; INITIALIZE REGISTERS FOR ADD WINDOW
;
MOV AH,09H
MOV AL,03H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR 'WSCTRL'
MOV DI,SEG AWRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET AWRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR ADD WINDOW SERVICE
;
INT 7AH
.
.
.
```

Window Management Service X'04': Change Window Position on Screen

Use this service to change the position of a window on the specified screen profile. The window's new position is determined by placing the upper left corner of the window at the row and column numbers specified in the parameter list.

Register Values

On Request

AH = X'09'
AL = X'04'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCtrl
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Screen profile number	Unchanged
4	1 byte	Window short name	Unchanged
5	1 byte	Row	Unchanged, or row
6	1 byte	Column	Unchanged, or column

Change Window Position on Screen

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the workstation control session.
- The screen profile number is the number (in ASCII) of the screen profile containing the specified window.
- The window short name is the 1-character ASCII name for the window being changed. Window short names must be alphabetic characters.
- “Row” is the new row position for the upper left corner of the window on the screen.
- “Column” is the new column position for the upper left corner of the window on the screen.

Note: Row and column numbers start at zero.

Completion Parameters:

- “Row” is the row number chosen by the workstation program when the row number in the parameter list on request caused the window to overlap the screen boundaries.
- “Column” is the column number chosen by the workstation program when the column number in the parameter list on request caused the window to overlap the screen boundaries.

Note: Row and column numbers start at zero.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'06'	Invalid screen ID.
X'07'	The window was not found on screen.
X'0C'	Byte 0 of the parameter list is not zero on request.
X'0E'	No windows exist on screen.
X'10'	The window overlapped the screen boundaries or did not fit on the screen.

See Appendix H, "Return Codes," for more information.

Usage Notes

- If the window overlaps the screen boundaries after it has been moved to the new position:
 - The window is moved to fit on the screen.
 - A return code of X'10' is returned in the parameter list.
 - The row and column numbers of the window position chosen by the workstation program are returned in the parameter list.

Change Window Position on Screen

Coding Example

```
;
;  PARAMETER LIST FOR CHANGE WINDOW POSITION ON SCREEN
;
CSRETNCDB DB 0 ; RETURN CODE
CSFXNID DB 0 ; FUNCTION NUMBER
CSSESSID DB 0 ; SESSION ID
CSSCREEN DB 0 ; SCREEN PROFILE NUMBER
CSWINDOW DB 0 ; WINDOW SHORT NAME
CSROW DB 0 ; ROW POSITION OF UPPER LEFT CORNER
; OF WINDOW
CSCOLUMN DB 0 ; COLUMN POSITION OF UPPER LEFT
; CORNER OF WINDOW
.
.
.
;
;  INITIALIZE PARAMETER LIST FOR CHANGE WINDOW POSITION ON SCREEN
;
MOV CSRETNCDB,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV CSFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID INTO THE LIST
MOV CSSESSID,AL
MOV CSSCREEN,'2' ; SCREEN NUMBER 2
MOV CSWINDOW,'B' ; WINDOW 'B' SHORT NAME
MOV AL,ROW ; ROW POSITION INTO THE LIST
MOV CSROW,AL
MOV AL,COL ; COLUMN POSITION INTO THE LIST
MOV CSCOLUMN,AL
;
;  INITIALIZE REGISTERS FOR CHANGE WINDOW POSITION ON SCREEN
;
MOV AH,09H
MOV AL,04H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR 'WSCTRL'
MOV DI,SEG CSRETNCDB ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET CSRETNCDB ; OFFSET OF PARAMETER LIST IN DI
;
;  SIGNAL WORKSTATION PROGRAM FOR CHANGE WINDOW POSITION ON SCREEN SERVICE
;
INT 7AH
.
.
```

Window Management Service X'05': Change Window Size

Use this service to change the size of a window on the specified screen profile. The window's new size is determined by the number of rows and columns specified in the parameter list.

Register Values

On Request

AH = X'09'
AL = X'05'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCTRL
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Screen profile number	Unchanged
4	1 byte	Window short name	Unchanged
5	1 byte	Rows	Unchanged, or rows
6	1 byte	Columns	Unchanged, or columns

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the work station control session.
- The screen profile number is the number (in ASCII) of the screen profile containing the specified window.
- The window short name is the 1-character ASCII name for the window being changed. Window short names must be alphabetic characters.

Change Window Size

- “Rows” is the number of rows in the new window size.
- “Columns” is the number of columns in the new window size.

Completion Parameters:

- “Rows” is the number of rows chosen by the workstation program when the number of rows in the parameter list on request caused the window to become too big to fit on the screen or the presentation space.
- “Columns” is the number of columns chosen by the workstation program when the number of columns in the parameter list on request caused the window to become too big to fit on the screen or the presentation space.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'06'	Invalid screen ID.
X'07'	The window was not found on screen.
X'0C'	Byte 0 of the parameter list is not zero on request.
X'0E'	No windows exist on the screen.
X'10'	The window overlapped the screen boundaries or did not fit on the screen, because of the row and column values sent in the parameter list. If the window size was too big, the number of rows and columns chosen by the workstation program is returned in the parameter list.

See Appendix H, “Return Codes,” for more information.

Usage Notes

- A value of 0 for either the number of rows or the number of columns in the window size is changed by the workstation program to be a value of 1.
- If the window overlaps the screen boundaries after it has been changed to the new size:
 - The window is moved to fit on the screen.
 - A return code of X'10' is returned in the parameter list.
- If the window overlaps the presentation space boundaries after it has been changed to the new size:
 - The window is moved to fit on the presentation space.
 - A return code of X'10' is returned in the parameter list.
- If the window is too big to fit on the screen or the presentation space after it has been changed to the new size:
 - The window size is reduced, and the window position is changed (if necessary), to allow the window to fit on the screen and the presentation space.
 - A return code of X'10' is returned in the parameter list.
 - The number of rows and columns in the window size chosen by the workstation program is returned in the parameter list.

Change Window Size

Coding Example

```
;
; PARAMETER LIST FOR CHANGE WINDOW SIZE
;
CZRETNCD DB 0 ; RETURN CODE
CZFXNID DB 0 ; FUNCTION ID
CZSESSID DB 0 ; SESSION ID
CZSCRPRO DB 0 ; SCREEN PROFILE NUMBER IN ASCII
CZWINDN DB 0 ; WINDOW SHORT NAME IN ASCII
CZNUMROW DB 0 ; NUMBER OF ROWS IN NEW WINDOW SIZE
CZNUMCOL DB 0 ; NUMBER OF COLUMNS IN NEW WINDOW SIZE

.
.
.

;
; INITIALIZE PARAMETER LIST FOR CHANGE WINDOW SIZE
;
        MOV     CZRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
        MOV     CZFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
        MOV     AL,SESSID ; SESSION ID OBTAINED FROM REQUEST
        MOV     CZSESSID,AL ; TO QUERY SESSION ID
        MOV     AL,'1' ; SCREEN PROFILE NUMBER 1
        MOV     CZSCRPRO,AL ; IN PARAMETER LIST
        MOV     AL,'P' ; WINDOW SHORT NAME 'P'
        MOV     CZWINDN,AL ; IN PARAMETER LIST
        MOV     AL,10 ; NUMBER OF ROWS IN THE NEW
        MOV     CZNUMROW,AL ; WINDOW SIZE
        MOV     AL,15 ; NUMBER OF COLUMNS IN THE
        MOV     CZNUMCOL,AL ; WINDOW SIZE

;
; INITIALIZE REGISTERS FOR CHANGE WINDOW SIZE
;
        MOV     AH,09H
        MOV     AL,05H
        MOV     BH,80H
        MOV     BL,20H
        MOV     CX,0FFH
        MOV     DX,WSCTRL ; RESOLVED VALUE FOR 'WSCTRL'
        MOV     DI,SEG CZRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
        MOV     ES,DI ; IN ES
        MOV     DI,OFFSET CZRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR CHANGE WINDOW SIZE SERVICE
;
        INT     7AH
.
.
.
```

Window Management Service X'06': Change Window Color

Use this service to change the foreground and background colors of a window on the specified screen profile.

Register Values

On Request

AH = X'09'
AL = X'06'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCTRL
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

 The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Screen profile number	Unchanged
4	1 byte	Window short name	Unchanged
5	1 byte	Foreground color value	Unchanged
6	1 byte	Background color value	Unchanged
7	1 byte	Base color	Unchanged

Change Window Color

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the work station control session.
- The screen profile number is the number (in ASCII) of the screen profile containing the specified window.
- The window short name is the 1-character ASCII name for the window being changed. Window short names must be alphabetic characters.
- The foreground and background color values are as follows:

Value		Color
0	=	Black
1	=	Blue
2	=	Red
3	=	Pink
4	=	Green
5	=	Turquoise
6	=	Yellow
7	=	White

If the foreground or background color value is greater than 7, the color value is selected using the formula: $\text{color} = \text{value} \text{ MOD } 8$.

- The base color is specified as follows:
 - A value of 1 indicates that the base colors are to be used to display the window.
 - A value other than 1 indicates that the specified foreground and background colors are to be used to display the window.

Note: Setting base colors for the window overrides the color values specified for the foreground and background.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
------	---------

X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'06'	Invalid screen ID.
X'07'	The window is not found on screen.
X'0C'	Byte 0 of the parameter list is not zero on request.
X'0E'	No windows exist on screen.
X'0F'	No colors can be set for a PC session.
X'12'	Foreground and background colors are the same. The window name and the window border become black on white.

See Appendix H, "Return Codes," for more information.

Change Window Color

Coding Example

```
;
; PARAMETER LIST FOR CHANGE WINDOW COLOR
;
CCRETNCD DB 0 ; RETURN CODE
CCFXNID DB 0 ; FUNCTION NUMBER
CCSESSID DB 0 ; SESSION ID
CCSCREEN DB 0 ; SCREEN PROFILE NUMBER
CCWINDOW DB 0 ; WINDOW SHORT NAME
CCFORGND DB 0 ; FOREGROUND COLOR VALUE
CCBAKGND DB 0 ; BACKGROUND COLOR VALUE
CCBASE DB 0 ; BASE COLOR

.
.
.

;
; INITIALIZE PARAMETER LIST FOR CHANGE WINDOW COLOR
;
MOV CCRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV CCFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID INTO PARAMETER LIST
MOV CCSESSID,AL ;
MOV CCSCREEN,'3' ; SCREEN NUMBER 3
MOV CCWINDOW,'S' ; WINDOW 'S' SHORT NAME
MOV CCFORGND,4 ; GREEN FOREGROUND
MOV CCBAKGND,0 ; BLACK BACKGROUND
MOV CCBASE,0 ; NO BASE COLORS

;
; INITIALIZE REGISTERS FOR CHANGE WINDOW COLOR
;
MOV AH,09H
MOV AL,06H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR 'WSCTRL '
MOV DI,SEG CCRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET CCRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR CHANGE WINDOW COLOR SERVICE
;
INT 7AH
.
.
.
```

Window Management Service X'07': Change Window Position on Presentation Space

Use this service to change the position of a window on the presentation space for the specified screen profile. The window's new position is determined by placing the upper left corner of the window at the row and column numbers specified in the parameter list.

Register Values

On Request

AH = X'09'
AL = X'07'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCTRL
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Screen profile number	Unchanged
4	1 byte	Window short name	Unchanged
5	1 byte	Row	Unchanged, or row
6	1 byte	Column	Unchanged, or column

Change Window Position on Presentation Space

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the workstation control session.
- The screen profile number is the number (in ASCII) of the screen profile containing the specified window.
- The window short name is the 1-character ASCII name for the window being changed. Window short names must be alphabetic characters.
- “Row” is the new row position for the upper left corner of the window on the presentation space.
- “Column” is the new column position for the upper left corner of the window on the presentation space.

Note: Row and column numbers start at zero.

Completion Parameters:

- “Row” is the row number chosen by the workstation program when the row number in the parameter list on request caused the window to overlap the presentation space boundaries.
- “Column” is the column number chosen by the workstation program when the column number in the parameter list on request caused the window to overlap the presentation space boundaries.

Note: Row and column numbers start at zero.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'06'	Invalid screen ID.
X'07'	The window is not found on screen.
X'0C'	Byte 0 of the parameter list is not zero on request.
X'0E'	No windows exist on screen.
X'10'	The window overlapped the presentation space boundaries or did not fit on the presentation space.

See Appendix H, "Return Codes," for more information.

Usage Notes

- If the window overlaps the presentation space boundaries after it has been moved to the new position:
 - The window is moved to fit on the presentation space.
 - A return code of X'10' is returned in the parameter list.
 - The row and column numbers of the window position chosen by the workstation program are returned in the parameter list.
- This service accepts only row and column positions as parameters, not PEL positions such as those used by windows in graphics mode.
- This service is similar to the Browse function on the keyboard. When the window becomes the top window of the screen and the screen is active, if the cursor is not within the area shown by the window, the window will be moved on the presentation space until the cursor is within the window area.

Change Window Position on Presentation Space

Coding Example

```
;
; PARAMETER LIST FOR CHANGE WINDOW POSITION ON PRESENTATION SPACE
;
CPRETNCD DB 0 ; RETURN CODE
CPFXNID DB 0 ; FUNCTION ID
CPSESSID DB 0 ; SESSION ID
CPSCRPRO DB 0 ; SCREEN PROFILE NUMBER IN ASCII
CPWINDN DB 0 ; WINDOW SHORT NAME IN ASCII
CPROWNUM DB 0 ; ROW NUMBER FOR NEW POSITION OF
; UPPER LEFT CORNER OF WINDOW
CPCOLNUM DB 0 ; COLUMN NUMBER FOR NEW POSITION OF
; UPPER LEFT CORNER OF WINDOW
.
.
.
;
; INITIALIZE PARAMETER LIST FOR CHANGE WINDOW POSITION ON PRESENTATION SPACE
;
MOV CPRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV CPFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID OBTAINED FROM REQUEST
MOV CPSESSID,AL ; TO QUERY SESSION ID SERVICE
MOV AL,'1' ; SCREEN PROFILE NUMBER 1
MOV CPSCRPRO,AL ; IN PARAMETER LIST
MOV AL,'P' ; WINDOW SHORT NAME 'P'
MOV CPWINDN,AL ; IN PARAMETER LIST
MOV AL,20 ; ROW NUMBER FOR NEW POSITION OF
MOV CPROWNUM,AL ; UPPER LEFT CORNER OF WINDOW
MOV AL,25 ; COLUMN NUMBER FOR NEW POSITION OF
MOV CPCOLNUM,AL ; UPPER LEFT CORNER
;
; INITIALIZE REGISTERS FOR CHANGE WINDOW POSITION ON PRESENTATION SPACE
;
MOV AH,09H
MOV AL,07H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR WSCTRL
MOV DI,SEG CPRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET CPRETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR CHANGE WINDOW POSITION ON PRESENTATION
; SPACE SERVICE
;
INT 7AH
.
.
.
```

Window Management Service X'08': Change Hidden State

Use this service to toggle the “hidden” state of a window on the specified screen profile. (A hidden window becomes not hidden, or a window that is not hidden becomes hidden.)

Register Values

On Request

AH = X'09'
AL = X'08'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCtrl
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Screen profile number	Unchanged
4	1 byte	Window short name	Unchanged

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the workstation control session.
- The screen profile number is the number (in ASCII) of the screen profile containing the specified window.
- The window short name is the 1-character ASCII name for the window being changed. Window short names must be alphabetic characters.

Change Hidden State

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'06'	Invalid screen ID.
X'07'	The window is not found on screen.
X'0A'	Only one window on screen; the window cannot be hidden.
X'0B'	All other windows are hidden; the next window has been unhidden and made the active window.
X'0C'	Byte 0 of the parameter list is not zero on request.
X'0E'	No windows exist on screen.

See Appendix H, "Return Codes," for more information.

Usage Notes

- If the requested window is the active window and all other windows are hidden, the next window on the screen profile becomes not hidden and is made the active window.

Coding Example

```

;
; PARAMETER LIST FOR CHANGE HIDDEN STATE
;
CHRETNCD DB 0 ; RETURN CODE
CHFXNID DB 0 ; FUNCTION NUMBER
CHSESSID DB 0 ; SESSION ID
CHSCREEN DB 0 ; SCREEN PROFILE NUMBER
CHWINDOW DB 0 ; WINDOW SHORT NAME

.
.
.

;
; INITIALIZE PARAMETER LIST FOR CHANGE HIDDEN STATE
;
MOV CHRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV CHFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID INTO THE
MOV CHSESSID,AL ; PARAMETER LIST
MOV CHSCREEN,'5' ; SCREEN NUMBER 5
MOV CHWINDOW,'T' ; WINDOW 'T' SHORT NAME

;
; INITIALIZE REGISTERS FOR CHANGE HIDDEN STATE
;
MOV AH,09H
MOV AL,08H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR 'WSCTRL '
MOV DI, SEG CHRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET CHRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR CHANGE HIDDEN STATE SERVICE
;
INT 7AH
.
.
.

```

Change Enlarge State

Window Management Service X'09': Change Enlarge State

Use this service to toggle the “enlarge” state of the display image. (An enlarged display image becomes normal, or a normal display image becomes enlarged.)

Register Values

On Request

AH = X'09'
AL = X'09'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCTRL
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the workstation control session.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'0C'	Byte 0 of the parameter list is not zero on request.

See Appendix H, "Return Codes," for more information.

Coding Example

```
;
; PARAMETER LIST FOR CHANGE ENLARGE STATE
;
CERETNCD DB 0 ; RETURN CODE
CEFXNID DB 0 ; FUNCTION ID
CESESSID DB 0 ; SESSION ID
.
.
.
;
; INITIALIZE PARAMETER LIST FOR CHANGE ENLARGE STATE
;
MOV CERETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV CEFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID OBTAINED FROM REQUEST
MOV CESESSID,AL ; TO QUERY SESSION ID SERVICE
;
; INITIALIZE REGISTERS FOR CHANGE ENLARGE STATE
;
MOV AH,09H
MOV AL,09H
MOV BH,80H
MOV BL,20H
MOV CX,OFFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR WSCTRL
MOV DI,SEG CERETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET CERETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR CHANGE ENLARGE STATE SERVICE
;
INT 7AH
.
.
.
```

Change Screen Background

Window Management Service X'0A': Change Screen Background

Use this service to change the background color of the specified screen profile.

Register Values

On Request

AH = X'09'
AL = X'0A'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCTRL
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Screen profile number	Unchanged
4	1 byte	Screen background color	Unchanged

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the workstation control session.
- The screen profile number is the number (in ASCII) of the screen profile being changed.

- The background color values are as follows:

Value		Color
0	=	Black
1	=	Blue
2	=	Red
3	=	Pink
4	=	Green
5	=	Turquoise
6	=	Yellow
7	=	White

If the background color value is greater than 7, the color value is selected using the formula: $\text{color} = \text{value} \text{ MOD } 8$.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'06'	Invalid screen ID.
X'0C'	Byte 0 of the parameter list is not zero on request.

See Appendix H, "Return Codes," for more information.

Change Screen Background

Coding Example

```
;
; PARAMETER LIST FOR CHANGE SCREEN BACKGROUND
;
CBRETNCD DB 0 ; RETURN CODE
CBFXNID DB 0 ; FUNCTION NUMBER
CBSESSID DB 0 ; SESSION ID
CBSCREEN DB 0 ; SCREEN PROFILE NUMBER
CBCOLOR DB 0 ; SCREEN BACKGROUND COLOR

.
.
.

;
; INITIALIZE PARAMETER LIST FOR CHANGE SCREEN BACKGROUND
;
MOV CBRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV CBFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID INTO THE
MOV CBSESSID,AL ; PARAMETER LIST
MOV CBSCREEN,'0' ; SCREEN NUMBER 0
MOV CBCOLOR,7 ; BACKGROUND COLOR = WHITE

;
; INITIALIZE REGISTERS FOR CHANGE SCREEN BACKGROUND
;
MOV AH,09H
MOV AL,0AH
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR 'WSCTRL '
MOV DI, SEG CBRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET CBRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR CHANGE SCREEN BACKGROUND SERVICE
;
INT 7AH
.
.
.
```

Window Management Service X'0B': Query Window Position on Screen

Use this service to obtain the position of a window on the specified screen profile. The window's position is given by the row and column numbers of the upper left corner of the window.

Register Values

On Request

AH = X'09'
AL = X'0B'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCTRL
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Screen profile number	Unchanged
4	1 byte	Window short name	Unchanged
5	1 byte	Reserved	Row
6	1 byte	Reserved	Column

Query Window Position on Screen

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the workstation control session.
- The screen profile number is the number (in ASCII) of the screen profile containing the specified window.
- The window short name is the 1-character ASCII name for the window being queried. Window short names must be alphabetic characters.

Completion Parameters:

- "Row" is the row number of the upper left corner of the window on the screen.
- "Column" is the column number of the upper left corner of the window on the screen.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'06'	Invalid screen ID.
X'07'	The window is not found on screen.
X'0C'	Byte 0 of the parameter list is not zero on request.
X'0E'	No windows exist on screen.

See Appendix H, "Return Codes," for more information.

Coding Example

```
;
; PARAMETER LIST FOR QUERY WINDOW POSITION ON SCREEN
;
QQRETNCD DB 0 ; RETURN CODE
QQFXNID DB 0 ; FUNCTION ID
QQSESSID DB 0 ; SESSION ID
QQSCRPRO DB 0 ; SCREEN PROFILE NUMBER IN ASCII
QQWINDN DB 0 ; WINDOW SHORT NAME IN ASCII
QQROWNUM DB 0 ; ROW NUMBER OF UPPER LEFT CORNER
QQCOLNUM DB 0 ; COLUMN NUMBER OF UPPER LEFT CORNER
.
.
.
;
; INITIALIZE PARAMETER LIST FOR QUERY WINDOW POSITION ON SCREEN
;
MOV QQRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV QQFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID OBTAINED FROM REQUEST
MOV QQSESSID,AL ; TO QUERY SESSION ID SERVICE
MOV AL,'1' ; SCREEN PROFILE NUMBER
MOV QQSCRPRO,AL ; IN PARAMETER LIST
MOV AL,'P' ; WINDOW SHORT NAME OBTAINED FROM
MOV QQWINDN,AL ; REQUEST TO QUERY SESSION ID SERVICE
;
; INITIALIZE REGISTERS FOR QUERY WINDOW POSITION ON SCREEN
;
MOV AH,09H
MOV AL,0BH
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR WSCTRL
MOV DI,SEG QQRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET QQRETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR QUERY WINDOW POSITION ON SCREEN SERVICE
;
INT 7AH
.
.
.
```

Window Management Service X'0C': Query Window Size

Use this service to obtain the size of a window on the specified screen profile. The window's size is given as the number of rows and columns in the window.

Register Values

On Request

AH = X'09'
AL = X'0C'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCTRL
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Screen profile number	Unchanged
4	1 byte	Window short name	Unchanged
5	1 byte	Reserved	Rows
6	1 byte	Reserved	Columns

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the workstation control session.
- The screen profile number is the number (in ASCII) of the screen profile containing the specified window.
- The window short name is the 1-character ASCII name for the window being queried. Window short names must be alphabetic characters.

Completion Parameters:

- “Rows” is the number of rows in the window size.
- “Columns” is the number of columns in the window size.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'06'	Invalid screen ID.
X'07'	The window is not found on screen.
X'0C'	Byte 0 of the parameter list is not zero on request.
X'0E'	No windows exist on screen.

See Appendix H, “Return Codes,” for more information.

Query Window Size

Coding Example

```
;
; PARAMETER LIST FOR QUERY WINDOW SIZE
;
QZRETNCD DB 0 ; RETURN CODE
QZFXNID DB 0 ; FUNCTION NUMBER
QZSESSID DB 0 ; SESSION ID
QZSCREEN DB 0 ; SCREEN PROFILE NUMBER
QZWINDOW DB 0 ; WINDOW SHORT NAME
QZROWS DB 0 ; NUMBER OF ROWS IN WINDOW SIZE
QZCOLUMNS DB 0 ; NUMBER OF COLUMNS IN WINDOW SIZE

.
.
.

;
; INITIALIZE PARAMETER LIST FOR QUERY WINDOW SIZE
;
MOV QZRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV QZFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID INTO THE
MOV QZSESSID,AL ; PARAMETER LIST
MOV QZSCREEN,'4' ; SCREEN NUMBER 4
MOV QZWINDOW,'D' ; WINDOW 'D' SHORT NAME INTO THE LIST

;
; INITIALIZE REGISTERS FOR QUERY WINDOW SIZE
;
MOV AH,09H
MOV AL,0CH
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR 'WSCTRL '
MOV DI,SEG QZRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET QZRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR QUERY WINDOW SIZE SERVICE
;
INT 7AH
.
.
.
```

Window Management Service X'0D': Query Window Colors

Use this service to obtain the foreground and background colors of a window on the specified screen profile.

Register Values

On Request

AH = X'09'
AL = X'0D'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCtrl
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Screen profile number	Unchanged
4	1 byte	Window short name	Unchanged
5	1 byte	Reserved	Foreground color
6	1 byte	Reserved	Background color
7	1 byte	Reserved	Color flag

Query Window Colors

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the work station control session.
- The screen profile number is the number (in ASCII) of the screen profile containing the specified window.
- The window short name is the 1-character ASCII name for the window being queried. Window short names must be alphabetic characters.

Completion Parameters:

- The foreground and background color values are as follows:

Value		Color
0	=	Black
1	=	Blue
2	=	Red
3	=	Pink
4	=	Green
5	=	Turquoise
6	=	Yellow
7	=	White

- The color flag is as follows:

0	=	Normal (foreground/background color values apply)
1	=	Base mode (base colors apply)
2	=	PC session

Note: If the color flag value is 1 or 2, the foreground and background color values are both 0.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'06'	Invalid screen ID.
X'07'	The window is not found on screen.
X'0C'	Byte 0 of the parameter list is not zero on request.
X'0E'	No windows exist on screen.

See Appendix H, "Return Codes," for more information.

Query Window Colors

Coding Example

```
;
; PARAMETER LIST FOR QUERY WINDOW COLORS
;
QCRETNCD DB 0 ; RETURN CODE
QCFXNID DB 0 ; FUNCTION ID
QCSESSID DB 0 ; SESSION ID
QCSCRPRO DB 0 ; SCREEN PROFILE NUMBER IN ASCII
QCWINDN DB 0 ; WINDOW SHORT NAME IN ASCII
QCFORCOL DB 0 ; FOREGROUND COLOR
QCBKCOL DB 0 ; BACKGROUND COLOR
QCCOLFLG DB 0 ; COLOR FLAG

.
.
.

;
; INITIALIZE PARAMETER LIST FOR QUERY WINDOW COLORS
;
MOV QCRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV QCFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID OBTAINED FROM REQUEST
MOV QCSESSID,AL ; TO QUERY SESSION ID SERVICE
MOV AL,'1' ; SCREEN PROFILE NUMBER 1
MOV QCSCRPRO,AL ; IN PARAMETER LIST
MOV AL,'P' ; WINDOW SHORT NAME 'P'
MOV QCWINDN,AL ; IN PARAMETER LIST

;
; INITIALIZE REGISTERS FOR QUERY WINDOW COLORS
;
MOV AH,09H
MOV AL,0DH
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR WSCTRL
MOV DI,SEG QCRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET QCRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR QUERY WINDOW COLORS SERVICE
;
INT 7AH
.
.
.
```

Window Management Service X'0E': Query Window Position on Presentation Space

Use this service to obtain the position of a window on the presentation space for the specified screen profile. The window's position is given by the row and column numbers of the upper left corner of the window.

Register Values

On Request

AH = X'09'
AL = X'0E'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCtrl
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Screen profile number	Unchanged
4	1 byte	Window short name	Unchanged
5	1 byte	Reserved	Row
6	1 byte	Reserved	Column

Query Window Position on Presentation Space

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the workstation control session.
- The screen profile number is the number (in ASCII) of the screen profile containing the specified window.
- The window short name is the 1-character ASCII name for the window being queried. Window short names must be alphabetic characters.

Completion Parameters:

- “Row” is the row number of the upper left corner of the window on the presentation space.
- “Column” is the column number of the upper left corner of the window on the presentation space.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'06'	Invalid screen ID.
X'07'	The window is not found on screen.
X'0C'	Byte 0 of the parameter list is not zero on request.
X'0E'	No windows exist on screen.

See Appendix H, “Return Codes,” for more information.

Coding Example

```
;
; PARAMETER LIST FOR QUERY WINDOW POSITION ON PRESENTATION SPACE
;
QPRETNCD DB 0 ; RETURN CODE
QPFYNID DB 0 ; FUNCTION NUMBER
QPSESSID DB 0 ; SESSION ID
QPSCREEN DB 0 ; SCREEN PROFILE NUMBER
QPWINDOW DB 0 ; WINDOW SHORT NAME
QPROW DB 0 ; ROW NUMBER OF UPPER LEFT CORNER
QPCOLUMN DB 0 ; COLUMN NUMBER OF UPPER LEFT CORNER

.
.
.

;
; INITIALIZE PARAMETER LIST FOR QUERY WINDOW POSITION ON
; PRESENTATION SPACE
;
MOV QPRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV QPFYNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID INTO THE
MOV QPSESSID,AL ; PARAMETER LIST
MOV QPSCREEN,'3' ; SCREEN NUMBER 3
MOV QPWINDOW,'J' ; WINDOW 'J' SHORT NAME

;
; INITIALIZE REGISTERS FOR QUERY WINDOW POSITION ON
; PRESENTATION SPACE
;
MOV AH,09H
MOV AL,0EH
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR 'WSCTRL '
MOV DI, SEG QPRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET QPRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR QUERY WINDOW POSITION ON
; PRESENTATION SPACE SERVICE
;
INT 7AH
.
.
.
```

Window Management Service X'0F': Query Hidden State

Use this service to obtain the “hidden” state of a window on the specified screen profile. (The hidden state tells whether the window is hidden or not hidden.)

Register Values

On Request

AH = X'09'
AL = X'0F'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCTRL
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers
AX, BX, DX, ES, and DI
are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Screen profile number	Unchanged
4	1 byte	Window short name	Unchanged
5	1 byte	Reserved	“Hidden” flag

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the workstation control session.
- The screen profile number is the number (in ASCII) of the screen profile containing the specified window.
- The window short name is the 1-character ASCII name for the window being queried. Window short names must be alphabetic characters.

Completion Parameters:

- The “hidden” flag is as follows:
 - A value of X'01' in the “hidden” flag means that the window is hidden.
 - A value of X'00' in the “hidden” flag means that the window is not hidden.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'06'	Invalid screen ID.
X'07'	The window is not found on screen.
X'0C'	Byte 0 of the parameter list is not zero on request.
X'0E'	No windows exist on screen.

See Appendix H, “Return Codes,” for more information.

Query Hidden State

Coding Example

```
;
; PARAMETER LIST FOR QUERY HIDDEN STATE
;
QHRETNCD DB 0 ; RETURN CODE
QHFXNID DB 0 ; FUNCTION ID
QHSESSID DB 0 ; SESSION ID
QHSCRPRO DB 0 ; SCREEN PROFILE NUMBER IN ASCII
QHWINDN DB 0 ; WINDOW SHORT NAME IN ASCII
QHHIDFLG DB 0 ; HIDDEN FLAG

.
.
.

;
; INITIALIZE PARAMETER LIST FOR QUERY HIDDEN STATE
;
MOV QHRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV QHFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID OBTAINED FROM REQUEST
MOV QHSESSID,AL ; TO QUERY SESSION ID SERVICE
MOV AL,'1' ; SCREEN PROFILE NUMBER
MOV QHSCRPRO,AL ; IN PARAMETER LIST
MOV AL,'P' ; WINDOW SHORT NAME
MOV QHWINDN,AL ; IN PARAMETER LIST

;
; INITIALIZE REGISTERS FOR QUERY HIDDEN STATE
;
MOV AH,09H
MOV AL,0FH
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR WSCTRL
MOV DI, SEG QHRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET QHRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR QUERY HIDDEN STATE SERVICE
;
INT 7AH
.
.
.
```

Window Management Service X'10': Query Enlarge State

Use this service to obtain the “enlarge” state of the display image. (The display image can be either enlarged or not enlarged. In an enlarged image, the active window is displayed on the entire screen.)

Register Values

On Request

AH = X'09'
AL = X'10'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCtrl
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code
 The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Reserved	“Enlarge” flag

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the workstation control session.

Completion Parameters:

- The “enlarge” flag is as follows:
 - A value of X'01' in the “enlarge” flag means that the display image is enlarged.
 - A value of X'00' in the “enlarge” flag means that the display image is not enlarged.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'0C'	Byte 0 of the parameter list is not zero on request.

See Appendix H, “Return Codes,” for more information.

Coding Example

```

;
; PARAMETER LIST FOR QUERY ENLARGE STATE
;
QERETNCD DB 0 ; RETURN CODE
QEFXNID DB 0 ; FUNCTION NUMBER
QESSESSID DB 0 ; SESSION ID
QEENLFLG DB 0 ; ENLARGE FLAG

.
.
.

;
; INITIALIZE PARAMETER LIST FOR QUERY ENLARGE STATE
;
MOV QERETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV QEFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID INTO THE
MOV QESESSID,AL ; PARAMETER LIST

;
; INITIALIZE REGISTERS FOR QUERY ENLARGE STATE
;
MOV AH,09H
MOV AL,10H
MOV BH,80H
MOV BL,20H
MOV CX,OFFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR 'WSCTRL '
MOV DI, SEG QERETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET QERETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR QUERY ENLARGE STATE SERVICE
;
INT 7AH
.
.
.

```

Query Screen Background Color

Window Management Service X'11': Query Screen Background Color

Use this service to obtain the background color of the specified screen profile.

Register Values

On Request

AH = X'09'
AL = X'11'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCTRL
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Screen profile number	Unchanged
4	1 byte	Reserved	Screen background color

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the work station control session.
- The screen profile number is the number (in ASCII) of the screen profile being queried.

Completion Parameters:

- The background color values are as follows:

Value		Color
0	=	Black
1	=	Blue
2	=	Red
3	=	Pink
4	=	Green
5	=	Turquoise
6	=	Yellow
7	=	White

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'06'	Invalid screen ID.
X'0C'	Byte 0 of the parameter list is not zero on request.

See Appendix H, "Return Codes," for more information.

Query Screen Background Color

Coding Example

```
;
; PARAMETER LIST FOR QUERY SCREEN BACKGROUND COLOR
;
QBRETNCD DB 0 ; RETURN CODE
QBFXNID DB 0 ; FUNCTION ID
QBSESSID DB 0 ; SESSION ID
QBSCRPRO DB 0 ; SCREEN PROFILE NUMBER IN ASCII
QBBACOL DB 0 ; SCREEN BACKGROUND COLOR

.
.
.

;
; INITIALIZE PARAMETER LIST FOR QUERY SCREEN BACKGROUND COLOR
;
MOV QBRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV QBFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID OBTAINED FROM REQUEST
MOV QBSESSID,AL ; TO QUERY SESSION ID SERVICE
MOV AL,'1' ; SCREEN PROFILE NUMBER
MOV QBSCRPRO,AL ; IN PARAMETER LIST

;
; INITIALIZE REGISTERS FOR QUERY SCREEN BACKGROUND COLOR
;
MOV AH,09H
MOV AL,11H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR WSCTRL
MOV DI,SEG QBRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET QBRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR QUERY SCREEN BACKGROUND COLOR SERVICE
;
INT 7AH
.
.
.
```

Window Management Service X'12': Query Window Names

Use this service to obtain the short names of all windows in the specified screen profile.

Register Values

On Request

AH = X'09'
AL = X'12'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCtrl
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code
 The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Screen profile number	Unchanged
4	1 byte	Reserved	Window short name #1
5	1 byte	Reserved	Window short name #2
6	1 byte	Reserved	Window short name #3
⋮			
21	1 byte	Reserved	Window short name #18
22	1 byte	Reserved	Window short name #19
23	1 byte	Reserved	Window short name #20

Query Window Names

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the workstation control session.
- The screen profile number is the number (in ASCII) of the screen profile being queried.

Completion Parameters:

- The window short name is the 1-character ASCII name for the window on the specified screen profile. Window short names are uppercase alphabetic characters.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'06'	Invalid screen ID.
X'0C'	Byte 0 of the parameter list is not zero on request.
X'0E'	No windows exist on the screen.

See Appendix H, "Return Codes," for more information.

Usage Notes

- The windows are listed in the parameter list in the order they appear on the screen.
- The parameter list is filled with blanks (X'20') after all window names have been given.

Coding Example

```

;
; PARAMETER LIST FOR QUERY WINDOW NAMES
;
QWRETNCD DB 0 ; RETURN CODE
QWFXNID DB 0 ; FUNCTION NUMBER
QWSESSID DB 0 ; SESSION ID
QWSCREEN DB 0 ; SCREEN PROFILE NUMBER
QWWNDLST DB 20 DUP(0) ; LIST OF WINDOW SHORT NAMES

.
.
.

;
; INITIALIZE PARAMETER LIST FOR QUERY WINDOW NAMES
;
MOV QWRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV QWFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID INTO THE
MOV QWSESSID,AL ; PARAMETER LIST
MOV QWSCREEN,'1' ; SCREEN NUMBER 1

;
; INITIALIZE REGISTERS FOR QUERY WINDOW NAMES
;
MOV AH,09H
MOV AL,12H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR 'WSCTRL '
MOV DI,SEG QWRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET QWRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR QUERY WINDOW NAMES SERVICE
;
INT 7AH
.
.
.

```


Clear Screen

Window Management Service X'13': Clear Screen

Use this service to delete all windows from the specified screen profile.
Windows cannot be deleted from screen profile 0.

Register Values

On Request

AH = X'09'
AL = X'13'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCtrl
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers
AX, BX, DX, ES, and DI
are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Screen profile number	Unchanged

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the workstation control session.
- The screen profile number is the number (in ASCII) of the screen profile being cleared.

Return Codes

- **System Return Codes:**

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- **Window Management Services Return Codes:**

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'06'	Invalid screen ID.
X'0C'	Byte 0 of the parameter list is not zero on request.

See Appendix H, "Return Codes," for more information.

Usage Notes

- Windows cannot be deleted from screen profile 0.

Clear Screen

Coding Example

```
;
; PARAMETER LIST FOR CLEAR SCREEN
;
CLRETNCD DB 0 ; RETURN CODE
CLFXNID DB 0 ; FUNCTION ID
CLSESSID DB 0 ; SESSION ID
CLSCRPRO DB 0 ; SCREEN PROFILE NUMBER IN ASCII
.
.
.
;
; INITIALIZE PARAMETER LIST FOR CLEAR SCREEN
;
MOV CLRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV CLFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID OBTAINED FROM REQUEST
MOV CLSESSID,AL ; TO QUERY SESSION ID SERVICE
MOV AL,'1' ; SCREEN PROFILE NUMBER
MOV CLSCRPRO,AL ; IN PARAMETER LIST
;
; INITIALIZE REGISTERS FOR CLEAR SCREEN
;
MOV AH,09H
MOV AL,13H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR WSCTRL
MOV DI,SEG CLRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET CLRETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR CLEAR SCREEN SERVICE
;
INT 7AH
.
.
.
```

Window Management Service X'14': Select Active Window

Use this service to select a window on the specified screen profile to become the active window.

Register Values

On Request

AH = X'09'
AL = X'14'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCtrl
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code
 The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Screen profile number	Unchanged
4	1 byte	Window short name	Unchanged

Select Active Window

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the workstation control session.
- The screen profile number is the number (in ASCII) of the screen profile containing the specified window.
- The window short name is the 1-character ASCII name for the window being made active. Window short names must be alphabetic characters.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'06'	Invalid screen ID.
X'07'	The window is not found on screen.
X'0C'	Byte 0 of the parameter list is not zero on request.
X'0E'	No windows exist on screen.

See Appendix H, "Return Codes," for more information.

Coding Example

```

;
; PARAMETER LIST FOR SELECT ACTIVE WINDOW
;
ACRETNCD DB 0 ; RETURN CODE
ACFXNID DB 0 ; FUNCTION NUMBER
ACSESSID DB 0 ; SESSION ID
ACSCREEN DB 0 ; SCREEN PROFILE NUMBER
ACWINDOW DB 0 ; WINDOW SHORT NAME

.
.
.

;
; INITIALIZE PARAMETER LIST FOR SELECT ACTIVE WINDOW
;
MOV ACRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV ACFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID INTO THE
MOV ACSESSID,AL ; PARAMETER LIST
MOV ASCREEN,'1' ; SCREEN NUMBER 1
MOV ACWINDOW,'C' ; WINDOW 'C' SHORT NAME

;
; INITIALIZE REGISTERS FOR SELECT ACTIVE WINDOW
;
MOV AH,09H
MOV AL,14H
MOV BH,80H
MOV BL,20H
MOV CX,OFFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR 'WSCTRL '
MOV DI, SEG ACRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET ACRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR SELECT ACTIVE WINDOW SERVICE
;
INT 7AH
.
.
.

```

Redraw Screen

Window Management Service X'15': Redraw Screen

Use this service to redraw the specified screen profile if it is the active screen.

Register Values

On Request

AH = X'09'
AL = X'15'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCtrl
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Screen profile number	Unchanged

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the workstation control session.
- The screen profile number is the number (in ASCII) of the screen profile being redrawn.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'06'	Invalid screen ID.
X'0C'	Byte 0 of the parameter list is not zero on request.
X'0D'	This screen is not the active screen.

See Appendix H, "Return Codes," for more information.

Usage Notes

- This service is necessary if the position or size of any window on the screen has been changed, or if a window has been enlarged, so that the change becomes visible.
- The Disconnect from Work Station Control service also redraws the screen.

Redraw Screen

Coding Example

```
;
; PARAMETER LIST FOR REDRAW SCREEN
;
RSRETNCD DB 0 ; RETURN CODE
RSFXNID DB 0 ; FUNCTION ID
RSSESSID DB 0 ; SESSION ID
RSSCRPRO DB 0 ; SCREEN PROFILE NUMBER IN ASCII
.
.
.
;
; INITIALIZE PARAMETER LIST FOR REDRAW SCREEN
;
MOV RSRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV RSFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID OBTAINED FROM REQUEST
MOV RSSESSID,AL ; TO QUERY SESSION ID
MOV AL,'1' ; SCREEN PROFILE NUMBER
MOV RSSCRPRO,AL ; IN PARAMETER LIST
;
; INITIALIZE REGISTERS FOR REDRAW SCREEN
;
MOV AH,09H
MOV AL,15H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR WSCTRL
MOV DI,SEG RSRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET RSRETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR REDRAW SCREEN SERVICE
;
INT 7AH
.
.
.
```

Window Management Service X'16': Redraw Window

Use this service to redraw a window on the specified screen profile if it is the active screen.

Register Values

On Request

AH = X'09'
AL = X'16'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCTRL
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

 The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Screen profile number	Unchanged
4	1 byte	Window short name	Unchanged

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the workstation control session.
- The screen profile number is the number (in ASCII) of the screen profile containing the specified window.
- The window short name is the 1-character ASCII name for the window being redrawn. Window short names must be alphabetic characters.

Redraw Window

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'06'	Invalid screen ID.
X'07'	The window is not found on screen.
X'0C'	Byte 0 of the parameter list is not zero on request.
X'0D'	Requested screen not active.
X'0E'	No windows exist on screen.

See Appendix H, "Return Codes," for more information.

Usage Notes

- This service is necessary to make the change visible on the screen when the contents of a window or its colors have changed, but the position or size of the window has not changed. The Redraw Screen service and the Disconnect from Work Station Control service have the same function, except that they redraw the entire screen.

Coding Example

```

;
; PARAMETER LIST FOR REDRAW WINDOW
;
RWRETNCD DB 0 ; RETURN CODE
RWFYNID DB 0 ; FUNCTION NUMBER
RWSESSID DB 0 ; SESSION ID
RWSCREEN DB 0 ; SCREEN PROFILE NUMBER
RWWINDOW DB 0 ; WINDOW SHORT NAME

.
.
.

;
; INITIALIZE PARAMETER LIST FOR REDRAW WINDOW
;
MOV RWRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV RWFYNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID INTO THE
MOV RWSESSID,AL ; PARAMETER LIST
MOV RWSCREEN,'7' ; SCREEN NUMBER 7
MOV RWWINDOW,'E' ; WINDOW 'E' SHORT NAME

;
; INITIALIZE REGISTERS FOR REDRAW WINDOW
;
MOV AH,09H
MOV AL,16H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR 'WSCTRL '
MOV DI,SEG RWRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET RWRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR REDRAW WINDOW SERVICE
;
INT 7AH
.
.
.

```

Window Management Service X'17': Delete Window

Use this service to delete a window from the specified screen profile.
Windows cannot be deleted from screen profile 0.

Register Values

On Request

AH = X'09'
AL = X'17'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCTRL
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers
AX, BX, DX, ES, and DI
are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Screen profile number	Unchanged
4	1 byte	Window short name	Unchanged

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the workstation control session.
- The screen profile number is the number (in ASCII) of the screen profile containing the specified window.
- The window short name is the 1-character ASCII name for the window being deleted. Window short names must be alphabetic characters.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'06'	Invalid screen ID.
X'07'	The window is not found on screen.
X'0C'	Byte 0 of the parameter list is not zero on request.
X'0E'	No windows exist on screen.

See Appendix H, "Return Codes," for more information.

Usage Notes

- Windows cannot be deleted from screen profile 0.
- If all the remaining windows on the specified screen profile are hidden, the next window on the chain will be unhidden and made the active window on the screen.

Query Active Window

Coding Example

```
;
; PARAMETER LIST FOR DELETE WINDOW
;
DDRETNCD DB 0 ; RETURN CODE
DDFXNID DB 0 ; FUNCTION ID
DDSESSID DB 0 ; SESSION ID
DDSCRPRO DB 0 ; SCREEN PROFILE NUMBER IN ASCII
DDWINDN DB 0 ; WINDOW SHORT NAME IN ASCII
.
.
.
;
; INITIALIZE PARAMETER LIST FOR DELETE WINDOW
;
MOV DDRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV DDFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID OBTAINED FROM REQUEST
MOV DDSESSID,AL ; TO QUERY SESSION ID SERVICE
MOV AL,'1' ; SCREEN PROFILE NUMBER
MOV DDSCRPRO,AL ; IN PARAMETER LIST
MOV AL,'P' ; WINDOW SHORT NAME
MOV DDWINDN,AL ; IN PARAMETER LIST
;
; INITIALIZE REGISTERS FOR DELETE WINDOW
;
MOV AH,09H
MOV AL,17H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR WSCTRL
MOV DI,SEG DDRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET DDRETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR DELETE WINDOW SERVICE
;
INT 7AH
.
.
.
```

Window Management Service X'18': Query Active Window

Use this service to obtain the short name of the active window in the specified screen profile.

Register Values

On Request

AH = X'09'
AL = X'18'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCtrl
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Screen profile number	Unchanged
4	1 byte	Reserved	Window short name

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the workstation control session.
- The screen profile number is the number (in ASCII) of the screen profile being queried.

Completion Parameters:

- The window short name is the 1-character ASCII name for the active window. Window short names are uppercase alphabetic characters.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'06'	Invalid screen ID.
X'0C'	Byte 0 of the parameter list is not zero on request.
X'0E'	No windows exist on the screen.

See Appendix H, "Return Codes," for more information.

Coding Example

```

;
; PARAMETER LIST FOR QUERY ACTIVE WINDOW
;
QNRETNCD DB 0 ; RETURN CODE
QNFXNID DB 0 ; FUNCTION NUMBER
QNSESSID DB 0 ; SESSION ID
QNSCREEN DB 0 ; SCREEN PROFILE NUMBER
QNWINDOW DB 0 ; ACTIVE WINDOW SHORT NAME

.
.
.

;
; INITIALIZE PARAMETER LIST FOR QUERY ACTIVE WINDOW
;
MOV QNRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV QNFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID INTO THE
MOV QNSESSID,AL ; PARAMETER LIST
MOV QNSCREEN,'0' ; SCREEN NUMBER 0

;
; INITIALIZE REGISTERS FOR QUERY ACTIVE WINDOW
;
MOV AH,09H
MOV AL,18H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR 'WSCTRL '
MOV DI, SEG QNRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET QNRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR QUERY ACTIVE WINDOW SERVICE
;
INT 7AH
.
.
.

```

Window Management Service X'19': Query Active Screen

Use this service to obtain the number of the active screen profile.

Register Values

On Request

AH = X'09'
AL = X'19'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCTRL
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers
AX, BX, DX, ES, and DI
are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Reserved	Screen profile number

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the workstation control session.

Completion Parameters:

- The screen profile number is the number (in ASCII) of the active screen profile.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'0C'	Byte 0 of the parameter list is not zero on request.

See Appendix H, "Return Codes," for more information.

Query Active Screen

Coding Example

```
;
; PARAMETER LIST FOR QUERY ACTIVE SCREEN
;
QARETNCD DB 0 ; RETURN CODE
QAFXNID DB 0 ; FUNCTION ID
QASESSID DB 0 ; SESSION ID
QASCRPRO DB 0 ; SCREEN PROFILE NUMBER IN ASCII
.
.
.
;
; INITIALIZE PARAMETER LIST FOR QUERY ACTIVE SCREEN
;
MOV QARETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV QAFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID OBTAINED FROM REQUEST
MOV QASESSID,AL ; TO QUERY SESSION ID
;
; INITIALIZE REGISTERS FOR QUERY ACTIVE SCREEN
;
MOV AH,09H
MOV AL,19H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR WSCTRL
MOV DI,SEG QARETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET QARETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR QUERY ACTIVE SCREEN SERVICE
;
INT 7AH
.
.
.
```

Window Management Service X'1A': Query Window Attributes

Use this service to obtain the following information about a window on the specified screen profile:

- The number of rows in the window
- The number of columns in the window
- The row number of the upper left corner of the window on the screen
- The column number of the upper left corner of the window on the screen
- Window colors
- Border colors
- Control flags
- The row number of the upper left corner of the window on the presentation space
- The column number of the upper left corner of the window on the presentation space.

Register Values

On Request

AH = X'09'
AL = X'1A'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCTRL
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers
AX, BX, DX, ES, and DI
are unpredictable.

Query Window Attributes

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Screen profile number	Unchanged
4	1 byte	Window short name	Unchanged
5	1 byte	Reserved	Rows
6	1 byte	Reserved	Columns
7	1 byte	Reserved	Row on screen
8	1 byte	Reserved	Column on screen
9	1 byte	Reserved	Window colors
10	1 byte	Reserved	Border colors
11	1 byte	Reserved	Control flags
12	1 byte	Reserved	Row on PS
13	1 byte	Reserved	Column on PS

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the work station control session.
- The screen profile number is the number (in ASCII) of the screen profile containing the specified window.
- The window short name is the 1-character ASCII name for the window being queried. Window short names must be alphabetic characters.

Completion Parameters:

- "Rows" is the hexadecimal number of rows in the window.
- "Columns" is the hexadecimal number of columns in the window.
- "Row on screen" is the row position of the upper left corner of the window on the screen.
- "Column on screen" is the column position of the upper left corner of the window on the screen.

- The window colors are specified as follows:

0 and 1	2 through 4	5 through 7
Not used	Foreground color	Background color

The foreground and background color values are as follows:

Value	Color
0	= Black
1	= Blue
2	= Red
3	= Pink
4	= Green
5	= Turquoise
6	= Yellow
7	= White

- The border colors will always be the same as the window colors (except where the window foreground and background colors are the same; in that case, the border colors will be white on black).
- The bits in the control flag are as follows:

0	1 and 2	3	4 and 5	6	7
Hidden	Reserved	Enlarge	Reserved	Base colors	Window colors

- Bit 0 set to 0 means that the window is not hidden.
If bit 0 is set to 1 and:
 1. Bit 3 is set to 0, the window is hidden.
 2. Bit 3 is set to 1, the window is not hidden (but will not be displayed on the screen, because the display is enlarged).
- Bit 6 set to 0 means that the session is not displayed in the base colors.
Bit 6 set to 1 means that the session is displayed in the base colors.
- Bit 7 set to 0 means that the session is not displayed in the foreground and background colors.
Bit 7 set to 1 means that the session is displayed in the foreground and background colors.

Note: Bits 6 and 7 cannot both be set to 1.

Query Window Attributes

- “Row on PS” is the row position of the upper left corner of the window on the presentation space.
- “Column on PS” is the column position of the upper left corner of the window on the presentation space.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'06'	Invalid screen ID.
X'07'	The window is not found on screen.
X'0C'	Byte 0 of the parameter list is not zero on request.
X'0E'	No windows exist on screen.

See Appendix H, “Return Codes,” for more information.

Coding Example

```

;
; PARAMETER LIST FOR QUERY WINDOW ATTRIBUTES
;
QTRETNCD DB 0 ; RETURN CODE
QTFXNID DB 0 ; FUNCTION NUMBER
QTSESSID DB 0 ; SESSION ID
QTSCREEN DB 0 ; SCREEN PROFILE NUMBER
QTWINDOW DB 0 ; WINDOW SHORT NAME
QTNUMROW DB 0 ; NUMBER OF ROWS IN THE WINDOW
QTNUMCOL DB 0 ; NUMBER OF COLUMNS IN THE WINDOW
QTLCRWSC DB 0 ; ROW NUMBER OF UPPER LEFT CORNER OF
; THE WINDOW ON THE SCREEN
QTLCLLSC DB 0 ; COLUMN NUMBER OF UPPER LEFT CORNER
; OF THE WINDOW ON THE SCREEN
QTWCOLOR DB 0 ; WINDOW COLOR ATTRIBUTES
QTBOLOR DB 0 ; BORDER COLOR ATTRIBUTES
QTCTLFLG DB 0 ; CONTROL FLAGS
QTLCRWPS DB 0 ; ROW NUMBER OF UPPER LEFT CORNER
; OF THE WINDOW ON THE
; PRESENTATION SPACE
QTLCLLPS DB 0 ; COLUMN NUMBER OF UPPER LEFT CORNER
; OF THE WINDOW ON THE
; PRESENTATION SPACE

.
.
.
;
; INITIALIZE PARAMETER LIST FOR QUERY WINDOW ATTRIBUTES
;
MOV QTRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV QTFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID INTO THE
MOV QTSESSID,AL ; PARAMETER LIST
MOV QTSCREEN,'0' ; SCREEN NUMBER 0
MOV QTWINDOW,'M' ; WINDOW 'M' SHORT NAME
;
; INITIALIZE REGISTERS FOR QUERY WINDOW ATTRIBUTES
;
MOV AH,09H
MOV AL,1AH
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR 'WSCTRL '
MOV DI,SEG QTRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET QTRETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR QUERY WINDOW ATTRIBUTES SERVICE
;
INT 7AH
.
.
.

```

Window Management Service X'1B': Change Window Attributes

Use this service to change the following information about a window on the specified screen profile:

- The number of rows in the window
- The number of columns in the window
- The row number of the upper left corner of the window on the screen
- The column number of the upper left corner of the window on the screen
- Window colors
- Border colors
- Control flags
- The row number of the upper left corner of the window on the presentation space
- The column number of the upper left corner of the window on the presentation space.

Register Values

On Request

AH = X'09'
AL = X'1B'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCtrl
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers
AX, BX, DX, ES, and DI
are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Screen profile number	Unchanged
4	1 byte	Window short name	Unchanged
5	1 byte	Rows	Unchanged *
6	1 byte	Columns	Unchanged *
7	1 byte	Row on screen	Unchanged *
8	1 byte	Column on screen	Unchanged *
9	1 byte	Window colors	Unchanged
10	1 byte	Border colors	Unchanged
11	1 byte	Control flags	Unchanged
12	1 byte	Row on PS	Unchanged *
13	1 byte	Column on PS	Unchanged *
* These values may be changed by the workstation program. See "Usage Notes" for more information.			

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the workstation control session.
- The screen profile number is the number (in ASCII) of the screen profile containing the specified window.
- The window short name is the 1-character ASCII name for the window being changed. Window short names must be alphabetic characters.
- "Rows" is the number of rows in the window.
- "Columns" is the number of columns in the window.
- "Row on screen" is the row position of the upper left corner of the window on the screen.
- "Column on screen" is the column position of the upper left corner of the window on the screen.

Change Window Attributes

- The window colors are specified as follows:

0, 1	2 through 4	5 through 7
Not used	Foreground color	Background color

The foreground and background color values are as follows:

Value	Color
0	= Black
1	= Blue
2	= Red
3	= Pink
4	= Green
5	= Turquoise
6	= Yellow
7	= White

- The border colors will always be the same as the window colors (except where the window foreground and background colors are the same; in that case, the border colors will be white on black).

Note: If the window and border color attributes do not match, the border color will be changed to match the window colors.

- The bits in the control flag are as follows:

0	1 and 2	3	4 and 5	6	7
Hidden	Reserved	Enlarge	Reserved	Base colors	Window colors

- Bit 0 set to 0 means that the window is not hidden.
If bit 0 is set to 1 and:
 1. Bit 3 is set to 0, the window is hidden.
 2. Bit 3 is set to 1, the window is not hidden (but will not be displayed on the screen, because the display is enlarged).
- Bit 6 set to 0 means that the session is not displayed in the base colors.
Bit 6 set to 1 means that the session is displayed in the base colors.
- Bit 7 set to 0 means that the session is not displayed in the foreground and background colors.
Bit 7 set to 1 means that the session is displayed in the foreground and background colors.

Note: Bits 6 and 7 cannot both be set to 1.

- “Row on PS” is the row position of the upper left corner of the window on the presentation space.
- “Column on PS” is the column position of the upper left corner of the window on the presentation space.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'06'	Invalid screen ID.
X'07'	The window is not found on screen.
X'0C'	Byte 0 of the parameter list is not zero on request.
X'0E'	No windows exist on screen.
X'11'	One or more values sent in the parameter list are not valid.

See Appendix H, “Return Codes,” for more information.

Usage Notes

- If the window does not exist on the specified screen, it is added to the screen, with the attributes specified in the parameter list.
- A value of 0 for either the number of rows or the number of columns in the window size is changed by the workstation program to a value of 1.
- If the window overlaps the screen or presentation space boundaries after it has been moved to the new position:
 - The window is moved to fit on the screen.
 - A return code of X'11' is returned in the parameter list.
 - The row and column numbers of the window position chosen by the workstation program are returned in the parameter list.

Change Window Attributes

- If the window overlaps the screen boundaries after it has been changed to the new size:
 - The window is moved to fit on the screen.
 - A return code of X'11' is returned in the parameter list.
- If the window is too big to fit on the screen after it has been changed to the new size:
 - The window size is reduced, and the window position is changed (if necessary), to allow the window to fit on the screen.
 - A return code of X'11' is returned in the parameter list.
 - The number of rows and columns in the window size chosen by the workstation program is returned in the parameter list.
- If the "hidden" bit in the control flag is 1 and the window is the only window on the screen, the workstation program changes the bit setting to 0.
- If the "hidden" bit in the control flag is 1 and all the other windows on the specified screen profile are hidden, then the next window on the chain will become not hidden and will be made the active window.
- If any of the reserved bits in the control flag are set to 1, the workstation program changes the bit setting to 0.
- If both the "base colors" and "window colors" bits in the control flag are the same value (both 1 or both 0), "base colors" is used as the default setting.
- This service places the specified window at the bottom of the chain on the specified screen profile.

Coding Example

```
;
; PARAMETER LIST FOR CHANGE WINDOW ATTRIBUTES
;
CTRETNCD DB 0 ; RETURN CODE
CTFXNID DB 0 ; FUNCTION ID
CTSESSID DB 0 ; SESSION ID
CTSCRPRO DB 0 ; SCREEN PROFILE NUMBER IN ASCII
CTWINDN DB 0 ; WINDOW SHORT NAME IN ASCII
CTROWNUM DB 0 ; NUMBER OF ROWS IN THE WINDOW
CTCOLNUM DB 0 ; NUMBER OF COLUMNS IN THE WINDOW
CTROWSCR DB 0 ; ROW NUMBER AND COLUMN NUMBER OF THE
CTCOLSCR DB 0 ; UPPER LEFT CORNER OF THE WINDOW
; ON THE SCREEN
CTWINDCO DB 0 ; WINDOW COLOR ATTRIBUTES
CTBORDCO DB 0 ; BORDER COLOR ATTRIBUTES
CTFLAG DB 0 ; FLAG BYTE
CTROWPS DB 0 ; ROW AND COLUMN NUMBER OF THE UPPER
CTCOLPS DB 0 ; LEFT CORNER OF THE WINDOW ON THE
; PRESENTATION SPACE
```

Change Window Attributes

```
.
.
.
;
; INITIALIZE PARAMETER LIST FOR CHANGE WINDOW ATTRIBUTES
;
MOV CTRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV CTFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID OBTAINED FROM REQUEST
MOV CTSESSID,AL ; TO QUERY SESSION ID
MOV AL,'1' ; SCREEN PROFILE NUMBER
MOV CTSCRPRO,AL ; IN PARAMETER LIST
MOV AL,'P' ; WINDOW SHORT NAME
MOV CTWINDN,AL ; IN PARAMETER LIST
MOV AL,10 ; NUMBER OF ROWS IN THE NEW
MOV CTROWNUM,AL ; WINDOW SIZE
MOV AL,10 ; NUMBER OF COLUMNS IN THE
MOV CTCOLNUM,AL ; WINDOW SIZE
MOV AL,15 ; ROW NUMBER AND COLUMN NUMBER OF THE
MOV CTROWSCR,AL ; UPPER LEFT CORNER OF THE
MOV AL,15 ; WINDOW ON SCREEN
MOV CTCOLSCR,AL ; IN THE PARAMETER LIST
MOV AL,00001000B ; FOREGROUND = BLUE AND
MOV CTWINDCO,AL ; BACKGROUND = BLACK
MOV AL,0 ; BORDER COLOR WILL BE THE SAME
MOV CTBORDCO,AL ; AS THE WINDOW COLOR
MOV AL,00000001B ; THE SESSION IS NOT HIDDEN AND IT
MOV CTFLAG,AL ; IS DISPLAYED IN FOREGROUND AND
; BACKGROUND COLORS
MOV AL,5 ; ROW NUMBER OF UPPER LEFT CORNER
MOV CTROWPS,AL ; OF THE WINDOW
; ON THE PRESENTATION SPACE
MOV AL,5 ; COLUMN NUMBER OF UPPER LEFT CORNER
MOV CTCOLPS,AL ; OF THE WINDOW
MOV CTROWPS,AL ; OF THE WINDOW
;
; INITIALIZE REGISTERS FOR CHANGE WINDOW ATTRIBUTES
;
MOV AH,09H
MOV AL,1BH
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,WSCTRL ; RESOLVED VALUE FOR WSCTRL
MOV DI,SEG CTRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET CTRETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR CHANGE WINDOW ATTRIBUTES SERVICE
;
INT 7AH
.
.
.
```


Select Active Screen

Window Management Service X'1C': Select Active Screen

Use this service to make the specified screen profile the active screen.

Register Values

On Request

AH = X'09'
AL = X'1C'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for WSCTRL
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'63')
2	1 byte	Session ID	Unchanged
3	1 byte	Screen profile number	Unchanged

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session currently connected to the workstation control session.
- The screen profile number is the number (in ASCII) of the screen profile being made active.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Window Management Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the window management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Window management return codes use a function ID of X'63'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'04'	The session is not connected for window management services.
X'06'	Invalid screen ID.
X'0C'	Byte 0 of the parameter list is not zero on request.

See Appendix H, "Return Codes," for more information.

Select Active Screen

Coding Example

```
;
; PARAMETER LIST FOR SELECT ACTIVE SCREEN
;
ASRETNCD DB 0 ; RETURN CODE
ASFXNID DB 0 ; FUNCTION NUMBER
ASSESSID DB 0 ; SESSION ID
ASSCREEN DB 0 ; SCREEN NUMBER

.
.
.

;
; INITIALIZE PARAMETER LIST FOR SELECT ACTIVE SCREEN
;
MOV ASRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV ASFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID INTO THE LIST
MOV ASSESSID,AL
MOV ASSCREEN,3 ; SCREEN NUMBER 3

;
; INITIALIZE REGISTERS FOR SELECT ACTIVE SCREEN
;
MOV AH,09H
MOV AL,1CH
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,SERVTYPE ; SERVICE TYPE IN DX
MOV DI,SEG ASRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET ASRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR SELECT ACTIVE SCREEN SERVICE
;
INT 7AH
.
.
.
```

Chapter 7. Coding Host Interactive Service Requests

Introduction	7-2
Requesting the Host Interactive Services	7-2
Return Codes for the Host Interactive Services	7-3
Host Interactive Service X'01': Connect to Host Session	7-4
Host Interactive Service X'02': Disconnect from Host Session	7-11
Host Interactive Service X'03': Read Structured Field	7-15
Host Interactive Service X'04': Write Structured Field	7-20
Host Interactive Service X'05': Define Buffer	7-25

Introduction

This chapter describes how to code requests for the host interactive services provided by the API.

The host interactive services allow communication between a personal computer application program and a host application program using destination/origin structured field protocol. The host interactive services also allow a personal computer application program to be notified when a host presentation space or operator information area is updated.

With CUT host sessions, **one** connection is allowed for only PS/OIA updates. For each DFT host session, all PC tasks may connect for both destination/origin and PS/OIA updates; however, a maximum of three connections at any one time are allowed. When destination/origin protocols are used, each PC task must be identified with a unique application name. When activated, the 3270 PC file transfer program uses one of the three connections. With the destination/origin protocol, the 3270 Workstation Program accepts Open (X'D000'), Close (X'D041'), Set Cursor (X'D045'), Get (X'D046'), and Insert and Insert Data structured fields (X'D047'). See Appendix B, "Destination/Origin Structured Fields," for more information.

The host interactive services provided by the API are:

- **Connect to Host Session Service:** Use this service to connect to the specified host session for host interactive services.
- **Disconnect from Host Session Service:** Use this service to disconnect from the specified host session.
- **Read Structured Field Service:** Use this service to read structured field data from the specified host session. This service is valid for DFT host sessions only.
- **Write Structured Field Service:** Use this service to write structured field data to the specified host session. This service is valid for DFT host sessions only.
- **Define Buffer Service:** Use this service to define a buffer that will be used to receive a message from the specified host session. This service is valid for DFT host sessions only.

Requesting the Host Interactive Services

To request any of the host interactive services, load the registers and the parameter list with the proper values, and use the INT 7AH instruction to signal the workstation program that it has a request to process.

Note: Before your application can request the host interactive services, it must request the Name Resolution service, using 'MFIC' as the gate name in the parameter list. (Remember that the gate name must be padded to the right with blanks if it is less than eight characters.)

Return Codes for the Host Interactive Services

Each host interactive service has two return codes associated with it: a system return code and a host interaction management return code. Both types of return codes are 2-byte values made up of a function ID and an error number. The function ID indicates the portion of the workstation program in which the error occurred. The error number indicates the specific type of error that has occurred. An error number of X'00' always indicates a successful acceptance or completion of the request.

- **System Return Codes:**

After your application has requested a host interactive service, the CH and CL registers contain a return code generated by the request processing portion of the workstation program. The function ID is in the CH register, and the error number is in the CL register. System return codes use a function ID of X'12'. The error codes that can appear are:

Code	Meaning
X'00'	Request accepted.
X'05'	Invalid index specified.
X'07'	Invalid reply specified.
X'08'	Invalid wait type specified.
X'0B'	RQE pool depleted.
X'0F'	Invalid environment access.
X'34'	Invalid gate entry.

These system return codes apply to all host interactive services.

- **Host Interactive Services Return Codes:**

After a requested host interactive service is completed, bytes 0 and 1 of the parameter list contain a return code generated by the host interaction management portion of the Workstation Program. The function ID is in byte 1, and the error number is in byte 0. Host interactive return codes use a function ID of X'32'. The error numbers that can appear are specific to the service that was requested and are included in the descriptions of each service.

See Appendix H, "Return Codes," for more information.

Host Interactive Service X'01': Connect to Host Session

Use this service to connect to the specified host session for host interactive services.

Register Values

On Request

AH = X'09'
AL = X'01'
BH = Synchronous or asynchronous *
BL = Synchronous or asynchronous *
CX = X'0000'
DX = Resolved value for MFIC
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

AX = Request ID
CH = X'12'
CL = Return code

The contents of registers BH, DX, ES, and DI are unpredictable.

* The values in these registers depend on whether you want the request to be processed synchronously or asynchronously. See the following description of request register values for more information.

- Request Register Values:

You can specify synchronous or asynchronous processing of the Connect to Host Session service. In synchronous processing, control is returned to your application program after the workstation program has completed the request. In asynchronous processing, control is returned to your application program before the workstation program has completed the request. You must use the Get Request Completion service to obtain the parameter list values on completion when you request asynchronous processing.

Synchronous processing:

There are two ways to specify synchronous processing:

1. Set the BH register to X'80' and the BL register to X'20'. When the request is completed, control is returned to your application program, and the registers and parameter list contain the values for completion of the request.
2. Set both the BH and BL registers to X'40'. When the request is completed, control is returned to your program, but the parameter list values for completion of the request are not obtained until you request the Get Request Completion service.

Asynchronous processing:

For asynchronous processing of the Connect to Host Session service request, set the BH register to X'40' and the BL register to X'00'. When asynchronous processing is specified, you must request the Get Request Completion service to obtain the results of the Connect to Host Session service.

- Completion Register Values:

If you specified asynchronous processing, or synchronous processing using X'40' in both the BH and BL registers on request, the AX register contains a request ID that the workstation program assigned to the request. Match this request ID with the results from the Get Request Completion service.

Parameter List Format to Connect for Structured Field Communications

Note: Connection for structured field communication is valid for DFT host sessions only.

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	X'32'
2	1 byte	Host session ID	Unchanged
3	1 byte	Must be zero	Unchanged
4	1 word	Fixed-length queue ID	Unchanged
6	1 byte	X'01'	Unchanged
7	1 byte	Must be zero	Unchanged
8	1 byte	Must be zero	Unchanged
9	1 byte	X'06'	Unchanged
10	1 word	Offset address of Query Reply data	Unchanged
12	1 word	Segment address of Query Reply data	Unchanged
14	1 word	Must be zero	Unchanged
16	1 word	Task ID	Unchanged
18	1 word	Must be zero	Unchanged
20 – 35	9 words	System work area	System work area

Connect to Host Session

Parameter Definitions

Request Parameters:

For connect for structured field communications:

- The session ID is the ID of the host session you will be communicating with using structured fields.
- The fixed-length queue ID is the ID of a fixed-length queue that the workstation program will use to post communication status information about the specified host session. Your application program must use the Dequeue Data service to obtain the communication status information before each Read Structured Field service request. The communication status information is described under “Usage Notes” in the Read Structured Field service description in this chapter.
- The format of the Query Reply data is as follows:

Offset	Length	Contents	Meaning
0	1 byte	Must be zero	Not used
1	1 byte	X'19'	Length of structure
2	1 byte	X'81'	Query Reply
3	1 byte	X'9D'	Query Reply type
4	1 byte	Must be zero	Reserved flags
5	1 byte	X'01'	Structured field exchange
6, 7	2 bytes	Up to maximum of X'0E00'	Maximum number of bytes allowed in an inbound transmission
8, 9	2 bytes	Up to maximum of X'0E00'	Maximum number of bytes allowed in an outbound transmission
10	1 byte	Must be X'0F'	Identifies the next two bytes as being the destination/origin ID
11	1 word	Must be zero	Destination/origin ID supplied by the 3270 workstation program
13 – 24	12 bytes	APLNME	Application name (in EBCDIC)

- The task ID is the ID of the task that is issuing the Connect to Host Session service request. It is used to identify the application to the API and must be the same for all Read Structured Field, Write Structured Field, and Define Buffer services that your application program requests. You can use the Query Active Task service to obtain the ID of your application program. The Query Active Task service is described in Chapter 17, “Coding Task State Modifier Services.”
- The system work area is used by the workstation program while it processes the request. This area must be provided in the parameter list.

Parameter List Format to Connect for PS/OIA Update Events

Note: Connection for PS/OIA update events is valid for both DFT and CUT host sessions.

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	X'32'
2	1 byte	Host session ID	Unchanged
3	1 byte	Must be zero	Unchanged
4	1 word	Fixed-length queue ID	Unchanged
6	1 byte	X'03'	Unchanged
7	1 byte	Must be zero	Unchanged
8	1 byte	Events	Unchanged
9	1 byte	Must be zero	Unchanged
10	1 word	Reserved	Reserved
12	1 word	Reserved	Reserved
14	1 word	Reserved	Reserved
16	1 word	Reserved	Reserved
18	1 word	Reserved	Reserved
20 – 35	9 words	System work area	System work area

Parameter Definitions

Request Parameters:

For connect for PS/OIA update events:

- The session ID is the ID of the host session for which you want to receive notification whenever PS/OIA information is updated. The session ID is one word in length.
- The fixed-length queue ID is the ID of a fixed-length queue that the workstation program will use to post update information about the PS/OIA of the specified host session. Your application program must use the Dequeue Data service to obtain the update information. The Dequeue Data service is described in Chapter 3, "Coding Supervisor Services." The format of the update information is in 4 bytes, 2 for the session ID and 2 for the update information. The update information is as follows:
 - X'1000' - Presentation space updated
 - X'2000' - OIA updated

Connect to Host Session

- The events that you want to be notified of are specified as follows:

Bits 0 – 4	Bit 5	Bit 6	Bit 7
Must be zero	PS	OIA	Must be zero

- Bits 0 through 4 are reserved and must be zero.
 - Bit 5 set to 1 indicates that you want to be notified of presentation space updates to the specified host session.
 - Bit 6 set to 1 indicates that you want to be notified of operator information area updates to the specified host session.
 - Bit 7 is reserved and must be zero.
- The system work area is used by the workstation program while it processes the request. This area must be provided in the parameter list.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Host Interactive Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the host interaction management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Host interactive return codes use a function ID of X'32'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'01'	The host session is not active (DFT only).
X'02'	Invalid service request parameter.
X'04'	The session is already connected.
X'08'	A system error has occurred.
X'10'	The limit of three requesters have already connected.
X'0C'	Byte 0 of the parameter list is not zero on request.

See Appendix H, "Return Codes," for more information.

Usage Notes

- Before you request this service, you must create a fixed-length queue entry using the Create Queue service.
- If you specified asynchronous processing, or synchronous processing using X'40' in both the BH and BL registers on request, you must use the Get Request Completion service to obtain the results in the parameter list when the Connect to Host Session service is completed.
- CUT host sessions can be connected to for PS/OIA update events only, not structured field communications.

Connect to Host Session

Coding Example

```
;
; PARAMETER LIST FOR CONNECT TO HOST SESSION
;
CFRETNCDB 0; RETURN CODE
CFFXNIDDB 0; FUNCTION NUMBER
CFSESSIDDB 0; SESSION ID
CFRESRV1DB 0; MUST BE 0
          DW 0
CFFLQIDDB 01; FIXED-LENGTH QUEUE ID MUST BE 01
          DB 00
CFEVNTSDB 00; EVENTS TO BE ENQUEUED - DFT
          DB 06
CFQRPLYDD QUERYREP; OFFSET AND SEGMENT OF THE QUERY REPLY
CFRESRV2DW 0; MUST BE 0
CFTASKIDDW 0; PC TASK ID
CFRESRV3DW 0; MUST BE 0
CFWORKDW 9 DUP(0); SYSTEM WORK AREA
;
; QUERY REPLY FOR DESTINATION/ORIGIN
;
QUERYREP DW 0019H; LENGTH OF THE STRUCTURE
          DB 81H; QUERY REPLY
          DB 9DH; ANOMALY IMPLEMENTATION
          DB 0; MUST BE 0
          DB 01H; STRUCTURED FIELD EXCHANGE
          DW 0008H; MAXIMUM NUMBER OF BYTES IN INBOUND TRANSMISSION
          DW 0008H; MAXIMUM NUMBER OF BYTES IN OUTBOUND TRANSMISSION
          DB 0FH; RESERVED
          DW 0; RESERVED
QRAPLNAM DB 8 DUP(0); PC APPLICATION NAME IN EBCDIC
          .
          .
;
; INITIALIZE PARAMETER LIST FOR CONNECT TO HOST SESSION
;
      MOV CFRETNCDB,00H; RETURN CODE MUST = 0 BEFORE REQUEST
      MOV CFFXNIDDB,00H; FUNCTION ID MUST = 0 BEFORE REQUEST
      MOV AL,SESSID; SESSION ID INTO THE LIST
      MOV CFSESSID,AL
      MOV AX,QUEUEID; FIXED-LENGTH QUEUE ID INTO THE LIST
      MOV CFFLQID,AX
      MOV AX,PCTASKID; PC TASK ID INTO THE LIST
      MOV CFTASKID,AX
;
; INITIALIZE REGISTERS FOR CONNECT TO HOST SESSION
;
      MOV AH,09H
      MOV AL,01H
      MOV BH,80H
      MOV BL,20H
      MOV CX,0FFH
      MOV DX,SERVTYPE; RESOLVED VALUE FOR 'MFIC'
      MOV DI,SEG CFRETNCDB; SEGMENT ADDRESS OF PARAMETER LIST
      MOV ES,DI; IN ES
      MOV DI,OFFSET CFRETNCDB; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR CONNECT TO HOST SESSION SERVICE
;
      INT 7AH
      .
      .
```

Host Interactive Service X'02': Disconnect from Host Session

Use this service to disconnect from the specified host session.

Register Values

On Request

AH = X'09'
AL = X'02'
BH = Synchronous or asynchronous *
BL = Synchronous or asynchronous *
CX = X'0000'
DX = Resolved value for MFIC
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

AX = Request ID
CH = X'12'
CL = Return code

The contents of registers BH, DX, ES, and DI are unpredictable.

* The values in these registers depend on whether you want the request to be processed synchronously or asynchronously. See the following description of request register values for more information.

- **Request Register Values:**

You can specify synchronous or asynchronous processing of the Disconnect from Host Session service. In synchronous processing, control is returned to your application program after the workstation program has completed the request. In asynchronous processing, control is returned to your application program before the workstation program has completed the request. You must use the Get Request Completion service to obtain the parameter list values on completion when you request asynchronous processing.

Synchronous Processing:

There are two ways to specify synchronous processing:

1. Set the BH register to X'80' and the BL register to X'20'. When the request is completed, control is returned to your application program, and the registers and parameter list contain the values for completion of the request.
2. Set both the BH and BL registers to X'40'. When the request is completed, control is returned to your program, but the parameter list values for completion of the request are not obtained until you request the Get Request Completion service.

Disconnect from Host Session

Asynchronous Processing:

For asynchronous processing of the Disconnect from Host Session service request, set the BH register to X'40' and the BL register to X'00'. When asynchronous processing is specified, you must request the Get Request Completion service to obtain the results of the Disconnect from Host Session service.

- Completion Register Values:

If you specified asynchronous processing, or synchronous processing using X'40' in both the BH and BL registers on request, the AX register contains a request ID that the workstation program assigned to the request. You use this request ID to match the results of the service obtained by the Get Request Completion service to the results of this service. That is, when the request ID in the AX register on completion of the Get Request Completion service, matches the request ID in the AX register on completion of this service, the results obtained by the Get Request Completion service pertain to this request.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	X'32'
2	1 byte	Session ID	Unchanged
3	1 byte	Must be zero	Unchanged
4	1 word	Reserved	Reserved
6	1 byte	Disconnect type	Unchanged
7	1 byte	Must be zero	Must be zero
8	1 word	Reserved	Reserved
10	1 word	Reserved	Reserved
12	1 word	Reserved	Reserved
14	1 word	Reserved	Reserved
16	1 word	Task ID	Unchanged
18	1 word	Reserved	Reserved
20 – 35	9 words	System work area	System work area

Parameter Definitions

Request Parameters:

- The session ID is the ID of the host session to disconnect from.
- The disconnect type is specified as follows:
 - X'01' to disconnect for structured field communications
 - X'03' to disconnect for PS/OIA update events

Note: Disconnect type X'01' can be specified for DFT host sessions only. Disconnect type X'03' can be specified for both DFT and CUT host sessions.

- The system work area is used by the workstation program while it processes the request. This area must be provided in the parameter list.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Host Interactive Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the host interaction management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Host interactive return codes use a function ID of X'32'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid service request parameter.
X'04'	The session is not connected.
X'08'	A system error has occurred.
X'0C'	Byte 0 of the parameter list is not zero on request.

See Appendix H, "Return Codes," for more information.

Usage Notes

- If you specified asynchronous processing, or synchronous processing using X'40' in both the BH and BL registers on request, you must use the Get Request Completion service to obtain the results in the parameter list when the Disconnect from Host Session service is completed.

Disconnect from Host Session

Coding Example

```
;
; PARAMETER LIST FOR DISCONNECT FROM HOST SESSION
;
DFRETNCD DB 0 ; RETURN CODE
DFFXNID DB 0 ; FUNCTION NUMBER
DFSESSID DB 0 ; SESSION ID
DFRESERV1 DB 0 ; RESERVED
          DW 0 ; NOT USED
DFTYPE DB 01 ; DISCONNECT TYPE - DESTINATION/ORIGIN
          DW 6 DUP(0) ; NOT USED
DFWORK DW 9 DUP(0) ; SYSTEM WORK AREA

.
.
.

;
; INITIALIZE PARAMETER LIST FOR DISCONNECT FROM HOST SESSION
;
      MOV DFRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
      MOV DFFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
      MOV AL,SESSID ; SESSION ID INTO THE LIST
      MOV DFSESSID,AL

;
; INITIALIZE REGISTERS FOR DISCONNECT FROM HOST SESSION
;
      MOV AH,09H
      MOV AL,02H
      MOV BH,80H
      MOV BL,20H
      MOV CX,0FFH
      MOV DX,SERVTYPE ; RESOLVED VALUE FOR 'MFIC '
      MOV DI,SEG DFRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
      MOV ES,DI ; IN ES
      MOV DI,OFFSET DFRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR DISCONNECT FROM HOST SESSION SERVICE
;
      INT 7AH
.
.
.
```

Host Interactive Service X'03': Read Structured Field

Use this service to read structured field data from the specified host session. This service is valid for DFT host sessions only.

Register Values

On Request

AH = X'09'
AL = X'03'
BH = Synchronous or asynchronous *
BL = Synchronous or asynchronous *
CX = X'0000'
DX = Resolved value for MFIC
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

AX = Request ID
CH = X'12'
CL = Return code

The contents of registers BH, DX, ES, and DI are unpredictable.

* The values in these registers depend on whether you want the request to be processed synchronously or asynchronously. See the following description of request register values for more information.

- Request Register Values:

You can specify synchronous or asynchronous processing of the Read Structured Field service. In synchronous processing, control is returned to your application program after the workstation program has completed the request. In asynchronous processing, control is returned to your application program before the workstation program has completed the request. You must use the Get Request Completion service to obtain the parameter list values on completion when you request asynchronous processing.

Synchronous Processing:

There are two ways to specify synchronous processing:

1. Set the BH register to X'80' and the BL register to X'20'. When the request is completed, control is returned to your application program and the registers and parameter list contain the values for completion of the request.
2. Set both the BH and BL registers to X'40'. When the request is completed, control is returned to your program, but the parameter list values for completion of the request are not obtained until you request the Get Request Completion service.

Read Structured Field

Asynchronous Processing:

For asynchronous processing of the Read Structured Field service request, set the BH register to X'40' and the BL register to X'00'. When asynchronous processing is specified, you must request the Get Request Completion service to obtain the results of the Read Structured Field service.

- Completion Register Values:

If you specified asynchronous processing, or synchronous processing using X'40' in both the BH and BL registers on request, the AX register contains a request ID that the workstation program assigned to the request. You use this request ID to match the results of the service obtained by the Get Request Completion service to the results of this service. That is, when the request ID in the AX register on completion of the Get Request Completion service matches the request ID in the AX register on completion of this service, the results obtained by the Get Request Completion service pertain to this request.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	X'32'
2	1 byte	Host session ID	Unchanged
3	1 byte	Must be zero	Unchanged
4	1 word	Reserved	Reserved
6	1 byte	X'01'	Unchanged
7	1 byte	Must be zero	Unchanged
8	1 word	Reserved	Reserved
10	1 word	Unchanged	Offset address of structured field data
12	1 word	Unchanged	Segment address of structured field data
14	1 word	Reserved	Reserved
16	1 word	Task ID	Unchanged
18	1 word	Reserved	Reserved
20 - 35	9 words	System work area	System work area

Parameter Definitions

Request Parameters:

- The session ID is the ID of the host session to read the structured field data from.
- The task ID must be the same task ID that was specified by the application program in the parameter list for the Connect to Host Session service.
- The system work area is used by the workstation program while it processes the request. This area must be provided in the parameter list.

Completion Parameters:

- The structured field data contains the application structured fields received from the host. Destination/origin structured field headers are removed by the workstation program before the structured field data reaches the application.

The structured field data format is as follows:

Offset	Length	Contents
0	1 word	X'0000'
2	1 word	m (message length, which is the number of bytes in the message). This length does not include the eight bytes used for the message buffer header.
4	1 word	n (buffer size - this is the number that you specified in the Define Buffer request).
6	1 word	X'C000'
8	1 word	p Number of bytes from byte 8 to the end of the message.
10	1 byte	First byte in the structured field message
11	1 byte	Second byte in the structured field message
		⋮
m	byte	Last byte in the structured field message

Bytes 0 through 7 are the buffer header. Bytes 8 and 9 contain the number of bytes in the message, including 2 bytes for bytes 8 and 9. Bytes 10 through **m** are used for the structured field message received from the host.

Read Structured Field

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Host Interactive Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the host interaction management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Host interactive return codes use a function ID of X'32'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion. Structured field data is available in the message buffer.
X'02'	Invalid service request parameter.
X'04'	The session is not connected.
X'08'	A system error has occurred.
X'0C'	Byte 0 of the parameter list is not zero on request.
X'14'	No structured field data is available.

See Appendix H, "Return Codes," for more information.

Usage Notes

- If you specified asynchronous processing, or synchronous processing using X'40' in both the BH and BL registers on request, you must use the Get Request Completion service to obtain the results in the parameter list when the Read Structured Fields service is completed.
- Before you request the Read Structured Field service, you must use the Dequeue Data service to check for the communication status information that indicates that a message is available from the host.

The first two bytes of the communication status information contain the session ID of the host session that the information pertains to. The second two bytes of the communication status information contain one of the following codes:

Code	Type	Meaning
X'04'	X'80'	A message from the host is available.
X'06'	X'80'	An outbound transmission from the host was canceled.
X'08'	X'00'	Lost contact with the host.
X'0A'	X'00'	Contact reestablished with the host.

Coding Example

```

;
; PARAMETER LIST FOR READ STRUCTURED FIELD
;
RSRETNCD  DB  0                ; RETURN CODE
RSFXNID   DB  0                ; FUNCTION NUMBER
RSHOSTID  DB  0                ; HOST SESSION ID
RSZERO    DB  0                ; UNCHANGED
          DW  0                ; NOT USED
          DB  01               ; STRUCTURED FIELD TYPE, (DEST/ORIG)
          DB  00               ; UNUSED
RSOFFSD   DW  0                ; OFFSET ADDRESS OF STRUCTURED FIELD DATA
RSSEGTID  DW  0                ; SEGMENT ADDRESS OF STRUCTURED FIELD DATA
          DW  0                ; UNUSED
RSTASKID  DW  0                ; PC TASK ID
          DW  0
          DW  9 DUP(0)         ; SYSTEM WORK AREA
          .
          .
          .
;
; INITIALIZE PARAMETER LIST FOR READ STRUCTURED FIELD
;
      MOV     RSRETNCD,00H      ; RSRETNCD MUST BE 0 BEFORE REQUEST
      MOV     RSFXNID,00H      ; RSFXNID MUST BE 0 BEFORE REQUEST
      MOV     AL,HOSTID        ; HOST ID IN
      MOV     RSHOSTID,AL      ; THE LIST
      MOV     AX,PCTSKID       ; PC TASK ID
      MOV     RSTASKID,AX      ; IN LIST
      MOV     RSZERO,0         ; THIS FIELD MUST BE ZEROED
;
; INITIALIZE REGISTERS FOR READ STRUCTURED FIELD
;
      MOV     AH,09H
      MOV     AL,03H
      MOV     BH,40H           ; REPLY TYPE IN BH
      MOV     BL,40H           ; WAIT TYPE IN BL
      MOV     CX,0             ; PRIORITY IN CX
      MOV     DX,MFIC          ; RESOLVED VALUE FOR 'MFIC'
      MOV     DI, SEG RSRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
      MOV     ES,DI            ; IN ES
      MOV     DI,OFFSET RSRETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR READ STRUCTURED FIELD SERVICE
;
      INT     7AH
      .
      .
      .

```

Host Interactive Service X'04': Write Structured Field

Use this service to write structured field data to the specified host session. This service is valid for DFT host sessions only.

Register Values

On Request

AH = X'09'
AL = X'04'
BH = Synchronous or asynchronous *
BL = Synchronous or asynchronous *
CX = X'0000'
DX = Resolved value for MFIC
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

AX = Request ID
CH = X'12'
CL = Return code

The contents of registers BH, DX, ES, and DI are unpredictable.

* The values in these registers depend on whether you want the request to be processed synchronously or asynchronously. See the following description of request register values for more information.

- Request Register Values:

You can specify synchronous or asynchronous processing of the Write Structured Field service. In synchronous processing, control is returned to your application program after the workstation program has completed the request. In asynchronous processing, control is returned to your application program before the workstation program has completed the request. You must use the Get Request Completion service to obtain the parameter list values on completion when you request asynchronous processing.

Synchronous Processing:

There are two ways to specify synchronous processing:

1. Set the BH register to X'80' and the BL register to X'20'. When the request is completed, control is returned to your application program, and the registers and parameter list contain the values for completion of the request.
2. Set both the BH and BL registers to X'40'. When the request is completed, control is returned to your program, but the parameter list values for completion of the request are not obtained until you request the Get Request Completion service.

Asynchronous Processing:

For asynchronous processing of the Write Structured Field service request, set the BH register to X'40' and the BL register to X'00'. When asynchronous processing is specified, you must request the Get Request Completion service to obtain the results of the Write Structured Field service.

- Completion Register Values:

If you specified asynchronous processing, or synchronous processing using X'40' in both the BH and BL registers on request, the AX register contains a request ID that the workstation program assigned to the request. You use this request ID to match the results of the service obtained by the Get Request Completion service to the results of this service. That is, when the request ID in the AX register on completion of the Get Request Completion service matches the request ID in the AX register on completion of this service, the results obtained by the Get Request Completion service pertain to this request.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	X'32'
2	1 byte	Host session ID	Unchanged
3	1 byte	Must be zero	Unchanged
4	1 word	Reserved	Reserved
6	1 byte	X'01'	Unchanged
7	1 byte	Must be zero	Unchanged
8	1 word	Reserved	Reserved
10	1 word	Offset address of structured field data	Unchanged
12	1 word	Segment address of structured field data	Unchanged
14	1 word	Reserved	Reserved
16	1 word	Task ID	Unchanged
18	1 word	Reserved	Reserved
20 – 35	9 words	System work area	System work area

Write Structured Field

Parameter Definitions

Request Parameters:

- The session ID is the ID of the host session to write the structured field data to.
- The task ID must be the same task ID that was specified by the application program in the parameter list for the Connect to Host Session service.
- The structured field data contains the application structured fields that are to be sent to the host. Destination/origin structured fields are added by the workstation program before the structured field data reaches the host.
- The system work area is used by the Workstation Program while it processes the request. This area must be provided in the parameter list.

The structured field data format is as follows:

Offset	Length	Contents
0	1 word	X'0000'
2	1 word	m (message length, which is the number of bytes in the message). This length does not include the eight bytes used for the message buffer header.
4	1 word	X'0000'
6	1 word	X'0000'
8	1 word	p Number of bytes from byte 8 to the end of the message.
10	1 byte	First byte in the structured field message must be X'D0'.
11	1 byte	Second byte in the structured field message must be X'00', X'41', X'45', X'46', X'47', and X'48'.
⋮		
m + 7	byte	Last byte in the structured field message

Bytes 0 through 7 are the buffer header. Bytes 8 and 9 contain the number of bytes in the message, including 2 bytes for bytes 8 and 9. Bytes 10 through **m** + 7 are used for the structured field message sent to the host.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Host Interactive Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the host interaction management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Host interactive return codes use a function ID of X'32'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion. The message has been sent to and acknowledged by the host.
X'02'	Invalid service request parameter.
X'04'	The session is not connected.
X'08'	A system error has occurred.
X'0C'	Byte 0 of the parameter list was not zero on request.
X'10'	The message you sent was rejected.

See Appendix H, "Return Codes," for more information.

Usage Notes

- If you specified asynchronous processing, or synchronous processing using X'40' in both the BH and BL registers on request, you must use the Get Request Completion service to obtain the results in the parameter list when the Write Structured Fields service is completed.
- Before you use the Write Structured Field service, you must use the Define Buffer service to define a buffer to use to receive the next transmission of structured field data sent from the host.

Write Structured Field

Coding Example

```
;
; PARAMETER LIST FOR WRITE STRUCTURED FIELD
;
WSRETNCD DB 0 ; RETURN CODE
WSFXNID DB 0 ; FUNCTION NUMBER
WSHOSTID DB 0 ; HOST SESSION ID
WSZERO DB 0 ; UNCHANGED
        DW 0 ; NOT USED
        DB 01 ; STRUCTURED FIELD TYPE, (DEST/ORIG)
        DB 00 ; UNUSED
WSOFFSD DW 0 ; OFFSET ADDRESS OF STRUCTURED FIELD DATA
WSSEGTD DW 0 ; SEGMENT ADDRESS OF STRUCTURED FIELD DATA
        DW 0 ; UNUSED
WSTASKID DW 0 ; PC TASK ID
        DW 0
        DW 9 DUP(0) ; SYSTEM WORK AREA
        .
        .
        .
;
; INITIALIZE PARAMETER LIST FOR WRITE STRUCTURED FIELD
;
        MOV WSRETNCD,00H ; WSRETNCD MUST BE 0 BEFORE REQUEST
        MOV WSFXNID,00H ; WSFXNID MUST BE 0 BEFORE REQUEST
        MOV AL,HOSTID ; HOST ID IN
        MOV WSHOSTID,AL ; THE LIST
        ; OFFSET AND SEGMENT OF DATA IN LIST
        MOV WSOFFSD,OFFSET STR$DATA
        MOV WSSEGTD,SEG STR$DATA
        MOV AX,PCTSKID ; PC TASK ID
        MOV WSTASKID,AX ; IN LIST
        MOV WSZERO,0 ; THIS FIELD MUST BE ZEROED
;
; INITIALIZE THE FIELDS IN THE STRUCTURED FIELD 8 BYTE HEADER.
; STR$DATA IS THE MEMORY LOCATION NAME OF THE BEGINNING OF THE
; STRUCTURED FIELD 8 BYTE HEADER.
;
        MOV STR$DATA,0
        MOV WORD PTR STR$DATA + 4,0
        MOV WORD PTR STR$DATA + 6,0
;
; INITIALIZE REGISTERS FOR WRITE STRUCTURED FIELD
;
        MOV AH,09H
        MOV AL,04H
        MOV BH,80H ; REPLY IS A COMPLETION SIGNAL
        MOV BL,20H ; WAIT FOR A COMPLETION SIGNAL
        MOV CX,0 ; PRIORITY IN CX
        MOV DX,MFIC ; RESOLVED VALUE FOR 'MFIC '
        MOV DI,SEG WSRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
        MOV ES,DI ; IN ES
        MOV DI,OFFSET WSRETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR WRITE STRUCTURED FIELD SERVICE
;
        INT 7AH
        .
        .
        .
```

Host Interactive Service X'05': Define Buffer

Use this service to define a buffer that will be used to receive a message from the specified host session. This service is valid for DFT host sessions only.

Register Values

On Request

AH = X'09'
AL = X'05'
BH = Synchronous or asynchronous *
BL = Synchronous or asynchronous *
CX = X'0000'
DX = Resolved value for MFIC
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

AX = Request ID
CH = X'12'
CL = Return code

The contents of registers BH, DX, ES, and DI are unpredictable.

* The values in these registers depend on whether you want the request to be processed synchronously or asynchronously. See the following description of request register values for more information.

- Request Register Values:

You can specify synchronous or asynchronous processing of the Define Buffer service. In synchronous processing, control is returned to your application program after the workstation program has completed the request. In asynchronous processing, control is returned to your application program before the workstation program has completed the request. You must use the Get Request Completion service to obtain the parameter list values on completion when you request asynchronous processing.

Synchronous Processing:

There are two ways to specify synchronous processing:

1. Set the BH register to X'80' and the BL register to X'20'. When the request is completed, control is returned to your application program, and the registers and parameter list contain the values for completion of the request.
2. Set both the BH and BL registers to X'40'. When the request is completed, control is returned to your program, but the parameter list values for completion of the request are not obtained until you request the Get Request Completion service.

Define Buffer

Asynchronous Processing:

For asynchronous processing of the Define Buffer service request, set the BH register to X'40' and the BL register to X'00'. When asynchronous processing is specified, you must request the Get Request Completion service to obtain the results of the Define Buffer service.

- Completion Register Values:

If you specified asynchronous processing, or synchronous processing using X'40' in both the BH and BL registers on request, the AX register contains a request ID that the workstation program assigned to the request. You use this request ID to match the results of the service obtained by the Get Request Completion service to the results of this service. That is, when the request ID in the AX register on completion of the Get Request Completion service matches the request ID in the AX register on completion of this service, the results obtained by the Get Request Completion service pertain to this request.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	X'32'
2	1 byte	Host session ID	Unchanged
3	1 byte	Must be zero	Unchanged
4	1 word	Reserved	Reserved
6	1 byte	X'01'	Unchanged
7	1 byte	Must be zero	Unchanged
8	1 word	Reserved	Reserved
10	1 word	Offset address of buffer	Unchanged
12	1 word	Segment address of buffer	Unchanged
14	1 word	Reserved	Reserved
16	1 word	Task ID	Unchanged
18	1 word	Reserved	Reserved
20 – 35	9 words	System work area	System work area

Parameter Definitions

Request Parameters:

- The session ID is the ID of the host session whose structured field data will be received in the buffer being defined.
- The task ID must be the same task ID that was specified by the application program in the parameter list for the Connect to Host Session service.
- The system work area is used by the workstation program while it processes the request. This area must be provided in the parameter list.
- The format of the buffer is as follows:

Offset	Length	Contents
0	1 word	Must be zero
2	1 word	Must be zero
4	1 word	n (buffer size). This includes the 8-byte prefix. The maximum buffer size allowed is 3592 bytes (decimal).
6	1 word	X'0000'
8	1 byte	Used for structured field data
9	1 byte	Used for structured field data
		• • •
n + 8	1 byte	Used for structured field data

Bytes 0 through 7 are the buffer header. Bytes 8 through $n + 8$ are used for the destination/origin structured field message received from the host.

- The length of the buffer is the number of bytes in the buffer. The maximum buffer size allowed is 3592 (decimal) bytes.

Return Codes

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

Define Buffer

- Host Interactive Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the host interaction management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Host interactive return codes use a function ID of X'32'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid service request parameter.
X'04'	The session is not connected.
X'08'	A system error has occurred.
X'0C'	Byte 0 of the parameter list was not zero on request.

See Appendix H, "Return Codes," for more information.

Usage Notes

- If you specified asynchronous processing, or synchronous processing using X'40' in both the BH and BL registers on request, you must use the Get Request Completion service to obtain the results in the parameter list when the Define Buffer service is completed.
- You must request the Define Buffer service at the following times:
 - Before the host application that communicates with your application is started, so that a message buffer is available in time to receive the first message from the host. (To start the host application program, your application program can use the keyboard services for sending keystrokes to the host.)
 - Before each Write Structured Field service request, so that a message buffer is available in time to receive the next message from the host session.

You do not have to use a different message buffer for each Write Structured Field service request (although you can if you wish), but you must reset the message buffer header as follows:

1. Set the first two words of the buffer header to zero.
2. Set the third word of the buffer header to the length of the message buffer (including the eight bytes of the buffer header).
3. Set the fourth word of the message buffer header to zero.

Coding Example

```

;
; PARAMETER LIST FOR DEFINE RECEIVE BUFFER
;
DBRETNCD DB 0 ; RETURN CODE
DBFXNID DB 0 ; FUNCTION NUMBER
DBHOSTID DB 0 ; HOST SESSION ID
          DB 0 ; UNCHANGED
          DW 0 ; NOT USED
          DB 01
          DB 00 ; UNUSED
DBOFFSET DW 0 ; SEGMENT AND OFFSET OF THE MESSAGE BUFFER
DBSEGMNT DW 0
          DW 0 ; UNUSED
DBTASKID DW 0 ; PC TASK ID
          DW 0
          DW 9 DUP(0) ; SYSTEM WORK AREA
          .
          .
          .
;
; INITIALIZE PARAMETER LIST FOR DEFINE RECEIVE BUFFER
;
      MOV DBRETNCD,00H ; DBRETNCD MUST BE 0 BEFORE REQUEST
      MOV DBFXNID,00H ; DBFXNID MUST BE 0 BEFORE REQUEST
      MOV AL,HOSTID ; HOST ID IN
      MOV DBHOSTID,AL ; THE LIST
      MOV AX,PCTSKID ; PC TASK ID
      MOV DBTASKID,AX ; IN THE LIST
      MOV AX,OFFSET BUFFER ; OFFSET OF MESSAGE BUFFER
      MOV DBOFFSET,AX ; IN THE LIST
      MOV AX,SEG BUFFER ; SEGMENT OF THE MESSAGE BUFFER
      MOV DBSEGMNT,AX ; IN THE LIST
;
; INITIALIZE THE 8 BYTE HEADER OF THE MESSAGE BUFFER.
; BUFFER IS THE MEMORY LOCATION NAME OF THE BEGINNING OF THE
; STRUCTURED FIELD 8 BYTE HEADER.
;
      MOV BUFFER,0
      MOV WORD PTR BUFFER + 2,0
      MOV WORD PTR BUFFER + 6,0
;
; INITIALIZE REGISTERS FOR DEFINE RECEIVE BUFFER
;
      MOV AH,09H
      MOV AL,05H
      MOV BH,40H ; REPLY
      MOV BL,40H ; WAIT TYPE IN BL
      MOV CX,0 ; PRIORITY IN CX
      MOV DX,MFIC ; RESOLVED VALUE FOR 'MFIC '
      MOV DI,SEG DBRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
      MOV ES,DI ; IN ES
      MOV DI,OFFSET DBRETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR DEFINE RECEIVE BUFFER SERVICE
;
      INT 7AH
      .
      .
      .

```


Chapter 8. Coding Presentation Space Service Requests

Introduction	8-2
Requesting the Presentation Space Services	8-2
Return Codes for the Presentation Space Services	8-2
Presentation Space Service X'01': Define Presentation Space	8-4
Presentation Space Service X'02': Delete Presentation Space	8-11
Presentation Space Service X'03': Display Presentation Space	8-14
Presentation Space Service X'04': Set Cursor Position	8-17
Presentation Space Service X'05': Switch Presentation Space	8-21

Introduction

This chapter describes how to code requests for the presentation space services provided by the API.

The presentation space services allow your application program to create and delete personal computer presentation spaces, to display them, and to control the position of the cursor in them.

The presentation space services provided by the API are:

- **Define Presentation Space Service:** Use this service to define a presentation space, and to obtain the session ID that the workstation program assigns to that presentation space.
- **Delete Presentation Space Service:** Use this service to delete a presentation space created by the Define Presentation Space service. Any window created on this presentation space is also deleted.
- **Display Presentation Space Service:** Use this service to display any changes made in the specified presentation space.
- **Set Cursor Service:** Use this service to set a cursor position in a presentation space.
- **Switch Presentation Space Service:** Use this service to specify a presentation space as the default presentation space for all DOS and BIOS updates.

Requesting the Presentation Space Services

To request any of the presentation space services, load the registers and the parameter list with the proper values, and use the INT 7AH instruction to signal the workstation program that it has a request to process.

Note: Before your application can request the presentation space services, it must request the Name Resolution service, using 'PCPSM ' as the gate name in the parameter list. (Remember that the gate name must be padded to the right with blanks if it is less than eight characters.)

Return Codes for the Presentation Space Services

Each presentation space service has two return codes associated with it: a system return code and a presentation space management return code. Both types of return codes are 2-byte values made up of a function ID and an error number. The function ID indicates the portion of the workstation program in which the error occurred. The error number indicates the specific type of error that has occurred. An error number of X'00' always indicates a successful acceptance or completion of the request.

- **System Return Codes:**

After your application has requested a presentation space service, the CH and CL registers contain a return code generated by the request processing portion of the workstation program. The function ID is in the CH register, and the error number is in the CL register. System return codes use a function ID of X'12'. The error codes that can appear are:

Code	Meaning
X'00'	Request accepted.
X'05'	Invalid index specified.
X'07'	Invalid reply specified.
X'08'	Invalid wait type specified.
X'0B'	RQE pool depleted.
X'0F'	Invalid environment access.
X'34'	Invalid gate entry.

These system return codes apply to all presentation space services.

- **Presentation Space Services Return Codes:**

After a requested presentation space service is completed, bytes 0 and 1 of the parameter list contain a return code generated by the presentation space management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Presentation space return codes use a function ID of X'69'. The error numbers that can appear are specific to the service that was requested and are included in the descriptions of each service.

See Appendix H, "Return Codes," for more information.

Define Presentation Space

Presentation Space Service X'01': Define Presentation Space

Use this service to define a presentation space and to obtain the session ID that the workstation program assigns to it. *This service is allowed only if Multi-DOS is selected at customization time.*

Register Values

On Request

AH = X'09'
AL = X'01'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for PCPSM
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'69')
2	1 byte	Reserved	Session ID
3	1 byte	Reserved	Reserved
4	1 word	Offset address of the presentation space work area	Unchanged
6	1 word	Segment address of the presentation space work area	Unchanged
8	1 word	Offset address of the presentation space data stream	Unchanged
10	1 word	Segment address of the presentation space data stream	Unchanged
12	1 byte	Must be zero	Unchanged
13	1 byte	Reserved	Window short name

Parameter Definitions

Request Parameters:

- The presentation space work area is a 1552-byte area that your application program must provide.
- The presentation space data stream is in the following format:
 - Header
 - Elements in the form of command data

The header is a 1-byte value that contains the number of commands you have coded in the data stream.

Each command is specified as a 1-byte value indicating the command number, followed by a variable-length amount of data.

Commands 01, 02, and 03 are required to define a presentation space, while 05 and 06 are optional. Command 07 is used only for a specific function—3270 keystroke emulation.

The commands available in the presentation space data stream, and their data format, are as follows:

- Command 01: Set Presentation Space Size

The only presentation space currently supported is 25 rows by 80 columns. (The presentation space is the buffer described for command 03.) The data format is one byte containing X'19', which indicates 25 rows in hexadecimal followed by one byte containing X'50', which indicates 80 columns in hexadecimal.

- Command 02: Set Presentation Space Type

The data format is one byte containing X'00', which indicates text indirect. The application program uses either a presentation space or the BIOS or DOS calls that must be routed to a presentation space.

- Command 03: Set Presentation Space Buffer

The data format is one word containing the offset address of the presentation space buffer, followed by one word containing the segment address of the presentation space buffer.

The size of the presentation space buffer must be twice the number of character positions in the presentation space. (Each character needs two bytes of information for it to be displayed.) Thus, for a presentation space of 25 rows with 80 columns each, the presentation space size must be 4000 bytes. (See Command 01.)

The space for the presentation space buffer must be provided by your application program.

Define Presentation Space

- Command 04: Reserved
- Command 05: Set Window Long Name

The data format is eight bytes containing the window long name. The window long name can be as many as eight ASCII characters long, and it must begin with an alphabetic character. If the window long name is less than eight characters long, it must be padded to the right with blanks. This command is optional.

- Command 06: Set Window Short Name

The data format is one byte containing the window short name. This command is optional. If the window short name is not specified, the workstation program assigns the first unused letter from A through Z as the window short name. If a short name is supplied, it must be a unique short name (one not already in use). If the short name supplied is currently in use, the Define Presentation Space request will not be completed successfully.

- Command 07: Set Session Attribute Buffer

The data format is one word containing the offset address of the session attribute buffer, followed by one word containing the segment address of the session attribute buffer. The size of the session attribute buffer must be twice the number of rows specified in command 01 of the presentation space data stream.

This buffer is an internal work space allocated in the user area; it should **not** be altered by the user. The workstation program uses this buffer for 3270 attribute algorithms.

This command should be used only if you intend to use 3270 keystroke emulation in this presentation space. For more information, refer to Chapter 9, “Coding 3270 Keystroke Emulation Service Requests.”

Completion Parameters:

- The session ID is the ID identifying this presentation space. Use this session ID for all further communication with this presentation space or its window.
- The window short name is the 1-character ASCII name of the window associated with the presentation space. This window short name is either provided in the presentation space data stream (command 06) or provided by the workstation program. If it is provided by the workstation program, it will be the first letter from A through Z that is not currently used as a short window name.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- You may receive return codes from the session information services, with a function code of X'6B'.
- Presentation Space Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the presentation space management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Presentation space return codes use a function ID of X'69'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'0A'	An invalid number of commands is in the presentation space data stream.
X'0B'	An invalid number of rows/columns is in the presentation space data stream.
X'0C'	Byte 0 of the parameter list was not zero on request.
X'0D'	There is invalid data in the Set Presentation Space Type data stream command.
X'0F'	A command that had no data was found in the presentation space data stream.
X'11'	Invalid parameter.
X'13'	The address of the work area was zero on request.
X'14'	The maximum number of personal computer presentation spaces has already been created, or DOS has been configured and you attempt to create an alternate presentation space (ALT PS).
X'15'	The Set Presentation Space Buffer command was missing from the presentation space data stream.
X'18'	The Set Presentation Space Size command was missing from the presentation space data stream.
X'19'	The Set Presentation Space Type command was missing from the presentation space data stream.

See Appendix H, "Return Codes," for more information.

Define Presentation Space

Usage Notes

- A window for the presentation space is created on screen profile 0 and also on the current screen. You can use the presentation space data stream to specify the window short and long names. If no short name is specified, a default short name is provided. The workstation program assigns the first unused character from A through Z as the default window short name. There is no default for the long name.
- The only type of presentation space that can be created is a personal computer presentation space.
- The presentation space that was the default remains the default for all DOS and BIOS updates until the switch occurs for the new presentation space just defined by the application program. After the switch occurs, the new presentation space becomes the default. To specify another presentation space as the default for DOS and BIOS updates, use the Switch Presentation Space service.
- Before exiting, your application program should use the Delete Presentation Space service to delete any presentation spaces it may have created with this service.
- For each presentation space that you create using this service, you must provide a unique 1552-byte work area.
- A session defined by your application program as a result of a Define Presentation Space service request requires that a Connect to Keyboard service request with an All key intercept option be issued in order to receive and process keystrokes. A second Connect to Keyboard service request can be issued relative to a Define Presentation Space session.

Coding Example

```

;
; PRESENTATION SPACE DATA STREAM FOR DEFINE PRESENTATION SPACE
;
PSDS      DB 5                      ; PSDS HEADER - 5 COMMANDS
SETSIZE   DB 01H                    ; COMMAND TO SET THE PS SIZE
PSROWS    DB 25                      ; 25 ROWS IN THE PS
PSCOLS    DB 80                      ; 80 COLUMNS IN THE PS
SETTYPE    DB 02H                    ; COMMAND TO SET THE PS TYPE
          DB 00H                      ; TYPE = TEXT INDIRECT
SETPSBUF   DB 03H                    ; COMMAND TO SET THE PS BUFFER
PSOFFSET   DW 0                      ; PS OFFSET
PSSEGMNT   DW 0                      ; PS SEGMENT
SETLNGNM   DB 05H                    ; COMMAND TO SET THE WINDOW LONG NAME
LONGNAME   DB 'SAMPLE '              ; WINDOW LONG NAME
SETSHNTM   DB 06H                    ; COMMAND TO SET WINDOW SHORT NAME
SHRTNAME   DB 'X'                    ; WINDOW SHORT NAME
;
; PARAMETER LIST FOR DEFINE PRESENTATION SPACE
;
DPRETNCD   DB 0                      ; RETURN CODE
DPFXNID    DB 0                      ; FUNCTION NUMBER
DPSESSID   DB 0                      ; SESSION ID
DPRESERV   DB 0                      ; RESERVED
DPBUFOFF   DW 0                      ; OFFSET ADDRESS OF THE 1552 BYTE WORK AREA
DPBUFSEG    DW 0                      ; SEGMENT ADDRESS OF THE 1552 BYTE WORK AREA
DPDSOFF    DW 0                      ; OFFSET OF DATA STREAM
DPDSSEG     DW 0                      ; SEGMENT OF DATA STREAM
          DB 0                      ; MUST BE 0
DPWINDOW    DB 0                      ; RETURNED WINDOW SHORT NAME
;
; NEW PRESENTATION SPACE
;
PS          DB 4000 DUP(0)
;
; KEYSTROKING BUFFER
;
KSBUF       DB 128 DUP(0)
;
;
; 1552 BYTE WORK AREA
;
WORKAREA    DB 1552 DUP(0)
;
.
.
.
;
; INITIALIZE PRESENTATION SPACE DATA STREAM FOR DEFINE PRESENTATION SPACE
;
          MOV     PSOFFSET,OFFSET PS    ; OFFSET OF PS INTO THE PSDS
          MOV     PSSEGMNT,SEG PS       ; SEGMENT OF PS INTO THE PSDS

```

Define Presentation Space

```
;
; INITIALIZE PARAMETER LIST FOR DEFINE PRESENTATION SPACE
;
    MOV     DPRETNCD,00H          ; RETURN CODE MUST = 0 BEFORE REQUEST
    MOV     DPFXNID,00H          ; FUNCTION ID MUST = 0 BEFORE REQUEST
    MOV     AX,OFFSET WORKAREA   ; WORK AREA OFFSET INTO THE LIST
    MOV     DPBUFOFF,AX
    MOV     AX,SEG WORKAREA      ; WORK AREA SEGMENT INTO THE LIST
    MOV     DPBUFSEG,AX
    MOV     AX,OFFSET PSDS       ; PSDS OFFSET INTO THE LIST
    MOV     DPDSOFF,AX
    MOV     AX,SEG PSDS          ; PSDS SEGMENT INTO THE LIST
    MOV     DPDSSEG,AX
;
; INITIALIZE REGISTERS FOR DEFINE PRESENTATION SPACE
;
    MOV     AH,09H
    MOV     AL,01H
    MOV     BH,80H
    MOV     BL,20H
    MOV     CX,0FFH
    MOV     DX,PCPSM             ; RESOLVED VALUE FOR 'PCPSM'
    MOV     DI,SEG DPRETNCD      ; SEGMENT ADDRESS OF PARAMETER LIST
    MOV     ES,DI                ; IN ES
    MOV     DI,OFFSET DPRETNCD   ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR DEFINE PRESENTATION SPACE SERVICE
;
    INT     7AH
    .
    .
    .
```

Presentation Space Service X'02': Delete Presentation Space

Use this service to delete a presentation space created by the Define Presentation Space service. *This service is allowed only if Multi-DOS is selected at customization time.* Any window created on this presentation space is also deleted.

Register Values

On Request

AH = X'09'
AL = X'02'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for PCPSM
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'69')
2	1 byte	Session ID	Unchanged
3	1 byte	Reserved	Reserved

Parameter Definitions

Request Parameters:

- The session ID is the ID that was assigned to the presentation space by the Define Presentation Space service.

Delete Presentation Space

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Presentation Space Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the presentation space management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0.

Presentation space return codes use a function ID of X'69'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'0C'	Byte 0 of the parameter list was not zero on request.
X'10'	The specified presentation space cannot be deleted.

See Appendix H, "Return Codes," for more information.

Usage Notes

- Any window created on this presentation space is deleted from all screen profiles on which it appears, as well as from screen profile 0.

Coding Example

```
;
; PARAMETER LIST FOR DELETE PRESENTATION SPACE
;
DYRETNCD DB 0 ; RETURN CODE
DYFXNID DB 0 ; FUNCTION NUMBER
DYSESSID DB 0 ; SESSION ID
DYRESERV DB 0 ; RESERVED
.
.
.
;
; INITIALIZE PARAMETER LIST FOR DELETE PRESENTATION SPACE
;
MOV DYRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV DYFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; HANDLE ID INTO THE LIST
MOV DYSESSID,AL
;
; INITIALIZE REGISTERS FOR DELETE PRESENTATION SPACE
;
MOV AH,09H
MOV AL,02H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,PCPSM ; RESOLVED VALUE FOR 'PCPSM'
MOV DI,SEG DYRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET DYRETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR DELETE PRESENTATION SPACE SERVICE
;
INT 7AH
.
.
.
```

Presentation Space Service X'03': Display Presentation Space

Use this service to display any changes made in the specified presentation space.

Register Values

On Request

AH = X'09'
AL = X'03'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for PCPSM
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'69')
2	1 byte	Session ID	Unchanged
3	1 byte	Reserved	Reserved
4	1 word	Starting offset	Unchanged
6	1 word	Length	Unchanged

Parameter Definitions

Request Parameters:

- The session ID is the ID that was assigned to the presentation space by the Define Presentation Space service.
- The starting offset is a character offset into the presentation space buffer, specifying the first character to display.
- The length is the number of characters to be displayed.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Presentation Space Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the presentation space management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0.

Presentation space return codes use a function ID of X'69'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'03'	The specified offset for display is not within the address of the presentation space.
X'09'	The specified length is invalid.
X'0C'	Byte 0 of the parameter list was not zero on request.
X'11'	Invalid parameter.

See Appendix H, "Return Codes," for more information.

Display Presentation Space

Coding Example

```
;
; PARAMETER LIST FOR DISPLAY PRESENTATION SPACE
;
RDRETNCD DB 0 ; RETURN CODE
RDFXNID DB 0 ; FUNCTION NUMBER
RDSESSID DB 0 ; SESSION ID
RDRESERV DB 0 ; RESERVED
RDCHROFF DW 0 ; STARTING CHARACTER OFFSET
RDLENGTH DW 0 ; NUMBER OF CHARACTERS TO DISPLAY

.
.
.

;
; INITIALIZE PARAMETER LIST FOR DISPLAY PRESENTATION SPACE
;
MOV RDRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV RDFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID INTO THE
MOV RDSESSID,AL ; PARAMETER LIST
MOV RDCHROFF,0 ; START AT THE 1ST CHARACTER
MOV RDLENGTH,2000 ; DISPLAY 2000 CHARACTERS

;
; INITIALIZE REGISTERS FOR DISPLAY PRESENTATION SPACE
;
MOV AH,09H
MOV AL,03H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,PCPSM ; RESOLVED VALUE FOR 'PCPSM'
MOV DI,SEG RDRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET RDRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR DISPLAY PRESENTATION SPACE SERVICE
;
INT 7AH
.
.
.
```

Presentation Space Service X'04': Set Cursor Position

Use this service to set a cursor position in a presentation space.

Register Values

On Request

AH = X'09'
AL = X'04'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for PCPSM
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'69')
2	1 byte	Session ID	Unchanged
3	1 byte	Reserved	Reserved
4	1 word	Cursor address	Unchanged
6	1 byte	Cursor type	Unchanged
7	1 byte	Reserved	Reserved

Parameter Definitions

Request Parameters:

- The session ID is the ID that was assigned to the presentation space by the Define Presentation Space service.
- The cursor address is an offset into the presentation space buffer, specifying character position to set the cursor.

Set Cursor Position

The character position for the cursor is derived from the following formula:

$$[\text{Row number} \times \text{number of columns}] + \text{column number}$$

where:

- “Row number” is the number of the row that you want the cursor to be positioned in (0 to 24).
 - “Number of columns” is the number of columns defined in the presentation space.
 - “Column number” is the number of the column that you want the cursor to be positioned in (0 to 79).
- The cursor type byte is as follows (where bit 0 is the high-order bit and bit 7 is the low-order bit):

0	Reserved
1	Reserved
2	Reserved
3	Inhibited cursor with autoscroll
4	Reserved
5	Inhibited cursor
6	Blinking cursor
7	Box cursor
 - A blinking underscore cursor appears as _
 - A blinking box cursor appears as ■
 - An inhibited cursor is not displayed. When the cursor position changes, the text in the window is not moved to keep the cursor inside the window borders.
 - An inhibited cursor with autoscroll is not displayed. When the cursor position changes, the text in the window is moved to keep the cursor inside the window borders.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Presentation Space Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the presentation space management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Presentation space return codes use a function ID of X'69'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'06'	Invalid cursor type.
X'07'	Invalid cursor address.
X'0C'	Byte 0 of the parameter list was not zero on request.
X'11'	Invalid parameter.

See Appendix H, "Return Codes," for more information.

Usage Notes

- Request this service each time you wish to change the position of the cursor.
- The cursor only appears in the active window, data autoscrolls if necessary to keep the cursor in view, and the cursor always blinks.

Set Cursor Position

Coding Example

```
;
; PARAMETER LIST FOR SET CURSOR POSITION
;
DCRETNCD DB 0 ; RETURN CODE
DCFXNID DB 0 ; FUNCTION NUMBER
DCSESSID DB 0 ; SESSION ID
DCRESRV1 DB 0 ; RESERVED
DCCURADD DW 0 ; CURSOR ADDRESS
DCCURTYP DB 0 ; CURSOR TYPE
DCRESRV2 DB 0 ; RESERVED

.
.
.

;
; INITIALIZE PARAMETER LIST FOR SET CURSOR POSITION
;
MOV DCRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV DCFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID INTO THE
MOV DCSESSID,AL ; PARAMETER LIST
MOV DCCURADD,0 ; DISPLAY CURSOR AT THE HOME POSITION
MOV DCCURTYP,01H ; CURSOR TYPE = BOX CURSOR

;
; INITIALIZE REGISTERS FOR SET CURSOR POSITION
;
MOV AH,09H
MOV AL,04H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,PCPSM ; RESOLVED VALUE FOR 'PCPSM '
MOV DI,SEG DCRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET DCRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR SET CURSOR POSITION SERVICE
;
INT 7AH
.
.
.
```

Presentation Space Service X'05': Switch Presentation Space

Use this service to specify a presentation space to become the default presentation space for all DOS and BIOS updates.

Register Values

On Request

AH = X'09'
AL = X'05'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for PCPSM
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'69')
2	1 byte	Session ID	Unchanged
3	1 byte	Reserved	Reserved

Parameter Definitions

Request Parameters:

- The session ID is the ID that was assigned to the presentation space by the Define Presentation Space service.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

Switch Presentation Space

- Presentation Space Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the presentation space management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Presentation space return codes use a function ID of X'69'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'0C'	Byte 0 of the parameter list was not zero on request.

See Appendix H, "Return Codes," for more information.

Coding Example

```
;
; PARAMETER LIST FOR SWITCH PRESENTATION SPACE
;
SPRETNCD DB 0 ; RETURN CODE
SPFXNID DB 0 ; FUNCTION NUMBER
SPSESSID DB 0 ; SESSION ID
SPRESERV DB 0 ; RESERVED
.
.
.
;
; INITIALIZE PARAMETER LIST FOR SWITCH PRESENTATION SPACE
;
MOV SPRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV SPFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID INTO THE LIST
MOV SPSESSID,AL
;
; INITIALIZE REGISTERS FOR SWITCH PRESENTATION SPACE
;
MOV AH,09H
MOV AL,05H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,PCPSM ; RESOLVED VALUE FOR 'PCPSM'
MOV DI,SEG SPRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET SPRETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR SWITCH PRESENTATION SPACE SERVICE
;
INT 7AH
.
.
.
```

Chapter 9. Coding 3270 Keystroke Emulation Service Requests

Introduction	9-2
Field Attribute Definition for 3270 Keystroke Emulation	9-2
Presentation Space Format for 3270 Keystroke Emulation	9-4
Requesting the 3270 Keystroke Emulation Services	9-5
Return Codes for the 3270 Keystroke Emulation Services	9-5
3270 Keystroke Emulation Service X'01': Connect for 3270 Keystroke Emulation	9-7
3270 Keystroke Emulation Service X'02': Disconnect for 3270 Keystroke Emulation	9-10
3270 Keystroke Emulation Service: Read Attention Identifier (AID)	9-13

Introduction

This chapter describes how to code requests for the 3270 keystroke emulation services provided by the API. *This service is allowed only if Multi-DOS is selected at customization time.*

The 3270 keystroke emulation services enable you to type into a personal computer presentation space as if it were a host presentation space. Keystrokes that previously were valid only for host sessions are processed by the 3270 keystroke emulation task for personal computer sessions as well.

The 3270 keystroke emulation services are activated in your personal computer session by issuing the INDEML command. To use the 3270 keystroke services, your application must first issue a Define Presentation Space command to define a presentation space. This presentation space should have no more than 24 rows or 80 columns. Screen row 25 is reserved for the operator information area (OIA). You must run the INDEML utility in each PC session for which you want to use 3270 Keystroke Emulation. See Chapter 10 in the *IBM 3270 Workstation Program User's Guide and Reference* for more information.

The format of a personal computer session defined to accept 3270 keystroke emulation is the same as the format of a standard personal computer session. However, the contents of that presentation space are interpreted and processed differently from other personal computer presentation spaces. A presentation space defined to accept 3270 keystroke emulation is interpreted as having 3270 field attributes as well as personal computer ASCII characters.

Field Attribute Definition for 3270 Keystroke Emulation

Field attributes are contained in two bytes and are defined as any personal computer ASCII character with a hexadecimal value between X'C0' and X'FF', and a character attribute of nondisplay (X'00'). This enables all 256 characters of the personal computer character set to be displayed with 3270 keystroke emulation. Field attributes occupy character positions within the presentation space. The first byte within a field is a field attribute character that defines the characteristics of the field. A field continues until the next field attribute is encountered in the presentation space. Fields within the presentation space can wrap from the bottom of the presentation space to the top of the presentation space. The 3270 keystroke emulation task interprets field attributes within the presentation space and applies the 3270 keystroke rule defined by the field attribute to all keystrokes entered into the presentation space field. The following table describes the field attribute character bit assignment. (Remember that bit 0 is the high-order, leftmost, bit in the byte, and bit 7 is the low-order, rightmost, bit in the byte.)

Note: Only the 3270 base attributes are supported.

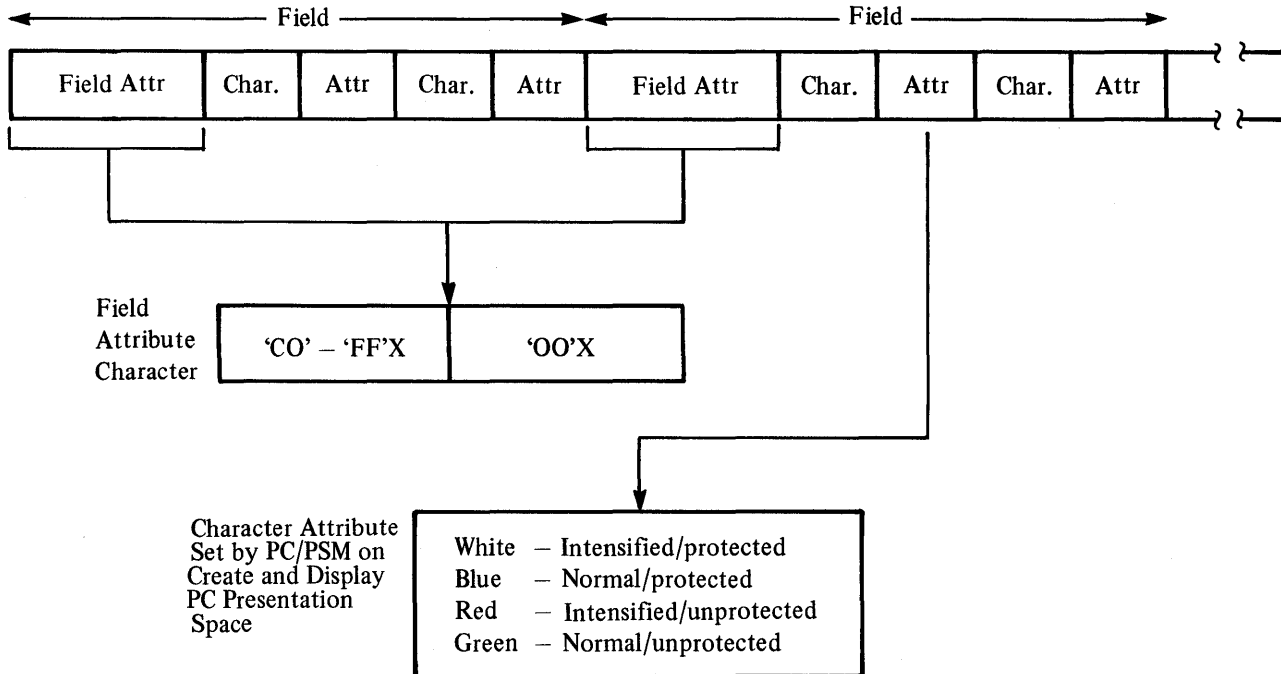
EBCDIC Bit	Field Characteristics
0, 1	11 = This byte is an attribute
2	0 = Unprotected 1 = Protected (see Note)
3	0 = Alphanumeric 1 = Numeric (if numeric lock capability is activated, causes automatic numeric shift of keyboard) (see Note)
4, 5	00 = Display not detectable by Cursor Select key 01 = Display detectable by Cursor Select key 10 = Intensified display detectable by Cursor Select key 11 = Nondisplay, nonprint, nondetectable
6	Reserved: always 0
7	Modified data tag (MDT); identifying modified fields during Read Modified command operation 0 = Field has not been modified 1 = Field has been modified by the operator. Can also be set by a program in the presentation space.
<i>Note: Binary 11 in bits 2 and 3 causes an automatic skip.</i>	

When a personal computer presentation space defined to accept 3270 keystroke emulation is defined or redisplayed using presentation space services, all character attributes in the presentation space are set to display the character in the color defined by that character's field attribute.

Introduction

Presentation Space Format for 3270 Keystroke Emulation

When defined to accept 3270 keystroke emulation, the presentation space is interpreted as follows:



The 3270 keystroke emulation services provided by the API are:

- **Connect for 3270 Keystroke Emulation Service:** Use this service to connect the 3270 keystroke emulation task to the session identified in the request.
- **Disconnect for 3270 Keystroke Emulation Service:** Use this service to disconnect the 3270 keystroke emulation task from the session identified in the request.
- **Read AID Key Service:** Use this service to enable operator input at the keyboard until a valid AID key is entered. The 3270 PC READ AID API allows you to choose how the application will receive AID keys. You can receive each AID key as a scan code/shift state or as a 2- or 4-byte ASCII mnemonic. Select the ASCII option by setting the high-order bit of byte 3 in the parameter list during READ API request. The ASCII mnemonic is returned in bytes 10 – 13 of the parameter list.

Your personal computer application program formats the presentation space by storing characters and field attributes directly in the presentation space buffer. After formatting the presentation space, use the Display Presentation Space and Display Cursor services to display the formatted presentation space.

After the presentation space has been formatted and displayed, request the Read AID Key service to enable operator input from the keyboard. When the Read AID Key service request is completed, your application program must interrogate the contents of the presentation space, or scan the field attributes for attributes with the modified data tag (MDT) bit set, to determine which fields have been modified. Your application should modify and, if necessary, redisplay the presentation space before the next Read AID Key service request.

Requesting the 3270 Keystroke Emulation Services

To request any of the 3270 keystroke emulation services, load the registers and the parameter list with the proper values, and use the INT 7AH instruction to signal the workstation program that it has a request to process.

Note: Before your application can request the 3270 keystroke emulation services, it must request the Name Resolution service, using '3270EML ' as the gate name in the parameter list. (Remember that the gate name must be padded to the right with blanks if it is less than eight characters.)

Return Codes for the 3270 Keystroke Emulation Services

Each 3270 keystroke emulation service has two return codes associated with it: a system return code and a 3270 keystroke emulation services return code. Both types of return codes are 2-byte values made up of a function ID and an error number. The function ID indicates the portion of the workstation program in which the error occurred. The error number indicates the specific type of error that has occurred. An error number of X'00' always indicates a successful acceptance or completion of the request.

- **System Return Codes:**

After your application has requested a 3270 keystroke emulation service, the CH and CL registers contain a return code generated by the request processing portion of the workstation program. The function ID is in the CH register, and the error number is in the CL register. System return codes use a function ID of X'12'. The error codes that can appear are:

Code	Meaning
X'00'	Request accepted.
X'05'	Invalid index specified.
X'07'	Invalid reply specified.
X'08'	Invalid wait type specified.
X'0B'	RQE pool depleted.
X'0F'	Invalid environment access.
X'34'	Invalid gate entry.

These system return codes apply to all 3270 keystroke emulation services.

- **3270 Keystroke Emulation Services Return Codes:**

After a requested 3270 keystroke emulation service is completed, bytes 0 and 1 of the parameter list contain a return code generated by the 3270 keystroke emulation management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. The 3270 keystroke emulation services return codes use a function ID of X'6E'. The error numbers that can appear are specific to the service that was requested and are included in the descriptions of each service.

See Appendix H, "Return Codes," for more information.

3270 Keystroke Emulation Service X'01': Connect for 3270 Keystroke Emulation

Use this service to attach a 3270 keystroke emulation task to your PC presentation space that has been defined to accept 3270 keystroke emulation. On successful completion of this service, operator input to the keyboard of the connected session is disabled, so that the operator cannot type keystrokes to that session from the keyboard.

Register Values

On Request

AH = X'09'
AL = X'01'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for 3270EML
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

BL = Return type
CH = X'12'
CL = Return code

The contents of registers AX, BH, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Reserved	Function ID (X'6E')
2	1 byte	Session ID	Unchanged
3	1 byte	X'00'	Unchanged
4	1 word	X'00'	Keystroke task ID
6	1 word	Work area offset	Unchanged
8	1 word	Work area segment	Unchanged

Parameter Definitions

Request Parameters:

- The session ID is the ID of the PC presentation space returned on the Define Presentation Space request. The PC presentation space must be defined to accept 3270 keystroke emulation.
- The work area is a 700-byte area of working storage that your application program must provide. The work area is allocated to the keystroke emulation task until the Disconnect 3270 Keystroke Emulation service request is issued.

Completion Parameters:

- The keystroke task ID is the task ID of the 3270 keystroke emulation task. This task ID is required on both the Read AID Key service and the Disconnect 3270 Keystroke Emulation service request. This is the task provided by the workstation program that performs the 3270 keystroke emulation for the specified presentation space.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- 3270 Keystroke Emulation Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the 3270 keystroke emulation management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. The 3270 keystroke emulation services return codes use a function ID of X'6E'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'08'	An unsuccessful return code was encountered while processing the request.
X'0C'	Byte 0 of the parameter list is not 0 on the request.

See Appendix H, "Return Codes," for more information.

Coding Example

```
;
; PARAMETER LIST FOR CONNECT FOR 3270 KEYSTROKE EMULATION
;
CERETNCD DB 0 ; RETURN CODE
CEFXNID DB 0 ; FUNCTION NUMBER
CESESSID DB 0 ; SESSION ID
CEZERO DB 0 ; MUST BE ZERO
CEKEY$ID DW 0 ; KEYSTROKE TASK ID
CEWRKOFF DW 0 ; WORK AREA OFFSET
CEWRKSEG DW 0 ; WORK AREA SEGMENT
.
.
.

;
; INITIALIZE PARAMETER LIST FOR CONNECT FOR 3270 KEYSTROKE EMULATION
;
MOV CERETNCD,00H ; CERETNCD MUST BE 0 BEFORE REQUEST
MOV CEFXNID,00H ; CEFXNID MUST BE 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID OBTAINED FROM DEFINE
MOV CESESSID,AL ; PRESENTATION SPACE API
MOV CEZERO,0 ; MUST BE ZERO
MOV AX,OFFSET WORKAREA ; OFFSET OF THE WORK AREA IN LIST
MOV CEWRKOFF,AX
MOV AX,SEG WORKAREA ; SEGMENT OF THE WORK AREA IN LIST
MOV CEWRKSEG,AX

;
; INITIALIZE REGISTERS FOR CONNECT FOR 3270 KEYSTROKE EMULATION
;
MOV AH,09H
MOV AL,01H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,3270EML ; RESOLVED VALUE FOR '3270EML '
MOV DI, SEG CERETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET CERETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR CONNECT FOR 3270 KEYSTROKE EMULATION SERVICE
;
INT 7AH
.
.
.
```


Disconnect for 3270 Keystroke Emulation

3270 Keystroke Emulation Service X'02': Disconnect for 3270 Keystroke Emulation

Use this service to detach the 3270 keystroke emulation task from your PC presentation space that has been defined to accept 3270 keystroke emulation.

Register Values

On Request

AH = X'09'
AL = X'02'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for 3270EML
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

BL = Return type
CH = X'12'
CL = Return code

The contents of registers AX, BH, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Reserved	Function ID (X'6E')
2	1 byte	Session ID	Unchanged
3	1 byte	X'00'	Unchanged
4	1 word	Keystroke task ID	Unchanged

Parameter Definitions

Request Parameters:

- The session ID is the ID of the PC presentation space that was specified on the Connect to 3270 Keystroke Emulation request.
- The keystroke task ID must be the ID of the 3270 keystroking task returned on the Connect for 3270 Keystroke request.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- 3270 Keystroke Emulation Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the 3270 keystroke emulation management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. The 3270 keystroke emulation services return codes use a function ID of X'6E'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'08'	An unsuccessful return code was encountered while processing the request.
X'0C'	Byte 0 of the parameter list is not 0 on the request.

See Appendix H, "Return Codes," for more information.

Usage Notes

- You cannot request the Disconnect for 3270 Keystroke Emulation service while you have a Read AID Key service request outstanding. That is, if you have requested the Read AID Key service and specified asynchronous processing, you must use the Get Request Completion service to obtain the values on completion in the parameter list before you can request the Disconnect for 3270 Keystroke Emulation service.

Disconnect for 3270 Keystroke Emulation

Coding Example

```
;
; PARAMETER LIST FOR DISCONNECT FOR 3270 KEYSTROKE EMULATION
;
DERETNCD DB 0 ; RETURN CODE
DEFXNID DB 0 ; FUNCTION NUMBER
DESESSID DB 0 ; SESSION ID
DEZERO DB 0 ; MUST BE ZERO
DEKEY$ID DW 0 ; KEYSTROKE TASK ID
.
.
.
;
; INITIALIZE PARAMETER LIST FOR DISCONNECT FOR 3270 KEYSTROKE EMULATION
;
MOV DERETNCD,00H ; DERETNCD MUST BE 0 BEFORE REQUEST
MOV DEFXNID,00H ; DEFXNID MUST BE 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID OBTAINED FROM REQUEST
MOV DESESSID,AL ; TO DEFINE PRESENTATION SPACE API
MOV DEZERO,0 ; MUST BE ZERO
MOV AX,KEYTSKID ; KEYSTROKE TASK ID IN LIST (THE ID IS RETURN
ED MOV DEKEY$ID,AX ; FROM CONNECT TO 3270 KEYSTROKE EMULATION)
;
; INITIALIZE REGISTERS FOR DISCONNECT FOR 3270 KEYSTROKE EMULATION
;
MOV AH,09H
MOV AL,02H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,3270EML ; RESOLVED VALUE FOR '3270EML '
MOV DI,SEG DERETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET DERETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR DISCONNECT FOR 3270 KEYSTROKE EMULATION SERVICE
;
INT 7AH
.
.
.
```

3270 Keystroke Emulation Service: Read Attention Identifier (AID) Key

Use this service to receive AID keystrokes from the 3270 keystroke emulation task that is performing 3270 keystroke emulation for the specified presentation space. The Read AID Key service begins keystroke processing by enabling operator input at the keyboard of the connected session. As keystrokes are entered, the presentation space is updated using 3270 keystroke rules until a valid AID key is entered. When an AID key is encountered, operator input to the connected session's keyboard is again disabled. The Read AID key service returns the AID key in the parameter list in one of two formats: scan code/shift state format or ASCII mnemonic format. Select the ASCII format by setting the ASCII option of byte 3 of the parameter list upon request. The READ AID key service also returns the current row and column position of the cursor.

Register Values

On Request

AH = X'09'
BH = Synchronous or asynchronous *
BL = Synchronous or asynchronous *
CX = X'00FF'
DX = Keystroke task ID
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

BL = Return type
CH = X'12'
CL = Return code

The contents of registers AX, BH, DX, ES, and DI are unpredictable.

* The values in these registers depend on whether you want the request to be processed synchronously or asynchronously. See the following description of request register values for more information.

• Request Register Values:

You can specify synchronous or asynchronous processing of the Read AID Key service. In synchronous processing, control is returned to your application program after the workstation program has completed the request. In asynchronous processing, control is returned to your application program before the workstation program has completed the request. You must use the Get Request Completion service to obtain the parameter list values on completion when you request asynchronous processing.

Synchronous Processing:

There are two ways to specify synchronous processing:

1. Set the BH register to X'80' and the BL register to X'20'. When the request is completed, control is returned to your application program and the registers and parameter list contain the values for completion of the request.

Read AID Key

2. Set both the BH and BL registers to X'40'. When the request is completed, control is returned to your program, but the parameter list values for completion of the request are not obtained until you request the Get Request Completion service.

Asynchronous Processing:

For asynchronous processing of the Read AID key service request, set the BH register to X'40' and the BL register to X'00'. When asynchronous processing is specified, you must request the Get Request Completion service to obtain the results of the Read AID Key service.

- Completion register values:

If you specified asynchronous processing, or synchronous processing using X'40' in both the BH and BL registers on request, the AX register contains a request ID that the workstation program assigned to the request. You use this request ID to match the results of the service obtained by the Get Request Completion service to the results of this service. That is, when the request ID in the AX register, on completion of the Get Request Completion service, matches the request ID in the AX register on completion of this service, the results obtained by the Get Request Completion service pertain to this request.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Reserved	Function ID (X'6E')
2	1 byte	Session ID	Unchanged
3	1 byte	Options byte	Unchanged
4	1 word	Reserved	Reserved
6	1 byte	X'00'	Scan code or length of mnemonic starting at byte 10
7	1 byte	X'00'	Shift state or unchanged
8	1 byte	X'00'	Cursor row
9	1 byte	X'00'	Cursor column
10-13	2-4 bytes	Reserved	ASCII mnemonic. May be 2 or 4 bytes long.

Parameter Definitions

Request Parameters:

- The session ID is the ID of the presentation space to which the keystroke emulation task is attached.
- The options byte has the following values:
 - X'00': Previous AID key was accepted, return AID keys in scan/shift.
 - X'01': Previous AID key was rejected, return AID keys in scan/shift.
 - X'80': Previous AID key was accepted, return AID keys in ASCII.
 - X'81': Previous AID key was rejected, return AID keys in ASCII.

Completion Parameters:

- The scan code or length of mnemonic field (byte 6 of the parameter list) is a hexadecimal value that could contain one of two values:
 - If the options byte was set to X'80' or X'81' upon request, byte 6 will contain the length of the ASCII mnemonic being returned. If this byte is X'02', then bytes 10–11 of the parameter list contain the 2-byte ASCII mnemonic being returned, and bytes 12–13 are unchanged. If this byte is X'04', then bytes 10–13 of the parameter list contain the 4-byte ASCII mnemonic being returned.
 - If the options byte was set to X'00' or X'01' upon request, byte 6 will contain the scan code of the AID key being reported. Bytes 10–13 are unchanged.

The AID keys, and their associated hexadecimal scan code and ASCII mnemonics, are shown in the table below:

AID Key	Hexadecimal Scan Code Returned	ASCII Mnemonic Returned
Enter or CrSel on Ampersand (&) *	X'58'	@E
Numeric Pad Enter	X'79'	@E
PF1	X'07'	@1
PF2	X'0F'	@2
PF3	X'17'	@3
PF4	X'1F'	@4
PF5	X'27'	@5

Read AID Key

AID Key	Hexadecimal Scan Code Returned	ASCII Mnemonic Returned
PF6	X'2F'	@6
PF7	X'37'	@7
PF8	X'3F'	@8
PF9	X'47'	@9
PF10	X'4F'	@a
PF11	X'56'	@b
PF12	X'5E'	@c
PF13	X'08'	@d
PF14	X'10'	@e
PF15	X'18'	@f
PF16	X'20'	@g
PF17	X'28'	@h
PF18	X'30'	@i
PF19	X'38'	@j
PF20	X'40'	@k
PF21	X'48'	@l
PF22	X'50'	@m
PF23	X'57'	@n
PF24	X'5F'	@o
CrSel on a Space or null *	X'03'	@A@J
PA1	X'67'	@x
PA2	X'6E'	@y
PA3	X'6F'	@z
Attn	X'0C'	@A@Q
Clear	X'06'	@C

* The 3270 Workstation Program uses the CrSel key the same way a light pen is used. If CrSel is pressed when the cursor is on a light pen detectable field, the workstation program may do one of four things:

1. It returns an Enter AID key if the field begins with an ampersand (&).
2. It returns the CrSel key itself if the field begins with a null or a space.
3. It returns no AID key if the field begins with a question mark (?). The question mark is, however, changed to a '>' and the modified data tag (MDT) bit in the field attribute is set on.
4. It returns no AID key if the field begins with a greater than sign (>). The greater than sign is, however, changed to a '?' and the modified data tag (MDT) bit in the field attribute is set off.

If the cursor is not on a light pen detectable field, an input-inhibit condition results.

- When the Clear key is pressed, the presentation space is cleared.
- When all other AID keys in this table are pressed, the MDT bit is set in the field attribute byte of all modified fields in the presentation space.
- The SysRq and Test keys are not supported by 3270 keystroke emulation.
- The shift state indicates the shift conditions that were active when the AID key was sent to your application program. The format of the shift byte is as follows:

0, 1	2	3	4	5	6	7
Reserved	Right shift	Left shift	Control key	ALT keys	Shift Lock	Upshift keys

- Bits 0 and 1 are reserved.
- Bit 2 represents the right upshift key.
- Bit 3 represents the left upshift key.
- Bit 4 represents the control shift state.
- Bit 5 represents the ALT shift state.
- Bit 6 represents the Shift Lock state.
- Bit 7 represents the upshift state. Bit 7 indicates that one of the two upshift keys was pressed. If your application program must distinguish between the right upshift key and the left upshift key, use bits 2 and 3.
- Lower shift is represented by a value of X'00'.
- “Cursor row” is the row position of the cursor on the specified presentation space. Cursor row positions start at zero.
- “Cursor column” is the column position of the cursor on the specified presentation space. Cursor column positions start at zero.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

Read AID Key

- 3270 Keystroke Emulation Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the 3270 keystroke emulation management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. The 3270 keystroke emulation services return codes use a function ID of X'6E'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion. The scan code for AID key was returned in parameter list.
X'02'	Invalid session ID.
X'08'	An unsuccessful return code was detected while processing the request.
X'0C'	Byte 0 of the presentation list is not 0 on the request.

See Appendix H, "Return Codes," for more information.

Usage Notes

- If you specified asynchronous processing, or synchronous processing using X'40' in both the BH and BL registers on request, you must use the Get Request Completion service to obtain the results in the parameter list when the Read AID Key service is completed.

Coding Example

```

;
; PARAMETER LIST FOR READ AID KEY
;
RARETNCD DB 0 ; RETURN CODE
RAFXNID DB 0 ; FUNCTION NUMBER
RASESSID DB 0 ; SESSION ID
RAACCRESJ DB 0 ; ACCEPT/REJECT AID
          DW 0 ; UNUSED
RASCNCDE DB 0 ; SCAN CODE
RASHFTST DB 0 ; SHIFTSTATE
RACURS$R DB 0 ; CURSOR ROW POSITION
RACURS$C DB 0 ; CURSOR COLUMN POSITION
          .
          .
          .
;
; INITIALIZE PARAMETER LIST FOR READ AID KEY
;
      MOV RARETNCD,00H ; RARETNCD MUST BE 0 BEFORE REQUEST
      MOV RAFXNID,00H ; RAFXNID MUST BE 0 BEFORE REQUEST
      MOV AL,SESSID ; SESSION ID OBTAINED FROM REQUEST
      MOV RASESSID,AL ; TO DEFINE PRESENTATION SPACE API
      MOV AL,ACC$REJ ; ACCEPT OR REJECT PREVIOUS AID IN LIST
      MOV RAACCRESJ,AL
;
; INITIALIZE REGISTERS FOR READ AID KEY
;
      MOV AH,09H
      MOV BH,80H ; REPLY TYPE
      MOV BL,20H ; WAIT TYPE
      MOV CX,0FFH ; PRIORITY
      MOV DX,KEYTSKID ; KEYSTROKE TASK ID RETURNED FROM CONNECT TO
                      ; TO 3270 KEYSTROKE EMULATION
      MOV DI, SEG RARETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
      MOV ES,DI ; IN ES
      MOV DI,OFFSET RARETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR READ AID KEY SERVICE
;
      INT 7AH
      .
      .
      .

```


Chapter 10. Coding Copy Service Requests

Introduction	10-2
Requesting the Copy Services	10-3
Return Codes for the Copy Services	10-4
Copy Service X'01': Copy String	10-5
Copy Service X'02': Copy Block	10-12
Copy Service X'03': Connect for Copy to PC Session	10-19
Copy Service X'04': Disconnect for Copy to PC Session	10-22

Introduction

This chapter describes how to code requests for the copy services provided by the API.

The copy services allow your application program to copy data into a personal computer window, as well as copy data from one area into another within the same personal computer window. The copy services also allow copying of data between any host and notepad sessions. (Before you can use a personal computer window as the target for a copy operation, you must use the Connect for Copy to PC Session service to designate the session as a valid copy target. You must also use the Disconnect for Copy to PC Session service when you no longer want your personal computer session to be a target for copy operations.)

The copy services also allow your application program to copy data from one presentation space or buffer to another. Copying graphics characters or program symbol set characters is not allowed.

When copying between sessions that have different character code types (i.e., PC characters and host/notepad characters), a translation will occur.

The copy services provided by the API are:

- **Copy String Service:** Use this service to copy a string from a specified presentation space or buffer into another specified presentation space or buffer.
- **Copy Block Service:** Use this service to copy a block from a specified presentation space or buffer into another specified presentation space or buffer.
- **Connect for Copy to PC Session Service:** Use this service to identify a personal computer session as being a valid target session for the copy services.
- **Disconnect for Copy to PC Session Service:** Use this service to identify a personal computer session as no longer being a valid target for the copy services.

Copy services are available for use only if you specify COPY = YES at customization time.

The major copy operations are: copy string and copy block. The copy string operation copies all characters beginning with the specified starting character up to and including the specified ending character, as shown below:

- Copy string

If the source is specified as:

```
Now is the time for all good  
women to come to the aid of their part y
```

the string copied to the target is:

```
Now is the time for all good  
women to come to the aid of their party
```

The copy block operation copies all characters in the block of text formed by the specified starting and ending characters, as shown below:

- Copy block

If the source is specified as:

```
Now is the time for all good  
women to come to the aid of their part y
```

the block copied to the target is:

```
Now is the time for all good  
men to come to the aid of their party
```

Requesting the Copy Services

To request any of the copy services, load the registers and the parameter list with the proper values, and use the INT 7AH instruction to signal the workstation program that it has a request to process.

Note: Before your application can request the copy services, it must request the Name Resolution service, using 'COPY ' as the gate name in the parameter list. (Remember that the gate name must be padded to the right with blanks if it is less than eight characters.)

Return Codes for the Copy Services

Each copy service has two return codes associated with it, a system return code and a copy service return code. Both types of return codes are 2-byte values made up of a function ID and an error number. The function ID indicates the portion of the workstation program in which the error occurred. The error number indicates the specific type of error that has occurred. An error number of X'00' indicates a successful acceptance or completion of the request.

- **System Return Codes:**

After your application has requested a copy service, the CH and CL registers contain a return code generated by the request processing portion of the workstation program. The function ID is in the CH register, and the error number is in the CL register. System return codes use a function ID of X'12'. The error codes that can appear are:

Code	Meaning
X'00'	Request accepted.
X'05'	Invalid index specified.
X'07'	Invalid reply specified.
X'08'	Invalid wait type specified.
X'0B'	RQE pool depleted.
X'34'	Invalid gate entry.

These system return codes apply to all the copy services.

- **Copy Services Return Codes:**

After a requested copy service has completed, bytes 0 and 1 of the parameter list contain a return code generated by the copy management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Copy services return codes use a function ID of X'64'. The error numbers that can appear are specific to the service that was requested, and are included in the descriptions of each service.

See Appendix H, "Return Codes," for more information.

Copy Service X'01': Copy String

Use this service to copy a string from one presentation space or buffer into another.

Register Values

On Request

AH = X'09'
AL = X'01'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for COPY
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'64')
2	1 byte	Source session ID or zero*	Unchanged
3	1 byte	Reserved - must be X'00'	Reserved
4	1 word	Offset address of source buffer or zero*	Unchanged
6	1 word	Segment address of source buffer*	Unchanged
8	1 byte	Source characteristics*	Unchanged
9	1 byte	Source session type*	Unchanged
10	1 word	Offset of source starting character	Unchanged
12	1 word	Offset of source ending character	Unchanged
14	1 byte	Target session ID* or zero	Unchanged

* The contents of this offset depend on whether a presentation space or a buffer is being used as the copy source/target. See the parameter definitions for more information.

Copy String

Offset	Length	Contents on Request	Contents on Completion
15	1 byte	Reserved - must be X'00'	Reserved
16	1 word	Offset address of target buffer*	Unchanged
18	1 word	Segment address of target buffer*	Unchanged
20	1 byte	Target characteristics*	Unchanged
21	1 byte	Target session type*	Unchanged
22	1 word	Offset of target starting character	Unchanged
24	1 byte	Copy mode	Unchanged
25	1 byte	Reserved	Reserved

* The contents of this offset depend on whether a presentation space or a buffer is being used as the copy source/target. See the parameter definitions for more information.

Parameter Definitions

Request Parameters:

- **If the copy source is a presentation space:**
 - The source session ID is the ID of the session containing the string to copy.
 - Offsets 3 through 9 of the parameter list must be zero.
 - The source starting character is the character offset into the presentation space of the starting character of the string to be copied. This is the number of characters, not including the attributes. Character offsets begin with zero.
 - The source ending character is the character offset into the presentation space of the ending character of the string to be copied. This is the number of characters, not including the attributes. Character offsets begin with zero.

- **If the copy source is a buffer:**
 - Offset 2 of the parameter list must be zero.
 - The source buffer contains the source string for the copy.
 - The source characteristics apply to DFT host sessions only as follows:
 - If bit 7 = 0, the source has base attributes.
 - If bit 7 = 1, the source has extended attributes.
 - The source session type is one of the following:
 - X'02' — DFT host session
 - X'03' — CUT host session
 - X'04' — notepad session
 - X'05' — PC session
 - The source starting character is the byte offset into the buffer of the starting character of the string to be copied. Byte offsets begin with zero.
 - The source ending character is the byte offset into the buffer of the ending character of the string to be copied. Byte offsets begin with zero.
- **If the copy target is a presentation space:**
 - The target session ID is the ID of the session to receive the copied string.
 - Offsets 15 through 21 of the parameter list must be zero.
 - The target starting character is the character offset into the presentation space of the character to place the beginning of the copied string. This is the number of characters not including the attributes. Character offsets begin with zero.

- If the copy target is a buffer:
 - The target buffer contains the target data area for the copy.
 - Offset 14 must be zero.
 - The target characteristics apply to DFT host sessions only, and are as follows:
 - If bit 7 = 0, the target has base attributes.
 - If bit 7 = 1, the target has extended attributes.
 - The target session type is one of the following:
 - X'02' — DFT host session
 - X'03' — CUT host session
 - X'04' — notepad session
 - X'05' — PC session
 - The target starting character is the byte offset into the buffer of the starting character of the string to be copied. Byte offsets begin with zero.
- The copy mode is specified as follows:
 - To force the desired color mode (refer to *IBM 3270 Workstation Program User's Guide and Reference* for the definition of the color modes):
 - X'00' = 4-color mode
 - X'80' = 7-color mode
 - If the target is a personal computer presentation and is connected for 3270 keystroke emulation or if the target is a PC buffer:
 - X'00' = field attributes not copied
 - X'40' = field attributes copied

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Copy Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the copy management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. The copy services return codes use a function ID of X'64'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'01'	Source not allowed.
X'02'	Invalid session ID.
X'03'	The target window is input-inhibited.
X'05'	The source and target areas overlap (copy performed).
X'06'	There is a missing or invalid parameter on the source definition.
X'07'	There is a missing or invalid parameter on the target definition.
X'09'	Truncation occurred (copy performed). You attempted to copy past the end of the presentation space.
X'0C'	Byte 0 of the parameter list was not zero on request.
X'0D'	The target was not allowed.
X'0E'	The target window is protected.
X'0F'	The copy of the field attributes was not allowed (copy not performed).

See Appendix H, "Return Codes," for more information.

Copy String

Coding Example

```
;
; PARAMETER LIST FOR COPY STRING
;
CSRETNCD DB 0 ; RETURN CODE
CSFXNID DB 0 ; FUNCTION ID
CSSSESID DB 0 ; SOURCE SESSION ID
CSRESRV1 DB 0 ; RESERVED - MUST BE X'00'
CSSRCOFF DW 0 ; OFFSET ADDRESS OF SOURCE BUFFER
CSSRCSEG DW 0 ; SEGMENT ADDRESS OF SOURCE BUFFER
CSSRCCHR DB 0 ; SOURCE CHARACTERISTICS
CSSRCTYP DB 0 ; SOURCE SESSION TYPE
CSSSTRTC DW 0 ; OFFSET OF SOURCE STARTING CHARACTER
CSSEND C DW 0 ; OFFSET OF SOURCE ENDING CHARACTER
CSTSEID DB 0 ; TARGET SESSION ID
CSRESRV2 DB 0 ; RESERVED - MUST BE X'00'
CSTRGOFF DW 0 ; OFFSET ADDRESS OF TARGET BUFFER
CSTRGSEG DW 0 ; SEGMENT ADDRESS OF TARGET BUFFER
CSTRGCHR DB 0 ; TARGET CHARACTERISTICS
CSTRGTYP DB 0 ; TARGET SESSION TYPE
CSTSTRTC DW 0 ; OFFSET OF TARGET STARTING CHARACTER
CSMODE DB 0 ; COPY MODE
CSRESRV3 DB 0 ; RESERVED

.
.
.

;
; COPY THE FIRST LINE OF A NOTEPAD TO A PC BUFFER
;
;
; INITIALIZE PARAMETER LIST FOR COPY STRING
;
MOV CSRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV CSFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SRCSID ; SOURCE SESSION ID INTO THE LIST
MOV CSSSESID,AL ;
MOV CSRESRV1,0 ; RESERVED - MUST BE 0
MOV CSSRCOFF,0 ; SOURCE OFFSET NOT USED
MOV CSSRCSEG,0 ; SOURCE SEGMENT NOT USED
MOV CSSRCCHR,00H ; SOURCE CHARACTERISTICS NOT USED
MOV CSSRCTYP,04H ; SOURCE TYPE = NOTEPAD SESSION
MOV CSSSTRTC,0 ; SOURCE STARTING CHARACTER OFFSET =0
MOV CSSEND C,80 ; SOURCE ENDING CHARACTER OFFSET = 80
;
MOV CSTSEID,0 ; TARGET SESSION ID NOT USED
MOV CSRESRV2,0 ; RESERVED - MUST BE 0
MOV AX,OFFSET TARGET ; TARGET OFFSET INTO THE LIST
MOV CSTRGOFF,AX ;
MOV AX,SEG TARGET ; TARGET SEGMENT INTO THE LIST
MOV CSTRGSEG,AX ;
MOV CSTRGCHR,0 ; TARGET CHARACTERISTICS NOT USED
MOV CSTRGTYP,05H ; TARGET TYPE = PC SESSION
MOV CSTSTRTC,0 ; TARGET STARTING CHARACTER OFFSET =0
MOV CSMODE,01000000B ; COPY MODE = 4-COLOR WITH FIELD ATT.
```

```
;
; INITIALIZE REGISTERS FOR COPY STRING
;
    MOV     AH,09H
    MOV     AL,01H
    MOV     BH,80H
    MOV     BL,20H
    MOV     CX,0FFH
    MOV     DX,COPY           ; RESOLVED VALUE FOR 'COPY      '
    MOV     DI, SEG CSRETNCD  ; SEGMENT ADDRESS OF PARAMETER LIST
    MOV     ES,DI             ; IN ES
    MOV     DI,OFFSET CSRETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR COPY STRING SERVICE
;
    INT     7AH
    .
    .
    .
```

Copy Block

Copy Service X'02': Copy Block

Use this service to copy a block from one presentation space or buffer into another.

Register Values

On Request

AH = X'09'
AL = X'02'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for COPY
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'64')
2	1 byte	Source session ID or zero*	Unchanged
3	1 byte	Reserved - must be X'00'	Reserved
4	1 word	Offset address of source buffer or zero*	Unchanged
6	1 word	Segment address of source buffer*	Unchanged
8	1 byte	Source characteristics*	Unchanged
9	1 byte	Source session type*	Unchanged
10	1 word	Offset of source starting character	Unchanged
12	1 word	Offset of source ending character	Unchanged
14	1 byte	Target session ID* or zero	Unchanged

* The contents of this offset depend on whether a presentation space or a buffer is being used as the copy source/target. See the parameter definitions for more information.

Offset	Length	Contents on Request	Contents on Completion
15	1 byte	Reserved - must be X'00'	Reserved
16	1 word	Offset address of target buffer*	Unchanged
18	1 word	Segment address of target buffer*	Unchanged
20	1 byte	Target characteristics*	Unchanged
21	1 byte	Target session type*	Unchanged
22	1 word	Offset of target starting character	Unchanged
24	1 byte	Copy mode	Unchanged
25	1 byte	Reserved	Reserved

* The contents of this offset depend on whether a presentation space or a buffer is being used as the copy source/target. See the parameter definitions for more information.

Parameter Definitions

Request Parameters:

- **If the copy source is a presentation space:**
 - The source session ID is the ID of the session containing the block to copy.
 - Offsets 3 through 9 of the parameter list must be zero.
 - The source starting character is the character offset into the presentation space of the starting character of the block to be copied. This is the number of characters, not including the attributes. Character offsets begin with zero.
 - The source ending character is the character offset into the presentation space of the ending character of the block to be copied. This is the number of characters, not including the attributes. Character offsets begin with zero.

- **If the copy source is a buffer:**

- Offset 2 of the parameter list must be zero.
- The source buffer contains the source block for the copy.
- The source characteristics apply to DFT host sessions only as follows:

If bit 7 = 0, the source has base attributes.

If bit 7 = 1, the source has extended attributes.

- The source session type is one of the following:

X'02' — DFT host session

X'03' — CUT host session

X'04' — notepad session

X'05' — PC session

- The source starting character is the byte offset into the buffer of the starting character of the block to be copied. Byte offsets begin with zero.
- The source ending character is the byte offset into the buffer of the ending character of the block to be copied. Byte offsets begin with zero.

- **If the copy target is a presentation space:**

- The target session ID is the ID of the session to receive the copied block.
- Offsets 15 through 21 of the parameter list must be zero.
- The target starting character is the character offset into the presentation space of the character to place the beginning of the copied block. This is the number of characters not including the attributes. Character offsets begin with zero.

- **If the copy target is a buffer:**
 - Offset 14 must be zero.
 - The target buffer contains the target data area for the copy.
 - The target characteristics apply to DFT host sessions only, and are as follows:
 - If bit 7 = 0, the source has base attributes.
 - If bit 7 = 1, the source has extended attributes.
 - The target session type is one of the following:
 - X'02' — DFT host session
 - X'03' — CUT host session
 - X'04' — notepad session
 - X'05' — PC session
 - The target starting character is the byte offset into the buffer of the starting character of the block to be copied. Byte offsets begin with zero.
- The copy mode is specified as follows:
 - To force the desired color mode (refer to *IBM 3270 Workstation Program User's Guide and Reference* for the definition of color modes):
 - X'00' = 4-color mode
 - X'80' = 7-color mode
 - If the target is a personal computer presentation and is connected for 3270 keystroke emulation or if the target is a PC buffer:
 - X'00' = Field attributes were not copied
 - X'40' = Field attributes were copied

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Copy Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the copy management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. The copy services return codes use a function ID of X'64'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'01'	Source not allowed.
X'02'	Invalid session ID.
X'03'	The target window is input-inhibited.
X'05'	The source and target areas overlap (copy performed).
X'06'	There is a missing or invalid parameter on source definition.
X'07'	There is a missing or invalid parameter on target definition.
X'09'	Truncation occurred (copy performed). You attempted to copy past the end of the presentation space.
X'0C'	Byte 0 of the parameter list was not zero on request.
X'0D'	The target was not allowed.
X'0E'	The target window is protected.
X'0F'	A copy of the field attributes was not allowed (copy not performed).

See Appendix H, "Return Codes," for more information.

Coding Example

```

;
; PARAMETER LIST FOR COPY BLOCK
;
CBRETNCD DB 0 ; RETURN CODE
CBFXNID DB 0 ; FUNCTION ID
CBSSESID DB 0 ; SOURCE SESSION ID
CBRESRV1 DB 0 ; RESERVED
CBSRCOFF DW 0 ; OFFSET ADDRESS OF SOURCE BUFFER
CBSRCSEG DW 0 ; SEGMENT ADDRESS OF SOURCE BUFFER
CBSRCCHR DB 0 ; SOURCE CHARACTERISTICS
CBSRCTYP DB 0 ; SOURCE SESSION TYPE
CBSSTRTC DW 0 ; OFFSET OF SOURCE STARTING CHARACTER
CBSEND DC DW 0 ; OFFSET OF SOURCE ENDING CHARACTER
CBTSESID DB 0 ; TARGET SESSION ID
CBRESRV2 DB 0 ; RESERVED
CBTRGOFF DW 0 ; OFFSET ADDRESS OF TARGET BUFFER
CBTRGSEG DW 0 ; SEGMENT ADDRESS OF TARGET BUFFER
CBTRGCHR DB 0 ; TARGET CHARACTERISTICS
CBTRGTYP DB 0 ; TARGET SESSION TYPE
CBTSTRTC DW 0 ; OFFSET OF TARGET STARTING CHARACTER
CBMODE DB 0 ; COPY MODE
CBRESRV3 DB 0 ; RESERVED

.
.
.

;
; COPY A BLOCK FROM THE HOST TO A NOTEPAD
;
; INITIALIZE THE PARAMETER LIST FOR COPY BLOCK
;
MOV CBRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV AL,SRCSID ; SOURCE SESSION ID INTO THE LIST
MOV CBSSESID,AL
MOV CBSRCOFF,0 ; SOURCE OFFSET NOT USED
MOV CBSRCSEG,0 ; SOURCE SEGMENT NOT USED
MOV CBSRCCHR,10000000B ; SOURCE CHARACTERISTICS = SOURCE HAS
; EXTENDED ATTRIBUTES
MOV CBSRCTYP,02H ; SOURCE TYPE = DFT HOST SESSION
MOV CBSSTRTC,0 ; SOURCE STARTING CHARACTER OFFSET =0
MOV CBSEND DC,835 ; SOURCE ENDING CHARACTER OFFSET =835
MOV AL,TRGSESID ; TARGET SESSION ID INTO THE LIST
MOV CBTSESID,AL
MOV CBTRGOFF,0 ; TARGET OFFSET NOT USED
MOV CBTRGSEG,0 ; TARGET SEGMENT NOT USED
MOV CBTRGCHR,00H ; TARGET CHARACTERISTICS NOT USED
MOV CBTRGTYP,04H ; TARGET TYPE = NOTEPAD SESSION
MOV CBTSTRTC,440 ; TARGET STARTING CHARACTER OFFSET
; = 440
MOV CBMODE,0 ; COPY MODE NOT USED

```

Copy Block

```
;
; INITIALIZE REGISTERS FOR COPY BLOCK
;
      MOV    AH,09H
      MOV    AL,02H
      MOV    BH,80H
      MOV    BL,20H
      MOV    CX,0FFH
      MOV    DX,COPY          ; RESOLVED VALUE FOR 'COPY '
      MOV    DI, SEG CBRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
      MOV    ES,DI           ; IN ES
      MOV    DI,OFFSET CBRETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR COPY BLOCK SERVICE
;
      INT     7AH
      .
      .
      .
```

Copy Service X'03': Connect for Copy to PC Session

Use this service to identify a personal computer session as being a valid target session for the copy services.

Register Values

On Request

AH = X'09'
AL = X'03'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for COPY
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code
 The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'64')
2	1 byte	Session ID	Unchanged
3	1 byte	Reserved — must be X'00'	Reserved

Parameter Definitions

Request Parameters:

- The session ID is the ID of a personal computer session that will be identified as a valid target session for the copy services. You can obtain the session ID through a request to the Query Base Window service if this personal computer session is not one created by the Define Presentation Space service.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Session Information Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the copy management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. The copy services return codes use a function ID of X'64'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	The specified session is not a personal computer session.
X'0C'	Byte 0 of the parameter list was not zero on request.

See Appendix H, "Return Codes," for more information.

Coding Example

```
;
; PARAMETER LIST FOR CONNECT FOR COPY TO PC SESSION
;
CCRETNCD DB 0 ; RETURN CODE
CCFXNID DB 0 ; FUNCTION NUMBER
CCSESSID DB 0 ; SESSION ID
CCRESERV DB 0 ; RESERVED -- MUST BE 0

.
.
.

;
; INITIALIZE PARAMETER LIST FOR CONNECT FOR COPY TO PC SESSION
;
MOV CCRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV CCFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID INTO THE LIST
MOV CCSESSID,AL

;
; INITIALIZE REGISTERS FOR CONNECT FOR COPY TO PC SESSION
;
MOV AH,09H
MOV AL,03H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,COPY ; RESOLVED VALUE FOR 'COPY '
MOV DI, SEG CCRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET CCRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR CONNECT FOR COPY TO PC SESSION SERVICE
;
INT 7AH
.
.
.
```


Disconnect for Copy to PC Session

Copy Service X'04': Disconnect for Copy to PC Session

Use this service to identify a personal computer session as no longer being a valid target session for the copy services.

Register Values

On Request

AH = X'09'
AL = X'04'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for COPY
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'64')
2	1 byte	Session ID	Unchanged
3	1 byte	Reserved — must be X'00'	Reserved

Parameter Definitions

Request Parameters:

- The session ID is the ID of a personal computer session that has been identified as a valid target session for the copy services. You can obtain the session ID through a request to the Query Base Window service if this personal computer session is not one created by the Define Presentation Space service.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Session Information Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the copy management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. The copy services return codes use a function ID of X'64'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	The specified session is not a personal computer session.
X'0C'	Byte 0 of the parameter list was not zero on request.

See Appendix H, "Return Codes," for more information.

Disconnect for Copy to PC Session

Coding Example

```
;
; PARAMETER LIST FOR DISCONNECT FOR COPY TO PC SESSION
;
DCRETNCD DB 0 ; RETURN CODE
DCFXNID DB 0 ; FUNCTION NUMBER
DCSESSID DB 0 ; SESSION ID
DCRESERV DB 0 ; RESERVED -- MUST BE 0
.
.
.
;
; INITIALIZE PARAMETER LIST FOR DISCONNECT FOR COPY TO PC SESSION
;
MOV DCRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV DCFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID INTO THE LIST
MOV DCSESSID,AL
;
; INITIALIZE REGISTERS FOR DISCONNECT FOR COPY TO PC SESSION
;
MOV AH,09H
MOV AL,04H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,COPY ; RESOLVED VALUE FOR 'COPY '
MOV DI,SEG DCRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET DCRETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR DISCONNECT FOR COPY TO PC SESSION
;
INT 7AH
.
.
.
```

Chapter 11. Coding Translate Service Requests

Introduction	11-2
Requesting the Translate Service	11-2
Return Codes for the Translate Service	11-2
Translate Service X'01': Translate Data	11-4

Introduction

This chapter describes how to code requests for the translate service provided by the API.

Data that is displayed in host and notepad presentation spaces is represented by numbers called host/notepad character codes. Data that is displayed in personal computer presentation spaces is represented by ASCII codes. The translate service allows your application program to translate the data in a buffer from one type of data representation to the other.

Notes:

1. *You cannot translate graphic characters or programmed symbol set characters.*
2. *If the input code does not have a matching output code, it is translated to a blank.*

The translate service provided by the API is:

- **Translate Data Service:** Use this service to translate the data in a buffer from ASCII codes to host/notepad character codes, or from host/notepad character codes to ASCII codes.

Requesting the Translate Service

To request the translate service, load the registers and the parameter list with the proper values, and use the INT 7AH instruction to signal the workstation program that it has a request to process.

Note: Before your application can request the translate service, it must request the Name Resolution service, using 'XLATE' as the gate name in the parameter list. (Remember that the gate name must be padded to the right with blanks if it is less than eight characters.)

Return Codes for the Translate Service

The translate service has two return codes associated with it: a system return code and a translate service return code. Both types of return codes are 2-byte values made up of a function ID and an error number. The function ID indicates the portion of the workstation program in which the error occurred.

The error number indicates the specific type of error that has occurred. An error number of X'00' always indicates a successful acceptance or completion of the request.

- **System Return Codes:**

After your application has requested a translate service, the CH and CL registers contain a return code generated by the request processing portion of the workstation program. The function ID is in the CH register, and the error number is in the CL register. System return codes use a function ID of X'12'. The error codes that can appear are:

Code	Meaning
X'00'	Request accepted.
X'05'	Invalid index specified.
X'07'	Invalid reply specified.
X'08'	Invalid wait type specified.
X'0B'	RQE pool depleted.
X'34'	Invalid gate entry.

- **Translate Service Return Codes:**

After a requested translate service is completed, bytes 0 and 1 of the parameter list contain a return code generated by the translation management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. The translate service return codes use a function ID of X'6C'. The error numbers that can appear are included in the description of the service.

See Appendix H, "Return Codes," for more information.

Translate Data

Translate Service X'01': Translate Data

Use this service to translate the data in a buffer from ASCII codes to host/notepad character codes, or from host/notepad character codes to ASCII codes.

Register Values

On Request

AH = X'09'
AL = X'01'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for XLATE
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'6C')
2	1 word	Offset address of source buffer	Unchanged
4	1 word	Segment address of source buffer	Unchanged
6	1 word	Offset address of target buffer	Unchanged
8	1 word	Segment address of target buffer	Unchanged
10	1 byte	Translate type	Unchanged
11	1 byte	Reserved	Reserved
12	1 word	Length	Unchanged

Parameter Definitions

Request Parameters:

- The source buffer contains the codes to be translated.
- The target buffer is where the translated codes are to be stored.

Note: The source and target buffers can be the same address.

- The translate type is specified as follows:

X'01'—ASCII to host/notepad

X'02'—Host/notepad to ASCII

- “Length” is the number of bytes to translate. If the length is 0, no translation occurs, but the request will not fail.

Note: If the input code does not have a matching output code, it will be translated to a blank. An ASCII blank is X'20', and a host and notepad blank is X'10'.

The following table shows the host and notepad character codes and the characters they represent.

	0x	1x	2x	3x	4x	5x	6x	7x	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx
x0	NUL	SP	0	&	à	ä	À	Ä	a	q	A	Q	↖	^	P	☒
x1	EM	=	1	—	è	ë	È	Ë	b	r	B	R	—		S	☒
x2	FF	'	2	.	ì	ï	Ì	Ï	c	s	C	S	z	■	→	↵
x3	NL	”	3	,	ò	ö	Ò	Ö	d	t	D	T	_	°	↑	↵
x4	STP	/	4	:	ù	ü	Ù	Ü	e	u	E	U	◊	°	⤴	Ⓜ
x5	CR	\	5	+	ã	â	Ã	Â	f	v	F	V	◊	—	↓	—
x6			6	¬	õ	ê	Õ	Ê	g	w	G	W	✕	⌈	⌋	—
x7			7	—	ÿ	î	Y	Î	h	x	H	X	■	⌈	⌋	▶
x8	>	?	8	°	à	ô	A	Ô	i	y	I	Y	←	1	µ	¿
x9	<	!	9		è	û	E	Û	j	z	J	Z	⬇	J	²	☒
xA	[\$	β	^	é	á	E	Á	k	ae	K	Æ	◊	⌈	³	□
xB]	¢	§	~	ì	é	I	É	l	ø	L	Ø	◊	⌈	▶	☒
xC)	£	#	..	ò	í	O	Í	m	à	M	À	Ⓐ	⌈	□	Ⓐ
xD	(¥	@	`	ù	ó	U	Ó	n	ç	N	C	Ⓔ	⌈	↔	□
xE	}	pts	%	'	ü	ú	Y	Ú	o	;	O	;	•	=	☒	i
xF	{	✱	—	5	ç	ñ	C	Ñ	p	*	P	*	■		●	Not Supported

Translate Data

Notes:

1. Values X'C0' through X'FF' are used as attributes in CUT and DFT host sessions, and as characters in notepad sessions.
2. Characters X'68' through X'6F' are replaced in the refresh buffer by X'E8', X'69', X'6A', X'F8', X'FE', X'D4', X'CE', and X'D3', respectively. These characters are used only in non-U.S. countries.

The following table shows the hexadecimal ASCII codes found in the personal computer presentation space and the characters they represent.

HEXA DECIMAL VALUE	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	BLANK (NULL)	▶	BLANK (SPACE)	0	@	P	'	p	Ç	É	á				∞	≡
1	☺	◀	!	1	A	Q	a	q	ü	æ	í				β	±
2	☹	↑	"	2	B	R	b	r	é	Æ	ó				Γ	≥
3	♥	!!	#	3	C	S	c	s	â	ô	ú				π	≤
4	♦	¶	\$	4	D	T	d	t	ä	ö	ñ				Σ	∫
5	♣	§	%	5	E	U	e	u	à	ò	Ñ				σ	∫
6	♠	■	&	6	F	V	f	v	å	û	ä				μ	÷
7	•	↓	'	7	G	W	g	w	ç	ù	ö				τ	≈
8	•	↑	(8	H	X	h	x	ê	ÿ	ï				ø	°
9	○	↓)	9	I	Y	i	y	ë	Ö	Γ				θ	•
A	○	→	*	:	J	Z	j	z	è	Ü	Γ				Ω	•
B	♂	←	+	;	K	I	k	{	ï	¢	½				δ	√
C	♀	└	,	<	L	\	l		î	£	¼				∞	n
D	♪	↔	—	=	M	I	m	}	ì	¥	ì				φ	²
E	♪	▲	.	>	N	^	n	~	Ä	R	«				∈	■
F	☼	▼	/	?	O	_	o	Δ	Å	f	»				∩	BLANK FF

For further information regarding the personal computer character set and attributes, refer to the *IBM Personal Computer Technical Reference*.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Translate Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the translate management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Translate services return codes use a function ID of X'6C'. The translate services return codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'01'	Invalid translate type.
X'0C'	Byte 0 in the parameter list was not zero on request.

See Appendix H, "Return Codes," for more information.

Translate Data

Coding Example

```
;
; PARAMETER LIST FOR TRANSLATE DATA
;
TLRETNCD DB 0 ; RETURN CODE
TLFXNID DB 0 ; FUNCTION NUMBER
TLSRCOFF DW 0 ; OFFSET ADDRESS OF SOURCE BUFFER
TLSRCSEG DW 0 ; SEGMENT ADDRESS OF SOURCE BUFFER
TLTRGOFF DW 0 ; OFFSET ADDRESS OF TARGET BUFFER
TLTRGSEG DW 0 ; SEGMENT ADDRESS OF TARGET BUFFER
TLTYPE DB 0 ; TRANSLATE TYPE
TLRESERV DB 0 ; RESERVED
TLLENGTH DW 0 ; LENGTH

.
.
.

;
; INITIALIZE PARAMETER LIST FOR TRANSLATE DATA
;
MOV TLRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV TLFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AX,OFFSET SOURCE ; SOURCE OFFSET INTO THE LIST
MOV TLSRCOFF,AX
MOV AX,SEG SOURCE ; SOURCE SEGMENT INTO THE LIST
MOV TLSRCSEG,AX
MOV AX,OFFSET TARGET ; TARGET OFFSET INTO THE LIST
MOV TLTRGOFF,AX
MOV AX,SEG TARGET ; TARGET SEGMENT INTO THE LIST
MOV TLTRGSEG,AX
MOV TLTYPE,01H ; TRANSLATE ASCII TO EBCDIC
MOV TLLENGTH,80 ; TRANSLATE 80 BYTES

;
; INITIALIZE REGISTERS FOR TRANSLATE DATA
;
MOV AH,09H
MOV AL,01H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,XLATE ; RESOLVED VALUE FOR 'XLATE '
MOV DI, SEG TLRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET TLRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR TRANSLATE DATA SERVICE
;
INT 7AH
.
.
.
```

Chapter 12. Coding Operator Information Area Service Requests

Introduction	12-2
Requesting the Operator Information Area Services	12-3
Return Codes for the Operator Information Area Services	12-3
Operator Information Area Service X'01': Read Operator Information Area Image	12-4
Operator Information Area Service X'02': Read Operator Information Area Group	12-7

Introduction

This chapter describes how to code requests for the operator information area (OIA) services provided by the API.

The OIA services allow your application program to determine the current status of a session as shown in the OIA.

The OIA services provided by the API are:

- **Read Operator Information Area Image Service:** Use this service to obtain an image of the OIA for the specified session.
- **Read Operator Information Area Group Service:** Use this service to obtain a bit string that indicates the current settings of a group of indicators in the OIA for the specified session.

The Read Operator Area Image service reads an image of the OIA into a buffer that you supply. In order for the image to be useful to you, you must be aware of the possible OIA indicators that can be displayed on the system you are connected to. To determine the presence of a particular OIA indicator, you must scan the buffer for the hexadecimal character that represents that character. The OIA image in the buffer is represented by host/notepad character codes.

The Read Operator Area Group service returns a bit mask that indicates the state of all the indicators in a specified group of OIA indicators. The states of each group are ordered so that the high-order bits represent the indicators of higher priority. Therefore, if more than one state is active within a group, the state with the highest priority is the active state within that group.

Notes:

1. *For CUT host sessions, the information in the OIA may not be updated to reflect the status of that session if the session has lost communication with the controller or the host.*
2. *For non-3270 PC hardware, no OIA is displayed for PC sessions. In addition, there is no host separation line in the OIA.*

For more information on the host OIA, refer to the *3278 Display Station Operator's Guide*.

For further information on the OIA, refer to the *IBM 3270 Workstation Program User's Guide and Reference*.

Requesting the Operator Information Area Services

To request the operator information area services, load the registers and the parameter list with the proper values, and use the INT 7AH instruction to signal the workstation program that it has a request to process.

Note: Before your application can request the operator information area services, it must request the Name Resolution service, using 'OIAM' as the gate name in the parameter list. (Remember that the gate name must be padded to the right with blanks if it is less than eight characters.)

Return Codes for the Operator Information Area Services

Each operator information area service has two return codes associated with it, a system return code and an operator information area services return code. Both types of return codes are 2-byte values made up of a function ID and an error number. The function ID indicates the portion of the workstation program in which the error occurred. The error number indicates the specific type of error that has occurred. An error number of X'00' always indicates a successful acceptance or completion of the request.

- **System Return Codes:**

After your application has requested an operator information area service, the CH and CL registers contain a return code generated by the request processing portion of the workstation program. The function ID is in the CH register, and the error number is in the CL register. System return codes use a function ID of X'12'. The error codes that can appear are:

Code	Meaning
X'00'	Request accepted.
X'05'	Invalid index specified.
X'07'	Invalid reply specified.
X'08'	Invalid wait type specified.
X'0B'	RQE pool depleted.
X'0F'	Invalid environment access.
X'34'	Invalid gate entry.

These system return codes apply to all the operator information area services.

- **Operator Information Area Services Return Codes:**

After a requested operator information area service is completed, bytes 0 and 1 of the parameter list contain a return code generated by the OIA management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Operator information area services return codes use a function ID of X'6D'. The error numbers that can appear are specific to the service that was requested and are included in the descriptions of each service.

See Appendix H, "Return Codes," for more information.

Read Operator Information Area Image

Operator Information Area Service X'01': Read Operator Information Area Image

Use this service to obtain an image of the operator information area for the specified host, notepad, or PC session (excluding WSCtrl mode).

Register Values

On Request

AH = X'09'
AL = X'01'
BH = X'80'
BL = X'20'
CX = X'00FF'
DX = Resolved value for OIAM
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'6D')
2	1 byte	Session ID	Unchanged
3	1 byte	Reserved	Reserved
4	1 word	Offset address of OIA buffer	Unchanged
6	1 word	Segment address of OIA buffer	Unchanged

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session being queried for OIA information.
- The OIA buffer is the buffer where the OIA image data will be returned. The OIA buffer must be 160 bytes long. Each character in the OIA is represented in the buffer by two bytes of information. The first byte contains the character in the OIA, and the second byte contains the character attribute for that character. Figures F-3 and F-5 in Appendix F, "Presentation Space Considerations," show the format of the character attributes and the hexadecimal codes for the OIA characters.

OIA Image Overlay

- Below are some useful byte definitions; their byte positions start with offset 0.

Byte	Meaning
36	Window short name.
38	Screen profile number.
104	Autokey play/record status.
106	Autokey abort/pause status.
110	Enlarge state.
118	Printer status.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Operator Information Area Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the OIA management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. The OIA return codes use a function ID of X'6D'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'0C'	Byte 0 of the parameter list was not zero on request.

See Appendix H, "Return Codes," for more information.

Usage Notes

- In order for the OIA image to be useful, you must be aware of the possible OIA indicators that can be displayed on the system you are connected to.
- To determine the presence of a particular OIA indicator, you must scan the buffer for the hexadecimal character that represents that character. The OIA image in the buffer is represented by host/notepad character codes.

Read Operator Information Area Image

- When this service is requested for a CUT host session, the following OIA indicators are not returned:
 - Autokey input inhibited (found in Group 8)
 - Autokey states (found in Groups 16 and 17)
 - Enlarge state (found in Group 18)

To obtain these indicators for a CUT host session, use the Read Operator Information Area Group service.

- For CUT host sessions, the information in the OIA may not be updated to reflect the status of that session if the session has lost communication with the controller or the host.

Coding Example

```
;
; PARAMETER LIST FOR READ OPERATOR INFORMATION AREA IMAGE
;
OIRETNCD DB 0 ; RETURN CODE
OIFXNID DB 0 ; FUNCTION ID
OISESSID DB 0 ; SESSION ID
OIRESVD DB 0 ; RESERVED
OIAOFFS DW 0 ; OFFSET ADDRESS OF OIA BUFFER
OIASEGM DW 0 ; SEGMENT ADDRESS OF OIA BUFFER
.
.
.
;
; INITIALIZE PARAMETER LIST FOR READ OPERATOR INFORMATION AREA IMAGE
;
MOV OIRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV OIFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SSID ; SESSION ID OBTAINED FROM REQUEST
MOV OISESSID,AL ; TO QUERY SESSION ID SERVICE
MOV AX,OFFSET OIABUFF ; OFFSET OF THE OIA BUFFER
MOV OIAOFFS,AX ; IN PARAMETER LIST
MOV AX,SEG OIABUFF ; SEGMENT ADDRESS OF OIA BUFFER
MOV OIASEGM,AX ; IN PARAMETER LIST
;
; INITIALIZE REGISTERS FOR READ OPERATOR INFORMATION AREA IMAGE
;
MOV AH,09H
MOV AL,01H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,OIAM ; NAME RESOLUTION FOR OIAM
MOV DI,SEG OIRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET OIRETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR READ OPERATOR INFORMATION AREA IMAGE
SERVICE
;
INT 7AH
.
.
.
```

Operator Information Area Service X'02': Read Operator Information Area Group

Use this service to obtain a bit string that indicates the current settings of a group of indicators in the operator information area for the specified session.

Register Values

On Request

AH = X'09'
AL = X'02'
BH = X'80'
BL = X'20'
XX = X'00FF'
DX = Resolved value for OIAM
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'6D')
2	1 byte	Session ID	Unchanged
3	1 byte	Reserved	Reserved
4	1 word	Offset address of OIA buffer	Unchanged
6	1 word	Segment address of OIA buffer	Unchanged
8	1 byte	OIA group number	Unchanged

Read Operator Information Area Group

Parameter Definitions

Request Parameters:

- The session ID is the ID of the session being queried for OIA information.
- The OIA buffer is the buffer where the OIA group data will be returned. The buffer sizes for each group are as follows:

Groups 1 through 7: 1 byte

Group 8: 5 bytes

Groups 9 through 18: 1 byte

All groups (group number X'FF'): 22 bytes

- The group number is the number of the requested OIA group. To obtain indicators for all OIA groups, specify group number X'FF'.

OIA Group Indicator Meanings

The states of each group are ordered so that the high-order bits represent the indicators of higher priority. Therefore, if more than one state is active within a group, the state with the highest priority is the active state within that group.

The meanings of the bits in each OIA group are as follows:

- Group 1: Online and screen ownership

Bit	Meaning
0	Setup mode
1	Test mode
2	SSCP-LU session owns screen
3	LU-LU session owns screen
4	Online and not owned
5	Subsystem ready
6, 7	Reserved

- Group 2: Character selection

Bit	Meaning
0	Extended select
1	APL
2	Kana
3	Alpha
4	Text
5 - 7	Reserved

- Group 3: Shift state

Bit	Meaning
0	Upper shift
1	Numeric
2 – 7	Reserved

- Group 4: PSS group 1

Bit	Meaning
0	Operator-selectable
1	Field inherit
2 – 7	Reserved

- Group 5: Highlight group 1

Bit	Meaning
0	Operator-selectable
1	Field inherit
2 – 7	Reserved

- Group 6: Color group 1

Bit	Meaning
0	Operator-selectable
1	Field inherit
2 – 7	Reserved

- Group 7: Insert

Bit	Meaning
0	Insert mode
1 – 7	Reserved

- Group 8: Input inhibited (5 bytes)

Bit	Meaning
0	Nonresettable machine check
1	Reserved for security key
2	Machine check
3	Communication check
4	Program check
5	Retry
6	Device not working
7	Device very busy

Read Operator Information Area Group

Bit	Meaning
0	Device busy
1	Terminal wait
2	Minus symbol
3	Minus function
4	Too much entered
5	Not enough entered
6	Wrong number
7	Numeric field

Bit	Meaning
0	Reserved
1	Operator unauthorized
2	Operator unauthorized, minus function
3	Invalid dead key combination
4	Wrong place
5 – 7	Reserved

Bit	Meaning
0	Message pending
1	Partition wait
2	System wait
3	Hardware mismatch
4	Logical terminal not configured at control unit
5 – 7	Reserved

Bit	Meaning
0	Autokey inhibit
1	Application program has operator input inhibited
2 – 7	Reserved

- Group 9: PSS group 2

Bit	Meaning
0	PS selected
1	PC display disable
2 – 7	Reserved

- Group 10: Highlight group 2

Bit	Meaning
0	Selected
1 – 7	Reserved

- Group 11: Color group 2

Bit	Meaning
0	Selected
1 – 7	Reserved

Read Operator Information Area Group

- Group 12: Communication error reminder

Bit	Meaning
0	Communication error
1	Response time monitor
2 – 7	Reserved

- Group 13: Printer status

Bit	Meaning
0	Print code not customized
1	Printer malfunction
2	Printer printing
3	Assign printer
4	What printer
5	Printer assignment
6 – 7	Reserved

- Group 14: Reserved group
- Group 15: Reserved group
- Group 16: Autokey play/record status

Bit	Meaning
0:	Play
1:	Record
2 – 7	Reserved

- Group 17: Autokey abort/pause state

Bit	Meaning
0	Recording overflow
1	Pause
2 – 7	Reserved

- Group 18: Enlarge state

Bit	Meaning
0	Window is enlarged
1 – 7	Reserved

Read Operator Information Area Group

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Operator Information Area Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the OIA management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. The OIA return codes use a function ID of X'6D'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'02'	Invalid session ID.
X'0C'	Byte 0 of the parameter list was not zero on request.

See Appendix H, "Return Codes," for more information.

Usage Notes

- For CUT host sessions, the information in the OIA may not be updated to reflect the status of that session if the session has lost communication with the controller or the host.

Read Operator Information Area Group

Coding Example

```
;
; PARAMETER LIST FOR READ OPERATOR INFORMATION AREA GROUP
;
OGRETNCD DB 0 ; RETURN CODE
OGFXNID DB 0 ; FUNCTION ID
OGSESSID DB 0 ; SESSION ID
OGRESVD DB 0 ; RESERVED
OGBUFOFF DW 0 ; OFFSET ADDRESS OF OIA BUFFER
OGBUFSEG DW 0 ; SEGMENT ADDRESS OF OIA BUFFER
OGGRPNUM DB 0 ; OIA GROUP NUMBER

.
.
.

;
; INITIALIZE PARAMETER LIST FOR READ OPERATOR INFORMATION AREA GROUP
;
MOV OGRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV OGFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID OBTAINED FROM REQUEST
MOV OGSESSID,AL ; TO QUERY SESSION ID SERVICE
MOV AX,OFFSET OIABUFF ; OFFSET OF THE OIA BUFFER
MOV OGBUFOFF,AX ; IN PARAMETER LIST
MOV AX,SEG OIABUFF ; SEGMENT ADDRESS OF OIA BUFFER
MOV OGBUFSEG,AX ; IN PARAMETER LIST
MOV AL,5 ; OIA GROUP NUMBER
MOV OGGRPNUM,AL ; IN PARAMETER LIST

;
; INITIALIZE REGISTERS FOR READ OPERATOR INFORMATION AREA GROUP
;
MOV AH,09H
MOV AL,02H
MOV BH,80H
MOV BL,20H
MOV CX,OFFH
MOV DX,OIAM ; NAME RESOLUTION FOR OIAM
MOV DI,SEG OGRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET OGRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR READ OPERATOR INFORMATION AREA GROUP
SERVICE
;
INT 7AH
.
.
.
```


Chapter 13. Coding Multi-DOS Support Service Requests

Introduction	13-2
Requesting the Multi-DOS Support Services	13-2
Return Codes for the Multi-DOS Support Services	13-3
Multi-DOS Support Service: Query Environment Size	13-4
Multi-DOS Support Service: Asynchronous DOS Function Requests	13-7
Multi-DOS Support Service X'01': Get Storage	13-12
Multi-DOS Support Service X'02': Free Storage	13-15
Multi-DOS Support Service X'03': Set Storage Allocation	13-18

Introduction

This chapter describes how to code requests for the Multi-DOS support services provided by the API.

The Multi-DOS support services allow your application program to query the size in paragraphs of a specified environment, and to request DOS INT 21H function calls asynchronously.

The Multi-DOS support services provided by the API are:

- **Query Environment Size:** Use this service to obtain the size in paragraphs of the specified environment, the address of the first paragraph of the environment, and a flag describing the state of the environment.
- **Asynchronous DOS Function Request:** Use this service to issue DOS function requests asynchronously.
- **Get Storage:** Use this service to allocate storage to your application program. This service performs the same function as the DOS INT 21H type 48 function call.
- **Free Storage:** Use this service to free storage from your application program. This service performs the same function as the DOS INT 21H type 49 function call.
- **Set Storage Allocation:** Use this service to modify the number of blocks of storage allocated to your application program. This service performs the same function as the DOS INT 21H type 4A function call.

Requesting the Multi-DOS Support Services

To request the Multi-DOS support services, load the registers and the parameter list with the proper values, and use the INT 7AH instruction to signal the workstation program that it has a request to process.

Note: Before your application can request the Multi-DOS support services, it must request the Name Resolution service, using 'INDJQRY ', 'INDJASY ', or 'MEMORY ' as the name in the parameter list. (Remember that the name must be padded to the right with blanks if it is less than eight characters.)

Return Codes for the Multi-DOS Support Services

Each Multi-DOS support service has two return codes associated with it, a system return code and a Multi-DOS support services return code. Both types of return codes are 2-byte values made up of a function ID and an error number. The function ID indicates the portion of the workstation program in which the error occurred. The error number indicates the specific type of error that has occurred. An error number of X'00' always indicates a successful acceptance or completion of the request.

- **System Return Codes:**

After your application has requested a Multi-DOS support service, the CH and CL registers contain a return code generated by the request processing portion of the workstation program. The function ID is in the CH register, and the error number is in the CL register. System return codes use a function ID of X'12'. The error codes that can appear are:

Code	Meaning
X'00'	Request accepted.
X'05'	Invalid index specified.
X'07'	Invalid reply specified.
X'08'	Invalid wait type specified.
X'0B'	RQE pool depleted.
X'0F'	Invalid environment access.
X'34'	Invalid gate entry.

These system return codes apply to all the Multi-DOS support services.

- **Multi-DOS Support Services Return Codes:**

After a requested Multi-DOS support service is completed, bytes 0 and 1 of the parameter list contain a return code generated by the Multi-DOS support management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. Multi-DOS support service return codes use function IDs X'22' and X'23'. The error numbers that can appear are specific to the service that was requested and are included in the descriptions of each service.

See Appendix H, "Return Codes," for more information.

Multi-DOS Support Service: Query Environment Size

Use this service to obtain the size in paragraphs of the specified environment, the address of the first paragraph of the environment, and a flag describing the state of the environment.

Register Values

On Request

AH = X'09'
BH = X'80'
BL = X'20'
CX = X'FF'
DX = Resolved value for INDJQRY
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12' or X'13'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

- Request Register Values:

The DX register contains the resolved value of the component name INDJQRY. Your application must request the Name Resolution service using INDJQRY as the name in the parameter list to obtain this resolved value.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Not used	Function ID (X'22')
2	1 byte	Environment ID	Unchanged
3	1 byte	Not used	Environment flag
4	1 word	Not used	Environment size
6	1 word	Not used	Environment address

Parameter Definitions

Request Parameters:

- The environment ID is the ID of the environment being queried.

Completion Parameters:

- The bits in the environment flag are as follows:

0	1, 2	3	4, 5	6	7
Stopping/ running	Reserved	DOS	Reserved	Base appl	Keybd wait

- Bit 0 set to 1 indicates that the environment is stopping, doing an IPL, or involved in an INDSPLIT or INDMERGE operation. Bit 0 set to 0 indicates that the environment is in a normal running state.
 - Bits 1 and 2 are reserved.
 - Bit 3 set to 1 indicates that COMMAND.COM is the base application.
 - Bits 4 and 5 are reserved.
 - Bit 6 set to 1 indicates that the base-level program (the application specified at customization time) is running.
 - Bit 7 set to 1 indicates that the program is in a keyboard wait.
- The environment size is the number of paragraphs in the environment.
 - The environment address is the segment address of the beginning of the environment.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Multi-DOS Support Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the Multi-DOS management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. The Multi-DOS support services return codes use a function ID of X'22'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'E6'	Byte 0 of the parameter list was not zero on request.
X'E7'	Invalid environment ID.

See Appendix H, "Return Codes," for more information.

Query Environment Size

Coding Example

```
;
; PARAMETER LIST FOR QUERY ENVIRONMENT SIZE
;
QDRETNCD DB 0 ; RETURN CODE
QDFXNID DB 0 ; FUNCTION ID
QDENVID DB 0 ; ENVIRONMENT ID
QDENVFLG DB 0 ; ENVIRONMENT FLAG
QDENVSIZ DW 0 ; ENVIRONMENT SIZE
QDENVADD DW 0 ; ENVIRONMENT ADDRESS
.
.
.

;
; INITIALIZE PARAMETER LIST FOR QUERY ENVIRONMENT SIZE
;
MOV QDRETNCD,00H ; QDRETNCD MUST BE 0 BEFORE REQUEST
MOV QDFXNID,00H ; QDFXNID MUST BE 0 BEFORE REQUEST
MOV AL,ENVID ; ENVIRONMENT ID
MOV QDENVID,AL ; IN LIST

;
; INITIALIZE REGISTERS FOR QUERY ENVIRONMENT SIZE
;
MOV AH,09H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,INDJQRY ; RESOLVED VALUE FOR INDJQRY
MOV DI, SEG QDRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET QDRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR QUERY ENVIRONMENT SIZE SERVICE
;
INT 7AH
.
.
.
```

Multi-DOS Support Service: Asynchronous DOS Function Requests

Use this service to issue DOS function requests.

The Multi-DOS portion of the workstation program provides a special task for personal computer applications to issue DOS interrupt 21H function calls asynchronously. For example, an application program could use this service to perform a DOS function call for file I/O, and continue processing without having to wait for the I/O to be completed.

On request, the parameter list for the Asynchronous DOS Function Requests service contains the register values needed for the DOS function being requested. On completion, the parameter list contains the register values that were set by the DOS function.

At least three separate requests have to be issued to use this function: a connect for asynchronous DOS function requests, one or more asynchronous DOS function requests, and a disconnect for asynchronous DOS function requests.

Note: DOS function calls 0H, 31H, 4B00H, 4CH, and 4DH cannot be requested using this service. Requests for these DOS function calls will fail with return code X'11', invalid function. This service should not be used for display or keyboard I/O, because the results will be unpredictable.

Register Values

On Request

AH = X'09'
BH = Synchronous or asynchronous *
BL = Synchronous or asynchronous *
CX = X'FF'
DX = Resolved value for INDJASY
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

AX = Return ID
BL = Return type
CH = X'12' or X'13'
CL = Return code

The contents of registers BH, DX, ES, and DI are unpredictable.

* The values in these registers depend on whether you want the request to be processed synchronously or asynchronously. See the description of request register values for more information.

• Request Register Values:

The DX register contains the resolved value of the name INDJASY. Your application must request the Name Resolution service using INDJASY as the name in the parameter list to obtain this resolved value.

Asynchronous DOS Function Requests

You can specify synchronous or asynchronous processing of the Asynchronous DOS Function Requests service. In synchronous processing, control is returned to your application program after the workstation program has completed the request. In asynchronous processing, control is returned to your application program before the workstation program has completed the request. You must use the Get Request Completion service to obtain the parameter list values on completion when you request asynchronous processing.

Synchronous Processing:

There are two ways to specify synchronous processing:

1. Set the BH register to X'80' and the BL register to X'20'. When the request is completed, control is returned to your application program and the registers and parameter list contain the values for completion of the request.
2. Set both the BH and BL registers to X'40'. When the request is completed, control is returned to your program, but the parameter list values for completion of the request are not obtained until you request the Get Request Completion service.

Asynchronous Processing:

For asynchronous processing of the Asynchronous DOS Function Requests service request, set the BH register to X'40' and the BL register to X'00'. When asynchronous processing is specified, you must request the Get Request Completion service to obtain the results of the Asynchronous DOS Function Requests service.

- Completion Register Values:

If you specified asynchronous processing, or synchronous processing using X'40' in both the BH and BL registers on request, the AX register contains a request ID that the workstation program assigned to the request. You use this request ID to match the results of the service obtained by the Get Request Completion service to the results of this service. That is, when the request ID in the AX register, on completion of the Get Request Completion service, matches the request ID in the AX register on completion of this service, the results obtained by the Get Request Completion service pertain to this request.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Not used	Function ID (X'23')
2	1 byte	Request type	Unchanged
3	1 byte	Reserved	Reserved
4	1 word	AX register value	Value determined by DOS function
6	1 word	BX register value	Value determined by DOS function
8	1 word	CX register value	Value determined by DOS function
10	1 word	DX register value	Value determined by DOS function
12	1 word	DS register value	Value determined by DOS function
14	1 word	ES register value	Value determined by DOS function
16	1 word	SI register value	Value determined by DOS function
18	1 word	DI register value	Value determined by DOS function
20	1 word	Flags register value	Value determined by DOS function

Parameter Definitions

Request Parameters:

- The request type can be one of the following:

X'00' – to connect for asynchronous DOS function requests

X'01' – to request a DOS function

X'02' – to disconnect for asynchronous DOS function requests

You must use request type X'00' to connect for asynchronous DOS function requests before you can use request type X'01' to request a DOS function. When your application program has completed all its DOS function requests, it must use request type X'02' to disconnect for asynchronous DOS function requests.

- The remainder of the parameter list must contain the values of registers AX, BX, CX, DX, DS, ES, SI, and DI and the flags register. The values in these registers should correspond to the values needed by the DOS function you are requesting. The registers that are not used by the DOS functions do not need to be set or initialized to any value.

Asynchronous DOS Function Requests

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Multi-DOS Support Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the Multi-DOS management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. The Multi-DOS support services return codes use a function ID of X'23'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion
X'01' through X'53'	DOS interrupt 21H errors
X'FD'	Not connected for asynchronous DOS requests

See Appendix H, "Return Codes," for more information.

Usage Notes

- If you specified asynchronous processing, or synchronous processing using X'40' in both the BH and BL registers on request, you must use the Get Request Completion service to obtain the results in the parameter list when the Asynchronous DOS Function Requests service is completed.
- If your system extension is going to use DOS function calls, it is recommended that you use the Multi-DOS support services, because those services do not use the interrupt vectors to issue the request, which allows ill-behaved application programs to run simultaneously.

Coding Example

```
;
; PARAMETER LIST FOR ASYNCHRONOUS DOS FUNCTION REQUESTS
;
ARRETNCD DB 0 ; RETURN CODE
ARFXNID DB 0 ; FUNCTION NUMBER
ARTYPE DB 0 ; ENVIRONMENT ID
ARRESRVD DB 0 ; RESERVED
AR$AX DW 0 ; AX REGISTER
AR$BX DW 0 ; BX REGISTER
AR$CX DW 0 ; CX REGISTER
AR$DX DW 0 ; DX REGISTER
AR$DS DW 0 ; DS REGISTER
AR$ES DW 0 ; ES REGISTER
AR$SI DW 0 ; SI REGISTER
AR$DI DW 0 ; DI REGISTER
ARFLAGS DW 0 ; FLAGS
.
.
.
;
; INITIALIZE PARAMETER LIST FOR ASYNCHRONOUS DOS FUNCTION REQUESTS
;
MOV ARRETNCD,00H ; SMRETNCD MUST BE 0 BEFORE REQUEST
MOV ARFXNID,00H ; SMFXNID MUST BE 0 BEFORE REQUEST
MOV AL,1 ; REQUEST TYPE IN THE LIST
MOV ARTYPE,AL
PUSHF ; PUT THE CURRENT FLAGS INTO THE LIST
POP ARFLAGS

MOV AR$AX,0900H ; PRINT A STRING
MOV AR$DX,OFFSET STRING ; PUT THE OFFSET AND SEGMENT OF THE STRING
MOV AR$DS,SEG STRING ; INTO THE LIST
;
; INITIALIZE REGISTERS FOR ASYNCHRONOUS DOS FUNCTION REQUESTS
;
MOV AH,09H
MOV BH,80H ; REPLY TYPE IN BH
MOV BL,20H ; WAIT TYPE IN BL
MOV CX,0FFH ; PRIORITY IN CX
MOV DX,INDJASY ; RESOLVED VALUE FOR 'INDJASY '
MOV DI,SEG ARRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET ARRETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR ASYNCHRONOUS DOS FUNCTION REQUESTS
SERVICE
;
INT 7AH
.
.
.
```

Multi-DOS Support Service X'01': Get Storage

Use this service to allocate storage to your application program. This service performs the same function as the DOS INT 21H type 48 function call.

Register Values

On Request

AH = X'09'
AL = X'01'
BH = X'80'
BL = X'20'
CX = X'FF'
DX = Resolved value for MEMORY
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12' or X'13'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

- Request Register Values:

The DX register contains the resolved value of the gate name MEMORY. Your application must request the Name Resolution service using MEMORY as the gate name in the parameter list to obtain this resolved value.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Not used	Function ID (X'23')
2	1 byte	Environment ID	Unchanged
3	1 byte	Must be zero	Reserved
4	1 word	Paragraphs	Unchanged or paragraphs free
6	1 word	Not used	Segment

Parameter Definitions

Request Parameters:

- The environment ID is the ID of the environment to perform the request in.
- “Paragraphs” is the number of 16-byte paragraphs of storage to allocate.

Completion Parameters:

- “Paragraphs free” is the number of 16-byte paragraphs that are free, if insufficient storage was available for the request. Otherwise, this value is unchanged from its request value.
- “Segment” is the segment address of the start of the allocated block of storage.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Multi-DOS Support Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the Multi-DOS management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. The Multi-DOS support services return codes use a function ID of X'23'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'07'	Storage control blocks destroyed.
X'08'	Insufficient storage.
X'E7'	Invalid environment ID.

See Appendix H, “Return Codes,” for more information.

Usage Notes

- An application running in a stoppable environment can only get storage from its own environment.

Get Storage

Coding Example

```
;
; PARAMETER LIST FOR GET STORAGE
;
GMRETNCD DB 0 ; RETURN CODE
GMFXNID DB 0 ; FUNCTION NUMBER
GMENVID DB 0 ; ENVIRONMENT ID
GMRESRVD DB 0 ; RESERVED
GMPARAGN DW 0 ; NUMBER OF PARAGRAPHS
GMSEGMNE DW 0 ; SEGMENT ADDRESS OF MEMORY
.
.
.
;
; INITIALIZE PARAMETER LIST FOR GET STORAGE
;
MOV GMRETNCD,00H ; GMRETNCD MUST BE 0 BEFORE REQUEST
MOV GMFXNID,00H ; GMFXNID MUST BE 0 BEFORE REQUEST
MOV AL,ENVID ; ENVIRONMENT ID
MOV GMENVID,AL ; IN LIST
MOV AX,NUMPARAG ; NUMBER OF PARAGRAPHS TO ALLOCATE
MOV GMPARAGN,AX ; IN LIST
;
; INITIALIZE REGISTERS FOR GET STORAGE
;
MOV AH,09H
MOV AL,01H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,MEMORY ; RESOLVED VALUE FOR 'MEMORY '
MOV DI,SEG GMRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET GMRETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR GET STORAGE SERVICE
;
INT 7AH
.
.
.
```

Multi-DOS Support Service X'02': Free Storage

Use this service to free storage from your application program. This service performs the same function as the DOS INT 21H type 49 function call.

Register Values

On Request

AH = X'09'
AL = X'02'
BH = X'80'
BL = X'20'
CX = X'FF'
DX = Resolved value for MEMORY
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12' or X'13'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

- Request Register Values:

The DX register contains the resolved value of the gate name MEMORY. Your application must request the Name Resolution service using MEMORY as the gate name in the parameter list to obtain this resolved value.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Not used	Function ID (X'23')
2	1 byte	Environment ID	Unchanged
3	1 byte	Must be zero	Reserved
4	1 word	Not used	Not used
6	1 word	Segment	Unchanged

Parameter Definitions

Request Parameters:

- The environment ID is the ID of the environment to perform the request in.
- "Segment" is the segment address of the start of the block of storage to free.

Free Storage

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Multi-DOS Support Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the Multi-DOS management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. The Multi-DOS support services return codes use a function ID of X'23'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'07'	Storage control blocks destroyed.
X'09'	Invalid storage block address.
X'E7'	Invalid environment ID.

See Appendix H, "Return Codes," for more information.

Usage Notes

- An application running in a stoppable environment can only free storage from its own environment.

Coding Example

```

;
;  PARAMETER LIST FOR FREE MEMORY
;
FMRETNCD  DB  0                ; RETURN CODE
FMFXNID   DB  0                ; FUNCTION NUMBER
FMENVID   DB  0                ; ENVIRONMENT ID
FMRESRVD  DB  0                ; RESERVED
FMPARAGN  DW  0                ; NUMBER OF PARAGRAPHS
FMSEGMNE  DW  0                ; SEGMENT ADDRESS OF MEMORY
.
.
.

;
;  INITIALIZE PARAMETER LIST FOR FREE MEMORY
;
      MOV     FMRETNCD,00H      ; FMRETNCD MUST BE 0 BEFORE REQUEST
      MOV     FMFXNID,00H      ; FMFXNID MUST BE 0 BEFORE REQUEST
      MOV     AL,ENVID         ; ENVIRONMENT ID
      MOV     FMENVID,AL       ;   IN LIST
      MOV     AX,SEGADDR       ; SEGMENT ADDRESS OF BLOCK
      MOV     FMSEGMNE,AX      ;   TO FREE

;
;  INITIALIZE REGISTERS FOR FREE MEMORY
;
      MOV     AH,09H
      MOV     AL,02H
      MOV     BH,80H
      MOV     BL,20H
      MOV     CX,0FFH
      MOV     DX,MEMORY        ; RESOLVED VALUE FOR 'MEMORY '
      MOV     DI, SEG FMRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
      MOV     ES,DI            ;   IN ES
      MOV     DI,OFFSET FMRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
;  SIGNAL WORKSTATION PROGRAM FOR FREE MEMORY SERVICE
;
      INT     7AH
.
.
.

```

Multi-DOS Support Service X'03': Set Storage Allocation

Use this service to modify the number of blocks of storage allocated to your application program. This service performs the same function as the DOS INT 21H type 4A function call.

Register Values

On Request

AH = X'09'
AL = X'03'
BH = X'80'
BL = X'20'
CX = X'FF'
DX = Resolved value for MEMORY
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'12' or X'13'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

- Request Register Values:

The DX register contains the resolved value of the gate name MEMORY. Your application must request the Name Resolution service using MEMORY as the gate name in the parameter list to obtain this resolved value.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Not used	Function ID (X'23')
2	1 byte	Environment ID	Unchanged
3	1 byte	Must be zero	Reserved
4	1 word	Paragraphs	Unchanged or paragraphs free
6	1 word	Segment	Unchanged

Parameter Definitions

Request Parameters:

- The environment ID is the ID of the environment to perform the request in.
- “Paragraphs” is the number of 16-byte paragraphs to set the block size to.
- “Segment” is the segment address of the block of storage to be set.

Completion Parameters:

- “Paragraphs free” is the maximum number of 16-byte paragraphs that the block can grow to, if insufficient storage was available for the request. Otherwise, this value is unchanged from its request value.

Return Codes

- System Return Codes:

Refer to the chapter introduction for a description of the system return codes found in the CH and CL registers.

- Multi-DOS Support Services Return Codes:

Bytes 0 and 1 of the parameter list contain a return code generated by the Multi-DOS management portion of the workstation program. The function ID is in byte 1, and the error number is in byte 0. The Multi-DOS support services return codes use a function ID of X'23'. The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion.
X'07'	Storage control blocks destroyed.
X'08'	Insufficient storage.
X'09'	Invalid storage block address.
X'E7'	Invalid environment ID.

See Appendix H, “Return Codes,” for more information.

Usage Notes

- An application running in a stoppable environment can only set storage from its own environment.

Set Storage Allocation

Coding Example

```
;
; PARAMETER LIST FOR SET MEMORY
;
SMRETNCD DB 0 ; RETURN CODE
SMFXNID DB 0 ; FUNCTION NUMBER
SMENVID DB 0 ; ENVIRONMENT ID
SMRESRVD DB 0 ; RESERVED
SMPARAGN DW 0 ; NUMBER OF PARAGRAPHS
SMSEGMNE DW 0 ; SEGMENT ADDRESS OF MEMORY
.
.
.

;
; INITIALIZE PARAMETER LIST FOR SET MEMORY
;
MOV SMRETNCD,00H ; SMRETNCD MUST BE 0 BEFORE REQUEST
MOV SMFXNID,00H ; SMFXNID MUST BE 0 BEFORE REQUEST
MOV AL,ENVID ; ENVIRONMENT ID
MOV SMENVID,AL ; IN LIST
MOV AX,NUMPARAG ; PARAGRAPH NUMBER
MOV SMPARAGN,AX ; IN LIST
MOV AX,SEGADDR ; SEGMENT ADDRESS
MOV SMSEGMNE,AX ; IN LIST

;
; INITIALIZE REGISTERS FOR SET MEMORY
;
MOV AH,09H
MOV AL,03H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,MEMORY ; RESOLVED VALUE FOR 'MEMORY '
MOV DI,SEG SMRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET SMRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR SET MEMORY SERVICE
;
INT 7AH
.
.
.
```

Part 3. Supervisor Services

This part contains information about the supervisor services of the application programming interface.

- Chapter 14, “Supervisor Services,” describes the types of supervisor services provided by the workstation program that your application program can use.
- Chapter 15, “Coding Supervisory Object Services,” describes the supervisory object services that your application program can use.
- Chapter 16, “Coding Request Services,” describes the task/component request services that your application program can use.
- Chapter 17, “Coding Task State Modifier Services,” describes the task state modifier services that your application program can use.
- Chapter 18, “Coding Semaphore Management Services,” describes the semaphore management request services that your application program can use.
- Chapter 19, “Coding Logical Timer Management Services,” describes the logical timer management request services that your application program can use.
- Chapter 20, “Coding Fixed-Length Queue Management Services,” describes the fixed-length queue management request services that your application program can use.
- Chapter 21, “Coding Interrupt Handler Management Services,” describes the interrupt handler management services that your application program can use.

-
- Chapter 22, “Environments and the Environment Manager,” describes environments, environment access restrictions, and resource managers.
 - Chapter 23, “Coding Environment Manager Services,” describes the environment manager services that your application program can use.
 - Chapter 24, “Coding System Extensions,” describes how to code and use system extensions as part of the control program.

Conventions Used in the API Service Descriptions

The following conventions are used in the descriptions of the API services:

- Hexadecimal numbers are represented in the notation X'nn' for byte values and X'nnnn' for word values.
- Offsets into data structures used by the API services are given as decimal numbers.
- Bits within a byte are numbered with the high-order (leftmost) bit as bit 0 and the low-order (rightmost) bit as bit 7, as follows:

0	1	2	3	4	5	6	7

This order of bit numbering follows the IBM 360/370 convention and is the reverse of the Intel 8088 bit-numbering convention.

Chapter 14. Supervisor Services

Introduction	14-2
Supervisory Object Creation and Deletion	14-2
Tasks	14-2
Components	14-3
Semaphores	14-4
Fixed-Length Queues	14-4
Gates	14-5
User Exit Tables	14-5
The Supervisor Call Instruction (SVC) Table	14-5
Creating Objects with Names	14-5
Supervisory Object Services Your Application Program Can Use ..	14-6
Task Requests	14-6
Use of Wait States	14-7
Sending a Request to Another Task	14-8
Receiving a Request from Another Task	14-8
Replying to a Request from Another Task	14-9
Obtaining Request Completion from Another Task	14-9
Task Request Services Your Application Program Can Use	14-9
Task State Modifiers	14-10
Dispatch Cycles	14-10
Task Dispatching Procedure	14-10
Task Dispatch Activity	14-11
Task Dispatcher States	14-11
Task State Modifier Services Your Application Program Can Use ..	14-12
Semaphore Management	14-13
Considerations for Using Code Serialization Semaphores	14-13
Restrictions on the Use of Semaphores	14-13
Semaphore Management Services Your Application Program Can	
Use	14-14
Logical Timer Management	14-14
Logical Timer Management Services Your Application Program	
Can Use	14-15
Fixed-Length Queue Management	14-15
Fixed-Length Queue Management Services Your Application	
Program Can Use	14-15
Interrupt Handler Management	14-15
Hardware Interrupt Handlers	14-16
Using the DOS Function Calls to Take Over Hardware	
Interrupts	14-16
Using the Install a Hardware Interrupt Handler Service to Take	
Over Hardware Interrupts	14-16
Using the Install an Interrupt Handler Service to Take Over	
Hardware Interrupts	14-17
Hardware Interrupt Handler Considerations	14-17
Software Interrupt Handlers	14-17
Using the DOS Function Calls to Take Over Software Interrupts	
Using the Install an Interrupt Handler Service to Take Over	
Software Interrupts	14-18
Local Software Interrupt Handlers	14-18

Global Software Interrupt Handlers	14-18
Software Interrupt Handler Considerations	14-18
Interrupt Handler Management Services Your Application Program Can Use	14-19

Introduction

The supervisor portion of the Workstation Program provides a set of system services that support a multitasking environment. Supervisor services are divided into the following categories:

- Supervisory object creation and deletion
- Task/component requests
- Task state modifiers
- Semaphore management
- Logical timer management
- Fixed-length queue management
- Interrupt handler management.

Each of these categories of supervisor services is discussed below.

Supervisory Object Creation and Deletion

The supervisor manages the following supervisory objects:

- Tasks
- Components
- Semaphores
- Fixed-length queues
- Gates
- User exit tables.

The supervisor manages supervisory objects through entries in the SVC table. A list of the supervisory object services that your application program can request is given at the end of this section.

Tasks

A task is a unit of dispatchable code. If the tasks are scheduled for execution by the dispatcher and are ready to run, they are guaranteed to execute (where components must be invoked). Conceptually, a task is a never-terminating program. Tasks can perform work on behalf of other system objects, including other tasks. They are capable of both synchronous and asynchronous communication.

All tasks run on their own stacks and are serially reusable (that is, not reentrant).

There are 64 priority levels in the system. When you write an application program, it runs at priority 60. Your application, however, can issue the Create Task Entry service to create a task that runs at a priority of your choice between 36 and 64. Your system extension may also create a task to run at any priority between 1 and 64.

For an example of a task or for the Create Task Entry service with coding instructions, see Chapter 15, "Coding Supervisory Object Services."

Components

Components represent shared code that can be dynamically invoked. Since they are shared, they should either be reentrant or use code serialization techniques (for example, semaphores).

Components are invoked through the Make Request system service. When they get control, the registers are set up just as if the component had done a Get Request. Once a component is invoked, it is running under the task thread of the invoking task. Because a component runs under a task thread, it can perform any function a task can perform. However, a component must be very careful in relying on the state of certain task resources, such as the request and completion queues. Since a task can do asynchronous processing, it may have outstanding signals that may arrive after a component has been invoked. Therefore, a component does not necessarily have free use of a task's wait states. In general, a component can freely wait on a fixed-length queue or a semaphore; but any other wait can interfere with a task's asynchronous processing.

If a component uses other services that require use of wait states, they should clearly document what restrictions are imposed on the invocation of the component. For example, if the component uses the Make Request service and waits for a completion signal, it must be clearly documented that no outstanding completion signals can be pending when this component is invoked.

As a general guideline, tasks should use completion signals only for synchronous services (request and wait) and should use completion queue signals for either synchronous or asynchronous requests. This ensures that components can use the completion signal facility to implement their required function.

Note: When a component is invoked through the Make Request service, the requested wait must be a completion signal, and the requested reply must be a completion signal. In implementation, the task is not put to sleep. Instead, it remains dispatchable. However, on each dispatch the component, running under the task thread, is dispatched. The invoking task's code will only be executed after the component returns.

Requests to components are passed by means of a parameter list. Components receive the address of the parameter list in the ES and DI registers and the requester's ID in the DX register.

See Chapter 15, "Coding Supervisory Object Services," for an example of how to create a component.

When a component is done executing, it must do a Return Far.

Semaphores

Semaphores are system objects that allow system designers to guarantee serial access to a resource. When a task owns a semaphore, it is assured that it is the only supervisory object that can access the resource protected by the semaphore.

There are two types of semaphores: resource semaphores and code serialization semaphores. Code serialization semaphores have some restrictions placed on them in respect to environment access. These restrictions are described under "Semaphore Management" in this chapter.

In essence, semaphores represent resource locks. However, this locking is only by agreement. That is, in order for a semaphore to be effective, all tasks must "agree" to claim the semaphore before accessing a desired resource, code, or data. If a task violates this agreement, the integrity of a system design may be in jeopardy.

Access to semaphores is granted on a first-in-first-out (FIFO) basis. Semaphore ownership is granted to tasks. This means that a component is not considered to own a semaphore. If a component claims a semaphore, that semaphore is owned by the task the component is running under.

Fixed-Length Queues

Fixed-length queues are queues that exist in an application program's address space but that are managed by the supervisor. Fixed-length queues are used by the supervisor to report events and pass keystrokes typed on the keyboard to your application program. They are also useful for application programs that must send small amounts of data between them. In a sense, fixed-length queues represent a "pipeline" between two cooperating tasks.

Fixed-length queues are managed on a FIFO basis, use a circular linking structure, and have a fixed size. You determine the structure and length of the elements enqueued and dequeued from a fixed-length queue, on the basis of the needs of your particular application program.

Gates

A gate is a grouping of services provided by a system extension that can be accessed by tasks or components. A gate contains a list of component or task IDs. Each component or task ID is associated with a specified service provided by the gate. The services provided by the gate are requested through the use of the Task/Component Request services, described under “Task Request Services Your Application Program Can Use” on page 14-9.

A gate should have a name assigned to it. Typically, the gate name indicates the type of services provided by the gate.

User Exit Tables

A user exit table is basically a branch table. It is made up of a series of 32-bit values, generally the segment and offset addresses of entry points to routines. Your application program can write routines to run whenever control is passed to their addresses in the user exit table. By giving a user exit table a name that an application program can name-resolve, your system extension can allow an application program to replace these routines with its own. This is useful for functions such as error handling.

The Supervisor Call Instruction (SVC) Table

The supervisor makes an entry in a table called the SVC table for each supervisory object defined in the system. The supervisor uses these entries to manage task and component requests, and to control access to semaphores, fixed-length queues, user exit tables, and gates. The supervisor identifies each supervisory object by a numeric index. This numeric index is referred to as the ID of the object.

Creating Objects with Names

Each entry in the SVC table can have a unique eight-byte name associated with it. If the name is less than eight characters, you should pad it to the right with blanks. All names must be unique, with one exception: an object in a stoppable environment may have a name that is a duplicate of a name in another stoppable environment. The name is useful for objects that are to be known outside their environment. The Name Resolution supervisor service returns the numeric ID of the supervisory object associated with a specified name. The ID Resolution supervisor service returns the name associated with the supervisory object specified by its numeric ID.

Supervisory Object Services Your Application Program Can Use

The supervisory object services provided by the API are:

- **Create Task Entry:** Use this service to create an entry in the SVC table for a task.
- **Create Component Entry:** Use this service to create an entry in the SVC table for a component.
- **Create Semaphore Entry:** Use this service to create an entry in the SVC table for a semaphore.
- **Create Fixed-Length Queue Entry:** Use this service to create an entry in the SVC table for a fixed-length queue.
- **Create Gate Entry:** Use this service to create an entry in the SVC table for a gate.
- **Create User Exit Table Entry:** Use this service to create an entry in the SVC table for a user exit table.
- **Install User Exit Table Entries:** Use this service to install entries in the specified user exit table.
- **Name Resolution:** Use this service to resolve the specified supervisory object name to its numeric index.
- **ID Resolution:** Use this service to resolve the specified supervisory ID name to its alphanumeric name.
- **Delete Entry:** Use this service to delete an entry in the SVC table representing the specified supervisory object.

These services are described in Chapter 15, “Coding Supervisory Object Services.”

Task Requests

Tasks, components, and interrupt handlers represent the only code in the system. In a multitasking system, it is necessary for a task, the primary object, to request services of other tasks.

This request structure is supported by a variable-length request queue and a variable-length completion queue in each task control block. Tasks pass information about requests and request completion through the use of request queue elements (RQEs).

Use of Wait States

A task can wait for one or more events. These events are typically referred to as *signals* (that is, a task waits for specific types of signals). The seven types of signals for which a task can wait are:

- 'Request queue' signal
- 'Completion queue' signal
- 'Completion' signal
- 'Semaphore' signal
- 'Timer' signal
- 'Generic' signal
- 'Data available' signal

Each of these signals is associated with a specific event. The first three signals ('request queue,' 'completion queue,' and 'completion') are all associated with the task request structure. Whenever a task uses the Make Request system service, it specifies what events it wants to wait for and what kind of reply should be sent to the task after the request has been completed. If it specifies a reply of 'completion queue' signal, the RQE associated with the request is placed on the task's completion queue, and a 'completion queue' signal is generated to the task when the request has been finished. If the task specifies a reply of 'completion' signal, the RQE associated with the request is not returned when the request has been finished. Therefore, it is not necessary to invoke a Get Request Completion system service. However, a 'completion' signal is generated to the task.

The 'request queue' signal is sent whenever a request (RQE) is placed onto a task's request queue. Therefore, a task can be in a "Get Request loop" waiting for a request to be placed on its request queue.

A 'semaphore' signal is generated whenever a task invokes the Claim a Semaphore system service with a "wait for semaphore free" wait state, and ownership of the semaphore is granted to the requesting task. Once the 'semaphore' signal is sent, the task is removed from the waiting state. The wait for semaphore free option is only meaningful when a task invokes the Claim a Semaphore system service.

Once a task has received a timer, it can use the Set Timer system service. After the timer is running, the task can specify a "wait for 'timer' signal" option on any system service that allows the wait option to be specified (this would normally be done when the timer is set). When the timer counts down, a 'timer' signal is sent to the task that owns the timer. If the task was waiting for a 'timer' signal, it is then removed from the wait state.

A task can wait for a ‘generic’ signal. Any task can send a ‘generic’ signal to any other task (environment restrictions apply). Therefore, a ‘generic’ signal is a means for two tasks to cooperate and communicate in a primitive manner. No meaning is attached to the ‘generic’ signal other than that which cooperating tasks assign to it.

Whenever a task requests data from a fixed-length queue, there may not be enough data in the queue to satisfy its request. If the request specifies a “wait for data” option, the task will be taken out of the wait state when some other task in the system enqueues enough data to satisfy the dequeue request. The “wait for data” option is only meaningful when the Dequeue system service is invoked.

Sending a Request to Another Task

When a task requests a service of another task, the supervisor obtains an RQE, fills it in with the details of the request (including a pointer to a parameter list and the type of reply desired), and enqueues the RQE on the requested task’s request queue. Parameter lists sent to a task must reserve the first two bytes as a return code field to be used by the workstation program.

The requesting task specifies what type of wait state it will be set to. The task can specify a multiple-wait state, which ends when any one of the wait conditions is satisfied. The requesting task also specifies what type of reply it expects when the request is completed. Possible reply types include no reply, reply via a ‘completion’ signal, or reply via an RQE on the requesting task’s completion queue. For example, a task can request a service, specifying that it will wait for an RQE on its completion queue, and expect a reply in the form of an RQE on its completion queue.

Requests can be made to tasks or components. When they are directed at components, the wait requested must be a ‘completion’ signal. The system will simply call the component, passing in registers the details of the request.

Receiving a Request from Another Task

A task that receives requests can be in a never-terminating loop, waiting to get requests. The task can do this by invoking the Get a Request service and specifying that it wants to wait until an RQE is enqueued on its request queue. When an RQE is enqueued on this task’s request queue, the task is set to the “dispatchable” state. The registers of this task will contain the details of the request.

Replying to a Request from Another Task

The requested task can now service the request and use the Reply to a Request service to send a reply and the request-completion information to the requesting task. When a task issues the Reply to a Request service, the supervisor removes the RQE from the task's request queue, examines it for the type of reply specified by the requesting task, and sends that reply to the requesting task.

Obtaining Request Completion from Another Task

The supervisor notifies the requesting task that the request has been completed by sending it the type of reply specified on the request. If the requesting task specified that it wanted an RQE placed on its completion queue, it must use the Get Request Completion service to obtain the completion values of the registers and parameter list.

Task Request Services Your Application Program Can Use

The task/component request services provided by the API are:

- **Make a Request:** Use this service to put a request queue element on a task's request queue, or to directly invoke a component.
- **Get a Request:** Use this service to obtain the contents of a request queue element (which includes a parameter list) on a task's request queue. A component does not have to do this. When a component gets control, the parameter list passed to it is pointed to by the ES and DI registers.
- **Reply to a Request:** Use this service to remove a specified request queue element from a task's request queue and to send the specified reply to the requester. A component does not need to do this. You must get a request before you can reply to it.
- **Get Request Completion:** Use this service to obtain the contents of a request queue element from a task's completion queue. A component does not need to do this. It simply checks the parameter list.
- **Send a Signal to a Task:** Use this service to send a signal to the specified task.

These services are described in Chapter 16, "Coding Request Services."

Task State Modifiers

Tasks are the only dispatchable units of code in the system. The dispatcher portion of the workstation program is responsible for scheduling and dispatching a task on each dispatch cycle.

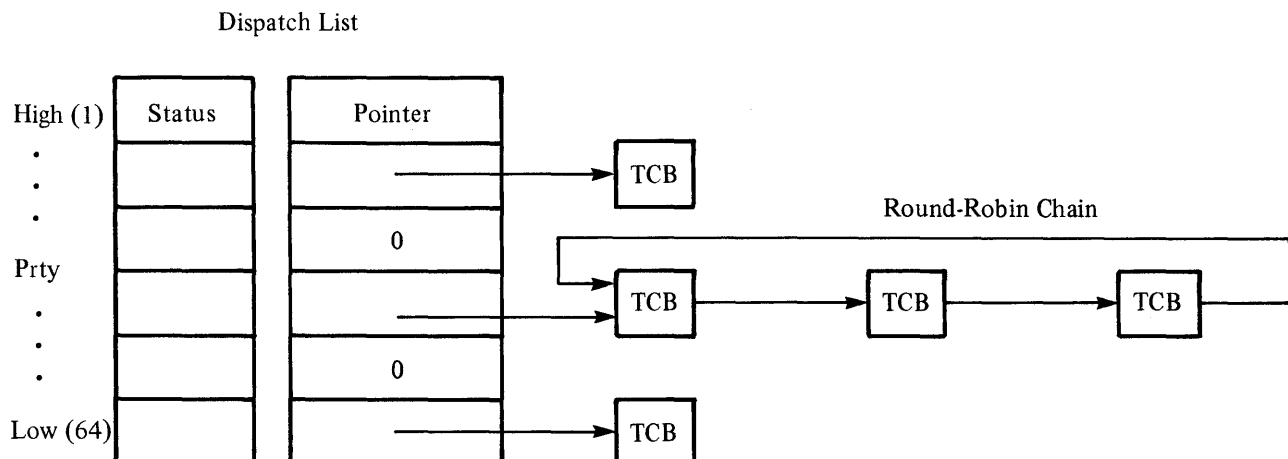
Dispatch Cycles

Dispatch cycles are driven by both hardware interrupts and voluntary relinquishment of the processor by a task. Because the workstation program is time-sliced, a dispatch cycle is guaranteed as often as hardware timer interrupts occur. A dispatch cycle occurs whenever:

- A first-level interrupt handler has finished processing.
- A task enters the “wait” or “unready” state.
- A task requests the Return to Dispatcher service, so that another task can be run.

Task Dispatching Procedure

Tasks are selected for dispatching by scanning the dispatch list. The dispatch list contains an entry for each task priority level (1 through 64). Each entry on the dispatch list consists of a status flag and a pointer to the first task control block (TCB) at that priority. The TCB of each task points to the TCB of the next task at the same priority. The last task in the TCB chain for each priority points to the first task in the chain. This type of chain is called a *round-robin* chain.



The dispatcher selects the next task to be dispatched as follows:

- The dispatcher first determines the highest dispatchable priority. If the last task to run at that priority is nonpreemptable within its round-robin and dispatchable, it is selected for dispatching. Otherwise, the next dispatchable task at that priority is selected.
- If the currently selected task has another task in its environment that had been running as nonpreemptable within the environment and did not voluntarily (by a requested wait or dispatch return request) give up control, that other task is selected instead for dispatching.

Task Dispatch Activity

Each time a task is dispatched, the following occurs:

1. The execution state (hardware flags and registers) is saved on the currently active task's stack.
2. The active task's SS and SP registers are saved.
3. The next task to be dispatched is selected.
4. The selected task's execution state is restored from its stack, and the task is set running through the use of an IRET instruction.

Task Dispatcher States

The dispatch status of each task is contained in the task's TCB. The possible dispatch states are as follows:

Dispatchable	The task is eligible to be dispatched. It is not in a wait, unready, or suspended state.
Wait	<p>The task cannot be dispatched. It is waiting for one or more of the following:</p> <ul style="list-style-type: none">• A request queue element in its request queue• A request queue element in its completion queue• A 'completion' signal• A 'semaphore claimed' signal• A 'timer tick' signal• A 'data available' signal• A 'generic' signal.
Unready	The task cannot be dispatched. The unready state ends when a task receives a 'ready' signal.

Suspend

The task cannot be dispatched. This state usually applies to all tasks within the same environment, while the unready state applies to a specific task only. The suspend state ends when the task becomes unsuspended.

Pending Unready

The resource manager puts the task in the “unready” state after it releases all code serialization semaphores.

Nonpreemptable within round robin

The task will continue to be selected for dispatching as long as no task at a higher priority becomes dispatchable.

Nonpreemptable within environment

The task will continue to be dispatched any time any task within its environment has been selected for dispatching. A task that is nonpreemptable within its environment is not nonpreemptable within its round-robin chain and, therefore, competes for the processor with tasks of equal and higher priority.

Task State Modifier Services Your Application Program Can Use

The task state modifier services provided by the API are:

- **Query Active Task:** Use this service to obtain the ID and priority of the currently active task.
- **Set Task “Ready”:** Use this service to set a specified task to the “ready” state.
- **Set Task “Unready”:** Use this service to set a specified task to the “unready” state.
- **Set Task “Preemptable”:** Use this service to set a specified task to the “preemptable” state.
- **Set Task “Nonpreemptable”:** Use this service to set a specified task to the “nonpreemptable” state.
- **Change Task’s Priority:** Use this service to change the specified task’s priority.
- **Return to Dispatcher:** Use this service to return to the dispatcher from the requesting task.

These services are described in Chapter 17, “Coding Task State Modifier Services.”

Semaphore Management

Semaphores are supervisory objects that allow your application program to control access to resources and the execution of nonreentrant code. Resource semaphores control access to resources, and code serialization semaphores control the execution of nonreentrant code.

Considerations for Using Code Serialization Semaphores

Code serialization semaphores protect segments of code in stoppable environments that should not be interrupted by stop or suspend functions. When a stop or suspend service is requested for an environment, the workstation program waits for all code serialization semaphores to be released by the tasks in the environment before honoring the stop or suspend request. Therefore, it is recommended that your application program release any code serialization semaphores it has claimed before it requests any service that causes the program to enter a dispatcher wait state.

In addition, there are times when the workstation program itself stops or suspends an environment. Your application program must release any code serialization semaphores before these stop or suspend requests can be completed. The workstation program issues stop or suspend requests on an environment at the following times:

- When you use the Split Environment or Merge Environment commands. The workstation program attempts to stop all environments involved in the split or merge operation.
- When you jump (either by using the window management services or the Jump key) to a window that contains an application that writes directly to the interrupt vectors. The workstation program attempts to suspend the application in the window you jumped from.
- When another application program becomes active, and the previously active application has claimed a code serialization semaphore and also violates any of the rules listed in Chapter 2, "Programming Considerations." The workstation program attempts to suspend the previously active application.

Restrictions on the Use of Semaphores

The supervisor imposes the following restrictions on the use of semaphores. Failure to follow these guidelines on the use of semaphores could result in system failure.

1. A task that owns a semaphore should avoid going into a dispatcher wait state. If a task that holds a resource semaphore representing a limited resource goes into a wait state, the performance of other tasks that need to access the same resource will be adversely affected.

-
2. A task that has claimed a code serialization semaphore must release that code serialization semaphore itself. Resource semaphores may be released by any task, even if the task requesting the release is not the task that claimed the semaphore.
 3. If more than one semaphore is required to access a resource, access to the semaphores should follow a defined order. An orderly access scheme for semaphore allocation reduces the possibility of resource deadlock. For example, without ordered semaphore access rules, a task that owns a semaphore may try to claim another semaphore that is owned by a task running at a higher priority. If the task running at a higher priority tries to claim the semaphore owned by the task running at the lower priority, neither task's request can be satisfied, causing both tasks to be in an infinite wait state (deadlock).

Semaphore Management Services Your Application Program Can Use

The semaphore management services provided by the API are:

- **Claim a Semaphore:** Use this service to claim a specified semaphore.
- **Release a Semaphore:** Use this service to release a specified semaphore.
- **Query a Semaphore:** Use this service to determine whether a specified semaphore is claimed or free.

These services are described in Chapter 18, "Coding Semaphore Management Services."

Logical Timer Management

The logical timer management services allow your application program to control time-dependent events through the use of logical timers.

The logical timers implemented by the workstation program use 18.2 timer ticks per second. Thus, to specify a logical timer interval of 1 second, you should specify a timer interval of 18, for a 2-second timer interval, you should specify a timer interval of 36, and so on.

Note: Reprogramming the PC timer on non-3270 PC systems could cause a host communication failure.

Logical Timer Management Services Your Application Program Can Use

The logical timer management services provided by the API are:

- **Get Logical Timer:** Use this service to get a logical timer for the specified task.
- **Set Logical Timer.** Use this service to set the timer interval for a specified logical timer.
- **Release Logical Timer:** Use this service to release a logical timer.

These services are described in Chapter 19, “Coding Logical Timer Management Services.”

Fixed-Length Queue Management

The fixed-length queue management services allow your application program to pass data to other tasks or components, and to receive data from other tasks or components, using the fixed-length queue as a “pipeline” for the data.

Fixed-Length Queue Management Services Your Application Program Can Use

The fixed-length queue management services provided by the API are:

- **Enqueue Data:** Use this service to enqueue data on the specified fixed-length queue.
- **Dequeue Data:** Use this service to dequeue data from the specified fixed-length queue.
- **Purge Queue Data:** Use this service to remove all data from the specified fixed-length queue.

These services are described in Chapter 20, “Coding Fixed-Length Queue Management Services.”

Interrupt Handler Management

Interrupt handlers are code that performs immediate functions. There are two types of interrupt handlers: hardware and software.

Hardware Interrupt Handlers

Hardware interrupt handlers generally perform action required to service a condition detected by the hardware. For example, a communication adapter may interrupt every time a character is received from the communication line. The interrupt handler would read the character received and perform any action required to allow the adapter to receive another character.

There are three ways for your application program to take over hardware interrupts. The first way is to use the DOS function calls X'35' and X'25', described in the *DOS Technical Reference* manual. The second way is to use the Install a Hardware Interrupt Handler service provided by the supervisor. The third way is to use the Install an Interrupt Handler service.

Using the DOS Function Calls to Take Over Hardware Interrupts

A program may use the DOS facilities to take over a hardware interrupt, provided there is no requirement to share the interrupt with a program in another environment. The interrupt handler cannot be removed (short of a system re-IPL) without endangering the integrity of the system.

Using the Install a Hardware Interrupt Handler Service to Take Over Hardware Interrupts

The Install a Hardware Interrupt Handler service allows system extensions or applications requiring hardware interrupt service to install an interrupt handler in the system, and to share hardware interrupt vectors in certain circumstances. This service provides enough information for the supervisor to remove the interrupt handler without disturbing other users of that level if the environment is stopped.

Programs that use the Install a Hardware Interrupt Handler service to share an interrupt vector must meet these restrictions:

- The device must be pollable at one address, where the byte of data will return nonzero when it is logically ANDed with a byte mask whenever the device is interrupting.
- It must be possible to disable the device by writing one byte of data to a single port address.
- The handler should return with the FAR option (not IRET) and should not call the previous handler.

Using the Install an Interrupt Handler Service to Take Over Hardware Interrupts

The Install an Interrupt Handler service allows system extensions or applications requiring hardware interrupt service to install an interrupt handler into the system. Using this service does not provide the supervisor with enough information to remove the interrupt handler without disturbing other users of that level if the environment is stopped. Therefore, use this service only if you cannot meet the restrictions for using the Install a Hardware Interrupt Handler service. Also, if a hardware interrupt handler installed by this service chooses to chain to the previous handler, it is imperative, before jumping to the previous handler, that the registers and stack be the same as when the interrupt handler gained control.

Hardware Interrupt Handler Considerations

Regardless of whether the application uses the DOS function calls or a supervisor service, no interrupt handler should ever create or destroy supervisory objects (such as gates, tasks, timers, or other interrupt handlers). Neither should a hardware interrupt handler attempt to use the Name Resolution service for supervisory objects.

Note: Application programs should not take over a hardware interrupt by writing directly to the interrupt vector table. The workstation program will support programs that do this, but only with significant performance degradation. See Chapter 2, "Programming Considerations," for more information.

Software Interrupt Handlers

Software interrupt handlers provide another means by which programs that are not linked together can communicate. One program can set an interrupt vector to point to a software interrupt handler within itself. When another program issues an INT instruction, the interrupt handler gains control. A software interrupt handler may be classified as either local or global. Local interrupt handlers may only receive interrupts originating within the same environment as the interrupt handler. Global interrupt handlers may receive interrupts originating from any environment.

There are two ways for your application program to take over software interrupts. The first way is to use the DOS function calls X'35' and X'25', described in the *DOS Technical Reference* manual. The second way is to use the Install an Interrupt Handler service provided by the supervisor.

Using the DOS Function Calls to Take Over Software Interrupts

A program may use the DOS facilities to take over a software interrupt, provided there is no requirement to share the interrupt with a program in another environment. The interrupt handler will only gain control if the INT instruction is executed from within the same environment.

Using the Install an Interrupt Handler Service to Take Over Software Interrupts

The supervisor provides a full set of interrupt vectors for each environment. When a software interrupt occurs, the supervisor fields the interrupt, uses its system's tables to determine the active environment, and gives control to the appropriate software interrupt handler. The Install an Interrupt Handler service allows programs to request that a software interrupt handler be installed. This service allows interrupt handlers to be defined as "local" or "global."

Local Software Interrupt Handlers

Local interrupt handlers only receive software interrupts on the requested vector when the interrupt originated in the handler's own environment.

Global Software Interrupt Handlers

Global interrupt handlers receive all software interrupts for a given vector, regardless of the environment that issued it.

An additional option for global software interrupt handlers is available. A program may request to service software interrupts as a "last resort." The requester will get the interrupt only if no other interrupt handlers are found to service it.

Generally, programs should avoid the use of global interrupt handlers if the purpose of the interrupt handler is to provide some service to other programs, since program environments can be stopped, killed, or suspended at any time during the software interrupt processing. To avoid this problem, the program should be written as a system extension and define gates, components, or tasks in order to receive requests. Another possibility is to have the interrupt handler claim a code serialization semaphore before processing the interrupt. Tasks holding code serialization semaphores are not stopped until the semaphore is released.

Note: Global or last-resort interrupt handlers cannot be installed in a stoppable environment.

Software Interrupt Handler Considerations

Application programs should not take over a software interrupt by writing directly to the interrupt vector table. The workstation program supports programs that do this, but only with significant performance degradation. See Chapter 2, "Programming Considerations," for more information.

Interrupt Handler Management Services Your Application Program Can Use

The interrupt handler management services provided by the API are:

- **Install a Hardware Interrupt Handler:** Use this service to identify an interrupt routine that is to gain control on hardware interrupts.
- **Install an Interrupt Handler:** Use this service to identify an interrupt routine that is to gain control on software or hardware interrupts. This service also returns the entry point of the previous interrupt handler. For hardware interrupt handlers, the Install a Hardware Interrupt Handler service is the recommended service to use if you can satisfy the restrictions for using it.
- **Query Interrupt Vector Contents:** Use this service to obtain the entry point address of the second-level interrupt handler currently installed for the specified interrupt vector.
- **Remove an Interrupt Handler:** Use this service to remove an interrupt handler that was installed through the Install a Hardware Interrupt Handler or Install an Interrupt Handler service request.

These services are described in Chapter 21, “Coding Interrupt Handler Management Services.”

Chapter 15. Coding Supervisory Object Services

Introduction	15-2
Supervisory Object Service X'92': Create Task Entry	15-4
Supervisory Object Service X'93': Create Component Entry	15-8
Supervisory Object Service X'94': Create Semaphore Entry	15-11
Supervisory Object Service X'04': Create Fixed-Length Queue Entry	15-14
Supervisory Object Service X'9A': Create Gate Entry	15-17
Supervisory Object Service X'97': Create User Exit Table Entry	15-21
Supervisory Object Service X'0E': Install User Exit Table Entries ..	15-24
Supervisory Object Service X'81': Name Resolution	15-27
Supervisory Object Service X'01': ID Resolution	15-30
Supervisory Object Service X'06': Delete Entry	15-32

Introduction

This chapter describes how to code requests for the supervisory object services provided by the API.

The supervisory object services allow your application program to create and delete tasks, components, semaphores, fixed-length queues, gates, and user exit tables. The supervisory object services also allow your application program to obtain the numeric ID of a supervisory object by specifying its alphanumeric name, or obtain the alphanumeric name of the supervisory object by specifying its numeric ID.

The supervisory object services provided by the API are:

- **Create a Task Entry:** Use this service to create an entry in the SVC table for a task.
- **Create a Component Entry:** Use this service to create an entry in the SVC table for a component.
- **Create a Semaphore Entry:** Use this service to create an entry in the SVC table for a semaphore.
- **Create a Fixed-Length Queue Entry:** Use this service to create an entry in the SVC table for a fixed-length queue.
- **Create a Gate Entry:** Use this service to create an entry in the SVC table for a gate.
- **Create a User Exit Table Entry:** Use this service to create an entry in the SVC table for a user exit table.
- **Install User Exit Table Entries:** Use this service to install entries in the specified user exit table.
- **Name Resolution:** Use this service to resolve the specified supervisory object name to its numeric index.
- **ID Resolution:** Use this service to resolve the specified supervisory ID name to its alphanumeric name.
- **Delete an Entry:** Use this service to delete an entry in the SVC table representing the specified supervisory object.

When created, an object can be optionally named. That name, however, must be unique.

Note: Following is a list of names used by the supervisor. Do not assign these same names to system objects that you create.

- *Names that begin with the letters IND or 3270KS*
- *SYSKILL*
- *MEMORY*
- *DOSINT21*
- *DOSIOR*
- *DOSBADP*
- *XLATE*
- *SESSMGR*
- *KEYBOARD*
- *WSCTRL*
- *OIAM*
- *CPYUET*
- *MFIC*
- *3270EML*
- *PCPSM*
- *COPY*
- *BSMUET*

Requesting the Supervisory Object Services

To request any of the supervisory object services, load the registers and the parameter list with the proper values, and use the INT 7AH instruction to signal the workstation program that it has a request to process.

Return Codes for the Supervisory Object Services

Return codes for the supervisory object services are 2-byte values made up of a function ID and an error code. The function ID indicates the portion of the workstation program in which the error occurred. The error code indicates the specific type of error that has occurred. An error code of X'00' indicates a successful acceptance or completion of the request.

After your application has requested a supervisory object service, the CH and CL registers contain a return code generated by the request processing portion of the workstation program. The function ID is in the CH register, and the error number is in the CL register. The return codes that can be generated by supervisory object services are called *system return codes*. The function ID for system return codes is X'12' or X'13'. The error codes that can appear are specific to the service that was requested and are included in the descriptions of each service.

See Appendix H, "Return Codes," for more information.

Create Task Entry

Supervisory Object Service X'92': Create Task Entry

Use this service to create an entry in the SVC table for a task.

Register Values

On Request

AH = X'92'
AL = X'00'
BH = Preemptive status
BL = 00 = no name / 01 = name / 02 = reset
CX = Task priority
DX = Task ID *
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = Function ID
CL = Return code
DX = Task ID

The contents of registers
AX, BX, ES, and DI are
unpredictable.

* The value coded in the DX register is dependent on the value coded in the BL register. See "Register Definitions" below for more information.

Register Definitions

Request Registers:

- The BH register indicates whether the new task is preemptable.

Possible values for the BH register are :

X'00' = The task is preemptable.
X'01' = The task is nonpreemptable within its priority level.
X'02' = The task is nonpreemptable within its environment.

If the BH register contains a value other than X'00', X'01', or X'02', the supervisor sets the preemptive status of the task to "preemptable."

- The BL register indicates whether the task has a name associated with it, or whether the task is to be reinitialized with the specified parameters.

Possible values for the BL register are :

X'00' = The task has no name.
X'01' = The task's name is in the parameter list.
X'02' = The task is to be reset to the specified parameters.

The "reset" option can only be used by requesters resetting tasks within the same environment. It is used for tasks previously created and still in the system. Use this option if you want to reinitialize the task. You must include all input as if you were creating the task for the first time. This option should only be used after a stop environment service that also used the reset option.

- The CX register indicates the dispatch priority of the task. If a system extension is issuing this request, valid dispatch priorities are 1 through 64. If an application program running in a stoppable environment is issuing this request, valid dispatch priorities are 36 through 64.
- The DX register indicates the task ID of the task to be reset. This register is used only if the BL register indicates that the task is to be reset.
- The ES register contains the segment address of the parameter list.
- The DI register contains the offset address of the parameter list.

Completion Registers:

- The DX register contains the ID of the task.

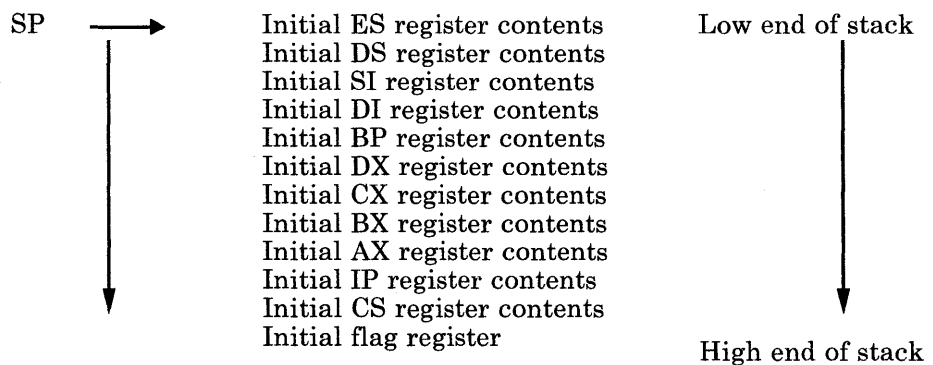
Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 word	Offset address of the task's stack	Unchanged
2	1 word	Segment address of the task's stack	Unchanged
4 - 11	8 bytes	Task name	Unchanged

Parameter Definitions

Request Parameters:

- The offset specified for the stack must point to the address containing the initial ES register value, so the stack should be set up as follows:



Note: When a task is dispatched, all the initial register values are placed into the corresponding registers and an IRET instruction is performed.

Create Task Entry

Note: When a task is dispatched, all the initial register values are placed into the corresponding registers and an IRET instruction is performed.

- The task name is an optional parameter and is needed only if the BL register is set to 1 on request. The task name can be a maximum of eight ASCII characters and should be padded to the right with blanks if it is less than eight characters.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received are:

Code	Meaning
X'00'	Successful completion of the request.
X'01'	The task name already exists.
X'02'	SVC table full.
X'03'	Name table full.
X'04'	No more free TCBs.
X'05'	Invalid task ID (on reset).
X'06'	Invalid priority.

Usage Notes

- A task is always created in the “unready” state. You must set the task to the “ready” state to make it dispatchable.
- A task is a continually executing thread. Therefore, care must be taken when it completes its function. In a stoppable environment, the only task that should return to DOS is the task under which the application first began running. All other tasks should be deleted. In a nonstoppable environment, most tasks will be in never-ending loops—typically, waiting for a unit of work, performing that work, and looping to the top, where it will wait again for the next unit of work.
- Whenever a task or component is providing a service to other applications, and a parameter list is required, the first two bytes of the parameter list should be used as the return code field.

Coding Example

```

;
; PARAMETER LIST FOR CREATE TASK
;
CTTASKSP DW 0 ; TASK'S STARTING SP
CTTASKSS DW 0 ; TASK'S STARTING SS
CTTSKNAM DB 8 DUP(0) ; TASK NAME
;
; THE TASK'S STACK
;
STACK DB 256 DUP(0)
.
.
.
;
; INITIALIZE THE TASK'S STACK
;
    PUSH DS ; SAVE DS
    MOV AX,SEG STACK ; GET THE TASK'S STACK SEGMENT
    MOV DS,AX
    MOV SI,OFFSET STACK ; DS:SI NOW POINT TO THE TASK STACK
    MOV WORD PTR [SI+254],0F242H ; SET FLAGS IN THE STACK
    MOV AX,SEG TASK
    MOV WORD PTR [SI+252],AX ; SET CODE SEGMENT OF TASK IN STACK
    MOV AX,OFFSET TASK
    MOV WORD PTR [SI+250],AX ; SET CODE OFFSET OF TASK IN STACK
    POP AX ; GET THE PROGRAM'S DS REGISTER INTO
    PUSH AX ; AX (PUT IT BACK ON THE STACK)
    MOV WORD PTR [SI+234],AX ; SET DATA SEGMENT IN STACK
    POP DS ; RESTORE DS
;
; INITIALIZE PARAMETER LIST FOR CREATE TASK
;
    MOV AX,OFFSET STACK ; GET THE OFFSET OF THE TASK'S STACK
    ADD AX,232 ; INCREMENT THE SP TO POINT AT THE
    ; START OF REGISTER RESTORE AREA
    MOV CTTASKSP,AX ; PUT TASK'S SP IN PARAMETER LIST
    MOV AX,SEG STACK ; PUT TASK'S SS IN PARAMETER LIST
    MOV CTTASKSS,AX
;
    MOV BL,0 ; NO NAME SPECIFIED
;
; INITIALIZE REGISTERS FOR CREATE TASK
;
    MOV AH,92H
    MOV AL,0
    MOV BH,PREEMPT ; PREEMPTION TYPE IN BH
    MOV CX,PRIORITY ; PRIORITY IN CX
    MOV DI,SEG CTTASKSP ; SEGMENT ADDRESS OF PARAMETER LIST
    MOV ES,DI ; IN ES
    MOV DI,OFFSET CTTASKSP ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR CREATE TASK SERVICE
;
    INT 7AH
.
.
.

```

Supervisory Object Service X'93': Create Component Entry

Use this service to create an entry in the SVC table for a component.

Register Values

On Request

AH = X'93'
AL = X'00'
BL = 00 = no name / 01 = name
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = Function ID
CL = Return code
DX = Component ID

The contents of registers AX, BX, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The BL register indicates whether the component has a name associated with it.

Possible values for the BL register are :

X'00' = The component has no name.
X'01' = The component's name is in the parameter list.

- The ES register contains the segment address of the parameter list.
- The DI register contains the offset address of the parameter list.

Completion Registers:

- The DX register contains the ID of the component.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 word	Offset address of the component's entry point	Unchanged
2	1 word	Segment address of the component's entry point	Unchanged
4 – 11	8 bytes	Component name	Unchanged

Parameter Definitions

Request Parameters:

- The entry point of the component is specified as the contents of the CS and IP registers when the component is invoked.
- The component name is an optional parameter and is needed only if the BL register is set to 1 on request. The component name can be a maximum of eight ASCII characters and should be padded to the right with blanks if necessary.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The system return codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion of the request.
X'01'	The component name already exists.
X'02'	SVC table full.
X'03'	Name table full.

Usage Notes

- When a component receives control, the pointer to the parameter list will be in the ES and DI registers.
- When a component receives control, interrupts are disabled.
- When a component is finished executing, it should use a RETURN FAR to return.

Create Component Entry

Coding Example

```
;
; PARAMETER LIST FOR CREATE A COMPONENT
;
CCCMPOFF  DW  0                ; COMPONENT ENTRY POINT OFFSET
CCCMPSEG  DW  0                ; COMPONENT ENTRY POINT SEGMENT
CCNAME    DB  8 DUP(0)        ; COMPONENT NAME
.
.
.

;
; INITIALIZE PARAMETER LIST FOR CREATE A COMPONENT
;
      MOV     AX,OFFSET CPMONET ; COMPONENT OFFSET INTO THE LIST
      MOV     CCCMPOFF,AX
      MOV     AX,SEG CPMONET    ; COMPONENT SEGMENT INTO THE LIST
      MOV     CCCMPSEG,AX

      MOV     AX,SEG CCNAME     ; [ES:DI] POINTS TO THE DESTINATION IN THE
      MOV     ES,AX            ;   PARAMETER LIST
      MOV     DI,OFFSET CCNAME
      MOV     SI,OFFSET COMPNAME ; [DS:SI] POINTS TO SOURCE OF THE NAME
      MOV     CX,8              ; MOVE 8 BYTES
      REP     MOVSB             ; COPY THE NAME INTO THE PARM LIST

;
; INITIALIZE REGISTERS FOR CREATE A COMPONENT
;
      MOV     AH,93H
      MOV     AL,0
      MOV     BL,1              ; BL = 1 SINCE A NAME IS SPECIFIED
      MOV     DI,SEG CCCMPOFF   ; SEGMENT ADDRESS OF PARAMETER LIST
      MOV     ES,DI             ;   IN ES
      MOV     DI,OFFSET CCCMPOFF ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR CREATE A COMPONENT SERVICE
;
      INT     7AH
      .
      .
      .
```

Supervisory Object Service X'94': Create Semaphore Entry

Use this service to create an entry in the SVC table for a semaphore.

Register Values

On Request

AH = '94'
BH = Semaphore type
BL = 00 = no name / 01 = name
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = Function ID
CL = Return code
DX = Semaphore ID

The contents of registers AX, BX, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The BH register indicates the semaphore type.

Possible values for the BH register are :

X'03' = The semaphore is a code serialization semaphore.
X'04' = The semaphore is a resource semaphore.

See Chapter 14, "Supervisor Services," and Chapter 2, "Programming Considerations," for guidelines on using semaphores.

- The BL register indicates whether the semaphore has a name associated with it.

Possible values for the BL register are :

X'00' = The semaphore has no name.
X'01' = The semaphore's name is in the parameter list.

- The ES register contains the segment address of the parameter list.
- The DI register contains the offset address of the parameter list.

Note: The parameter list is needed only if the BL register is set to 1.

Completion Registers:

- The DX register contains the ID of the semaphore.

Create Semaphore Entry

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0 – 7	8 bytes	Semaphore name	Unchanged

Parameter Definitions

Request Parameters:

The semaphore name is an optional parameter and is needed only if the BL register is set to 1 on request. The semaphore name can be a maximum of eight ASCII characters and should be padded to the right with blanks if necessary.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received are:

Code	Meaning
X'00'	Successful completion of the request.
X'01'	The semaphore name already exists.
X'02'	SVC table full.
X'03'	Name table full.
X'3C'	Invalid semaphore type.

Coding Example

```
;
; PARAMETER LIST FOR CREATE A SEMAPHORE ENTRY
;
CSNAME      DB  8 DUP(0)          ; SEMAPHORE NAME
.
.
.

;
; INITIALIZE PARAMETER LIST FOR CREATE A SEMAPHORE ENTRY
;
      MOV     SI,OFFSET SEMNAME    ; [DS:SI] POINTS TO SOURCE OF THE NAME
      MOV     AX,SEG CSNAME        ; [ES:DI] POINTS TO DESTINATION IN PARM
      MOV     ES,AX                ; LIST
      MOV     DI,OFFSET CSNAME
      MOV     CX,8                 ; MOVE 8 BYTES
      REP     MOVSB                ; COPY THE NAME INTO THE PARM LIST

;
; INITIALIZE REGISTERS FOR CREATE A SEMAPHORE ENTRY
;
      MOV     AH,94H
      MOV     BH,03H              ; SEMAPHORE TYPE = CODE SERIALIZATION
      MOV     BL,1                ; BL = 1 SINCE A NAME IS SPECIFIED
      MOV     DI, SEG CSNAME       ; SEGMENT ADDRESS OF PARAMETER LIST
      MOV     ES,DI               ; IN ES
      MOV     DI,OFFSET CSNAME    ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR CREATE A SEMAPHORE ENTRY SERVICE
;
      INT     7AH
.
.
.
```


Create Fixed-Length Queue Entry

Supervisory Object Service X'04': Create Fixed-Length Queue Entry

Use this service to create an entry in the SVC table for a fixed-length queue.

Register Values

On Request

AH = X'04'
BL = 00 = no name / 01 = name
CX = Queue length
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = Function ID
CL = Return code
DX = Queue ID

The contents of registers AX, BX, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The BL register indicates whether the queue has a name associated with it.

Possible values for the BL register are:

X'00' = The queue has no name.

X'01' = The queue's name is in the parameter list.

- The CX register contains the number of bytes your application program has reserved for the fixed-length queue. The queue must be greater than 10 bytes long, because the first 10 bytes of the queue are reserved for use by the workstation program.
- The ES register contains the segment address of the parameter list.
- The DI register contains the offset address of the parameter list.

Completion Registers:

- The DX register contains the ID of the fixed-length queue.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 word	Offset address of the queue	Unchanged
2	1 word	Segment address of the queue	Unchanged
4 – 11	8 bytes	Queue name	Unchanged

Parameter Definitions

Request Parameters:

The queue name is an optional parameter and is needed only if the BL register is set to 1 on request. The queue name can be a maximum of eight ASCII characters and should be padded to the right with blanks if necessary.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received are:

Code	Meaning
X'00'	Successful completion of the request.
X'01'	The queue name already exists.
X'02'	SVC table full.
X'03'	Name table full.
X'41'	Invalid queue length.

Usage Notes

The fixed-length queue resides in the requester's environment.

Create Fixed-Length Queue Entry

Coding Example

```
;
; DEFINE PARAMETER LIST FOR CREATE QUEUE
;
CQQOFFS  DW    0
CQSEGM   DW    0
CQQNAME  DB    8 DUP(' ')
.
.
.
;
; INITIALIZE FIRST 2 ENTRIES OF PARAMETER LIST
;
      MOV     CQQOFFS,OFFSET Q ; OFFSET OF QUEUE
      MOV     CQSEGM,SEG Q      ; SEGMENT OF QUEUE
;
; THE USER HAS A QUEUE NAME
;
      MOV     BL,01H            ; INDICATE A QNAME IS SPECIFIED
      CLD                     ; BEGIN MOVING QNAME TO THE PARAM LIST
      MOV     CX,4              ; QNAME IS FOUR WORDS LONG
      MOV     SI,OFFSET QNAME   ; SOURCE OFFSET OF QUEUE
      MOV     DI,OFFSET CQQNAME; DESTINATION OFFSET IS CQQNAME
      REP     MOVSW             ; MOVE QNAME TO PARAMETER LIST
;
; INITIALIZE REGISTERS FOR CREATE QUEUE
;
      MOV     AH,04H
      MOV     CX,50             ; CX = NUMBER OF BYTES FOR QUEUE
      MOV     DI,SEG CQQOFFS    ; ADDRESSABILITY TO
      MOV     ES,DI             ; PARAMETER LIST
      MOV     DI,OFFSET CQQOFFS; USING ES:DI
;
; SIGNAL WORKSTATION PROGRAM FOR CREATE QUEUE SERVICE
;
      INT     7AH
.
.
.
```

Supervisory Object Service X'9A': Create Gate Entry

Use this service to create an entry in the SVC table for a gate.

Register Values

On Request

AH = X'9A'
BL = 00 = no name / 01 = name
CX = Number of services
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = Function ID
CL = Return code
DX = Gate ID

The contents of registers
AX, BX, ES, and DI are
unpredictable.

Register Definitions

Request Registers:

- The BL register indicates whether the gate has a name associated with it.

Possible values for the BL register are :

X'00' = The gate has no name.
X'01' = The gate's name is in the parameter list.

- The CX register indicates the number of services provided by the gate.
- The ES register contains the segment address of the parameter list.
- The DI register contains the offset address of the parameter list.

Completion Registers:

- The DX register contains the ID of the gate.

Create Gate Entry

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 word	Offset address of the service entry array	Unchanged
2	1 word	Segment address of the service entry array	Unchanged
4 – 11	8 bytes	Gate name	Unchanged

Parameter Definitions

Request Parameters:

- The format of the service entry array is as follows:

Offset	Length	Contents on Request	Contents on Completion
0	1 word	Service entry 1	Unchanged
2	1 word	Service entry 2	Unchanged
4	1 word	Service entry 3	Unchanged
⋮			
2n-2	1 word	Service entry n, where n is the number of service entries specified in the CX register on request.	Unchanged

A service entry is the ID of the task or component that will provide the service. This ID was obtained when the task or component was created.

- The gate name is an optional parameter. The gate name can be a maximum of eight ASCII characters and should be padded to the right with blanks if necessary.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received are:

Code	Meaning
X'00'	Successful completion of the request.
X'01'	The gate name already exists.
X'02'	SVC table full.
X'03'	Name table full.
X'0F'	Invalid environment access.
X'34'	The service entry is not a task or component ID, or the number of services specified in the gate is invalid.
X'3B'	Gate table full.

Usage Notes

- Only nonstoppable environments (system extensions) can create gates.
- Service entries may not be changed, added to, or deleted from a gate after it has been initialized.
- Requests through gates can be made from any environment.

Create Gate Entry

Coding Example

```
;
; PARAMETER LIST FOR CREATE A GATE ENTRY
;
CGENTRYS DD NTRYARRAY          ; POINTER TO ENTRY ARRAY
CGNAME   DB  8 DUP(0)          ; GATE NAME
;
; ENTRY ARRAY
;
NTRYARRAY DW 10 DUP(0)          ; ENTRY ARRAY FOR 10 ENTRIES
.
.
.
;
; INITIALIZE PARAMETER LIST FOR CREATE A GATE ENTRY
;
        MOV     AX,SEG CGNAME    ; ES:DI POINT TO DESTINATION IN PARM
        MOV     ES,AX            ; LIST
        MOV     DI,OFFSET CGNAME
        MOV     SI,OFFSET GATENAME ; DS:SI POINT TO SOURCE OF THE NAME
        MOV     CX,8             ; MOVE 8 BYTES
        REP     MOVSB            ; COPY THE NAME INTO THE PARM LIST
;
; INITIALIZE REGISTERS FOR CREATE A GATE ENTRY
;
        MOV     AH,9AH
        MOV     BL,1              ; BL = 1 SINCE A NAME IS SPECIFIED
        MOV     CX,10             ; CX = NUMBER OF ENTRIES (10)
        MOV     DI, SEG CGENTRYS  ; SEGMENT ADDRESS OF PARAMETER LIST
        MOV     ES,DI            ; IN ES
        MOV     DI,OFFSET CGENTRYS ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR CREATE A GATE ENTRY SERVICE
;
        INT     7AH
.
.
.
```

Supervisory Object Service X'97': Create User Exit Table Entry

Use this service to create an entry in the SVC table for a user exit table.

Register Values

On Request

AH = X'97'
BL = 00 = no name / 01 = name
CX = Number of entries
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = Function ID
CL = Return code
DX = User exit table ID

The contents of registers AX, BX, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The BL register indicates whether the user exit table has a name associated with it.

Possible values for the BL register are :

X'00' = The user exit table has no name.

X'01' = The user exit table's name is in the parameter list.

- The CX register contains the maximum number of entries that the user exit table will be able to contain.
- The ES register contains the segment address of the parameter list.
- The DI register contains the offset address of the parameter list.

Completion Register:

- The DX register contains the ID of the user exit table.

Create User Exit Table Entry

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 word	Offset address of the user exit table	Unchanged
2	1 word	Segment address of the user exit table	Unchanged
4 – 11	8 bytes	User exit table name	Unchanged

Parameter Definitions

Request Parameters:

- The format of the user exit table is as follows:

Offset	Length	Contents
0	1 word	Offset address of user exit table entry 0
2	1 word	Segment address of user exit table entry 0
4	1 word	Offset address of user exit table entry 1
6	1 word	Segment address of user exit table entry 1
8	1 word	Offset address of user exit table entry 2
10	1 word	Segment address of user exit table entry 2
12	1 word	Offset address of user exit table entry 3
14	1 word	Segment address of user exit table entry 3
⋮		
4n-4	1 word	Offset address of user exit table entry n-1
4n-2	1 word	Segment address of user exit table entry n-1

In this table, n is the number of entries in the user exit table. User exit table entries are numbered from 0 to $n-1$. The contents of a user exit entry are the address of code to be given control when it is invoked by means of a CALL FAR instruction.

- The user exit table name is an optional parameter. The user exit table name can be a maximum of eight ASCII characters and should be padded to the right with blanks if necessary.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received are:

Code	Meaning
X'00'	Successful completion of the request.
X'01'	The name already exists.
X'02'	SVC table full.
X'03'	Name table full.
X'35'	The user exit table size cannot be zero.

Usage Notes

- Access to user exit tables is not valid between stoppable environments.

Coding Example

```
;
; PARAMETER LIST FOR CREATE A USER EXIT TABLE ENTRY
;
CUUETOFF  DW  0                      ; OFFSET OF USER EXIT TABLE
CUUETSEG  DW  0                      ; SEGMENT OF USER EXIT TABLE
CUNAME    DB  8 DUP(0)              ; USER EXIT TABLE NAME
;
; USER EXIT TABLE
;
UET       DD  7 DUP(?)              ; THE USER EXIT TABLE
.
.
.
;
; INITIALIZE PARAMETER LIST FOR CREATE A USER EXIT TABLE ENTRY
;
      MOV     AX,OFFSET UET          ; OFFSET OF USER EXIT TABLE INTO THE LIST
      MOV     CUUETOFF,AX
      MOV     AX,SEG UET             ; SEGMENT OF USER EXIT TABLE INTO THE LIST
      MOV     CUUETSEG,AX
;
; INITIALIZE REGISTERS FOR CREATE A USER EXIT TABLE ENTRY
;
      MOV     AH,97H
      MOV     BL,0                  ; BL = 0 SINCE NO NAME IS SPECIFIED
      MOV     CX,7                  ; NUMBER OF ENTRIES IN USER EXIT TABLE = 7
      MOV     DI,SEG CUUETOFF       ; SEGMENT ADDRESS OF PARAMETER LIST
      MOV     ES,DI                ; IN ES
      MOV     DI,OFFSET CUUETOFF    ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR CREATE A USER EXIT TABLE ENTRY SERVICE
;
      INT     7AH
.
.
.
```

Install User Exit Table Entries

Supervisory Object Service X'0E': Install User Exit Table Entries

Use this service to install up to n entries in the specified user exit table.

Register Values

On Request

AH = X'0E'
DX = User exit table ID
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = Function ID
CL = Return code
The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The DX register contains the ID of the user exit table where the entries are to be installed.
- The ES register contains the segment address of the parameter list.
- The DI register contains the offset address of the parameter list.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 word	n (number of user exit table entries)	Unchanged
2	1 word	User exit table entry number	Unchanged
4	1 word	Offset address of user exit	Previous contents
6	1 word	Segment address of user exit	Previous contents
⋮			
6n-4	1 word	User exit table entry number	Unchanged
6n-2	1 word	Offset address of user exit	Previous contents
6n	1 word	Segment address of user exit	Previous contents

Parameter Definitions

Request Parameters:

- The format of the user exit table is as follows:

Offset	Length	Contents
0	1 word	Offset address of user exit entry 0
2	1 word	Segment address of user exit entry 0
4	1 word	Offset address of user exit entry 1
6	1 word	Segment address of user exit entry 1
8	1 word	Offset address of user exit entry 2
10	1 word	Segment address of user exit entry 2
12	1 word	Offset address of user exit entry 3
14	1 word	Segment address of user exit entry 3
⋮		
4n-4	1 word	Offset address of user exit entry n-1
4n-2	1 word	Segment address of user exit entry n-1

Where n is the number of entries in the user exit table. User exit table entries must start with entry 0 as the first entry.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received are:

Code	Meaning
X'00'	Successful completion of the request.
X'05'	Invalid user exit table ID.
X'15'	The user exit table entry number is out of range.
X'16'	An incorrect number of user exit table entries was specified.
X'1A'	The user exit table address space is not available to the requester.

Usage Notes

This service may be requested between nonstoppable environments or within the requester's environment.

Install User Exit Table Entries

Coding Example

```
;
; PARAMETER LIST FOR INSTALL USER EXIT TABLE ENTRIES
;
IUNUMENT DW 0 ; NUMBER OF USER EXIT TABLE ENTRIES
IUUET1NO DW 0 ; USER EXIT TABLE ENTRY NUMBER
IUUET1OF DW 0 ; OFFSET OF USER EXIT TABLE ENTRY
IUUET1SG DW 0 ; SEGMENT OF USER EXIT TABLE ENTRY
IUUET2NO DW 0 ; USER EXIT TABLE ENTRY NUMBER
IUUET2OF DW 0 ; OFFSET OF USER EXIT TABLE ENTRY
IUUET2SG DW 0 ; SEGMENT OF USER EXIT TABLE ENTRY
IUUET3NO DW 0 ; USER EXIT TABLE ENTRY NUMBER
IUUET3OF DW 0 ; OFFSET OF USER EXIT TABLE ENTRY
IUUET3SG DW 0 ; SEGMENT OF USER EXIT TABLE ENTRY

.
.
.

;
; INITIALIZE PARAMETER LIST FOR INSTALL USER EXIT TABLE ENTRIES
;
MOV IUNUMENT,3 ; 3 ENTRIES TO INSTALL IN THE USER EXIT TABLE

MOV AX,ENT1NO ; NUMBER OF THE 1ST ENTRY IN THE LIST
MOV [BX].IUNUMBER,AX
MOV AX,OFFSET ENT1 ; OFFSET OF THE 1ST ENTRY IN THE LIST
MOV [BX].IUUETOFF,AX
MOV AX,SEG ENT1 ; SEGMENT OF THE 1ST ENTRY IN THE LIST
MOV [BX].IUUETSEG,AX

MOV AX,ENT2NO ; NUMBER OF THE 2ND ENTRY IN THE LIST
MOV [BX].IUNUMBER,AX
MOV AX,OFFSET ENT2 ; OFFSET OF THE 2ND ENTRY IN THE LIST
MOV [BX].IUUETOFF,AX
MOV AX,SEG ENT2 ; SEGMENT OF THE 2ND ENTRY IN THE LIST
MOV [BX].IUUETSEG,AX

MOV AX,ENT3NO ; NUMBER OF THE 3RD ENTRY IN THE LIST
MOV [BX].IUNUMBER,AX
MOV AX,OFFSET ENT3 ; OFFSET OF THE 3RD ENTRY IN THE LIST
MOV [BX].IUUETOFF,AX
MOV AX,SEG ENT3 ; SEGMENT OF THE 3RD ENTRY IN THE LIST
MOV [BX].IUUETSEG,AX

;
; INITIALIZE REGISTERS FOR INSTALL USER EXIT TABLE ENTRIES
;
MOV AH,0EH
MOV DX,UETID ; USER EXIT TABLE ID IN DX
MOV DI, SEG IUNUMENT ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET IUNUMENT ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR INSTALL USER EXIT TABLE ENTRIES SERVICE
;
INT 7AH
.
.
.
```

Supervisory Object Service X'81': Name Resolution

Use this service to resolve the specified supervisory object name to its numeric index.

Register Values

On Request

AH = X'81'
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

BH = Object type
CH = Function ID
CL = Return code
DX = Resolved name

The contents of registers AX, BL, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The ES register contains the segment address of the parameter list.
- The DI register contains the offset address of the parameter list.

Completion Registers:

- The BH register indicates the type of the specified supervisory object. Possible supervisory object types are as follows:
 - X'00' – Task
 - X'01' – Component
 - X'03' – Code serialization semaphore
 - X'04' – Resource semaphore
 - X'05' – Fixed-length queue
 - X'06' – User exit table
 - X'07' – Gate.
- The DX register contains the supervisory object ID of the resolved name, which is the numeric representation of the alphanumeric supervisory object name.

Name Resolution

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0 - 7	8 bytes	Supervisory object name	Unchanged

Parameter Definitions

Request Parameters:

The supervisory object name must be ASCII characters and also be padded to the right with blanks if it is less than eight characters long.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received are:

Code	Meaning
X'00'	Successful completion of the request.
X'0F'	The specified object is in an inaccessible environment.
X'2E'	The name is not found.

Usage Notes

This service does not allow names to be resolved across stoppable environments or allow code serialization semaphore names to be resolved from a stoppable environment to a nonstoppable environment. Also, the name of a component in a stoppable environment may never be resolved outside that environment.

Coding Example

```
;
; PARAMETER LIST FOR NAME RESOLUTION
;
SERVNAME  DB 'KEYBOARD'

          .
          .
          .

;
; INITIALIZE REGISTERS FOR NAME RESOLUTION
;
          MOV     AH,81H           ; AH = X'81'
          MOV     DI,SEG SERVNAME  ; SEGMENT ADDRESS OF PARM LIST
          MOV     ES,DI           ; ES = SEGMENT ADDRESS OF PARM LIST
          MOV     DI,OFFSET SERVNAME ; DI = OFFSET ADDR. OF PARM LIST

;
; SIGNAL WORKSTATION PROGRAM FOR NAME RESOLUTION SERVICE
;
          INT     7AH
          .
          .
          .
```


Supervisory Object Service X'01': ID Resolution

Use this service to resolve the specified supervisory object ID to its alphanumeric name.

Register Values

On Request

AH = X'01'
DX = Supervisory object ID
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = Function ID
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The DX register contains the ID of the supervisory object.
- The ES register contains the segment address of the parameter list.
- The DI register contains the offset address of the parameter list.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0 - 7	8 bytes	Reserved	Supervisory object name

Parameter Definitions

Completion Parameters:

- The supervisory object name is the alphanumeric name assigned to the specified supervisory object when an entry for it was created in the SVC table. The supervisory object name can be a maximum of eight characters and is padded to the right with blanks if necessary.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received are:

Code	Meaning
X'00'	Successful completion of the request.
X'05'	Invalid supervisory object ID.
X'0F'	The specified object is in an inaccessible environment.
X'2E'	The ID was not found in the SVC table.

Usage Notes

A task or component can resolve the ID of a supervisory object either within its own environment or within a nonstopable environment.

Coding Example

```
      .  
      .  
      .  
;   
; INITIALIZE REGISTERS FOR ID RESOLUTION  
;  
      MOV     AH,01H  
      MOV     DX,OBJECTID      ; OBJECT ID IN DX  
      MOV     DI, SEG OBJNAME  ; SEGMENT ADDRESS OF PARAMETER LIST  
      MOV     ES,DI            ; IN ES  
      MOV     DI,OFFSET OBJNAME ; OFFSET OF PARAMETER LIST IN DI  
;  
; SIGNAL WORKSTATION PROGRAM FOR ID RESOLUTION SERVICE  
;  
      INT     7AH  
      .  
      .  
      .
```

Delete Entry

Supervisory Object Service X'06': Delete Entry

Use this service to delete the entry in the SVC table representing the specified supervisory object.

Register Values

On Request

AH = X'06'
DX = Supervisory object ID

On Completion

CH = Function ID
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Register Definitions

Request Registers:

The DX register contains the ID of the supervisory object to be deleted from the SVC table. This object can only be a task, component, fixed-length queue, or semaphore within the requester's environment.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received are:

Code	Meaning
X'00'	Successful completion of the request.
X'05'	Invalid supervisory object ID.
X'0F'	The specified object is in an inaccessible environment.
X'30'	Cannot delete a task, fixed-length queue, or semaphore that has pending requests.
X'31'	Cannot delete a task that has timers.
X'3F'	Cannot delete a gate.

Usage Notes

- If the entry to be removed is a task, semaphore, or fixed-length queue and there are outstanding requests for the entry, then the entry will not be removed and an error indicator will be returned.
- An application program running in a stoppable environment can only delete entries in its own environment.

Coding Example

```
      .  
      .  
      .  
;   
; INITIALIZE REGISTERS FOR DELETE AN ENTRY REQUEST  
;  
      MOV    AH,06H  
      MOV    DX,QUE$ID      ; DX = FIXED LENGTH QUEUE ID  
;  
; SIGNAL WORKSTATION PROGRAM FOR DELETE AN ENTRY SERVICE  
;  
      INT    7AH  
      .  
      .  
      .
```


Chapter 16. Coding Request Services

Introduction 16-2

 Requesting the Request Services 16-2

 Return Codes for the Request Services 16-2

Request Service X'09': Make a Request 16-3

Request Service X'96': Get a Request 16-8

Request Service X'82': Reply to a Request 16-11

Request Service X'83': Get Request Completion 16-14

Request Service X'12': Send a Signal to a Task 16-17

Introduction

This chapter describes how to code requests for request services provided by the API.

The request services allow your application program to write tasks that request services of other tasks, and tasks that respond to requests from other tasks.

The request services provided by the API are:

- **Make a Request:** Use this service to put a request queue element on the specified task's request queue, or to directly invoke a component.
- **Get a Request:** Use this service to obtain the contents of a request queue element on a task's request queue.
- **Reply to a Request:** Use this service to remove a specified request queue element from a task's request queue and to send the specified reply to the requester.
- **Get Request Completion:** Use this service to obtain the contents of a request queue element from a task's completion queue.
- **Send a Signal to a Task:** Use this service to send a signal to the specified task.

Requesting the Request Services

To use any of the request services, load the registers and the parameter list with the proper values, and use the INT 7AH instruction to signal the workstation program that it has a request to process.

Return Codes for the Request Services

Return codes for the request services are 2-byte values made up of a function ID and an error code. The function ID indicates the portion of the workstation program in which the error occurred. The error number indicates the specific type of error that has occurred. An error number of X'00' indicates a successful acceptance or completion of the request.

After your application has requested a request service, the CH and CL registers contain a return code generated by the request processing portion of the workstation program. The function ID is in the CH register, and the error number is in the CL register. The return codes that can be generated by request services are called *system return codes*. The function ID for system return codes is X'12' or X'13'. The error numbers that can appear are specific to the service that was requested and are included in the descriptions of each service.

See Appendix H, "Return Codes," for more information.

Request Service X'09': Make a Request

Use this service to put a request queue element (RQE) on the specified task's request queue, or to directly invoke a component.

Register Values

On Request

AH = X'09'
AL = Service number
BH = Reply type
BL = Wait type
CX = Request priority
DX = Task, component, or gate ID
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

AX = Request ID
BL = Return type
CH = Function ID
CL = Return code

The contents of registers BH, DX, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The AL register contains the service entry number of the requested service, if the request is being made to a gate. This register is not used on input if the request is being used to directly invoke a component or a task.
- The BH register specifies the type of reply your application program receives when the request is completed. Possible reply types are as follows:

X'80' Request completion is indicated by a 'completion' signal. Any existing 'completion' signal to the application program is canceled.

X'40' Request completion is indicated by an RQE on the application program's completion queue.

X'20' No notification of request completion is received.

X'10' No notification of request completion is received, and the parameter list is copied into a 10-byte area so that the parameter list data area can be reused. This is intended for interrupt handler use.

Make a Request

- The BL register specifies the type of wait state your application program will go into until the request is completed. The type of wait is specified through a bit mask. When more than one type of wait is specified, the wait state ends when any one of the conditions is satisfied. The bits in the wait type mask are as follows:

0	1	2	3	4	5	6	7
Request queue	Comp queue	Comp signal	Sema-phore	Timer	Signal	Data	Reserved

- If bit 0 is set to 1, your application program waits until there is a request queue element in its request queue. If there is already an RQE in its request queue, the application stays dispatchable.
- If bit 1 is set to 1, your application program waits until there is a request queue element in its completion queue. If there is already an RQE in its completion queue, the application stays dispatchable.
- If bit 2 is set to 1, your application program waits until it receives a ‘completion’ signal.
- If bit 3 is set to 1, your application program waits until it receives a ‘semaphore claimed’ signal.
- If bit 4 is set to 1, your application program waits until it receives a ‘timer tick’ signal.
- If bit 5 is set to 1, your application program waits until it receives a ‘generic’ signal.
- If bit 6 is set to 1, your application program waits until it receives a ‘data available’ signal.
- Bit 7 is reserved and must be set to 0.

Note: X'00' specifies “no wait.” A “wait” for semaphore or data is inappropriate for this service.

- The CX register indicates the priority of the request. Valid request priorities are 0 through 255, with 0 being the highest priority.
- The DX register indicates the ID of the task to which the request is to be sent, or the ID of the component being directly invoked, or a gate ID through which a request is being made.
- The ES register contains the segment address of the parameter list.
- The DI register contains the offset address of the parameter list.

Completion Registers:

- The AX register contains the ID of the RQE that was placed on the requested task's request queue. You can use this request ID to match outstanding requests with incoming completion queue elements.
- The BL register indicates the type of wait condition that was satisfied to return control to your application program. The return type is specified through a bit mask. The bits in the return type mask have the same meaning as the bits in the wait type mask.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID
⋮			

You determine the format of the remainder of the parameter list on the basis of the needs of your particular application.

Parameter Definitions

Request Parameters:

- Bytes 0 and 1 of the parameter must be X'00' on request.
- You determine the format of the remainder of the parameter list on the basis of the needs of your particular application program.

Completion Parameters:

- Byte 1 of the parameter list contains a 2-digit function ID that indicates the portion of the requested task that processed the request.
- Byte 0 of the parameter list contains a 2-digit error number that indicates what type of error (if any) caused request processing to fail. An error number of zero is used to indicate successful completion of the request.
- You determine the format of the remainder of the parameter list on the basis of the needs of your particular application program.

Make a Request

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received are:

Code	Meaning
X'00' *	Successful completion of the request.
X'05'	Invalid index specified.
X'07'	Invalid reply type specified.
X'08'	Invalid wait type specified.
X'0B'	System RQE pool depleted.
X'34'	Invalid service number specified.

* When the error number returned is X'00', you must also check the return code in the parameter list to be sure that return code X'1314' is not returned in the parameter list. See the note below for more information.

You can use bytes 0 and 1 of the parameter list for return codes and function IDs from the requested task or component. You determine the return codes and function IDs that pertain to the request on the basis of the needs of your particular application program.

Note: Your application may have used the Make a Request service to send a request to another task in an environment that is stopped, reset, or deleted before the request could be acted upon. In this case, bytes 0 and 1 of the parameter list will contain return code X'1314'. This return code indicates that the request cannot be completed.

Usage Notes

- This service cannot be requested between stoppable environments.
- If you are doing a Make Request with a parameter list, the first two bytes should be used as a return code function code field.
- Generally, a task doing asynchronous processing should specify a reply of 'completion queue' signal. Tasks doing synchronous processing can specify a reply of 'completion' signal or a 'completion queue' signal.

Coding Example

```
      .  
      .  
      .  
;   
; INITIALIZE REGISTERS FOR MAKE A REQUEST  
;  
      MOV     AH,09H  
      MOV     BH,80H           ; REPLY TYPE = COMPLETION SIGNAL  
      MOV     BL,20H           ; WAIT TYPE = COMPLETION SIGNAL  
      MOV     CX,60            ; PRIORITY = 60  
      MOV     DX,TASKID        ; TASK OR COMPONENT ID IN DX  
      MOV     DI, SEG PARMLIST ; SEGMENT ADDRESS OF PARAMETER LIST  
      MOV     ES,DI            ; IN ES  
      MOV     DI,OFFSET PARMLIST ; OFFSET OF PARAMETER LIST IN DI  
;  
; SIGNAL WORKSTATION PROGRAM FOR MAKE A REQUEST SERVICE  
;  
      INT     7AH  
      .  
      .  
      .
```

Request Service X'96': Get a Request

Use this service to obtain the contents of a request queue element (RQE) on a task's request queue.

Register Values

On Request

AH = X'96'
BL = Wait type

On Completion

AX = Request ID
BL = Return type
CH = Function ID
CL = Return code
DX = Task ID
ES = Segment address of
the parameter list
DI = Offset address of
the parameter list

The contents of register
BH are unpredictable.

Register Definitions

Request Registers:

- The BL register specifies the type of wait state your application program goes into until the request is completed. The type of wait is specified through a bit mask. If more than one type of wait is specified, the wait state ends when any one of the conditions is satisfied. The bits in the wait type mask are as follows:

0	1	2	3	4	5	6	7
Request queue	Comp queue	Comp signal	Sema-phore	Timer	Signal	Data	Reserved

- If bit 0 is set to 1, your application program waits until there is a request queue element in its request queue. If there is already an RQE in its request queue, the application stays dispatchable.
- If bit 1 is set to 1, your application program waits until there is a request queue element in its completion queue. If there is already an RQE in its completion queue, the application stays dispatchable.
- If bit 2 is set to 1, your application program waits until it receives a 'completion' signal.
- If bit 3 is set to 1, your application program waits until it receives a 'semaphore claimed' signal.

- If bit 4 is set to 1, your application program waits until it receives a ‘timer tick’ signal.
- If bit 5 is set to 1, your application program waits until it receives a ‘generic’ signal.
- If bit 6 is set to 1, your application program waits until it receives a ‘data available’ signal.
- Bit 7 is reserved and must be set to 0.

Note: X'00' specifies “no wait.” A “wait” for semaphore or data is inappropriate for this service.

Completion Registers:

- The AX register contains the ID of the RQE that was returned.
- The BL register indicates the type of wait condition that was satisfied to return control to the requesting task. The return type is specified through a bit mask. The bits in the return type mask have the same meaning as the bits in the wait type mask.
- The DX register contains the ID of the task that made the request.
- The ES register contains the segment address of the parameter list.
- The DI register contains the offset address of the parameter list.

Parameter List Format

You define the format of the parameter list on the basis of the needs of your particular application program.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received are:

Code	Meaning
X'00'	Successful completion of the request.
X'09'	The request queue is empty.
X'36'	No request queue elements were on the request queue.

Get a Request

Usage Notes

- If the request queue is empty and a wait type of “no wait” is specified, the request ID in the AX register on completion is set to zero and a “request queue empty” is received in the CL register.
- If the request queue is empty and a wait type other than “no wait” is specified, the task is set to the specified wait state until the request is satisfied.

Coding Example

```
      .  
      .  
      .  
;   
; INITIALIZE REGISTERS FOR GET A REQUEST  
;  
      MOV    AH,96H  
      MOV    BL,80H           ; WAIT TYPE = REQUEST QUEUE  
;  
; SIGNAL WORKSTATION PROGRAM FOR GET A REQUEST SERVICE  
;  
      INT    7AH  
      .  
      .  
      .
```

Request Service X'82': Reply to a Request

Use this service to remove a specified request queue element (RQE) from a task's request queue and send the specified reply to the requester. The process of removing the RQE and sending the reply is called *completing the request*.

Register Values

On Request

AH = X'82'
BL = Wait type
DX = RQE ID

On Completion

BL = Return type
CH = Function ID
CL = Return code

The contents of registers AX, BH, DX, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The BL register specifies the type of wait state your application program goes into until the request is completed. The type of wait is specified through a bit mask. If more than one type of wait is specified, the wait state ends when any one of the conditions is satisfied. The bits in the wait type mask are as follows:

0	1	2	3	4	5	6	7
Request queue	Comp queue	Comp signal	Sema-phore	Timer	Signal	Data	Reserved

- If bit 0 is set to 1, your application program waits until there is a request queue element in its request queue. If there is already an RQE in its request queue, the application stays dispatchable.
- If bit 1 is set to 1, your application program waits until there is a request queue element in its completion queue. If there is already an RQE in its completion queue, the application stays dispatchable.
- If bit 2 is set to 1, your application program waits until it receives a 'completion' signal.
- If bit 3 is set to 1, your application program waits until it receives a 'semaphore claimed' signal.
- If bit 4 is set to 1, your application program waits until it receives a 'timer tick' signal.

Reply to a Request

- If bit 5 is set to 1, your application program waits until it receives a ‘generic’ signal.
- If bit 6 is set to 1, your application program waits until it receives a ‘data available’ signal.
- Bit 7 is reserved and must be set to 0.

Note: X'00' specifies “no wait.” A “wait” for semaphore or data is inappropriate for this service.

- The DX register specifies the ID of the request queue element to be completed. A value of X'0000' indicates that the first element on the request queue is to be completed.

Completion Registers:

- The BL register indicates the type of wait condition that was satisfied to return control to the requesting task. The return type is specified through a bit mask. The bits in the return type mask have the same meaning as the bits in the wait type mask.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received are:

Code	Meaning
X'00'	Successful completion of the request
X'09'	The request queue is empty
X'36'	The specified request queue element was not on request queue.

Usage Notes

- The specified RQE is removed from the task's request queue.
- If the request was issued with a reply type of X'40' (“completion queue”), the reply is added to the requesting task's completion queue.
- If the request was issued with a reply type of X'80' (“completion’ signal”), the requesting task is sent a ‘completion’ signal. The RQE is returned to the system RQE pool.
- If the request was issued with a reply type of X'20' or X'10' (“none” or “none—copy parameter list”), no reply is sent to the requesting task. The RQE is returned to the system RQE pool.

Coding Example

```
      .  
      .  
      .  
;   
; INITIALIZE REGISTERS FOR REPLY TO A REQUEST  
;  
      MOV     AH,82H  
      MOV     BL,00H           ; WAIT TYPE = NO WAIT  
      MOV     DX,0             ; 1ST RQE ON QUEUE  
;  
; SIGNAL WORKSTATION PROGRAM FOR REPLY TO A REQUEST SERVICE  
;  
      INT     7AH  
      .  
      .  
      .
```

Request Service X'83': Get Request Completion

Use this service to obtain the contents of a request queue element (RQE) from a task's completion queue.

Register Values

On Request

AH = X'83'
BL = Wait type

On Completion

AX = RQE ID
BL = Return type
CH = Function ID
CL = Return code
ES = Segment address of the parameter list
DI = Offset address of the parameter list

The contents of registers BH and DX are unpredictable.

Register Definitions

Request Registers:

- The BL register specifies the type of wait state your application program goes into until the request is completed. The type of wait is specified through a bit mask. If more than one type of wait is specified, the wait state ends when any one of the conditions is satisfied. The bits in the wait type mask are as follows:

0	1	2	3	4	5	6	7
Request queue	Comp queue	Comp signal	Sema-phore	Timer	Signal	Data	Reserved

- If bit 0 is set to 1, your application program waits until there is a request queue element in its request queue. If there is already an RQE in its request queue, the application stays dispatchable.
- If bit 1 is set to 1, your application program waits until there is a request queue element in its completion queue. If there is already an RQE in its completion queue, the application stays dispatchable.
- If bit 2 is set to 1, your application program waits until it receives a 'completion' signal.
- If bit 3 is set to 1, your application program waits until it receives a 'semaphore claimed' signal.

- If bit 4 is set to 1, your application program waits until it receives a ‘timer tick’ signal.
- If bit 5 is set to 1, your application program waits until it receives a ‘generic’ signal.
- If bit 6 is set to 1, your application program waits until it receives a ‘data available’ signal.
- Bit 7 is reserved and must be set to 0.

Note: X'00' specifies “no wait.” A “wait” for semaphore or data is inappropriate for this service.

Completion Registers:

- The AX register contains the ID of the RQE that was returned.
- The BL register indicates the type of wait condition that was satisfied to return control to the requesting task. The return type is specified through a bit mask. The bits in the return type mask have the same meaning as the bits in the wait type mask.
- The ES register contains the segment address of the parameter list.
- The DI register contains the offset address of the parameter list.

Parameter List Format

You define the format of the parameter list on the basis of the needs of your particular application program.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received are:

Code	Meaning
X'00'	Successful completion of the request
X'09'	The completion queue is empty.

Usage Notes

- If the request queue is empty and a wait type other than “no wait” is specified, the task is set to the specified wait state until the request is satisfied.

Get Request Completion

Coding Example

```
      .  
      .  
      .  
;     ;  
; INITIALIZE REGISTERS FOR GET REQUEST COMPLETION  
;  
      MOV    AH,83H  
      MOV    BL,40H           ; WAIT TYPE = COMPLETION QUEUE  
  
      ; SIGNAL WORKSTATION PROGRAM FOR GET REQUEST COMPLETION SERVICE  
      INT    7AH  
      .  
      .  
      .
```

Request Service X'12': Send a Signal to a Task

Use this service to send a signal to the specified task.

Register Values

On Request

AH = X'12'
BL = Wait type
DX = Task ID or X'0000'

On Completion

BL = Return type
CH = Function ID
CL = Return code

The contents of registers
AX, BH, DX, ES, and DI
are unpredictable.

Register Definitions

Request Registers:

- The BL register specifies the type of wait state your application program goes into until the request is completed. The type of wait is specified through a bit mask. If more than one type of wait is specified, the wait state ends when any one of the conditions is satisfied. The bits in the wait type mask are as follows:

0	1	2	3	4	5	6	7
Request queue	Comp queue	Comp signal	Sema-phore	Timer	Signal	Data	Reserved

- If bit 0 is set to 1, your application program waits until there is a request queue element in its request queue. If there is already an RQE in its request queue, the application stays dispatchable.
- If bit 1 is set to 1, your application program waits until there is a request queue element in its completion queue. If there is already an RQE in its completion queue, the application stays dispatchable.
- If bit 2 is set to 1, your application program waits until it receives a 'completion' signal.
- If bit 3 is set to 1, your application program waits until it receives a 'semaphore claimed' signal.
- If bit 4 is set to 1, your application program waits until it receives a 'timer tick' signal.
- If bit 5 is set to 1, your application program waits until it receives a 'generic' signal.

Send a Signal to a Task

- If bit 6 is set to 1, your application program waits until it receives a 'data available' signal.
- Bit 7 is reserved and must be set to 0.

Note: X'00' specifies "no wait." A "wait" for semaphore or data is inappropriate for this service.

- The DX register contains the ID of the task to be signaled. A value of X'0000' indicates that the currently active task should be sent the signal. (This results in your application program's sending a signal to itself.)

Completion Registers:

- The BL register indicates the type of wait condition that was satisfied to return control to the requesting task. The return type is specified via a bit mask. The bits in the return type mask have the same meaning as the bits in the wait type mask.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received are:

Code	Meaning
X'00'	Successful completion of the request.
X'05'	An invalid index was specified.

Usage Notes

A program running in a stoppable environment may send a signal only to tasks within its own environment.

Coding Example

```

      .
      .
      .
;
; INITIALIZE REGISTERS FOR SEND A SIGNAL TO A TASK
;
      MOV     AH,12H
      MOV     BL,04H           ; WAIT TYPE = SIGNAL
      MOV     DX,TASK$ID       ; TASK ID IN DX
;
; SIGNAL WORKSTATION PROGRAM FOR SEND A SIGNAL TO A TASK SERVICE
;
      INT     7AH
      .
      .
      .
```

Chapter 17. Coding Task State Modifier Services

Introduction	17-2
Requesting the Task State Modifier Services	17-2
Return Codes for the Task State Modifier Services	17-2
Task State Modifier Service X'9C': Query Active Task	17-3
Task State Modifier Service X'02': Set Task "Ready"	17-4
Task State Modifier Service X'05': Set Task "Unready"	17-7
Task State Modifier Service X'08': Set Task "Preemptable"	17-10
Task State Modifier Service X'07': Set Task "Nonpreemptable"	17-12
Task State Modifier Service X'03': Change Task's Priority	17-14
Task State Modifier Service X'95': Return to Dispatcher	17-16

Introduction

This chapter describes how to code requests for the task state modifier services provided by the API.

The task state modifier services allow your application program to change the dispatch state or priority of a task.

The task state modifier services provided by the API are:

- **Query Active Task:** Use this service to obtain the ID and priority of the currently active task.
- **Set Task “Ready”:** Use this service to set a specified task to the “ready” state.
- **Set Task “Unready”:** Use this service to set a specified task to the “unready” state.
- **Set Task “Preemptable”:** Use this service to set a specified task to the “preemptable” state.
- **Set Task “Nonpreemptable”:** Use this service to set a specified task to the “nonpreemptable” state.
- **Change Task’s Priority:** Use this service to change the specified task’s priority.
- **Return to Dispatcher:** Use this service to return to the dispatcher from the requesting task.

Requesting the Task State Modifier Services

To request any of the task state modifier services, load the registers and the parameter list with the proper values, and use the INT 7AH instruction to signal the workstation program that it has a request to process.

Return Codes for the Task State Modifier Services

Return codes for the Task State Modifier services are 2-byte values made up of a function ID and an error code. The function ID indicates the portion of the workstation program in which the error occurred. The code number indicates the specific type of error that has occurred. A code number of X'00' always indicates a successful acceptance or completion of the request.

After your application has requested a task state modifier service, the CH and CL registers contain a return code generated by the request processing portion of the workstation program. The function ID is in the CH register, and the error code is in the CL register. The return codes that can be generated by semaphore management services are called *system return codes*. The function ID for system return codes is X'12' or X'13'. The error codes that can appear are specific to the service that was requested and are included in the descriptions of each service.

See Appendix H, “Return Codes,” for more information.

Task State Modifier Service X'9C': Query Active Task

Use this service to obtain the ID and priority of the currently active task.

On Request

AH = X'9C'

On Completion

AL = Priority
CH = Function ID
CL = Return code
DX = Task ID

The contents of registers AH, BX, ES, and DI are unpredictable.

Register Definitions

Completion Registers:

- The AL register contains the priority of the currently active task.
- The DX register contains the ID of the currently active task.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' (found in the CH register). The system error code that can be received for this service is:

Code	Meaning
X'00'	Successful completion of the request.

Coding Example

```
      .  
      .  
      .  
;     ;  
; INITIALIZE REGISTERS FOR QUERY TASK  
;     ;  
      MOV     AH,9CH  
;     ;  
; SIGNAL WORKSTATION PROGRAM FOR QUERY TASK SERVICE  
;     ;  
      INT     7AH  
      .  
      .  
      .
```

Set Task “Ready”

Task State Modifier Service X'02': Set Task “Ready”

Use this service to set a specified task to the “ready” state.

Register Values

On Request

AH = X'02'
BL = Wait type
DX = Task ID

On Completion

BL = Return type
CH = Function ID
CL = Return code

The contents of registers
AX, BH, DX, ES, and DI
are unpredictable.

Register Definitions

Request Registers:

- The BL register specifies the type of wait state your application program goes into until the request is completed. The type of wait is specified through a bit mask. If more than one type of wait is specified, the wait state ends when any one of the conditions is satisfied. The bits in the wait type mask are as follows:

0	1	2	3	4	5	6	7
Request queue	Comp queue	Comp signal	Sema-phore	Timer	Signal	Data	Reserved

- If bit 0 is set to 1, your application program waits until there is a request queue element in its request queue. If there is already an RQE in its request queue, the application stays dispatchable.
- If bit 1 is set to 1, your application program waits until there is a request queue element in its completion queue. If there is already an RQE in its completion queue, the application stays dispatchable.
- If bit 2 is set to 1, your application program waits until it receives a ‘completion’ signal.
- If bit 3 is set to 1, your application program waits until it receives a ‘got semaphore’ signal.
- If bit 4 is set to 1, your application program waits until it receives a ‘timer tick’ signal.

•

- If bit 5 is set to 1, your application program waits until it receives a ‘generic’ signal.
- If bit 6 is set to 1, your application program waits until it receives a ‘data available’ signal.
- Bit 7 is reserved and must be set to 0.

Note: X'00' specifies “no wait.” A “wait” for semaphore or data is inappropriate for this service.

- The DX register contains the ID of the task to be set to the “ready” state.

Completion Registers:

- The BL register indicates the type of wait condition that was satisfied to return control to your application program. The return type is specified via a bit mask. The bits in the return type have the same meaning as the bits in the wait type mask.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion of the request.
X'05'	An invalid index was specified.
X'0F'	Invalid environment access.

Usage Notes

- A program running in a stoppable environment cannot set tasks in other environments to the ready state.
- The specified task is removed from the “unready” or “pending unready” state.
- The specified task is made dispatchable if it is not waiting and not suspended.

Set Task "Ready"

Coding Example

```
      .  
      .  
      .  
;     ;  
; INITIALIZE REGISTERS FOR SET TASK "READY"  
;     ;  
      MOV     AH,02H  
      MOV     BL,04H           ; WAIT TYPE = SIGNAL  
      MOV     DX,TASKID       ; TASK ID IN DX  
;     ;  
; SIGNAL WORKSTATION PROGRAM FOR SET TASK "READY" SERVICE  
;     ;  
      INT 7AH  
      .  
      .  
      .
```

Task State Modifier Service X'05': Set Task “Unready”

Use this service to set a specified task to the “unready” state.

Register Values

On Request

AH = X'05'
BL = Wait type
DX = Task ID or X'0000'

On Completion

BL = Return type
CH = Function ID
CL = Return code

The contents of registers AX, BH, DX, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The BL register specifies the type of wait state your application program goes into until the request is completed. The type of wait is specified via a bit mask. If more than one type of wait is specified, the wait state ends when any one of the conditions is satisfied. The bits in the wait type mask are as follows:

0	1	2	3	4	5	6	7
Request queue	Comp queue	Comp signal	Sema-phore	Timer	Signal	Data	Reserved

- If bit 0 is set to 1, your application program waits until there is a request queue element in its request queue. If there is already an RQE in its request queue, the application stays dispatchable.
- If bit 1 is set to 1, your application program waits until there is a request queue element in its completion queue. If there is already an RQE in its completion queue, the application stays dispatchable.
- If bit 2 is set to 1, your application program waits until it receives a ‘completion’ signal.
- If bit 3 is set to 1, your application program waits until it receives a ‘got semaphore’ signal.
- If bit 4 is set to 1, your application program waits until it receives a ‘timer tick’ signal.

Set Task “Unready”

- If bit 5 is set to 1, your application program waits until it receives a ‘generic’ signal.
- If bit 6 is set to 1, your application program waits until it receives a ‘data available’ signal.
- Bit 7 is reserved and must be set to 0.

Note: X’00’ specifies “no wait.” A “wait” for semaphore or data is inappropriate for this service.

- The DX register contains the ID of the task to be set to the “unready” state. If the value of the DX register is X’0000’, the workstation program uses the ID of the currently executing task.

Completion Registers:

- The BL register indicates the type of wait condition that was satisfied to return control to your application program. The return type is specified via a bit mask. The bits in the return type have the same meaning as the bits in the wait type mask.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X’12’ or X’13’ (found in the CH register). The error codes that can be received for this service are:

Code	Meaning
X’00’	Successful completion of the request.
X’05’	An invalid index was specified.
X’0F’	Invalid environment access.

Usage Notes

- A program running in a stoppable environment cannot set tasks in other environments to the unready state.
- If the specified task is already in the “unready” or “pending unready” state, there is no change.
- If the specified task does not own any code serialization semaphores, it is set to the “unready” state. In the “unready” state, the task is not dispatchable.
- If the specified task owns one or more code serialization semaphores, it is set to the “pending unready” state. In the “pending unready” state, dispatching is not affected. Once the task’s code serialization semaphores are released, the task is set to the “unready” state.
- If the specified task is the currently active task, the dispatcher takes control as soon as the task sets itself unready.

Coding Example

```
      .  
      .  
      .  
;     ;  
; INITIALIZE REGISTERS FOR SET TASK "UNREADY"  
;     ;  
      MOV     AH,05H  
      MOV     BL,20H           ; WAIT TYPE = COMPLETION SIGNAL  
      MOV     DX,TASKID       ; TASK ID IN DX  
;     ;  
; SIGNAL WORKSTATION PROGRAM FOR SET TASK "UNREADY" SERVICE  
;     ;  
      INT 7AH  
      .  
      .  
      .
```


Set Task “Preemptable”

Task State Modifier Service X'08': Set Task “Preemptable”

Use this service to set a specified task to the “preemptable” state.

Register Values

On Request

AH = X'08'
DX = Task ID or X'0000'

On Completion

CH = Function ID
CL = Return code

The contents of registers
AX, BX, DX, ES, and DI
are unpredictable.

Register Definitions

Request Registers:

- The DX register contains the ID of the task to be set to preemptable. If the value of the DX register is X'0000', the workstation program uses the ID of the currently executing task.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The system return codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion of the request.
X'05'	An invalid index specified.
X'0F'	Invalid environment access.

Usage Notes

- A program running in a stoppable environment cannot set tasks in other environments to the preemptable state.
- The preemptable state takes effect when the dispatcher makes the specified task the active task. The dispatcher selects the task to become active using the rules described at the beginning of this chapter. The task becomes preemptable until it goes into a wait state or becomes “unready.” When the task becomes active again, it will again become preemptable.

Coding Example

```
      .  
      .  
;     .  
; INITIALIZE REGISTERS FOR SET TASK "PREEMPTABLE"  
;  
      MOV     AH,08H  
      MOV     DX,TASKID           ; TASK ID IN DX  
;  
; SIGNAL WORKSTATION PROGRAM FOR SET TASK "PREEMPTABLE" SERVICE  
;  
      INT 7AH  
      .  
      .  
      .
```

Set Task “Nonpreemptable”

Task State Modifier Service X'07': Set Task “Nonpreemptable”

Use this service to set a specified task to the “nonpreemptable” state. The task can be set either “nonpreemptable within priority” or “nonpreemptable within environment.”

Register Values

On Request

AH = X'07'
BH = Nonpreemptable type
DX = Task ID or X'0000'

On Completion

CH = Function ID
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The BH register indicates the nonpreemptable type, which is specified as follows:

X'01' = Nonpreemptable within priority
X'02' = Nonpreemptable within environment
- The DX register contains the ID of the task to be set to the “nonpreemptable” state. If the value of the DX register is X'0000', the workstation program uses the ID of the currently executing task.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion of the request.
X'05'	An invalid index was specified.
X'0A'	Invalid nonpreemptable state.
X'0F'	Invalid environment access.

Usage Notes

- A program running in a stoppable environment cannot set tasks in other environments to the nonpreemptable state.
- The nonpreemptable state takes effect when the dispatcher makes the specified task the active task. The dispatcher selects the task to become active using the rules described at the beginning of this chapter. The task becomes nonpreemptable until it goes into a wait state or becomes “unready.” When the task becomes active again, it again becomes nonpreemptable.

Coding Example

```
      .  
      .  
      .  
;   
; INITIALIZE REGISTERS FOR SET TASK "NONPREEMPTABLE"  
;  
      MOV     AH,07H  
      MOV     BH,01H           ; NONPREEMPTABLE WITHIN PRIORITY  
      MOV     DX,TASKID       ; TASK ID IN DX  
;  
; SIGNAL WORKSTATION PROGRAM FOR SET TASK "NONPREEMPTABLE" SERVICE  
;  
      INT 7AH  
      .  
      .  
      .
```

Task State Modifier Service X'03': Change Task's Priority

Use this service to change the specified task's priority. The task's priority can be set within the range of 36 through 64.

Register Values

On Request

AH = X'03'
CX = Task priority
DX = Task ID or X'0000'

On Completion

CH = Function ID
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The CX register contains the specified task's priority. The priority must be in the range of 36 through 64.
- The DX register contains the ID of the task to be set to the specified priority. If the DX register contains X'0000', the supervisor uses the ID of the currently running task.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion of the request.
X'05'	An invalid index was specified.
X'06'	An invalid priority was specified.
X'0F'	Invalid environment access.

Usage Notes

- A program running in a stoppable environment cannot change the priority of tasks in other environments.
- The new task priority takes effect. The nonpreemptable state takes effect when the dispatcher makes the specified task the active task. The dispatcher selects the task to become active using the rules described at the beginning of this chapter.

Coding Example

```
      .  
      .  
      .  
;   
; INITIALIZE REGISTERS FOR CHANGE TASK'S PRIORITY  
;  
      MOV     AH,03H  
      MOV     CX,PRIORITY      ; TASK PRIORITY IN CX  
      MOV     DX,TASKID        ; TASK ID IN DX  
;  
; SIGNAL WORKSTATION PROGRAM FOR CHANGE TASK'S PRIORITY SERVICE  
;  
      INT 7AH  
      .  
      .  
      .
```

Task State Modifier Service X'95': Return to Dispatcher

Use this service to return to the dispatcher from the requesting task.

Register Values

On Request

AH = X'95'
BL = Wait type

On Completion

BL = Return type
CH = Function ID
CL = Return code

The contents of registers
AX, BH, DX, ES, and DI
are unpredictable.

Register Definitions

Request Registers:

- The BL register specifies the type of wait state your application program goes into until the request is completed. The type of wait is specified through a bit mask. If more than one type of wait is specified, the wait state ends when any one of the conditions is satisfied. The bits in the wait type mask are as follows:

0	1	2	3	4	5	6	7
Request queue	Comp queue	Comp signal	Sema-phore	Timer	Signal	Data	Reserved

- If bit 0 is set to 1, your application program waits until there is a request queue element in its request queue. If there is already an RQE in its request queue, the application stays dispatchable.
- If bit 1 is set to 1, your application program waits until there is a request queue element in its completion queue. If there is already an RQE in its completion queue, the application stays dispatchable.
- If bit 2 is set to 1, your application program waits until it receives a 'completion' signal.
- If bit 3 is set to 1, your application program waits until it receives a 'got semaphore' signal.
- If bit 4 is set to 1, your application program waits until it receives a 'timer tick' signal.

- If bit 5 is set to 1, your application program waits until it receives a ‘generic’ signal.
- If bit 6 is set to 1, your application program waits until it receives a ‘data available’ signal.
- Bit 7 is reserved and must be set to 0.

Note: X'00' specifies “no wait.” A “wait” for semaphore or data is inappropriate for this service.

Completion Registers:

- The BL register indicates the type of wait condition that was satisfied to return control to the requesting task. The return type is specified via a bit mask. The bits in the return type have the same meaning as the bits in the wait type mask.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error code that can be received for this service is:

Code	Meaning
X'00'	Successful completion of the request.

Coding Example

```
.  
. .  
;  
; INITIALIZE REGISTERS FOR RETURN TO DISPATCHER  
;  
    MOV    AH,95H  
    MOV    BL,04H           ; WAIT TYPE = SIGNAL  
;  
; SIGNAL WORKSTATION PROGRAM FOR RETURN TO DISPATCHER SERVICE  
;  
    INT    7AH  
.  
.  
.
```


Chapter 18. Coding Semaphore Management Services

Introduction 18-2

 Requesting the Semaphore Management Services 18-2

 Return Codes for the Semaphore Management Services 18-2

Semaphore Management Service X'0D': Claim a Semaphore 18-3

Semaphore Management Service X'0A': Release a Semaphore 18-6

Semaphore Management Service X'0B': Query a Semaphore 18-8

Introduction

This chapter describes how to code requests for the semaphore management request services provided by the API.

The semaphore management services allow your application program to control access to resources and the execution of nonreentrant code.

The semaphore management services provided by the API are:

- **Claim a Semaphore:** Use this service to claim a specified semaphore.
- **Release a Semaphore:** Use this service to release a specified semaphore.
- **Query a Semaphore:** Use this service to determine whether a specified semaphore is claimed or free.

Requesting the Semaphore Management Services

To request any of the semaphore management services, load the registers and the parameter list with the proper values, and use the INT 7AH instruction to signal the workstation program that it has a request to process.

Return Codes for the Semaphore Management Services

Return codes for the semaphore management services are 2-byte values made up of a function ID and an error code. The function ID indicates the portion of the workstation program in which the error occurred. The error code indicates the specific type of error that has occurred. An error code of X'00' always indicates a successful acceptance or completion of the request.

After your application has requested a semaphore management service, the CH and CL registers contain a return code generated by the request processing portion of the workstation program. The function ID is in the CH register, and the error code is in the CL register. The return codes that can be generated by semaphore management services are called *system return codes*. The function ID for system return codes is X'12' or X'13'. The error codes that can appear are specific to the service that was requested and are included in the descriptions of each service.

See Appendix H, "Return Codes," for more information.

Semaphore Management Service X'0D': Claim a Semaphore

Use this service to claim a specified semaphore.

Register Values

On Request

AH = X'0D'
BL = Wait type
DX = Semaphore ID

On Completion

BL = Return type
CH = Function ID
CL = Return code

The contents of registers AX, BH, DX, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The BL register specifies the type of wait state your application program goes into until the request is completed. The type of wait is specified through a bit mask. If more than one type of wait is specified, the wait state ends when any one of the conditions is satisfied. The bits in the wait type mask are as follows:

0	1	2	3	4	5	6	7
Request queue	Comp queue	Comp signal	Sema-phore	Timer	Signal	Data	Reserved

- If bit 0 is set to 1, your application program waits until a request queue element is in its request queue. If an RQE is already in its request queue, the application stays dispatchable.
- If bit 1 is set to 1, your application program waits until a request queue element is in its completion queue. If an RQE is already in its completion queue, the application stays dispatchable.
- If bit 2 is set to 1, your application program waits until it receives a 'completion' signal.
- If bit 3 is set to 1, your application program waits until it receives a 'got semaphore' signal.
- If bit 4 is set to 1, your application program waits until it receives a 'timer tick' signal.
- If bit 5 is set to 1, your application program waits until it receives a 'generic' signal.

Claim a Semaphore

- If bit 6 is set to 1, your application program waits until it receives a 'data available' signal.
- Bit 7 is reserved and must be set to 0.

Note: X'00' specifies "no wait." A "wait" for data is inappropriate for this service.

- The DX register contains the ID of the semaphore to be claimed.

Completion Registers:

- The BL register indicates the type of wait condition that was satisfied to return control to the requesting task. The return type is specified via a bit mask. The bits in the return type have the same meaning as the bits in the wait type mask.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion of the request.
X'05'	Invalid index specified.
X'0B'	System RQE pool depleted.
X'2C'	Semaphore not claimed on wait.
X'3D'	Semaphore already claimed, no wait specified.

Usage Notes

- A program running in a stoppable environment may not claim semaphores in other stoppable environments.
- A program running in a stoppable environment may claim a resource semaphore from a nonstoppable environment, but it will not be allowed to request the Name Resolution service for code serialization semaphores in a nonstoppable environment.

Coding Example

```
      .  
      .  
      .  
;   
; INITIALIZE REGISTERS FOR CLAIM A SEMAPHORE  
;  
      MOV     AH,0DH  
      MOV     BL,10H           ; WAIT FOR SEMAPHORE SIGNAL  
      MOV     DX,SEMID        ; SEMAPHORE ID  
;  
; SIGNAL WORKSTATION PROGRAM FOR CLAIM A SEMAPHORE SERVICE  
;  
      INT     7AH  
      .  
      .  
      .
```

Semaphore Management Service X'0A': Release a Semaphore

Use this service to release a specified semaphore.

Register Values

On Request

AH = X'0A'
DX = Semaphore ID

On Completion

CH = Function ID
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The DX register contains the ID of the semaphore to be released.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion of the request.
X'05'	Invalid index specified.
X'0D'	Semaphore is already free.

Usage Notes

- A program running in a stoppable environment may not release semaphores in other stoppable environments.
- The workstation program does not check that the semaphore is being released by the task that claimed it. This means that, for example, task A could claim a semaphore and then request to claim it again. The second claim request causes task A to be put in a wait state. Task B could, on completion of some action, release the semaphore. This action would cause task A's second claim request to be honored, and task A could continue.

Coding Example

```
      .  
      .  
      .  
;   
; INITIALIZE REGISTERS FOR RELEASE A SEMAPHORE  
;  
      MOV    AH,0AH  
      MOV    DX,SEMID          ; SEMAPHORE ID  
;  
; SIGNAL WORKSTATION PROGRAM FOR RELEASE A SEMAPHORE SERVICE  
;  
      INT    7AH  
      .  
      .  
      .
```


Semaphore Management Service X'0B': Query a Semaphore

Use this service to determine whether a specified semaphore is claimed or free.

Register Values

On Request

AH = X'0B'
DX = Semaphore ID

On Completion

BH = Semaphore type
CH = Function ID
CL = Return code
DX = Task ID or X'0000'

The contents of registers AX, BL, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The DX register contains the ID of the semaphore to be queried.

Completion Registers:

- The BH register indicates the type of the specified semaphore, as follows:
 - X'03' = Resource semaphore
 - X'04' = Code serialization semaphore
- The DX register contains the ID of the task that claimed the semaphore, or X'0000' if the semaphore is free.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion of the request.
X'05'	Invalid index specified.

Usage Notes

- A program running in a stoppable environment may not query semaphores in other stoppable environments.

Coding Example

```
      .  
      .  
      .  
;   
; INITIALIZE REGISTERS FOR QUERY A SEMAPHORE  
;  
      MOV     AH,OBH  
      MOV     DX,SEMID           ; SEMAPHORE ID  
;  
; SIGNAL WORKSTATION PROGRAM FOR QUERY A SEMAPHORE SERVICE  
;  
      INT     7AH  
      .  
      .  
      .
```


Chapter 19. Coding Logical Timer Management Services

Introduction	19-2
Requesting the Logical Timer Management Services	19-2
Return Codes for the Logical Timer Management Services	19-2
Logical Timer Management Service X'85': Get Logical Timer	19-3
Logical Timer Management Service X'84': Set Logical Timer	19-5
Logical Timer Management Service X'8A': Release Logical Timer ..	19-8

Introduction

This chapter describes how to code requests for the logical timer management request services provided by the API.

The logical timer management services allow your application program to control time-dependent events through the use of logical timers.

The logical timer management services provided by the API are:

- **Get Logical Timer:** Use this service to get a logical timer for the specified task.
- **Set Logical Timer:** Use this service to set the timer interval for a specified logical timer.
- **Release Logical Timer:** Use this service to release a logical timer.

Requesting the Logical Timer Management Services

To request any of the logical timer management services, load the registers and the parameter list with the proper values, and use the INT 7AH instruction to signal the workstation program that it has a request to process.

Return Codes for the Logical Timer Management Services

Return codes for the logical timer management services are 2-byte values made up of a function ID and an error code. The function ID indicates the portion of the workstation program in which the error occurred. The error code indicates the specific type of error that has occurred. An error code of X'00' always indicates a successful acceptance or completion of the request.

After your application has requested a logical timer queue management service, the CH and CL registers contain a return code generated by the request processing portion of the workstation program. The function ID is in the CH register, and the error code is in the CL register. The return codes that can be generated by logical timer management services are called *system return codes*. The function ID for system return codes is X'12' or X'13'. The error codes that can appear are specific to the service that was requested and are included in the descriptions of each service.

See Appendix H, "Return Codes," for more information.

Logical Timer Management Service X'85': Get Logical Timer

Use this service to get a logical timer for the specified task.

Register Values

On Request

AH = X'85'
BH = Timer interval
DX = Task ID or X'0000'
ES = Segment address of the timer flag
DI = Offset address of the timer flag

On Completion

CH = Function ID
CL = Return code
DX = Logical timer ID

The contents of registers AX, BX, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The BH register contains the timer interval for the logical timer. The logical timers implemented by the workstation program use 18.2 timer ticks per second. Thus, to specify a logical timer interval of 1 second, you should specify a timer interval of 18; for a 2-second timer interval, you should specify a timer interval of 36, and so on.
- The DX register contains the ID of the task that will receive a 'timer tick' signal when the timer interval has elapsed. If the value of the DX register is X'0000', the workstation program uses the ID of the currently executing task.
- The ES and DI registers point to the timer flag, which is a 1-byte data area provided by your application program, that will be set to X'FF' when the timer interval has elapsed.

Completion Registers:

- The DX register contains the logical timer ID.

Get Logical Timer

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion of the request.
X'05'	Invalid index specified.
X'11'	Logical timer pool depleted.

Usage Notes

- You must request the Set Logical Timer service to set the timer interval the first time.

After the Set Logical Timer service is requested and the timer interval elapses once, the timer interval specified on the Get Logical Timer service will be used to reset the timer interval until the timer is released. The Set Timer service can be used at any time to override this reset value.

If the timer interval specified on the Get Timer service was X'00', your application program must request the Set Logical Timer service to reset the timer interval each time the timer interval has elapsed.

- Your application program can own multiple timers. You can determine which timer interval has elapsed by checking the 1-byte timer flag that is associated with each logical timer.
- The task specified by its task ID is the task that receives the 'timer tick' signal when the timer interval has elapsed.

Coding Example

```
.
.
.
;
; INITIALIZE REGISTERS FOR GET LOGICAL TIMER
;
      MOV     AH,85H
      MOV     BH,0           ; DO NOT RESET THE TIMER
      MOV     DX,0           ; USE THE CURRENTLY EXECUTING TASK'S ID
      MOV     DI,SEG TIMERFLG ; ES:DI POINT TO THE TIMER FLAG
      MOV     ES,DI
      MOV     DI,OFFSET TIMERFLAG
;
; SIGNAL WORKSTATION PROGRAM FOR GET A LOGICAL TIMER SERVICE
;
      INT     7AH
.
.
```

Logical Timer Management Service X'84': Set Logical Timer

Use this service to set the timer interval for a specified logical timer. The specified task will receive a 'timer tick' signal when the timer interval has elapsed.

Register Values

On Request

AH = X'84'
BH = Timer interval
BL = Wait type
DX = Logical timer ID

On Completion

BL = Return type
CH = Function ID
CL = Return code

The contents of registers AX, BH, DX, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The BL register specifies the type of wait state your application program goes into until the request is completed. The type of wait is specified through a bit mask. If more than one type of wait is specified, the wait state ends when any one of the conditions is satisfied. The bits in the wait type are as follows:

0	1	2	3	4	5	6	7
Request queue	Comp queue	Comp signal	Sema-phore	Timer	Signal	Data	Reserved

- If bit 0 is set to 1, your application program waits until a request queue element is in its request queue. If an RQE is already in its request queue, the application stays dispatchable.
- If bit 1 is set to 1, your application program waits until a request queue element is in its completion queue. If an RQE is already in its completion queue, the application stays dispatchable.
- If bit 2 is set to 1, your application program waits until it receives a 'completion' signal.
- If bit 3 is set to 1, your application program waits until it receives a 'got semaphore' signal.
- If bit 4 is set to 1, your application program waits until it receives a 'timer tick' signal.

Set Logical Timer

- If bit 5 is set to 1, your application program waits until it receives a 'generic' signal.
- If bit 6 is set to 1, your application program waits until it receives a 'data available' signal.
- Bit 7 is reserved and must be set to 0.

Note: X'00' specifies "no wait." A "wait" for semaphore or data is inappropriate for this service.

- The BH register contains the timer interval for the logical timer. The logical timers implemented by the workstation program use 18.2 timer ticks per second. Thus, to specify a logical timer interval of 1 second, you should specify a timer interval of 18; for a 2-second timer interval, you should specify a timer interval of 36, and so on.
- The DX register contains the ID of the timer to be set.

Completion Registers:

- The BL register indicates the type of wait condition that was satisfied to return control to the requesting task. The return type is specified through a bit mask. The bits in the return type have the same meaning as the bits in the wait type mask.
- The DX register contains the logical timer ID.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion of the request.
X'05'	Invalid index specified.

Usage Notes

- After the specified timer interval has elapsed, the timer interval specified on the Get Logical Timer service is used to reset the timer interval until the timer is released.

If the timer interval is specified as X'00' on the Get Logical Timer service, your application program must request the Set Logical Timer service to reset the timer interval each time the timer interval has elapsed.

- The task specified on the Get Logical Timer service is the task that receives the 'timer tick' signal when the timer interval has elapsed.

Coding Example

```
.  
.   
.   
;  
; INITIALIZE REGISTERS FOR SET A LOGICAL TIMER  
;  
        MOV    AH,84H  
        MOV    BH,36          ; TIMER INTERVAL, APPROXIMATELY 2 SECONDS  
        MOV    BL,08H        ; WAIT TYPE, WAIT FOR TIMER TICK SIGNAL  
        MOV    DX,TIMERID    ; TIMER ID  
;  
; SIGNAL WORKSTATION PROGRAM FOR SET A LOGICAL TIMER SERVICE  
;  
        INT     7AH  
.  
.  
.
```

Logical Timer Management Service X'8A': Release Logical Timer

Use this service to release a logical timer. Logical timers may be released only by the task that was specified on the Get Logical Timer request.

Register Values

On Request

AH = X'8A'
DX = Logical timer ID

On Completion

CH = Function ID
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The DX register contains the ID of the logical timer to be released.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion of the request.
X'05'	Invalid index specified.
X'10'	Requesting task does not own timer.

Usage Notes

Logical timers may be released only by the task that was specified on the Get Logical Timer request.

Coding Example

```
      .  
      .  
      .  
;     ;  
; INITIALIZE REGISTERS FOR RELEASE LOGICAL TIMER  
;  
      MOV     AH,8AH  
      MOV     DX,TIMERID      ; TIMER ID  
;  
; SIGNAL WORKSTATION PROGRAM FOR RELEASE LOGICAL TIMER SERVICE  
;  
      INT     7AH  
      .  
      .  
      .
```


Chapter 20. Coding Fixed-Length Queue Management Services

Introduction	20-2
Requesting the Fixed-Length Queue Management Services	20-2
Return Codes for the Fixed-Length Queue Management Services ..	20-2
Fixed-Length Queue Management Service X'0C': Enqueue Data	20-3
Fixed-Length Queue Management Service X'13': Dequeue Data	20-5
Fixed-Length Queue Management Service X'0F': Purge Queue Data ..	20-8

Introduction

This chapter describes how to code requests for the fixed-length queue management request services provided by the API.

The fixed-length queue management services allow your application program to pass data to other tasks or components, and to receive data from other tasks or components, using the fixed-length queue as a “pipeline” for the data.

The fixed-length queue management services provided by the API are:

- **Enqueue Data:** Use this service to enqueue data on the specified fixed-length queue.
- **Dequeue Data:** Use this service to dequeue data from the specified fixed-length queue.
- **Purge Queue Data:** Use this service to remove all data from the specified fixed-length queue.

Requesting the Fixed-Length Queue Management Services

To request any of the fixed-length queue management services, load the registers and the parameter list with the proper values, and use the INT 7AH instruction to signal the workstation program that it has a request to process.

Return Codes for the Fixed-Length Queue Management Services

Return codes for the fixed-length queue management services are 2-byte values made up of a function ID and an error code. The function ID indicates the portion of the workstation program in which the error occurred. The error code indicates the specific type of error that has occurred. An error code of X'00' always indicates a successful acceptance or completion of the request.

After your application has requested a fixed-length queue management service, the CH and CL registers contain a return code generated by the request processing portion of the workstation program. The function ID is in the CH register, and the error code is in the CL register. The return codes that can be generated by fixed-length queue management services are called *system return codes*. The function ID for system return codes is X'12' or X'13'. The error codes that can appear are specific to the service that was requested and are included in the descriptions of each service.

See Appendix H, “Return Codes,” for more information.

Fixed-Length Queue Management Service

X'0C': Enqueue Data

Use this service to enqueue data on the specified fixed-length queue.

Register Values

On Request

AH = X'0C'
CX = Number of bytes
DX = Fixed-length queue ID
ES = Segment address of data
DI = Offset address of data

On Completion

AX = Number of bytes
CH = Function ID
CL = Return code

The contents of registers
 BX, DX, ES, and DI are
 unpredictable.

Register Definitions

Request Registers:

- The CX register contains the number of bytes to be enqueued on the specified fixed-length queue.
- The DX register contains the ID of the fixed-length queue to receive the data.
- The ES and DI registers point to the beginning of the data to be enqueued.

Completion Registers:

- The AX register contains the number of unused bytes currently left on the fixed-length queue if the number of bytes to be enqueued was too large.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received for this service are:

Code	Meaning
------	---------

X'00'	Successful completion of the request.
X'05'	Invalid index specified.
X'39'	Not enough room on fixed-length queue to enqueue the specified data.

Enqueue Data

Usage Notes

Programs running in stoppable environments cannot enqueue data to fixed-length queues in other stoppable environments.

Coding Example

```

      .
      .
      .
;
; INITIALIZE REGISTERS FOR ENQUEUE
;
      MOV     AH,0CH
      MOV     CX,2           ; NUMBER OF BYTES
      MOV     DX,QUEUEID     ; QUEUE ID
      MOV     DI, SEG DATANAME ; SEGMENT ADDRESS OF DATA
      MOV     ES,DI          ; IN ES
      MOV     DI,OFFSET DATANAME ; OFFSET OF DATA IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR ENQUEUE SERVICE
;
      INT     7AH
      .
      .
      .
```

Fixed-Length Queue Management Service X'13': Dequeue Data

Use this service to dequeue data from the specified fixed-length queue.

Register Values

On Request

AH = X'13'
BL = Wait type
CX = Number of bytes
DX = Fixed-length queue ID
ES = Segment address of data
DI = Offset address of data

On Completion

BL = Return type
CH = Function ID
CL = Return code
DX = Number of bytes

The contents of registers
 BH, AX, ES, and DI are
 unpredictable.

Register Definitions

Request Registers:

- The BL register specifies the type of wait state your application program goes into until the request is completed. The type of wait is specified through a bit mask. If more than one type of wait is specified, the wait state ends when any one of the conditions is satisfied. The bits in the wait type mask are as follows:

0	1	2	3	4	5	6	7
Request queue	Comp queue	Comp signal	Sema-phore	Timer	Signal	Data	Reserved

- If bit 0 is set to 1, your application program waits until a request queue element is in its request queue. If an RQE is already in its request queue, the application stays dispatchable.
- If bit 1 is set to 1, your application program waits until a request queue element is in its completion queue. If an RQE is already in its completion queue, the application stays dispatchable.
- If bit 2 is set to 1, your application program waits until it receives a 'completion' signal.
- If bit 3 is set to 1, your application program waits until it receives a 'got semaphore' signal.
- If bit 4 is set to 1, your application program waits until it receives a 'timer tick' signal.

Dequeue Data

- If bit 5 is set to 1, your application program waits until it receives a 'generic' signal.
- If bit 6 is set to 1, your application program waits until it receives a 'data available' signal.
- Bit 7 is reserved and must be set to 0.

Note: X'00' specifies "no wait." A "wait" for semaphore is inappropriate for this service.

- The CX register contains the number of bytes to be dequeued from the specified fixed-length queue.
- The DX register contains the ID of the fixed-length queue.
- The ES and DI registers point to the beginning of a data area provided by your application to contain the dequeued data.

Completion Registers:

- The DX register contains the number of bytes remaining on the fixed-length queue.
- The BL register indicates the type of wait condition that was satisfied to return control to the requesting task. The return type is specified via a bit mask. The bits in the return type have the same meaning as the bits in the wait type mask.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion of the request.
X'05'	Invalid index specified.
X'09'	The fixed-length queue is empty.
X'13'	Number of bytes requested is too large.
X'37'	Not your turn to dequeue.

Usage Notes

- Programs running in stoppable environments cannot dequeue data from fixed-length queues in other stoppable environments.
- If two or more tasks request the Dequeue Data service for the same fixed-length queue, the supervisor processes the requests in first-in-first-out (FIFO) order.

Coding Example

```
;
; DATA AREA FOR DEQUEUE
;
DQSESSID DB 4 DUP(0) ; DATA AREA TO RECEIVE 4 BYTES FROM THE
; DEQUEUE

.
.
.

;
; INITIALIZE REGISTERS FOR DEQUEUE
;
MOV AH,13H
MOV BL,02H ; WAIT UNTIL INFORMATION IS AVAILABLE
MOV CX,0004H ; DEQUEUE 4 BYTES
MOV DX,QUEUEID ; FIXED-LENGTH QUEUE ID IN DX
MOV DI,SEG DQSESSID ; SEGMENT ADDRESS OF DATA AREA IN ES
MOV ES,DI
MOV DI,OFFSET DQSESSID ; OFFSET ADDRESS OF DATA AREA IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR DEQUEUE SERVICE
;
INT 7AH
.
.
.
```

Fixed-Length Queue Management Service X'0F': Purge Queue Data

Use this service to remove all data from the specified fixed-length queue.

Register Values

On Request

AH = X'0F'
DX = Fixed-length queue ID

On Completion

CH = Function ID
CL = Return code

The contents of registers
AX, BX, DX, ES, and DI
are unpredictable.

Register Definitions

Request Registers:

- The DX register contains the ID of the fixed-length queue.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion of the request.
X'05'	Invalid index specified.

Usage Notes

Programs running in stoppable environments cannot purge data from fixed-length queues in other stoppable environments.

Coding Example

```
      .  
      .  
      .  
;   
; INITIALIZE REGISTERS FOR PURGE QUEUE  
;  
      MOV    AH,0FH  
      MOV    DX,QUEUEID      ; QUEUE ID  
;  
; SIGNAL WORKSTATION PROGRAM FOR PURGE QUEUE SERVICE  
;  
      INT    7AH  
      .  
      .  
      .
```

Purge Queue Data

Chapter 21. Coding Interrupt Handler Management Services

Introduction	21-2
Requesting the Interrupt Handler Management Services	21-2
Return Codes for the Interrupt Handler Management Services ...	21-3
Interrupt Handler Management Service X'86': Install a Hardware Interrupt Handler	21-4
Interrupt Handler Management Service X'87': Install an Interrupt Handler	21-7
Interrupt Handler Management Service X'88': Query Interrupt Vector Contents	21-10
Interrupt Handler Management Service X'89': Remove an Interrupt Handler	21-12

Introduction

This chapter describes how to code requests for the interrupt handler management services provided by the API.

The interrupt handler management services allow environments to share the interrupt vector on a cooperative basis. On hardware interrupts, a device handler in any environment can receive control. On software interrupts, control is passed only to the software interrupt handler in the same environment as the code that issued the software interrupt, unless the interrupt handler was installed as a “global” software interrupt handler.

The interrupt handler management services provided by the API are:

- **Install a Hardware Interrupt Handler:** Use this service to identify an interrupt routine that is to gain control on hardware interrupts.
- **Install an Interrupt Handler:** Use this service to identify an interrupt routine to gain control on software or hardware interrupts. This service also returns the entry point of the previous interrupt handler. For hardware interrupt handlers, the Install a Hardware Interrupt Handler service is the recommended service to use.
- **Query Interrupt Vector Contents:** Use this service to obtain the entry point address of the second-level interrupt handler currently installed for the specified interrupt vector.

Note: If you are querying the contents of an interrupt vector for interrupt handler chaining, the entry point returned by the Query Interrupt Vector Contents service should not be used. For interrupt handler chaining purposes, always use the entry point returned by the Install an Interrupt Handler service.

- **Remove an Interrupt Handler:** Use this service to remove an interrupt handler that was installed through the Install a Hardware Interrupt Handler or Install an Interrupt Handler service request.

Requesting the Interrupt Handler Management Services

To request any of the interrupt handler management services, load the registers and the parameter list with the proper values, and use the INT 7AH instruction to signal the workstation program that it has a request to process.

Return Codes for the Interrupt Handler Management Services

Return codes for the interrupt handler management services are 2-byte values made up of a function ID and an error code. The function ID indicates the portion of the workstation program in which the error occurred. The error code indicates the specific type of error that has occurred. An error code of X'00' always indicates a successful acceptance or completion of the request.

After your application has requested an interrupt handler management service, the CH and CL registers contain a return code generated by the request processing portion of the workstation program. The function ID is in the CH register, and the error code is in the CL register. The return codes that can be generated by interrupt handler management services are called *system return codes*. The function ID for system return codes is X'12' or X'13'. The error codes that can appear are specific to the service that was requested and are included in the descriptions of each service.

See Appendix H, "Return Codes," for more information.

Install a Hardware Interrupt Handler

Interrupt Handler Management Service X'86': Install a Hardware Interrupt Handler

Use this service to identify an interrupt routine to gain control on hardware interrupts for devices that:

- Share a level
- Can be polled
- Can be disabled.

Register Values

On Request

AH = X'86'
BL = Interrupt mask
CL = Interrupt level
DX = Status register
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = Function ID
CL = Return code
DX = Interrupt handler ID

The contents of registers AX, BX, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The BL register contains an 8-bit mask that is logically ANDed with the contents of the status register on hardware interrupts. A nonzero result indicates that the device is interrupting. If your device does not have a port that returns an interrupt status, install your handler using DOS function calls (X'35', X'25') or the Install an Interrupt Handler service.
- The CL register contains the hardware interrupt level, which can be in the range X'00' through X'07'.
- The DX register contains the address of a status register I/O port.
- The ES register contains the segment address of the parameter list.
- The DI register contains the offset address of the parameter list.

Completion Registers:

- The DX register contains the ID of the interrupt handler.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 word	Offset address of the interrupt handler	Unchanged
2	1 word	Segment address of the interrupt handler	Unchanged
4	1 word	I/O port address	Unchanged
6	1 byte	Byte to write on device shutdown	Unchanged

Parameter Definitions

Request Parameters:

- The first two words of the parameter list must point to the interrupt handler's entry point.
- The I/O port address is the address of the port used to shut the device down.
- A port address of '0000X' will prevent the device from being shut down on IPL.
- The byte to write on shutdown is written to the port on device shutdown.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion of the request.
X'0E'	Invalid interrupt vector.
X'20'	Interrupt handler control block pool depleted.

Install a Hardware Interrupt Handler

Usage Notes

Handlers installed with Install a Hardware Interrupt Handler service:

- Get control with interrupts disabled.
- Should never issue an EOI instruction.
- Must do a device reset if the hardware requires it.
- Should use a RETURN FAR instruction to return.
- Should not swap stacks and then enable interrupts on an XMA system; this causes failure unless INDIBM2.SIF has been updated. See the *User's Guide* for more information on updating INDIBM2.SIF.

Coding Example

```
;
; PARAMETER LIST FOR INSTALL A HARDWARE INTERRUPT HANDLER
;
IHHANOFF  DW  0                ; OFFSET ADDRESS OF THE INTERRUPT HANDLER
IHHANSEG  DW  0                ; SEGMENT ADDRESS OF THE INTERRUPT HANDLER
IHPORTAD  DB  0                ; I/O PORT ADDRESS FOR SHUT-DOWN
IHBYTE    DB  0                ; BYTE TO WRITE ON DEVICE SHUT-DOWN
.
.
.
;
; INITIALIZE PARAMETER LIST FOR INSTALL A HARDWARE INTERRUPT HANDLER
;
      MOV     AX,OFFSET INTHANDL ; INTERRUPT HANDLER OFFSET INTO THE LIST
      MOV     IHHANOFF,AX
      MOV     AX,SEG INTHANDL    ; INTERRUPT HANDLER SEGMENT INTO THE LIST
      MOV     IHHANSEG,AX
      MOV     AL,SHUTPORT        ; SHUT-DOWN PORT ADDRESS INTO THE LIST
      MOV     IHPORTAD,AL
      MOV     AL,SHUTBYTE        ; SHUT-DOWN BYTE INTO THE LIST
      MOV     IHBYTE,AL
;
; INITIALIZE REGISTERS FOR INSTALL A HARDWARE INTERRUPT HANDLER SERVICE
;
      MOV     AH,86H
      MOV     BL,MASK            ; STATUS REGISTER MASK IN BL
      MOV     CL,INTLEVEL        ; INTERRUPT LEVEL IN CL
      MOV     DX,STATADDR        ; STATUS REGISTER IN DX
      MOV     DI,SEG IHHANOFF    ; SEGMENT ADDRESS OF PARAMETER LIST
      MOV     ES,DI              ; IN ES
      MOV     DI,OFFSET IHHANOFF ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR INSTALL A HARDWARE INTERRUPT HANDLER
SERVICE
;
      INT     7AH
.
.
.
```

Interrupt Handler Management Service X'87': Install an Interrupt Handler

Use this service to identify an interrupt routine that is to gain control on software or hardware interrupts. This service also returns the entry point of the previous interrupt handler to allow sharing a given interrupt.

Register Values

On Request

AH = X'87'
BL = Flags
CL = Interrupt vector
ES = Segment address of
the interrupt handler
DI = Offset address of
the interrupt handler

On Completion

CH = Function ID
CL = Return code
DX = Interrupt
handler ID
ES = Segment address of
the previous interrupt
handler
DI = Offset address of
the previous interrupt
handler

The contents of registers AX and BX
are unpredictable.

Register Definitions

Request Registers:

- The BL register contains a flag byte that indicates how the interrupt handler will get control in the following format:

0 through 5	6 and 7
Reserved	Control options

- Bits 0 through 5 are reserved.
- Bits 6 and 7 are specified as follows:

B'00' – local. This interrupt handler gets control only if the interrupt originated in the requester's environment and if no "global" handlers, who do not chain, service it first.

B'10' – global. This interrupt handler always gets control for interrupts on this level, regardless of the issuing environment.

B'01' – global last resort. This interrupt handler gets control only if no other interrupt handlers service the interrupt. It services interrupts for all environments.

Install an Interrupt Handler

- The CL register contains the interrupt vector the requester wishes to claim. Valid interrupt vector values are 0 through 255, excluding vectors X'50' through X'57' and vector X'7A'.
- The ES register contains the segment address of the interrupt handler's entry point.
- The DI register contains the offset address of the interrupt handler's entry point.

Completion Registers:

- The DX register contains the ID of the interrupt handler.
- The ES register contains the segment address of the entry point of the interrupt handler previously recorded for this interrupt vector.
- The DI register contains the offset address of the entry point of the interrupt handler previously recorded for this interrupt vector.

Note: If both the ES and DI registers contain X'0000', do not chain to that address. Instead, use an IRET (interrupt return) instruction at the end of the interrupt handler code.

For hardware interrupts, ES and DI always contain the entry point of a routine that chains to the next interrupt handler for this level.

For software interrupts, ES and DI contain the entry point of the interrupt handler that previously would have been given control for an interrupt coming from the requester's environment, on the basis of the values specified in the flag byte on request. For example, if the "global" option was specified and no other global handlers have been installed, ES and DI contain the entry point of the software first-level interrupt handler. If another global handler exists, ES and DI will contain the entry point of that handler. If the "local" option was specified, ES and DI contain the address of the handler that was servicing interrupts for this environment.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion of the request.
X'0E'	Invalid interrupt vector.
X'0F'	Invalid environment access.
X'20'	Interrupt handler control block pool depleted.
X'26'	Maximum number of software interrupt vectors already taken.

Usage Notes

- If your interrupt handler is a local interrupt handler and it decides not to service an interrupt, it should jump to the address returned by this service in the ES and DI registers. If the ES and DI registers contain a value of zero, your interrupt handler should use the IRET instruction.
- If your interrupt handler is a global interrupt handler and it decides not to service an interrupt, it should jump to the address returned by this service in ESDI.
- If your interrupt handler services an interrupt, it should use the IRET instruction to return and thus discontinue chaining the interrupt. A hardware handler about to IRET must first do a RESET if hardware requires it and issue an EOI.
- If your interrupt handler attempts to chain to the previous handler with a PUSHF and far CALL instruction combination, the registers and the stack must first be restored.
- Your interrupt handler should not swap stacks and then enable interrupts on an XMA system; this causes failure unless INDIBM2.SIF has been updated. See the *User's Guide* for more information on updating INDIBM2.SIF.

Coding Example

```

      .
      .
      .
;
; INITIALIZE REGISTERS FOR INSTALL AN INTERRUPT HANDLER
;
      MOV     AH,87H
      MOV     BL,FLAGS           ; FLAGS
      MOV     CL,INTVECT        ; INTERRUPT VECTOR
      MOV     DI, SEG INTHNDLR   ; SEGMENT ADDRESS OF PARAMETER LIST
      MOV     ES,DI             ; IN ES
      MOV     DI,OFFSET INTHNDLR ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR INSTALL AN INTERRUPT HANDLER
;
      INT     7AH
      .
      .
      .
```


Interrupt Handler Management Service X'88': Query Interrupt Vector Contents

Use this service to obtain the entry point address of the second-level interrupt handler currently installed for the specified interrupt vector. Do not use this service for global chaining.

Register Values

On Request

AH = X'88'
CL = Interrupt vector

On Completion

CH = Function ID
CL = Return code
ES = Segment address of
the interrupt handler
DI = Offset address of
the interrupt handler

The contents of registers AX, BX, and DX are unpredictable.

Register Definitions

Request Registers:

- The CL register contains the interrupt vector the requester wishes to query. Valid interrupt vector values are 0 through 255.

Completion Registers:

- The ES register contains the segment address of the entry point of the interrupt handler currently installed for this interrupt vector.
- The DI register contains the offset address of the entry point of the interrupt handler currently installed for this interrupt vector.

Note: If both the ES and DI registers contain X'0000', do not chain to that address. Instead use an IRET (Interrupt Return) instruction at the end of the interrupt handler code.

For hardware interrupts, ES and DI always contain the entry point of a routine that chains to the next interrupt handler for this level.

For software interrupts, ES and DI contain the entry point of the interrupt handler that previously would have been given control for an interrupt coming from the requester's environment, after the global interrupt handlers had checked the interrupt. That is, if a local interrupt handler was installed, ES and DI contain the address of this local handler. Otherwise, ES and DI contain the address of the original contents of the interrupt vector before any local handlers were installed.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error code that can be received for this service is:

Code	Meaning
X'00'	Successful completion of the request.

Usage Notes

If you are querying the contents of an interrupt vector for interrupt handler chaining purposes, the entry point returned by the Query Interrupt Vector Contents service should not be used. For interrupt handler chaining purposes, always use the entry point returned by the Install a Software Interrupt Handler service.

Coding Example

```
.  
. .  
;  
; INITIALIZE REGISTERS FOR QUERY INTERRUPT VECTOR CONTENTS  
;  
    MOV    AH,88H  
    MOV    CL,9          ; INTERRUPT VECTOR IN BL  
;  
; SIGNAL WORKSTATION PROGRAM FOR QUERY INTERRUPT VECTOR CONTENTS  
;  
    INT    7AH  
.  
.  
.
```

Remove an Interrupt Handler

Interrupt Handler Management Service X'89': Remove an Interrupt Handler

Use this service to remove an interrupt handler that was installed through the Install a Hardware Interrupt Handler or Install an Interrupt Handler service request.

If the interrupt handler was installed through the Install a Hardware Interrupt Handler service, the device is shut down according to the information supplied when the hardware interrupt handler was installed.

Register Values

On Request

AH = X'89'
DX = Interrupt handler ID

On Completion

CH = Function ID
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The DX register contains the ID of the interrupt handler to be removed.

Return Codes

The CH and CL registers contain a return code generated by the workstation program. System return codes use a function ID of X'12' or X'13' (found in the CH register). The error codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion of the request.
X'05'	Invalid ID specified.
X'0F'	Invalid environment access.

Coding Example

```
      .  
      .  
      .  
;     ;  
; INITIALIZE REGISTERS FOR REMOVE AN INTERRUPT HANDLER  
;     ;  
      MOV     AH,89H  
      MOV     DX,HNDLR$ID      ; INTERRUPT HANDLER ID  
;     ;  
; SIGNAL WORKSTATION PROGRAM FOR REMOVE AN INTERRUPT HANDLER  
;     ;  
      INT     7AH  
      .  
      .  
      .
```


Chapter 22. Environments and the Environment Manager

Introduction	22-2
Environments	22-2
Stoppable Environments	22-2
Nonstoppable Environments	22-3
Environment Access Restrictions	22-3
Environment Management Services Your Program or System Extension Can Use to Control Environments	22-4
Resource Managers	22-4
Environment Management Services Your System Extension Can Use to Control Resource Management	22-6

Introduction

The environment manager portion of the workstation program allows you to divide memory, resources, and supervisory objects into several groupings, called *environments*. You can designate an environment to be used to run DOS or an application program. Environments used to run DOS and DOS applications are called *stoppable* environments. You can alternatively designate an environment to be used to run a system extension that is loaded as part of the workstation program. Environments used in this way are called *nonstoppable* environments.

The environment manager allows you to stop the program running in a stoppable environment, releasing any resources that were added to the environment by that program's request. After you stop the program running in a stoppable environment, you can restart a different program in that environment without re-IPLing the entire system. The Stop/Reset service corresponds to the Ctrl-Alt-Del key sequence offered by DOS in base PC mode.

The environment manager also provides services that allow you to suspend or resume an environment. When you suspend an environment, the supervisor sets all the tasks running in that environment to the unready state until you resume that environment. When you resume an environment, the supervisor sets all the tasks in that environment to the ready state. The Suspend/Resume service corresponds to the Ctrl-NumLock key sequence offered by DOS in base PC mode.

Environments

An environment may contain memory, system control blocks, and system data areas created or accessed by user program requests. For example, an application program that creates multiple windows will have a record of each of those windows in its environment definition.

Stoppable Environments

A stoppable environment is used for running DOS or personal computer application programs. Stoppable environments can be used for any program that can be removed from the system without causing other programs to fail. That is, programs in stoppable environments must not offer services to programs running in any other environments. Programs running in stoppable environments can be stopped by any program running in the same environment including itself, by a program running in a nonstoppable environment, or by the user by pressing the Ctrl-Alt-Del keys in that sequence.

Nonstopable Environments

Nonstopable environments are used to run system extensions, which are loaded as part of the workstation program and start running automatically when the workstation program is IPLed. System extensions cannot be removed from the system without recustomizing. A system extension may offer services that other programs can use.

Environment Access Restrictions

A program in a stoppable environment should never offer services or supervisory objects that other programs may depend on. To prevent this, restrictions on access rights of stoppable environments will be enforced. These restrictions are as follows:

1. Stoppable environments are only allowed to access system resources created within that environment, or within a nonstopable environment. For instance, a stoppable environment may not claim another stoppable environment's semaphore, or access its fixed-length queue, or send a work request to a task in another stoppable environment. This restriction is enforced at name resolution time. Nonstopable environments are allowed to access all system resources, but resources belonging to stoppable environments must be managed by a resource manager.
2. A program in a stoppable environment may not change the priority, preemptability, or dispatch status of a task outside that environment. This cannot be enforced at name resolution time. Instead, it will be enforced when the specific request is made.
3. A component within a stoppable environment may not be invoked by any other environment. Access restrictions to the component ID are enforced at name resolution time.
4. A stoppable environment cannot create a gate. This is enforced when the environment attempts to create the gate.
5. A stoppable environment cannot stop, suspend, or resume an environment other than its own. It can never reset any environment, including its own.
6. A stoppable environment cannot name-resolve a code serialization semaphore that belongs to a nonstopable environment. This is enforced at name resolution time.
7. A nonstopable environment can only reset its own environment, but it can stop any stoppable environment. A nonstopable environment can also suspend or resume any environment.

Resource Managers

Environment Management Services Your Program or System Extension Can Use to Control Environments

Application programs and system extensions can use the following services to control environments. Application programs must observe the environment access restrictions listed on page 22-3. The following services and their coding examples may be found in Chapter 23, “Coding Environment Manager Services.”

- **Suspend/Resume Environment:** Use this service to suspend or resume all tasks in the specified environment. You can also use this service to suspend or resume all tasks in the specified environment except the task that is requesting the service.
- **Stop/Reset Environment:** Use this service to stop or reset the specified environment.
- **Query Task’s Environment ID:** Use this service to obtain the ID of the environment associated with the specified task.
- **Query Environment Characteristics:** Use this service to obtain a list of characteristics associated with a specified environment.

Resource Managers

A resource is anything, such as control blocks, storage, or physical devices (for example, a printer), that a system extension allocates or deallocates. A resource is generally represented as a control block or data area. A resource chain is a grouping of these data areas or control blocks. Resources are chained per environment.

You can code a system extension to provide services or allocate resources to stoppable environments. A resource manager can only exist in a nonstoppable environment. The portion of a system extension that allocates and deallocates a specified resource is called a *resource manager*. A resource manager must be identified to the environment manager before it can begin its resource management operations. You can identify more than one resource manager per environment. Alternatively, each resource manager may have more than one resource chain (one per environment).

To act as a resource manager, your system extension should do the following:

- Request the Create Component Entry service to create a component called the *cleanup* component, which the environment manager calls to recover the resources allocated to an environment when that environment is stopped.
- Obtain a resource manager ID by requesting the Identify Resource Manager service. Your resource manager must use this ID in all subsequent interaction with the environment manager.

- Request the Add Resource service each time a resource is allocated to an application program. The environment manager will add the resource to the top of the environment's chain.
- Request the Delete Resource service each time a resource is deallocated normally from the application.

To add resources to or delete resources from a resource chain, the resource manager uses the Add Resource or Delete Resource service. Each resource is assumed to be a data area that contains as its first element a 2- or 4-byte field that the environment manager uses to link the resources. The size of the field is determined by a parameter that is required on the Identify Resource Manager request.

A resource manager has a component that is called by the environment manager when a stoppable environment to which the system extension has allocated resources is being stopped. This allows the resource manager to reclaim resources outstanding in that stoppable environment. In order for your resource manager's cleanup component to be notified when a PC environment is stopped (so your resource manager can reclaim resources used for that PC environment), it must have added at least one resource to the PC environment's chain.

When a stoppable environment is re-IPLed or stopped, the environment manager notifies the cleanup component by sending the address of each resource on the resource chain. This allows the cleanup component to recover the resources that have been allocated to the stoppable environment. See "The Cleanup Component Interface" on page 23-32 for more information.

If the application is stopped (either voluntarily or involuntarily) and there are outstanding resources allocated to the application, the environment manager will call the cleanup component once for each resource on the chain of allocated resources, passing a parameter list. The cleanup component should request the Delete Resource service each time it is called, to delete the resource. In this way, the cleanup component recovers all the resources allocated by the application program.

When a PC environment is stopped, the environment manager will notify *all* the resource manager's cleanup components that have a resource on the PC environment chain that the environment is being stopped. Thus it is important for your resource manager to add a resource to any PC environment about which it is to be notified. If your resource manager that handles requests from a PC is a **component**, then it is running under the PC task that is in the PC environment, and it can simply add a resource to the current active task environment.

If the resource manager that handles requests from a PC is a **task**, then it is executing in a different environment from the PC. When it received the request through the Make a Request service, the DX register contained the task ID of the task that made the request. Your resource manager will add a resource to the environment of the task (in the DX register) that made the request.

Environment Management Services Your System Extension Can Use to Control Resource Management

Your system extension can use the following services to control resource management:

- **Identify Resource Manager:** Use this service to identify a resource manager to the environment management portion of the workstation program.
- **Add Resource:** Use this service to add a resource to the top of the resource chain, or to move a resource already on the chain to the top of the chain for a specified environment.
- **Delete Resource:** Use this service to delete a resource from a resource chain.
- **Query Resource:** Use this service to obtain the address of the first resource in a specified resource chain.

These services and their coding examples are described in Chapter 23, “Coding Environment Manager Services.”

Chapter 23. Coding Environment Manager Services

Introduction	23-2
Requesting the Environment Manager Services	23-3
Return Codes for the Environment Manager Services	23-3
Environment Manager Service X'10': Identify Resource Manager ...	23-4
Environment Manager Service X'8E': Add Resource	23-8
Environment Manager Service X'8B': Delete Resource	23-12
Environment Manager Service X'8C': Query Resource	23-15
Environment Manager Service X'90': Suspend/Resume Environment	23-17
Environment Manager Service X'99': Stop/Reset Environment	23-23
Environment Manager Service X'11': Query Task's Environment ID .	23-34
Environment Manager Service X'8D': Query Environment Characteristics	23-36

Introduction

This chapter describes how to code requests for the environment manager services provided by the API. Before using it, you should have read the introductory discussion in Chapter 22, “Environments and the Environment Manager.”

The environment manager services allow your application program to control its environment, and allow a system extension to act as a resource manager to control the allocation and deallocation of resources to application programs running in stoppable environments.

Environment manager services that your application program and system extension can use are:

- **Suspend/Resume Environment:** Use this service to suspend or resume all tasks in the specified environment. You can also suspend or resume all tasks in the environment except the task that is requesting the service.
- **Stop/Reset Environment:** Use this service to stop or reset the environment. Reset should only be used by system extensions.
- **Query Task’s Environment ID:** Use this service to obtain the ID of the environment associated with a task.
- **Query Environment Characteristics:** Use this service to obtain a list of characteristics associated with an environment.

Additional environment manager services that your system extension can use to control resource management are:

- **Identify Resource Manager:** Use this service to identify a resource manager to the environment management portion of the workstation program.
- **Add Resource:** Use this service to add a resource to the top of the resource chain, or to move a resource already on the chain to the top of the chain for an environment.
- **Delete Resource:** Use this service to delete a resource from a resource chain.
- **Query Resource:** Use this service to obtain the address of the first resource in a resource chain.

Requesting the Environment Manager Services

To request any of the environment manager services, load the registers and the parameter list with the proper values, and use the INT 7AH instruction to signal the workstation program that it has a request to process.

Return Codes for the Environment Manager Services

Return codes for the environment manager services are 2-byte values made up of a function ID and an error code. The function ID indicates the portion of the workstation program in which the error occurred. The error number indicates the specific type of error that has occurred. An error code of X'00' always indicates a successful acceptance or completion of the request.

After your application has requested an environment manager service, the CH and CL registers contain a return code generated by either the request processing portion or the environment management portion of the workstation program. The function ID is in the CH register, and the error code is in the CL register. Environment manager services that require a parameter list have additional return codes in bytes 0 and 1 of the parameter list on completion. The function ID is in byte 1, and the error code is in byte 0. The function ID for system return codes is X'12'. The function ID for environment manager return codes is X'13'. The error codes that can appear are specific to the service that was requested and are included in the descriptions of each service.

See Appendix H, "Return Codes," for more information.

Environment Manager Service X'10': Identify Resource Manager

Your system extension can identify itself as a resource manager to the environment manager portion of the workstation program. A resource manager resides in a nonstoppable system extension and allocates resources to stoppable environments. The environment manager assigns an ID to the resource manager and establishes a unique resource chain for the resource manager.

Before you can request this service, you must create a component by using the Create Component service. This component, called the *cleanup* component, is invoked whenever an environment that has resources allocated by this resource manager is stopped, reset, or deleted.

Register Values

On Request

AH = X'10'
BH = Priority of the cleanup component
CX = X'0000' or segment value
DX = ID of the cleanup component

On Completion

CH = X'12' or X'13'
CL = Return code
DL = Resource manager ID

The contents of registers AX, BX, DH, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The BH register contains the priority at which to run the cleanup component. This priority must be in the range 1 through 64. If a task in the resource manager receives requests on a fixed-length queue from an application program, the cleanup component should run at a lower priority than that task, since the Stop/Reset Environment service does not track or clean up items placed on a fixed-length queue by a terminating environment. Running the cleanup component at a lower priority ensures that the data enqueued by the application is dequeued by the system extension before the cleanup component runs.
- The CX register indicates whether the resource manager's resource chain will be linked by 1-word or doubleword pointers. If a segment value is coded in the CX register on request, the resource manager is indicating that all resources that will be added to its resource chain reside in the specified segment and, thus, are linked by 1-word pointers.

The environment manager uses the specified segment value for Query Resource, Add Resource, and Delete Resource requests. If X'0000' is coded in the CX register on request, the resource manager is indicating that all resources that will be added to its resource chain will be linked by doubleword pointers.

If the resources in the resource chain are linked by 1-word pointers, each resource data area or control block must begin with a 1-word field that will be used by the environment manager for chaining. If the resources in the resource chain are linked by doubleword pointers, each resource control block must begin with a doubleword field that will be used by the environment manager for chaining.

- The DX register contains the ID of the cleanup component, which will reclaim all of this resource manager's resources whenever an environment using them is stopped, reset, or deleted. The cleanup component is given a data chain pointer and an indication of the type "stop." The cleanup component runs under the environment manager's task at the specified priority, and also runs off the environment manager's task's stack. This stack only has 80 bytes available for the cleanup component to use. If 80 bytes is not sufficient, the cleanup component switches to its own stack when it gets control.

The cleanup component is invoked once for each resource on the resource chain for the environment being stopped, reset, or deleted. The environment manager notifies the cleanup component when an environment is stopped, reset, or deleted. See "The Cleanup Component Interface" on page 23-32 for more information.

The cleanup component must request the Delete Resource service for each resource it has cleaned up in order to remove the resources from the stopping environment.

Completion Registers:

- The DL register contains the resource manager ID assigned to this resource manager. This ID identifies a unique chain for a resource manager within each environment.

Return Codes

The CH and CL registers contain a return code generated either by the supervisor portion of the workstation program or by the environment manager portion of the workstation program.

Identify Resource Manager

- Supervisor return code:

The supervisor return codes use a function ID of X'12' (found in the CH register). The error code that can be received is found in the CL register:

Code	Meaning
------	---------

X'05'	Invalid SVC ID.
-------	-----------------

- Environment manager return codes:

Environment manager return codes use a function ID of X'13' (found in the CH register). The error codes that can be received are found in the CL register:

Code	Meaning
------	---------

X'00'	Successful completion.
-------	------------------------

X'05'	The specified ID is not a component.
-------	--------------------------------------

X'06'	Invalid priority.
-------	-------------------

X'0F'	Invalid environment access. This service must be requested from a nonstoppable environment.
-------	---

X'25'	The maximum number of resource managers has already been defined.
-------	---

See Appendix H, "Return Codes," for more information.

Usage Notes

- This service may be requested only by a nonstoppable system extension.
- This service needs to be requested only by a resource manager that wants to be notified when a stoppable environment is being stopped, reset, or deleted, so that it may clean up any resources that it allocated on the stoppable environment's behalf.
- This service should be requested once by a resource manager when it is brought into the system, usually in the system extension's initialization code.
- Before you request the Identify Resource Manager service, you must request the Create Component service to create the component that will clean up any resources allocated by the resource manager.
- The cleanup component is invoked once for each of its resources. The component must issue a Delete Resource to have that resource deleted.

Coding Example

```
      .  
      .  
      .  
;   
; INITIALIZE REGISTERS FOR IDENTIFY RESOURCE MANAGER  
;  
      MOV     AH,10H  
      MOV     BH,50           ; PRIORITY OF THE CLEANUP COMPONENT  
      MOV     CX,SEGVAL      ; SEGMENT VALUE  
      MOV     DX,CLEANCID    ; CLEANUP COMPONENT ID  
;  
; SIGNAL WORKSTATION PROGRAM FOR IDENTIFY RESOURCE MANAGER SERVICE  
;  
      INT     7AH  
      .  
      .  
      .
```

Environment Manager Service X'8E': Add Resource

Use this service to add a resource to the top of a resource chain, or to move a resource already on the chain to the top of the chain for a specified environment.

When a PC application wants to add a resource to its chain, it should make a request to the resource manager. The resource manager then issues the Add Resource service to add the resource to the PC environment.

Register Values

On Request

AH = X'8E'
BL = Options
DX = ID of the task or X'0000' * or
DL = Environment ID or X'00' *
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'13'
CL = Return code
DL = Environment ID

The contents of registers
AX, BX, DH, ES, and DI
are unpredictable.

* The value coded in the DX or DL register depends on the value coded in the BL register.
See "Register Definitions" below for more information.

Register Definitions

Request Registers:

- The BL register indicates the following options:
 - Whether the resource is to be added to the resource chain or moved to the top of the resource chain
 - Whether a task ID or an environment ID is to be used to identify the environment that requested the Add Resource service from the resource manager.

The format of the options flag is as follows:

Bit 0	Bits 1 - 6	Bit 7
0 = Add resource 1 = Move resource	Must be zero	0 = Environment ID in DL 1 = Task ID in DX

- If bit 7 of the BL register is set to 1, the DX register contains the ID of the task that requested the Add Resource service from the resource manager. If the DX register contains X'0000', the environment manager uses the ID of the currently executing task.

- If bit 7 of the BL register is set to 0, the DL register contains the environment ID of the task that requested the Add Resource service from the resource manager. If the DL register contains X'00', the environment manager uses the environment ID of the currently executing task.
- The ES register contains the segment address of the parameter list.
- The DI register contains the offset address of the parameter list.

Completion Registers:

- The DL register contains the ID of the environment to which the resource was added or moved.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 word	Offset address of resource	Unchanged
2	1 word	Segment address of resource	Unchanged
4	1 word	Reserved	Offset address of resource chain
6	1 word	Reserved	Segment address of resource chain
8	1 byte	Resource manager ID	Unchanged
9	1 byte	Reserved	Reserved

Parameter Definitions

Request Parameters:

- The segment and offset addresses of the resource are used as a pointer to the resource being added or moved.

If you are chaining with 1-word pointers, the same segment specified on the Identify Resource Manager is always used when that resource manager issues an add resource. This is true regardless of the segment address that was entered in the parameter list.

- The resource manager ID (returned by the Identify Resource Manager service) indicates which resource chain to use to add or move the specified resource.

Add Resource

Completion Parameters:

- The segment and offset addresses of the resource chain indicate the resource that was at the top of the resource chain before the Add Resource service request was completed. If the resource pointer passed was the first one on the resource chain, this value is zero.

Return Codes

The CH and CL registers contain a return code generated by the environment manager portion of the workstation program. Environment manager return codes use a function ID of X'13' (found in the CH register). The error codes that can be received are found in the CL register:

Code	Meaning
X'00'	Successful completion.
X'05'	Invalid SVC ID.
X'21'	Invalid environment ID.
X'24'	Invalid resource manager ID.
X'33'	Specified resource not found.

See Appendix H, "Return Codes," for more information.

Usage Notes

- This service may be requested only by a nonstopable system extension.
- Before you request the Add Resource service, you must request the Identify Resource Manager to identify the resource manager to the environment manager.
- When a resource is added to the resource chain, it is always added to the top (beginning) of the chain. The resource is assumed to be a data area or control block that has as its first field either a 1-word or a doubleword pointer. The environment manager uses this pointer field to chain together all the resources recorded by the resource manager in the specified environment, by setting the pointer field to point to the next resource in the resource chain. The environment manager records the newly added resource as the first resource in the chain.

The resource chain enables a resource manager to track the resources it has allocated to a particular environment so that it may reclaim those resources if a request to stop an environment is received. When a stop request is received, each item on the resource chain is sent to the resource manager's cleanup component with an indication that the environment is being stopped, reset, or deleted.

- A resource manager can add different "types" of control blocks to its resource chain. If it does, the resource control block should contain a "type" field so that your resource manager can distinguish between the resource "types" when notified of a stop.

Coding Example

```
;
; PARAMETER LIST FOR ADD RESOURCE
;
AROFFSRS  DW  0                ; OFFSET ADDRESS OF RESOURCE
ARSEGRS   DW  0                ; SEGMENT ADDRESS OF RESOURCE
AROFFSCH  DW  0                ; OFFSET ADDRESS OF RESOURCE CHAIN
ARSEGCH   DW  0                ; SEGMENT ADDRESS OF RESOURCE CHAIN
ARRMID    DB  0                ; RESOURCE MANAGER ID
ARRESRVD  DB  0                ; RESERVED
.
.
.

;
; INITIALIZE PARAMETER LIST FOR ADD RESOURCE
;
        MOV     AROFFSRS,OFFSET RESOURCE    ; OFFSET ADDRESS OF RESOURCE
        MOV     ARSEGRS,SEG RESOURCE        ; SEGMENT ADDRESS OF RESOURCE
        MOV     AL,RESMGRID                 ; RESOURCE MANAGER ID
        MOV     ARRMID,AL                   ; IN LIST

;
; INITIALIZE REGISTERS FOR ADD RESOURCE
;
        MOV     AH,8EH
        MOV     BL,01H                    ; ADD RESOURCE, AND TASK ID IS IN DX
        MOV     DX,0                      ; USE ID OF THE CURRENT TASK
        MOV     DI, SEG AROFFSRS           ; SEGMENT ADDRESS OF PARAMETER LIST
        MOV     ES,DI                      ; IN ES
        MOV     DI,OFFSET AROFFSRS         ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR ADD RESOURCE SERVICE
;
        INT     7AH
.
.
.
```

Delete Resource

Environment Manager Service X'8B': Delete Resource

Use this service to delete a resource from a resource chain.

Register Values

On Request

AH = X'8B'
BL = Options
DX = ID of the task or X'0000' * or
DL = Environment ID or X'00' *
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

CH = X'13'
CL = Return code

The contents of registers
AX, BX, DX, ES, and DI
are unpredictable.

* The value coded in the DX or DL register depends on the value coded in the BL register.
See "Register Definitions" below for more information.

Register Definitions

Request Registers:

- The BL register indicates whether a task ID or an environment ID is to be used to identify the environment that requested the Delete Resource service from the resource manager. Possible values of the BL register are as follows:

X'00' = Environment ID in the DL register
X'01' = Task ID in the DX register
- If the value of the BL register is X'01', the DX register contains the ID of the task that requested the Delete Resource service from the resource manager. If the DX register contains X'0000', the environment manager uses the ID of the currently executing task.
- If the value of the BL register is X'00', the DL register contains the environment ID of the task that requested the Delete Resource service from the resource manager. If the DL register contains X'00', the environment manager uses the environment ID of the currently executing task.
- The ES register contains the segment address of the parameter list.
- The DI register contains the offset address of the parameter list.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 word	Offset address of resource	Unchanged
2	1 word	Segment address of resource	Unchanged
4	1 word	Reserved	Offset address of resource chain
6	1 word	Reserved	Segment address of resource chain
8	1 byte	Resource manager ID	Unchanged
9	1 byte	Reserved	Reserved

Parameter Definitions**Request Parameters:**

- The segment and offset addresses of the resource are used as a pointer to the resource being deleted.
- The resource manager ID (returned by the Identify Resource Manager service) indicates which resource chain to use to delete the specified resource.

Completion Parameters:

- The segment and offset addresses of the resource chain indicate the resource that was at the top of the resource chain before the Delete Resource service request was completed.

Return Codes

The CH and CL registers contain a return code generated by the environment manager portion of the workstation program. Environment manager return codes use a function ID of X'13' (found in the CH register). The error codes that can be received are found in the CL register:

Code	Meaning
X'00'	Successful completion.
X'05'	Invalid SVC ID.
X'21'	Invalid environment ID.
X'24'	Invalid resource manager ID.
X'33'	Specified resource not found.

See Appendix H, "Return Codes," for more information.

Delete Resource

Usage Notes

- This service may be requested only by a nonstopable system extension.
- Before you request the Delete Resource service, you must request the Identify Resource Manager to identify the resource manager to the environment manager, and use the Add Resource service to add resources to the resource chain.
- When the last resource is deleted from the resource chain, the environment manager will no longer notify the resource manager if the environment is stopped, reset, or deleted.
- A resource manager should request this service whenever it removes any resources from an environment. The resource manager may choose to delete any item—not necessarily the first—on the chain. However, on a Stop Environment request, each time the environment manager invokes your cleanup component, it will pass the address of one resource, traversing in order through the chain starting with the top or first item.

Coding Example

```
;
; PARAMETER LIST FOR DELETE RESOURCE
;
DROFFSRS DW 0 ; OFFSET ADDRESS OF RESOURCE
DRSEGRS DW 0 ; SEGMENT ADDRESS OF RESOURCE
DROFFSCH DW 0 ; OFFSET ADDRESS OF RESOURCE CHAIN
DRSEGCH DW 0 ; SEGMENT ADDRESS OF RESOURCE CHAIN
DRRMID DB 0 ; RESOURCE MANAGER ID
DRRESRVD DB 0 ; RESERVED
.
.

;
; INITIALIZE PARAMETER LIST FOR DELETE RESOURCE
;
MOV DROFFSRS,OFFSET RESOURCE ; OFFSET ADDRESS OF RESOURCE
MOV DRSEGRS,SEG RESOURCE ; SEGMENT ADDRESS OF RESOURCE
MOV AL,RESMGRID ; RESOURCE MANAGER ID
MOV DRRMID,AL ; IN LIST

;
; INITIALIZE REGISTERS FOR DELETE RESOURCE
;
MOV AH,8BH
MOV BL,01H ; TASK ID IN THE DX REGISTER
MOV DX,0 ; USE ID OF THE CURRENTLY EXECUTING TASK
MOV DI,SEG DROFFSRS ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET DROFFSRS ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR DELETE RESOURCE SERVICE
;
INT 7AH
.
.
.
```

Environment Manager Service X'8C': Query Resource

Use this service to obtain the address of the first resource in a specified resource chain.

Register Values

On Request

AH = X'8C'
BL = Options
CL = Resource manager ID
DX = ID of the task or X'0000' * or
DL = Environment ID or X'00' *

On Completion

CH = X'13'
CL = Return code
ES = Segment address
of the resource
pointer
DI = Offset address of
the resource
pointer

The contents of registers
AX BX, and DX are
unpredictable.

* The value coded in the DX or DL register depends on the value coded in the BL register.
See "Register Definitions" below for more information.

Register Definitions

Request Registers:

- The BL register indicates whether a task ID or an environment ID is to be used to identify the task that requested the resource from the resource manager. Possible values of the BL register are as follows:

X'00' = Environment ID in the DL register
X'01' = Task ID in the DX register

- The CL register contains the resource manager ID (returned by the Identify Resource Manager service), which indicates which resource chain to query.
- If the value of the BL register is X'01', the DX register contains the ID of the task that requested the Query Resource service from the resource manager. If the DX register contains X'0000', the environment manager uses the ID of the currently executing task.
- If the value of the BL register is X'00', the DL register contains the environment ID of the task that requested the Query Resource service from the resource manager. If the DL register contains X'00', the environment manager uses the environment ID of the currently executing task.

Query Resource

Completion Registers:

- The ES register contains the segment address of the resource at the top of the resource chain.
- The DI register contains the offset address of the resource at the top of the resource chain.

Return Codes

The CH and CL registers contain a return code generated by the environment manager portion of the workstation program. Environment manager return codes use a function ID of X'13' (found in the CH register). The error codes that can be received are found in the CL register:

Code	Meaning
X'00'	Successful completion.
X'05'	Invalid SVC ID.
X'21'	Invalid environment ID.
X'24'	Invalid resource manager ID.

See Appendix H, "Return Codes," for more information.

Usage Notes

- This service may be requested only by a nonstopable system extension.
- Before you request the Query Resource service, you must request the Identify Resource Manager to identify the resource manager to the environment manager.
- If there are no resources on the resource chain, the resource chain pointer is zero.

Coding Example

```

      .
      .
      .
;
; INITIALIZE REGISTERS FOR QUERY RESOURCE
;
      MOV     AH,8CH
      MOV     BL,01H           ; TASK ID IN THE DX REGISTER
      MOV     CL,MANAGRID      ; RESOURCE MANAGER ID
      MOV     DX,0             ; USE ID OF THE CURRENTLY EXECUTING TASK
;
; SIGNAL WORKSTATION PROGRAM FOR QUERY RESOURCE SERVICE
;
      INT     7AH
      .
      .
      .
```

Environment Manager Service X'90': Suspend/Resume Environment

Use this service to suspend or resume all tasks in the specified environment.

Register Values

On Request

AH = X'90'
BH = Reply type
BL = Wait type
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

AX = Request ID
BL = Return type
CH = Function ID
CL = Return code

The contents of registers BH, DX, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The BH register specifies the type of reply your application program will receive when the request is completed. Possible reply types are as follows:

X'80'	Request completion is indicated by a 'completion' signal. Any existing 'completion' signal to the application program is canceled.
X'40'	Request completion is indicated by an RQE on the application program's completion queue.
X'20'	No notification of request completion is received.
X'10'	No notification of request completion is received, and the parameter list is copied into a 10-byte area, so that the parameter list data area can be reused. This is intended for interrupt handler usage.

Suspend/Resume Environment

- The BL register specifies the type of wait state your application program goes into until the request is completed. The type of wait is specified through a bit mask. When more than one type of wait is specified, the wait state ends when any one of the conditions is satisfied. The bits in the wait type mask are as follows:

0	1	2	3	4	5	6	7
Request queue	Comp queue	Comp signal	Sema-phore	Timer	Signal	Data	Reserved

- If bit 0 is set to 1, your application program waits until there is a request queue element in its request queue. If there is already an RQE in its request queue, the application stays dispatchable.
- If bit 1 is set to 1, your application program waits until there is a request queue element in its completion queue. If there is already an RQE in its completion queue, the application stays dispatchable.
- If bit 2 is set to 1, your application program waits until it receives a ‘completion’ signal.
- If bit 3 is set to 1, your application program waits until it receives a ‘semaphore claimed’ signal.
- If bit 4 is set to 1, your application program waits until it receives a ‘timer tick’ signal.
- If bit 5 is set to 1, your application program waits until it receives a ‘generic’ signal.
- If bit 6 is set to 1, your application program waits until it receives a ‘data available’ signal.
- Bit 7 is reserved and must be set to 0.

Note: X'00' specifies “no wait.” A “wait” for semaphore or data is not appropriate for this service.

- The ES register contains the segment address of the parameter list.
- The DI register contains the offset address of the parameter list.

Completion Registers:

- The AX register contains the ID of the RQE used by the supervisor for this request.
- The BL register indicates the type of wait condition that was satisfied to return control to your application program. The return type is specified via a bit mask. The bits in the return type have the same meaning as the bits in the wait type.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'13')
2	1 byte	Request type	Unchanged
3	1 byte	Flags	Unchanged

If using a task ID to specify the environment to be suspended or resumed:

Offset	Length	Contents on Request	Contents on Completion
4	1 word	Task ID	May be changed

If using an environment ID to specify the environment to be suspended or resumed:

Offset	Length	Contents on Request	Contents on Completion
4	1 byte	Environment ID	May be changed
5	1 byte	Reserved	Reserved

Parameter Definitions

Request Parameters:

- The request type indicates whether the specified environment is to be suspended or resumed as follows:

X'05' – Suspend the environment

X'06' – Resume the environment

- The flags are as follows:

Bits 0 – 5	Bit 6	Bit 7
Must be zero	0 = All 1 = All except requester	0 = Env ID used 1 = Task ID used

Suspend/Resume Environment

- Bits 0 through 5 are reserved and must be zero.
- Bit 6 indicates whether all tasks in the specified environment are to be suspended:

Bit 6 = 0 – Suspend all tasks in the environment.

Bit 6 = 1 – Suspend all tasks in the environment except the requesting task.

- Bit 7 indicates how the environment to be suspended or resumed is specified in the parameter list:

Bit 7 = 0 – Environment ID in parameter list

Bit 7 = 1 – Task ID in parameter list. The task's environment is the one to be suspended or resumed.

- If bit 7 of the flag byte is 0, byte 4 of the parameter list must contain the environment ID of the environment to be suspended or resumed. Byte 5 of the parameter list is reserved.
- If bit 7 of the flag byte is 1, word 4 of the parameter list must contain the ID of the task whose environment is to be suspended or resumed.

Return Codes in the CH and CL Registers

The CH and CL registers contain a return code generated by the supervisor portion of the workstation program or the environment management portion of the workstation program.

Supervisor return codes use a function ID of X'12' (found in the CH register). The error codes that can be received are:

Code	Meaning
X'00'	Successful completion of the request.
X'07'	Invalid reply type specified.
X'0B'	System RQE pool depleted.

Environment management return codes use a function ID of X'13' (found in the CH register). The error codes that can be received are:

Code	Meaning
X'00'	Successful completion of the request.
X'05'	Invalid task ID.
X'0C'	Byte 0 of the parameter list was nonzero.
X'17'	Stoppable environment cannot stop, reset, suspend, or resume an environment other than its own.
X'21'	Invalid environment ID.
X'43'	Invalid request type (not suspend or resume).

Return Codes in the Parameter List

Bytes 0 and 1 of the parameter list contain a return code generated by the environment management portion of the workstation program. The function ID is in byte 1, and the error code is in byte 0. Environment management return codes use a function ID of X'13'. The error codes that can be received are:

Code	Meaning
X'00'	Successful completion of the request.
X'29'	Environment not suspended.

Usage Notes

- A program in a stoppable environment can suspend or resume only its own environment. A program in a nonstoppable environment can suspend or resume any environment. If a task suspends or resumes its own environment, all tasks will be suspended or resumed except for the requesting task.
- When your application or system extension suspends an environment, the supervisor sets all the tasks running in that environment to the unready state. When you resume, the supervisor sets all the tasks in that environment to the ready state.
- On a suspend request, if a task in the environment holds any code serialization semaphores, the suspend processor will wait for the semaphores to be released. (Code serialization semaphores are to be used to protect I/O operations that cannot be interrupted by a suspend or stop request.) A deadlock may occur if the task has gone into a wait state while holding the semaphore. A deadlock will certainly occur if the task is waiting upon another task in the same environment, since all other tasks may be suspended. All application programs should observe the semaphore restrictions described in Chapter 14, "Supervisor Services."
- On a suspend request, any tasks within the environment that are found waiting for code serialization semaphores are removed from the semaphore wait queue. This prevents other environments from having to wait merely because they claim a semaphore allocated to a suspended environment.
- On a resume request, all tasks waiting on code serialization semaphores (that were saved on the suspend request) are restored to the correct semaphore wait queues.
- If an environment holds, or is waiting on, a resource semaphore when it is suspended, any environment requesting the semaphore will have to wait for the suspended environment to resume.

Suspend/Resume Environment

- More than one suspend request can be made for the same environment before any resume requests are made. However, an equal number of resume requests must be issued before the environment will actually be resumed.
- For the reasons noted above, the Suspend/Resume Environment service should be used sparingly.

Coding Example

```
;
; PARAMETER LIST FOR SUSPEND/RESUME ENVIRONMENT
;
SSRETNCD  DB  0                ; RETURN CODE
SSFXNID   DB  0                ; FUNCTION NUMBER
SSTYPE    DB  0                ; REQUEST TYPE
SSFLAGS   DB  0                ; FLAGS
SSTASKID  DW  0                ; TASK ID (OR 1-BYTE ENVIRONMENT ID)
.
.
.
;
; INITIALIZE PARAMETER LIST FOR SUSPEND/RESUME ENVIRONMENT
;
      MOV     SSRETNCD,00H      ; RETURN CODE MUST = 0 BEFORE REQUEST
      MOV     SSFXNID,00H      ; FUNCTION ID MUST = 0 BEFORE REQUEST
      MOV     SSTYPE,05H       ; REQUEST TYPE = SUSPEND
      MOV     SSFLAGS,03H      ; FLAGS = ALL EXCEPT REQUESTER, TASK ID USED
      MOV     AX,TASKID        ; TASK ID INTO THE LIST
      MOV     SSTASKID,AX
;
; INITIALIZE REGISTERS FOR SUSPEND/RESUME ENVIRONMENT
;
      MOV     AH,90H
      MOV     BH,80H           ; REPLY TYPE = COMPLETION SIGNAL
      MOV     BL,20H           ; WAIT TYPE = COMPLETION SIGNAL
      MOV     DI, SEG SSRETNCD  ; SEGMENT ADDRESS OF PARAMETER LIST
      MOV     ES,DI            ; IN ES
      MOV     DI,OFFSET SSRETNCD ; OFFSET OF PARAMETER LIST IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR SUSPEND/RESUME ENVIRONMENT SERVICE
;
      INT     7AH
.
.
.
```

Environment Manager Service X'99': Stop/Reset Environment

Use this service to stop or reset the specified environment. This service corresponds to the Ctrl-Alt-Del key sequence offered by DOS in base PC mode.

- Stopping an environment:

The stop feature of the Stop/Reset Environment service can be used by a system extension to stop all programs that are loaded in a stoppable environment. It can also be used by an application to stop the environment it is running in. A program running in a stoppable environment cannot stop programs running in other stoppable environments.

A stop request asks the environment manager to stop the program(s) in the specified environment, releasing any noninitial resources that it currently owns, and freeing all storage acquired during its loading and execution. It also results in the termination and removal of any programs that were loaded and exited but remain resident in the environment. On a stop request, the environment's storage is cleared, and any alternate presentation spaces and their associated windows are removed. Only the base presentation space and base window remain. The environment's window is cleared, and COMMAND.COM is running, waiting for input.

- Resetting an environment:

The reset feature of the Stop/Reset Environment service can be used to request that an environment's resources be reset to the state they were in when they were first created. For example, all fixed-length queues are purged, and all semaphores are released. The environment's storage is not cleared. The environment can reinitialize itself without having to be reloaded. All supervisory objects that were created by programs running in the environment still exist with the same IDs that were assigned to them when they were created.

This feature is designed for system extensions. For example, system extensions that provide communications services from a host session that can get a reset from a controller may want to use the reset feature.

Stop/Reset Environment

Register Values

On Request

AH = X'99'
BH = Reply type
BL = Wait type
ES = Segment address of the parameter list
DI = Offset address of the parameter list

On Completion

AX = Request ID
BL = Return type
CH = Function ID
CL = Return code

The contents of registers
BH, DX, ES, and DI are
unpredictable.

Register Definitions

Request Registers:

- The BH register specifies the type of reply your application program receives when the request is completed. Possible reply types are as follows:
 - X'80'** Request completion is indicated by a 'completion' signal. Any existing 'completion' signal to the application program is canceled.
 - X'40'** Request completion is indicated by an RQE on the application program's completion queue.
 - X'20'** No notification of request completion is received.
 - X'10'** No notification of request completion is received, and the parameter list is copied into a 10-byte area so that the parameter list data area can be reused. This is intended for interrupt handler usage.
- The BL register specifies the type of wait state your application program will go into until the request is completed. The type of wait is specified through a bit mask. When more than one type of wait is specified, the wait state ends when any one of the conditions is satisfied. The bits in the wait type mask are as follows:

0	1	2	3	4	5	6	7
Request queue	Comp queue	Comp signal	Sema-phore	Timer	Signal	Data	Reserved

- If bit 0 is set to 1, your application program waits until there is a request queue element in its request queue. If there is already an RQE in its request queue, the application stays dispatchable.
- If bit 1 is set to 1, your application program waits until there is a request queue element in its completion queue. If there is already an RQE in its completion queue, the application stays dispatchable.

- If bit 2 is set to 1, your application program waits until it receives a ‘completion’ signal.
- If bit 3 is set to 1, your application program waits until it receives a ‘semaphore claimed’ signal.
- If bit 4 is set to 1, your application program waits until it receives a ‘timer tick’ signal.
- If bit 5 is set to 1, your application program waits until it receives a ‘generic’ signal.
- If bit 6 is set to 1, your application program waits until it receives a ‘data available’ signal.
- Bit 7 is reserved and must be set to 0.

Note: X'00' specifies “no wait.” A “wait” for semaphore or data is inappropriate for this service.

- The ES register contains the segment address of the parameter list.
- The DI register contains the offset address of the parameter list.

Completion Registers:

- The AX register contains the ID of the RQE used by the supervisor for this request.
- The BL register indicates the type of wait condition that was satisfied to return control to your application program. The return type is specified by a bit mask. The bits in the return type have the same meaning as the bits in the wait type.

Parameter List Format

Offset	Length	Contents on Request	Contents on Completion
0	1 byte	Must be zero	Return code
1	1 byte	Must be zero	Function ID (X'13')
2	1 byte	Request type	Unchanged
3	1 byte	Flags	Unchanged

Stop/Reset Environment

If using a task ID to specify the environment to be stopped:

Offset	Length	Contents on Request	Contents on Completion
4	1 word	Task ID	May be changed

If using an environment ID to specify the environment to be stopped:

Offset	Length	Contents on Request	Contents on Completion
4	1 byte	Environment ID	May be changed
5	1 byte	Reserved	Reserved

Parameter Definitions

Request Parameters:

- The request type indicates whether the specified environment is to be reset or stopped as follows:

X'02' = Reset the environment

X'03' = Stop the environment

- The flags are as follows:

Bits 0 – 6	Bit 7
Must be zero	0 = Env ID used 1 = Task ID used

- Bits 0 through 6 are reserved and must be zero.
- Bit 7 indicates how the environment to be stopped or reset is specified in the parameter list:
 - Bit 7 = 0 – Environment ID in parameter list
 - Bit 7 = 1 – Task ID in parameter list. The task's environment is the one to be stopped or reset.
- If bit 7 of the flag byte is 0, byte 4 of the parameter list must contain the environment ID of the environment to be stopped or reset. Byte 5 of the parameter list is reserved.
- If bit 7 of the flag byte is 1, word 4 of the parameter list must contain the ID of the task whose environment is to be stopped or reset.

Return Codes in the CH and CL Registers

The CH and CL registers contain a return code generated by the supervisor portion of the workstation program or the environment management portion of the workstation program.

Supervisor return codes use a function ID of X'12' (found in the CH register). The error codes that can be received are:

Code	Meaning
X'00'	Successful completion of the request.
X'07'	Invalid reply type specified.
X'0B'	System RQE pool depleted.

Environment management return codes use a function ID of X'13' (found in the CH register). The error codes that can be received are:

Code	Meaning
X'00'	Successful completion of the request.
X'05'	Invalid SVC ID.
X'17'	Stoppable environment cannot stop, reset, suspend, or resume an environment other than its own.
X'21'	Invalid environment ID.
X'32'	The environment is nonstoppable.
X'40'	A request to delete the environment using INDSPLIT or INDMERGE is already in progress.
X'43'	Invalid request type (not stop or reset).

Return Codes in the Parameter List

Bytes 0 and 1 of the parameter list contain a return code generated by the environment management portion of the workstation program. The function ID is in byte 1, and the error code is in byte 0. Environment management return codes use a function ID of X'13'. The error codes that can be received are:

Code	Meaning
X'00'	Successful completion of the request.
X'0C'	Byte 0 of the parameter list was nonzero.
X'22'	Some resources were not successfully released.
X'0F'	Invalid environment access.
X'27'	Time-out occurred.

Stop/Reset Environment

Usage Notes

- A stoppable environment cannot stop an environment other than its own. A nonstoppable environment can stop any stoppable environment, but it can only reset its own environment.
- If a system extension uses the Make a Request service to send a request to a task in a stoppable environment, the system extension must be aware that the environment can be stopped or deleted at any time.

If the Make a Request service has been issued, the task in the stoppable environment has not started to work on the request, and a stop, reset, or delete environment request occurs, the requester's parameter list is returned to the requester with return code X'1314'. Return code X'1314' indicates that the work request was not completed, because the environment was stopped or deleted before the request could be acted on.

- If the suspend request is in process, a stop or reset request waits for the suspend request to be completed.
- A stop request asks the environment manager to stop the program(s) in the specified environment, releasing any noninitial resources that it currently owns and freeing all storage acquired during its load and execution. It also results in the termination and removal of any programs that were loaded and exited but remain resident in the environment. On a stop request, the environment's storage is cleared, and any alternate presentation spaces and their associated windows are removed. Only the base presentation space and base window remain. The environment's window is cleared, and COMMAND.COM is running, waiting for input.

After the environment processes the stop request, it is the responsibility of the requester to ensure that any initial resources deleted by the stopped program are recreated.

When a stop request is issued, an environment manager task performs all supervisor cleanup and will drive all other resource managers for cleanup as well.

To complete cleanup operations, the environment manager task prevents initiation of new work by stopping all tasks in the environment. It then recovers or waits for completion of all outstanding requests. Next, it recovers all resources of external resource managers, and finally it recovers its own system resources.

The system resources that the environment manager task is concerned with during cleanup are:

- Outstanding request queue elements (RQEs) issued by the terminating task
- Semaphores requested by the terminating task
- Completion RQEs queued to the terminating task
- Request RQEs queued to the terminating task
- Logical timers claimed by the terminating task
- Second-level interrupt handlers created by the terminating task
- User exit tables created by the terminating task
- Fixed-length queues created by the terminating task.

The environment manager ensures that the terminating environment cannot do new work by marking all the tasks terminated and unready. Those tasks, however, that hold code serialization semaphores will be marked *pending unready* and will be allowed to continue. It is not acceptable simply to free the semaphore, as most serialized code initiates I/O that should be allowed to complete processing. When the semaphore is freed, the environment manager continues with the cleanup.

Note: It is assumed that code serialization semaphores are only claimed and released by well-tested code. Therefore, it is assumed that the semaphore will be released at some time, and the environment manager does not try to force the release of the semaphore.

The environment manager releases all resource semaphores held by the tasks in the environment being stopped.

RQEs on the completion queue are removed from tasks in the environment being stopped.

The environment manager waits for all accessed RQEs originating from each task in the environment being stopped to be released. Accessed RQEs are all RQEs from the environment being stopped that were sent out to tasks outside that environment, and that a task has already started to work on.

Next, an environment manager task runs through the record of resource managers interested in the environment being stopped and issues Make a Request services to the resource managers' cleanup components. The environment manager passes a parameter list to the cleanup component that contains a pointer to the resource that is to be cleaned up and an indication that the environment is being stopped. The format of the information sent to the cleanup components is described under the heading "The Cleanup Component Interface" on page 23-32.

Stop/Reset Environment

The cleanup component will run under the environment manager task at the priority specified on the Identify Resource Manager service request. The cleanup component is invoked once for each resource that is on the resource manager's chain for the environment being stopped. (Resources are added to a resource chain through the use of the Add Resource service.)

All tasks, fixed-length queues, semaphores, user exit tables, and components created by the environment being stopped may be released.

If any wait state during stop extends beyond a normal period of time (10 seconds), the environment manager sends a return code to the requester and displays a return code to indicate to the user that a time-out occurred. At this point the user may wish to take some action to lighten the system workload, so cleanup operations may be completed. However, when an environment has not been fully cleaned up, it may be that a serious system error occurred, or that some resource manager (or its device) has hung. If a bad return code is returned by any of the external resource managers, a return code is displayed that indicates some resources were not successfully released. In this case, the user may have to take some corrective action.

After issuing the error message, the workstation program attempts to continue cleanup operations. If cleanup operations are eventually completed after the user has been notified, the environment manager displays a return code that indicates the cleanup operations have been completed. At this time, the environment may be reused.

- A reset request asks the environment manager to reset an environment's resources to their state when they were first created. In this case only, the environment's storage is not cleared. The environment may now reinitialize itself without having to reload. All supervisor resources such as fixed-length queues and tasks still exist, with the same IDs assigned to them by the supervisor when they were created.

A task can only do a reset for its own environment.

Reset processing is similar to stop processing in that the environment manager will mark all tasks in the environment being reset as *terminating* and *unready*. Those tasks that hold code serialization semaphores are allowed to continue until the semaphores are released.

The environment manager then releases all resource semaphores held by the tasks in the environment being reset.

All hardware and software second-level interrupt handlers belonging to the environment being reset are left as they were. All timers belonging to the environment are stopped. Request queue elements (RQEs) on the completion queue are removed from tasks in the environment being reset. RQEs going to other tasks in the system are removed. As during stop processing, the environment manager waits for all accessed RQEs to be freed. All RQEs coming into the tasks in the environment being reset are returned to the requester with error code X'1314'. Fixed-length queues are purged.

The environment manager notifies all resource managers interested in the environment that the environment is being reset. The format of the information sent to the resource manager is described under the heading "The Cleanup Component Interface" on page 23-32.

Tasks are marked as *nonterminating* when the reset has been completed by the environment manager.

Only the task that initially requested the reset is set ready. It is recommended that the requesting task use the Create Task Entry service with the reset option for all other tasks in the environment. This will reset each task state to what it was when it was originally created. The requesting task must then set ready other tasks in the environment.

- If an application program uses the Make a Request service to send a request to a task in a system extension that can be reset, the application program must be aware that the system extension's environment can be reset at any time.

If the Make a Request service has been invoked, the task in the system extension has not started to work on the request, and a reset request occurs, the requester's parameter list is returned to the requester with a return code of X'1314'. This code indicates that the request was not completed, because the environment was reset before the request could be acted on.

The Cleanup Component Interface

When an environment is stopped, reset, or deleted, the environment manager sends a Make a Request service to the cleanup component of each resource manager that was interested in the environment for each of its resources on the chain. The Make a Request service sends information to the cleanup component in the following format:

Offset	Length	Contents
0	1 byte	Return code
1	1 byte	Function ID
4	1 word	Offset address of resource
2	1 word	Segment address of resource
6	1 byte	Environment ID
7	1 byte	Action

- The offset and segment address of the resource is a resource that was added to the resource manager's resource chain through the use of the Add Resource service. Each time the environment manager invokes your cleanup component, it will pass the address of one resource, traversing in order through the chain, starting with the top of the resource chain.
- The environment ID is the ID of the environment being reset, stopped, or deleted.
- The Action indicates whether the environment is being reset or stopped as follows:

X'02' = Reset

X'03' or X'04' = Stopped

The above information is sent to the cleanup component for each resource on the resource manager's chain whenever an environment that has resources allocated by this resource manager is reset, stopped, or deleted. The cleanup component will run under an environment manager task at the priority specified for it on the Identify Resource Manager service request. This allows a resource manager to clean up any resources it may have allocated at an application's request when an environment is reset, stopped, or deleted. It is the resource manager's responsibility to issue the Delete Resource request as necessary to remove resources it has cleaned up from the stopping environment.

If the cleanup component is unable to release all resources or detects an error, it must set a nonzero value in the return code portion of the parameter list. See the heading "System Extension Return Codes" in Chapter 24, "Coding System Extensions," for more information.

Coding Example

```
;
; PARAMETER LIST FOR STOP/RESET ENVIRONMENT
;
STRETNCD DB 0 ; RETURN CODE
STFXNID DB 0 ; FUNCTION NUMBER
STTYPE DB 0 ; REQUEST TYPE
STFLAGS DB 0 ; FLAGS
STTASKID DW 0 ; TASK ID (OR 1 BYTE ENV. ID)

.
.
.

;
; INITIALIZE PARAMETER LIST FOR STOP/RESET ENVIRONMENT
;
MOV STRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV STFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV STTYPE,03H ; REQUEST TYPE = STOP
MOV STFLAGS,01H ; FLAGS = TASK ID USED
MOV AX,TASKID ; TASK ID INTO THE LIST
MOV STTASKID,AX

;
; INITIALIZE REGISTERS FOR STOP/RESET ENVIRONMENT
;
MOV AH,99H
MOV BH,80H ; REPLY TYPE = COMPLETION SIGNAL
MOV BL,20H ; WAIT TYPE = COMPLETION SIGNAL
MOV DI, SEG STRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET STRETNCD ; OFFSET OF PARAMETER LIST IN DI

;
; SIGNAL WORKSTATION PROGRAM FOR STOP/RESET ENVIRONMENT SERVICE
;
INT 7AH
.
.
.
```

Environment Manager Service X'11': Query Task's Environment ID

Use this service to obtain the environment ID associated with a specified task.

Register Values

On Request

AH = X'11'
DX = Task ID or X'0000'

On Completion

CH = X'12' or X'13'
CL = Return code
DL = Environment ID

The contents of registers AX, BX, DH, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The DX register contains the ID of the task being queried. If the DX register contains X'0000', the environment manager uses the ID of the currently executing task.

Completion Registers:

- The DL register contains the environment ID associated with the specified task.

Return Codes

The CH and CL registers contain a return code generated by either the supervisor or environment manager portion of the workstation program.

- Supervisor Return Code:

The supervisor return code uses a function ID of X'12' (found in the CH register). The error code that can be received is found in the CL register:

Code	Meaning
X'05'	Invalid SVC ID.

- Environment Manager Return Codes:

Environment manager return codes use a function ID of X'13' (found in the CH register). The error codes that can be received are found in the CL register:

Code	Meaning
X'00'	Successful completion.
X'05'	The specified SVC ID is not a task.

See Appendix H, "Return Codes," for more information.

Usage Notes

A system extension or application can use this service to determine which environment issued a particular request. If the system extension or application is a task, this is the task to which completion status must be posted. If it is a component, the task is the one the component is currently running under.

Coding Example

```
.  
.  
.  
;  
; INITIALIZE REGISTERS FOR QUERY TASK'S ENVIRONMENT ID  
;  
    MOV    AH,11H  
    MOV    DX,0                ; USE ID OF THE CURRENTLY EXECUTING TASK  
;  
; SIGNAL WORKSTATION PROGRAM FOR QUERY TASK'S ENVIRONMENT ID SERVICE  
;  
    INT    7AH  
.  
.  
.
```

Environment Manager Service X'8D': Query Environment Characteristics

Use this service to obtain a list of characteristics associated with a specified environment. The characteristics obtained indicate whether the environment is:

- Stoppable or nonstoppable
- A user environment or a system environment
- Allocated or available for use.

Register Values

On Request

AH = X'8D'
BL = Options flag
CX = Output buffer length
DX = ID of the task or X'0000' * or
DL = Environment ID or X'FF' *
ES = Segment address of the output buffer
DI = Offset address of the output buffer

On Completion

BH = Number of environments
CH = X'13'
CL = Return code
The contents of registers AX, BL, DX, ES, and DI are unpredictable.

* The value coded in the DX or DL register depends on the value coded in the BL register. See "Register Definitions" below for more information.

Register Definitions

Request Registers:

- The BL register indicates whether a task ID or an environment ID is to be used to identify the environment being queried. Possible values of the BL register are as follows:

X'00' = Environment ID in the DL register
X'01' = Task ID in the DX register
- The CX register contains the number of bytes in the output buffer, which will contain the environment characteristics on completion of the request. If the value in the CX register is zero, the total number of environments in the system is returned in the BH register.
- If the value of the BL register is X'01', the DX register contains the ID of a task in the environment being queried. If the DX register contains X'0000', the environment manager uses the ID of the currently executing task.

- If the value of the BL register is X'00', the DL register contains the ID of the environment being queried. If the DL register contains X'00', the environment manager uses the ID of the currently executing task to identify the environment being queried. If the DL register contains X'FF', the environment manager returns the characteristics of all environments.
- The ES register contains the segment address of the output buffer.
- The DI register contains the offset address of the output buffer.

Completion Registers:

- The BH register contains the number of environment descriptions returned.

The BH register will contain the total number of environments in the system in the following cases:

- If information was requested for all environments and the buffer was not large enough for the information
- If information was requested for one environment and the buffer was not large enough for the information
- If the CX register contains zero on request.

In these circumstances, only the request for information on all environments returns an unsuccessful return code if there is not enough room in the output buffer.

Output Buffer Format

The environment information is returned in a variable-length buffer area that your application program must provide. In the format of the output buffer, offsets of 0 and 1 as shown below must be repeated for as many environments as can be returned for the request.

Offset	Length	Contents on Request	Contents on Completion
0 *	1 byte	Reserved	Environment ID
1 *	1 byte	Reserved	Environment characteristics

* The environment ID is the ID of the environment whose characteristics are given in the following byte.

Query Environment Characteristics

The environment characteristics are indicated by the following bit settings:

Bit 0	Bit 1	Bit 2	Bits 3 - 7
0 = Nonstoppable 1 = Stoppable	0 = System 1 = User	0 = Available 1 = In use	Reserved

Return Codes

The CH and CL registers contain a return code generated by the environment manager portion of the workstation program. Environment manager return codes use a function ID of X'13' (found in the CH register). The error codes that can be received are found in the CL register:

Code	Meaning
------	---------

X'00'	Successful completion.
X'05'	Invalid SVC ID.
X'21'	Invalid environment ID.
X'28'	The output buffer is too small.

See Appendix H, "Return Codes," for more information.

Coding Example

```
.
.
.
;
; INITIALIZE REGISTERS FOR QUERY ENVIRONMENT CHARACTERISTICS
;
      MOV     AH,8DH
      MOV     BL,01H           ; TASK ID IN THE DX REGISTER
      MOV     CX,2             ; SIZE OF BUFFER AREA
      MOV     DX,0             ; USE ID OF THE CURRENTLY EXECUTING TASK
      MOV     DI, SEG OUTBUFF  ; SEGMENT ADDRESS OF BUFFER AREA
      MOV     ES,DI            ; IN ES
      MOV     DI,OFFSET OUTBUFF ; OFFSET OF BUFFER AREA IN DI
;
; SIGNAL WORKSTATION PROGRAM FOR QUERY ENVIRONMENT CHARACTERISTICS
SERVICE
;
      INT     7AH
.
.
.
```

Chapter 24. Coding System Extensions

Introduction	24-2
How to Create a System Extension	24-3
Resident Code	24-3
Fixed Data	24-4
Initialization Code	24-4
How to Tell the Workstation Program about Your System Extension	24-5
Customization Procedure	24-5
Creating and Modifying System Information Files (SIFs)	24-9
How to Determine the Numbers to Use for Your System Information File	24-10
How a System Extension Is Loaded	24-13
System Extension Messages and Return Codes	24-15
System Extension Return Codes	24-15
The System Extension Message Service	24-16
Identifying Error Return Codes with the System Extension Message Service	24-16
Requesting Error Messages with the System Extension Message Service	24-17
Requesting Informational Messages with the System Extension Message Service	24-17
Coding the System Extension Message Service to Identify Return Codes	24-18
Coding the System Extension Message Service to Request Error Messages	24-20
Coding the System Extension Message Service to Request Informational Messages	24-23
Managing Resources	24-26
Design Considerations for System Extensions and the XMA Card ...	24-26
Components	24-27
Tasks	24-27
Fixed-Length Queues	24-27
General Notes	24-27

Introduction

You can code a system extension that will run as part of the workstation program. System extensions must be well-behaved. The system extension runs in its own nonstoppable environment, created by the DOS subsystem when the system extension is loaded into memory. System extensions do not have a logical screen or keyboard. A system extension should not write to the display buffer or accept input from the keyboard unless it has used the API to establish a presentation space or keyboard definition. System extensions should provide services to other programs in stoppable environments and to tasks or system extensions in nonstoppable environments.

A system extension has greater flexibility than normal application programs because it runs in a nonstoppable environment. System extensions are a permanent part of the system and cannot be removed until the system unit is turned off or re-IPLed. System extensions must follow the guidelines described in Chapter 2, "Programming Considerations." Considerations for system extensions that will act as resource managers are described in Chapter 22, "Environments and the Environment Manager."

When system extensions create parameter lists to pass data between tasks and components, they should reserve the first word of the parameter list for the return code.

Notes:

1. *Some languages are by nature poorly behaved and should not be used.*
2. *In XMA systems, application spaces are **not** all addressed simultaneously. If system extensions process parameter lists from PC applications, they should process them when they receive the request (because, at that time, they will be running in the PC's bank). If the system extensions put the parameter list on a work queue, and on a subsequent redispach they process the work queue, the system extensions will not be in the correct bank to be looking at the user's parameter list.*
3. *System extensions should use the Enqueue Data service and Dequeue Data service to enqueue and dequeue data. These services are not recommended for passing parameter lists from one task to another. Assume a PC application in an XMA environment enqueues a parameter list to a user system extension; when the user system extension was given control after a Dequeue Data service, the system extension would not be running in the bank of the PC application and, thus, could not look at the user's parameter list.*

How to Create a System Extension

A system extension can be either a DOS format .EXE file or a DOS format .COM file. Typically, a system extension consists of three parts:

1. Resident code
2. Fixed data
3. Initialization code.

By keeping the initialization code separate from the resident code, it is possible to make the storage occupied by the one-time initialization code available for other system extensions or DOS environments.

Following is a source code example of a .COM file. The initialization code is separate from both the resident code and the data.

```
START      JMP      INITCODE

            RESIDENT CODE

            •
            •
            •

            FIXED DATA

            •
            •
            •

INITCODE:

            INITIALIZATION CODE

            •
            •
            •

            RETURN TO DOS VIA INTERRUPT X'21'
            AND FUNCTION CODE X'31'.
```

Resident Code

The resident code part of the system extension should include the code to be run when tasks or components are invoked.

Note: If your system extension is going to use DOS function calls, it is recommended that you use the API DOS function request so that the function does not use the interrupt vectors to issue the request, allowing poorly behaved applications to run simultaneously.

Create a System Extension

Fixed Data

The fixed data part of the system extension should include any data declarations and data structures needed by the system extension.

Initialization Code

The initialization code part of the system extension should create all the system objects (such as tasks, components, or queues) needed by the system extension. The initialization code should also request the Set Task Ready service for all tasks that need to be started at that time. The initial task that the initialization code is running under will be deleted when the initialization of the system extension is completed.

If the system extension needs to allocate any variable or dynamic data, it can do so by having the initialization code relocate itself to the high end of storage. Once relocated, the initialization code can create control blocks and data areas in the area of storage where it used to reside. There are several advantages to this approach, since the data areas can be in the same segment of storage as the original data areas. This approach results in better performance, since the data areas can be accessed by the offset address only, instead of both the segment and offset addresses. Another advantage to this approach is that the number of control blocks can be determined from the configuration information. It is also generally faster to build control blocks, rather than read them in from a disk.

Following is an example of initialization code written to move itself to the high end of memory and execute there.

```
AAA:      PROC FAR
          MOV     ES, TOADR          ; ES-DI IS THE ADDRESS YOU ARE
                                     ; RELOCATING THE CODE TO.  TOADR
                                     ; IS FOUND BY SUBTRACTING THE SIZE
                                     ; OF THE MODULE IN PARAGRAPHS FROM
                                     ; THE TOP OF STORAGE, WHICH IS
                                     ; FOUND IN THE PSP

          SUB     DI,DI              ; DS-SI IS THE ADDRESS YOU ARE
          SUB     SI,SI              ; RELOCATING THE CODE FROM.  THE
                                     ; DS REGISTER IS SET UP ON ENTRY.

          MOV     CX,MODSIZE         ; MODSIZE IS THE SIZE OF THE
          CLD                        ; MODULE IN WORDS (USING WORDS
                                     ; FOR THE SIZE IS FASTER THAN
                                     ; USING BYTES)
```

```
REP MOVSB WORD PTR[DI],WORD PTR[SI] ; MOVE TO HIGH STORAGE

PUSH     ES                      ; SET UP THE STACK TO DO A FAR
LEA      BX, RELOCAT             ; RETURN AND GIVE CONTROL TO
PUSH     BX                      ; LABEL RELOCAT
RET

RELOCAT:
    •
    •
    •
;
; AT THIS POINT YOU ARE RUNNING
; CODE RESIDING AT THE HIGH END OF MEMORY
;
```

After the initialization code is completed, a system extension should return to DOS by using interrupt 21H with function code X'31'. This function allows you to tell DOS the number of paragraphs to keep resident, thus allowing the initialization code storage to be reused. (See note 3 under Panel 8.1 later in this chapter.)

How to Tell the Workstation Program about Your System Extension

In order to tell the workstation program that your system extension exists, you must:

- Supply information in the customization process
- Create a SIF for the extension.

Customization Procedure

You tell the workstation program to load user-supplied system extensions through the customization process. This process is described in the *IBM 3270 Workstation Program User's Guide and Reference* manual. A system extension can be added to or removed from the system only by customizing.

On the following customization home panel there is a question asking you for the number of user-supplied system extensions to be included in the workstation program. Enter the number of user-supplied system extensions you have written.

System Extensions

Home panel

IBM CORPORATION
3270 WORKSTATION CUSTOMIZATION

Level 1.00 Copyright IBM Corp. 1984, 1987

Move cursor under the desired option.
Press PF2 to select, or type the required information

DEFAULTS	<u>None</u>	IBM-supplied default values
	A B C	Drive from which previously customized system values are read
TARGET	<u>B</u>	Drive to which customized system is written
XMA CARD	<u>Yes</u> No	The expanded memory adapter card is installed
STORAGE	1 M	Amount of storage on XMA card (1 - 2 M)
SYSTEM EXTENSIONS	0	Number of user-supplied system extensions (0-29)

PF1=Help PgDn=Next

End=Summary Esc=Quit

Later in the customization process, Panel 8.1 (shown below) asks you to enter data about your system extensions. At this time you can specify which drive the system extension module will be loaded from. You can choose to put your system extension on the customized system diskette, on a different diskette, or on the hard disk. For information on entering data in panel 8.1, see the *IBM 3270 Workstation Program User's Guide and Reference manual*.

Panel 8.1		USER-SUPPLIED SYSTEM EXTENSION OPTIONS
Move cursor under the desired option. Press PF2 to select, or type the required information.		
Customized system is 375K allocated, XXXXX free		
NAME	A: _____	Name of system extension
STORAGE	1_K	Storage required for system extension
DOS	Yes <u>No</u>	System extension uses DOS functions
DEFAULT DRIVE	A	Default disk drive for this extension
DEFAULT DIRECTORY		
	A: _____	
	B: _____	
	C: _____	
	D: _____	
	E: _____	
	F: _____	
PF1=Help PgDn=Next PgUp=Prev Home=Home End=Summary Esc=Quit		

The following list explains the User-Supplied System Extension Options:

1. If your system extension uses DOS function calls, you should reply "yes" to DOS and the Multi-DOS feature will be selected for you. Multi-DOS=yes will protect you from the PC application and the system extension doing a file I/O at the same time. (This combination causes problems.) If you choose not to select the Multi-DOS feature, then you must protect yourself from the PC and system extension doing file I/O at the same time. (For example, assume your reply is DOS=no, even if your system extension uses DOS. The PC application can issue the Make Request service to the system extension with a WAIT option. The system extension will do all DOS functions and not reply to the PC application until all DOS functions complete processing.) If your system extensions use DOS function calls only during initialization, you can answer DOS=no.

Again, DOS is not reentrant and can handle function calls only on a serial basis. The Multi-DOS feature ensures that DOS gets all requests serially. Without the Multi-DOS feature, the application and system extension must ensure that DOS gets all function requests serially. Your indication on Panel 8.1 that your system extension uses DOS will automatically select the Multi-DOS feature.

2. If a PC application takes over interrupt X'21', then system extensions must use the DOS asynchronous service (discussed in Chapter 13, "Coding Multi-DOS Support Service Requests") to make all DOS interrupt X'21' function calls. This ensures that DOS gets your request and not the application that took over interrupt X'21'. (This action is available only if the Multi-DOS feature is selected at customization time.)
3. If your system is running Multi-DOS and it takes over interrupts, it must use the Install a Hardware Interrupt Handler service or Install an Interrupt Handler service to do so. If it uses DOS to take over the interrupts, only the environment that used DOS to take over the interrupt will see the interrupt. System extensions run in their own environment. PC applications run in different environments. If an application is running and causes a software interrupt that is taken over by a system extension using DOS calls, the handler installed by the system extension will not get control. It would detect a 000 vector.

If your system extension is running without the Multi-DOS feature, then DOS can be used to take over interrupts without any problems. (The supervisor API will also work without problems.)

4. Storage Required:
 - a. If you reply "yes" to using DOS, the storage size is the size of your system extension .COM or .EXE file (including initialization code), plus 4K bytes for DOS control blocks, plus the size of any variable data allocated by the system extension. The specified storage size is actually allocated by the workstation program for this system extension. When initialization code is completed, it should issue DOS interrupt X'21' and function code X'31' to terminate and stay resident, and to specify the number of paragraphs to remain resident. This frees storage to be used only by the system extension. This extra storage is not available to the system or DOS.
 - b. If your system extension replies "no" to using DOS, the storage size is the size of your system extension .COM or .EXE file (including initialization) plus the size of any variable data allocated by your system extension (if applicable).

In this case, storage is not actually allocated by the workstation program for this system extension. The system extension is simply loaded by DOS, and initialization is run. When it is completed, it issues the DOS function call to terminate and stay resident, specifying the number of paragraphs to remain resident. This frees storage to be used by the entire system and DOS.

Creating and Modifying System Information Files (SIFs)

You use the INDSPIF utility to create and modify SIFs. The INDSPIF utility is provided on your 3270 Workstation program diskettes.

To use the INDSPIF utility, follow these steps:

1. Determine the name of the EXE or COM file you will use. This will be the name of the SIF file.
2. For system extensions, determine how many of each type of control block are needed for the system extension to run.
3. At the DOS prompt, enter the INDSPIF command by typing INDSPIF followed by the Enter key.
4. On the Home panel, you may enter the module name or the path name.
 - a. To create a System Information File, press PF2.
 - b. To read an existing System Information File, press PF3.
5. Depending on your choice, you will see the SIF panel.
6. Complete all items on the panel. When you are done, press PF3 to save the SIF on diskette.
7. You may then press either Home, to return to the Home panel, or Esc, to quit the INDSPIF utility, or you may change any information on the panel and save it again.

System Extensions

How to Determine the Numbers to Use for Your System Information File

The panel that must be completed to create a SIF for a system extension is as follows:

Panel 1 of 1		SYSTEM INFORMATION FILE OPTIONS	
Type the required information.			
PROGRAM NAME:			
DIRECTORY: C:\			
ECB	0	How many Environment Control Blocks?	
RESM	0	How many Resource Managers?	
TCB	0	How many Task Control Blocks?	
RQE	0	How many Request Queue Entries?	
GATE	0	How many Gate Entries?	
NAME	0	How many Name Table Entries?	
SVC	0	How many SVC Entries?	
SLIH	0	How many Second Level Interrupt Handlers?	
TIMER	0	How many Logical Timers?	
STACK	0	How many Stacks for the FLIH?	
Press PF3 to write the System Information File.			
PF3=Write		Home=Home Panel	Esc=Quit

The following list explains each System Extension Option and how to determine what number to enter on the panel.

1. ECB: Environment Control Blocks

This is the environment your system extension will be running in. This number must be set to 1, since each user system extension runs in its own environment. Do not change this number.

2. RESM: Resource Managers

This is the number of resource managers created by your system extension. A resource manager is created by an Identify Resource Manager service request. Determine the maximum number of Identify Resource Manager service requests your code issues and enter this number in the RESM field.

3. TCB: Task Control Blocks

This is the number of tasks created by your system extension. A TCB is created by a Create Task Entry service request. Determine the number of Create Task Entry service requests your code issues and enter this number in the TCB field.

4. RQE: Request Queue Entries

This is an estimate of the number of request queue entries likely to be used by the system extension at any given time. Every time your code issues one of the following, at least one RQE is used and later returned to the free pool:

- a. The Make a Request service (one RQE)
- b. A Make a Request service with a reply type of X'10' (no notification of request completion, copy parameter list) (two RQEs)
- c. The Claim Semaphore service (one RQE)
- d. The Dequeue Data service (one RQE)
- e. All of the following service requests use one RQE each:
 - 1) Session information services
 - 2) Keyboard services
 - 3) Window management services
 - 4) Host interactive services
 - 5) Presentation space services
 - 6) 3270 keystroke emulation services
 - 7) Copy services
 - 8) Translate service
 - 9) Operator information area services
 - 10) Multiple DOS services
 - 11) The Stop Environment service
 - 12) The Suspend Environment service

Determine the maximum number of these service requests that may be pending at any time. Enter this value in the RQE field.

5. GATE: Gate Entries

This is the total number of gate service entries put in the gate table by your module. For example, if five gates are created, each with two gate service entries, then 10 is the number you specify for this System Extension Option. A gate is created by a Create Gate Entry service request. Add up the number of service entries you specify on each Create Gate Entry service request your code will issue. Enter this number in the GATE field.

6. NAME: Name Table Entries

This is the number of names put in the name table by your module. A name is created every time you issue a supervisory object service request that uses the name option to assign a name to the object. Add up all the supervisory object service requests that your system extension issues with the name option and enter this number in the NAME field.

7. SVC: Services

This is the total number of entries put into the SVC table by your module. The following items take an entry in the SVC table:

- a. Tasks
- b. Components
- c. Fixed-Length Queues
- d. User Exit Tables
- e. Gates (not the entries in the gate table, but the actual gates)
- f. Semaphores

To determine this value, add up all the Create Task Entry, Create Component Entry, Create Semaphore Entry, Create Fixed-Length Queue Entry, Create User Exit Table Entry, and Create Gate Entry service requests your system extension will issue. Enter that value in the SVC field.

8. SLIH: Second-Level Interrupt Handlers

This is the number of unique second-level interrupt handlers installed by your module. A second-level interrupt handler is created every time you request the Install a Hardware Interrupt Handler service or the Install an Interrupt Handler service. Determine the maximum number of these requests your code will issue and enter this number in the SLIH field.

9. TIMER: Logical Timers

This is the number of logical timers created by your module. A TIMER is created every time you issue a Get Logical Timer service request. Determine the maximum number of Get Logical Timer service requests your code will issue and enter this number in the TIMER field.

10. STACKS:

Note: You should ignore this field unless you are updating INDIBM2.SIF.

For systems that have an XMA card installed, you may want to specify one or more STACKS. You must do so if your PC applications install interrupt handlers that swap stacks and then enable. The STACK field will not be multiplied by the number of PC sessions you have customized for.

Note: You should not increase this field unless absolutely necessary, since it allocates large areas of storage for stack use.

For more information on using the SPIF facility to create SIFs, see *IBM 3270 Workstation Program User's Guide and Reference*.

How a System Extension Is Loaded

After you complete the customization process, you will have a customized diskette that you can use to IPL the workstation program. When you IPL the workstation program, the DOS loader loads all the system extensions from low storage to high storage.

The DOS loader reads the configuration file that was produced by customization (INDCFG.DAT) and loads the system extensions that were specified in the file. User-supplied system extensions are loaded after the workstation program system extensions, but before the remaining storage is divided among the personal computer environments. If you are using the XMA card, the loader will find a place to install your system extension. This may be anywhere in the 1-megabyte address space of the PC.

The system extension is loaded and started the same way that DOS is loaded, as follows:

- The system extension is loaded in storage, and control is passed to the entry point of the initialization code.
- Both the ES and DS registers contain the segment address of the program segment prefix (PSP), and the offset is always zero. Refer to the *DOS Technical Reference* manual for the format of the PSP.
- The amount of memory available to the system extension is recorded in its program segment prefix.

The location $\text{PSP} + \text{X}'82'$ will contain the address of the entry within the configuration file that describes the system extension being loaded. The first two bytes of the entry are used as a return code when control is passed back to the DOS loader after initialization is completed. The 2-byte return code is made up of a 1-byte function code and a 1-byte error number. If the error code is nonzero, message ST004 will be issued by the DOS loader and will contain the system extension name, the return code, and an option to take a dump or continue bringing up the system.

It is recommended that system extensions use this return code in the event that initialization fails. For more information about the return codes that your system extension can use, see "System Extension Messages and Return Codes" on page 24-15.

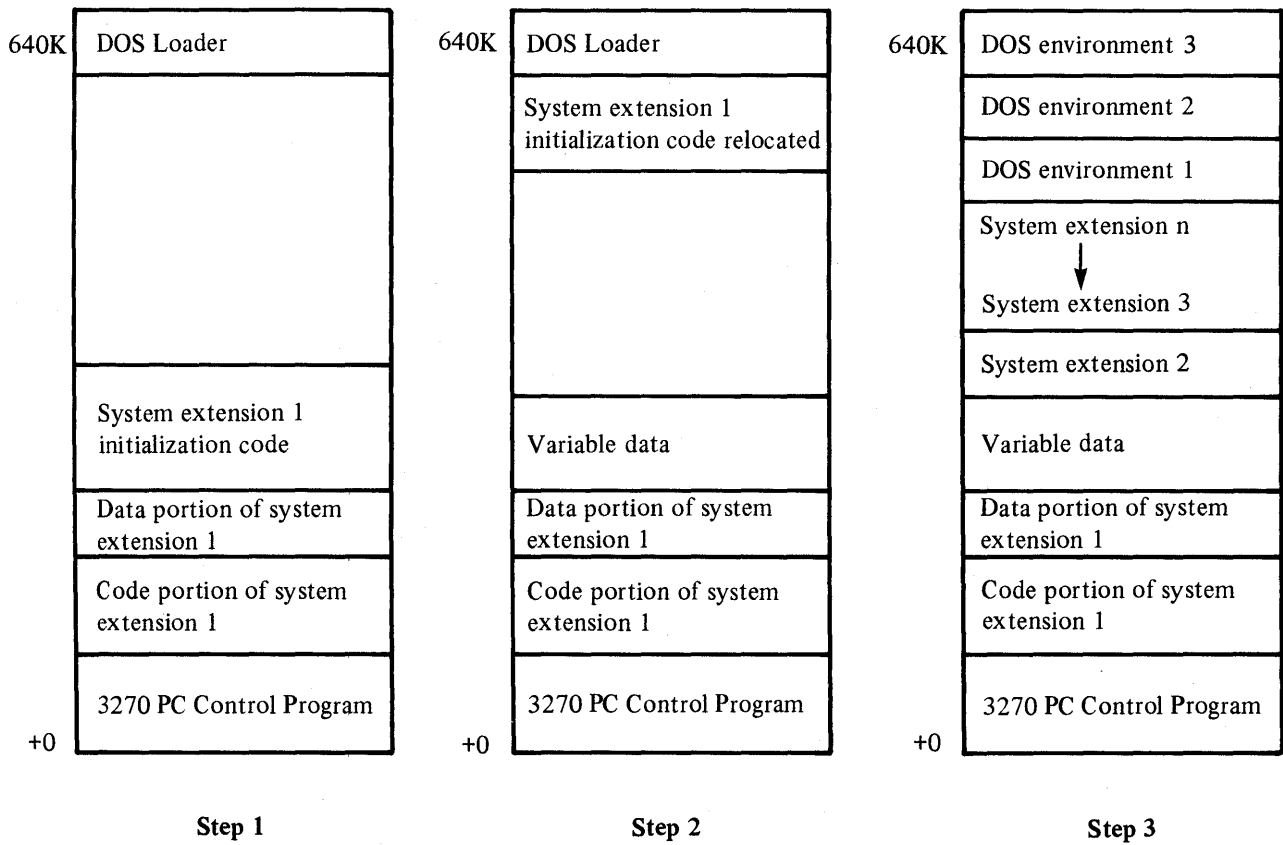
Loading a System Extension

The following diagram shows storage allocation as system extensions are loaded into memory.

Step 1 illustrates a system extension loaded into low memory with resident code separate from initialization code. The initialization code gets control to execute first.

Step 2 illustrates a system extension relocating its initialization code to the high end of storage. The initialization code runs in high storage, allocating variable data as an extension of the fixed data. When the initialization code is completed, the system extension will return to DOS. The return to DOS causes only the resident code and the fixed and variable data to remain in the system. In effect, the initialization code is deleted from the system.

Step 3 illustrates storage allocation after all system extensions have been loaded, the initialization code has been run and deleted from the system, variable data has been allocated, and the personal computer environments have been allocated.



System Extension Messages and Return Codes

Your system extension may want to issue messages to the terminal user to inform him of a normal event that is happening in the system or to tell him of an error that occurred while the system extension was running.

A system extension can issue messages by using the System Extension Message API service. This service supports error messages with three possible levels of severity, and also supports informational messages. The error messages that can be issued allow you to include a return code at the end of the message. All messages are displayed on the bottom of the screen in reverse video. The System Extension Message service is described under the heading “The System Extension Message Service.”

It is recommended that all messages issued by the system extension start with a unique identifier.

System Extension Return Codes

The workstation program uses a standard format for return codes. Return codes are 2 bytes long. The first byte is a function ID, and the second byte is an error code. The function ID indicates the portion of the workstation program in which the error occurred. The error code indicates the specific type of error that has occurred.

The workstation program has reserved the following function IDs for use by system extensions:

- X'Dx' = Vendor-supplied system extensions
- X'Ex' = User-supplied system extensions
- X'Fx' = IBM-supplied system extensions

where x is the ID of the environment that the system extension is running in. The environment ID can be obtained by requesting the Query Environment ID service.

The error code can be any number from 0 to 255. An error code of X'00' always indicates a successful acceptance or completion of the request.

The System Extension Message Service

You can use the System Extension Message service to do the following:

1. Identify the error return codes that will be used by the system extension as part of message INDSY001, INDSY002, or INDSY003
2. Issue error message INDSY001, INDSY002, or INDSY003
3. Issue informational messages written by the system extension.

Each of these functions is described in the following sections.

Note: The System Extension Message service is available for use by system extensions only, not by application programs running in stoppable environments.

Identifying Error Return Codes with the System Extension Message Service

Each return code that can be issued by your system extension as part of message INDSY001, INDSY002, or INDSY003 must be identified to the error handler portion of the workstation program before it can be displayed using the System Extension Message service. This needs to be done only once for each system extension, so that it is recommended that it be done in initialization code. When you identify a return code to the error handler, you provide the following information:

- The function ID and error code of the return code
- The threshold value for the error message
- The severity level of the error message.

The function ID and the error number must conform to the standard described under the heading "System Extension Return Codes" above.

The threshold value for the error message indicates the number of times the error can occur before an action is taken by the error service.

The severity level of the error message identifies the message that will be displayed when the error threshold is reached. There are three levels of error severity, as described below:

- **Severity 1:** A severity 1 error is an unrecoverable system error. The message that is displayed is INDSY001. The user can either press D to take a dump or press any other key to re-IPL the system.
- **Severity 2:** A severity 2 error is a less serious error. The message that is displayed is INDSY002. The user can either press D to take a dump or press any other key to continue.
- **Severity 3:** A severity 3 error is a minor error. The message that is displayed is INDSY003. The system continues running after the user presses another key.

Requesting Error Messages with the System Extension Message Service

Your system extension can request the System Extension Message service to display error message INDSY001, INDSY002, or INDSY003 in the event of an error, along with a 2-byte return code and, optionally, 2 bytes of data. The error message is displayed when the threshold level associated with the return code has been reached.

- **Severity 1 errors:**

When the System Extension Message service is requested for a severity 1 error and the threshold associated with the error is reached, the following message is displayed:

INDSY001 Unrecoverable system error - xxxxxxxx Press D to take a dump or any other key to Re-IPL

- **Severity 2 errors:**

When the System Extension Message service is invoked for a severity 2 error and the threshold associated with the error is reached, the following message is displayed:

INDSY002 Component error - xxxxxxxx Press D to take a dump or any other key to continue

- **Severity 3 errors:**

When the System Extension Message service is invoked for a severity 3 error and the threshold associated with the error is reached, the following message is displayed:

INDSY003 Component information - xxxxxxxx Press any key to continue.

“xxxxxxx” contains the 2-byte return code for the information, followed by 2 bytes of data. The 2 bytes of data can be unique for the system extension.

Requesting Informational Messages with the System Extension Message Service

Your system extension can use the System Extension Message service to issue its own informational message. The length of the message string must be 160 characters, and the message must be coded in host/notepad character format. See Appendix F, “Presentation Space Considerations,” for more information on host/notepad character formats. You can request the Translate Data service to translate an ASCII message to host/notepad character format before invoking the System Extension Message service. See Chapter 11, “Coding Translate Service Requests,” for a description of the Translate Data service.

Coding to Identify Return Codes

When the System Extension Message service is requested to display a message string, the error handler displays the string and waits for a keystroke to be pressed by the user before continuing. The 3270 converged keyboard scan code of the key pressed is returned to the requester in the AL register. Be sure to include in your message a request to the user to press a certain key or any key to continue.

Coding the System Extension Message Service to Identify Return Codes

The format of the System Extension Message service to identify an error return code that will be used by your system extension as part of message INDSY001, INDSY002, or INDSY003 is shown below:

Register Values on Request

AX = X'91'
CH = Function ID of return code
CL = Error number of return code
DH = Threshold value
DL = Severity value

Register Values on Completion

CH = X'72'
CL = Return code

The contents of registers AX, BX, DX, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The CH register contains the function ID of the return code.
- The CL register contains the error number of the return code.
- The DH register indicates the threshold value of the error, which is the number of times the error can occur before error message INDSY001, INDSY002, or INDSY003 is displayed.

Note: Threshold values should start at 1 to have the message displayed. If the threshold is specified as 0, a message will never be issued when the error occurs.

- The DL register contains the severity value of the error. Possible severity values are:

X'01' – A severity 1 error is an unrecoverable system error. The message that is displayed is INDSY001. The user can either press D to take a dump or press any other key to re-IPL the system.

X'02' – A severity 2 error is a less serious error. The message that is displayed is INDSY002. The user can either press D to take a dump or press any other key to continue.

X'03' – A severity 3 error is a minor error. The message that is displayed is INDSY003. The system continues running after the user presses another key.

Requesting the System Extension Message Service

To request the System Extension Message service, use the INT 7AH instruction to signal the workstation program that it has a request to process.

Return Codes

The CH and CL registers contain a return code generated by the error handler portion of the workstation program. Error handler return codes use a function ID of X'72' (found in the CH register). The error handler return codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion of the request.
X'02'	The error table is full.
X'03'	Invalid severity specified.

Usage Notes

- Your system extension needs to identify the return codes that it will issue as part of message INDSY001, INDSY002, or INDSY003 only once. Typically, all return code identification can be done in the initialization portion of the system extension code.
- Each return code must be identified one at a time. You cannot use a list to identify return codes.

Coding Example

```
.
.
.
;
; INITIALIZE REGISTERS FOR SYSTEM EXTENSION MESSAGE SERVICE TO IDENTIFY
; A RETURN CODE THAT WILL BE ISSUED ALONG WITH MESSAGE INDSY001,
; INDSY002, OR INDSY003.
;
      MOV     AH,91H
      MOV     CH,12H           ; FUNCTION ID IS X'12'
      MOV     CL,02H           ; ERROR NUMBER IS X'02'
      MOV     DL,01H           ; SEVERITY IS X'01'
      MOV     DH,01H           ; THRESHOLD IS X'01'
      MOV     ES,DI
;
; SIGNAL WORKSTATION PROGRAM FOR SYSTEM EXTENSION MESSAGE SERVICE
;
      INT     7AH
      .
      .
      .
```

Coding to Request Error Messages

Coding the System Extension Message Service to Request Error Messages

The format of the System Extension Message service to request error message INDSY001, INDSY002, or INDSY003 is shown below:

Register Values on Request

AH = X'91'
AL = X'00' or X'80'
BX = Two bytes of data or zero
CH = Function ID of return code
CL = Error number of return code
DX = X'00'
DI = SP register value

Register Values on Completion

AL = Scan code
CH = X'72'
CL = Return code

The contents of registers
AH, BX, DX, ES, and DI
are unpredictable.

Register Definitions

Request Registers:

- The AH register indicates whether you have pushed registers on the stack to be included in the dump data if a dump is taken.
 - X'00' indicates that the registers are not to be displayed if a dump is taken.
 - X'80' indicates that the registers are to be displayed if a dump is taken.
- The BX register must contain 2 bytes of data that are to be included in message INDSY001, INDSY002, or INDSY003, or may contain all zeros. These 2 bytes of data can be used as an extended return code.
- The CH register contains the function ID of the return code.
- The CL register contains the error number of the return code.
- The DI register must contain the value of the SP register before issuing INT 7AH.

Completion Registers:

- The AL register contains the scan code for the key pressed by the user in response to message INDSY001, INDSY002, or INDSY003. The error handler first intercepts this key to determine whether some action must be taken. If the keystroke causes a dump to be taken or the system to re-IPL, that action will be taken by the system. All other keys are passed to your system extension. See Appendix A, "Scan-Code/Shift-State and ASCII/ASCII-Mnemonic Values," for scan code values.

Requesting the System Extension Message Service

To request the System Extension Message service, use the INT 7AH instruction to signal the workstation program that it has a request to process.

Return Codes

The CH and CL registers contain a return code generated by the error handler portion of the workstation program. Error handler return codes use a function ID of X'72' (found in the CH register). The error handler return codes that can be received for this service are:

Code	Meaning
X'00'	Successful completion of the request.
X'01'	Return code was not previously identified.

Usage Notes

- The System Extension Message service request will not be completed successfully unless the return code associated with the error has been previously identified to the error handler.

Coding Example

```
      .  
      .  
      .  
;   
; EXTENDED RETURN CODE DECLARATION  
;  
EXT      DW      1306H  
      .  
      .  
      .  
;  
; PUSH ALL REGISTERS FOR DUMP  
;  
      PUSH      AX  
      PUSH      BX  
      PUSH      CX  
      PUSH      DX  
      PUSH      BP  
      PUSH      SI  
      PUSH      DI  
      PUSH      DS  
      PUSH      ES
```

Coding to Request Error Messages

```
;
; INITIALIZE REGISTERS FOR SYSTEM EXTENSION MESSAGE SERVICE TO REQUEST
; ERROR MESSAGE INDSY001, INDSY002, OR INDSY003.
;
    MOV     AH,91H
    MOV     AL,80H                ; REGISTERS HAVE BEEN SAVED
                                   ; ON THE STACK FOR DUMP
    MOV     BX,EXT                ; TWO BYTES OF DATA TO FOLLOW
                                   ; THE MESSAGE
    MOV     CH,12H                ; FUNCTION ID IS X'12'
    MOV     CL,02H                ; ERROR NUMBER IS X'02'
                                   ; NOTE THAT THIS ERROR CODE WAS
                                   ; IDENTIFIED IN THE PREVIOUS EXAMPLE
    MOV     DX,00H                ; MUST BE X'00'
    MOV     DI,SP                ; POINT TO REGISTERS SAVED ON
                                   ; THE STACK
;
; SIGNAL WORKSTATION PROGRAM FOR SYSTEM EXTENSION MESSAGE SERVICE
;
    INT     7AH
    .
    .
    .
;
; POP ALL SAVED REGISTERS
;
    POP     ES
    POP     DS
    POP     DI
    POP     SI
    POP     BP
    POP     DX
    POP     CX
    POP     BX
    POP     AX
```

Coding the System Extension Message Service to Request Informational Messages

The format of the System Extension Message service to request informational messages is shown below:

Register Values on Request

AH = X'91'
AL = X'00'
CX = X'7FFF'
ES = Segment address of the message string
DI = SP register value
DX = Offset address of the message string

Register Values on Completion

AL = Scan code
CH = X'72'
CL = Return code

The contents of registers AL, BX, DX, ES, and DI are unpredictable.

Register Definitions

Request Registers:

- The ES and DX registers contain the segment and offset addresses of the message string to be displayed. The length of the message string must be 160 characters, and the message must be coded in host/notepad character format. You can request the Translate Data service to translate an ASCII message to host/notepad character format before requesting the System Extension Message service. Refer to Chapter 11, "Coding Translate Service Requests," for the host characters. It is suggested that the first character of the message be a blank.
- The DI register must contain the value of the SP register before issuing INT 7AH.

Completion Registers:

- The AL register contains the scan code of the key that was pressed by the user in response to the message. Scan code values are found in Appendix A, "Scan-Code/Shift-State and ASCII/ASCII-Mnemonic Values."

Note: When the System Extension Message service is requested to display a message string, the error handler displays the string and waits for a keystroke to be pressed by the user before continuing. This keystroke is passed directly to your system extension. Be sure to include in your message a request to the user to press a certain key or any key to continue.

Coding to Request Informational Messages

Requesting the System Extension Message Service

To request the System Extension Message service, use the INT 7AH instruction to signal the workstation program that it has a request to process.

Return Codes

The CH and CL registers contain a return code generated by the error handler portion of the workstation program. Error handler return codes use a function ID of X'72' (found in the CH register). The error handler return code that can be received for this service is:

Code	Meaning
X'00'	Successful completion of the request.

Usage Notes

- The message displayed by this service is treated as a severity 3 message. After the message is displayed, the error handler waits for a keystroke from the user and then returns control to your system extension. Because of this, you should include in your message a prompt asking the user to press a key.
- The message string must be 160 characters long. It is suggested that the first character of each message be a blank.
- If any return codes are issued as part of this message, they do not have to be identified with the error handler.

Coding Example

```

      .
      .
      .
;
; THE USER-DEFINED MESSAGE
;
MSG      DB      10H,B3H,87H,88H,92H,10H,88H,92H,10H,80H,8DH,10H
          DB      84H,97H,80H,8CH,8FH,8BH,84H,10H,8EH,85H,10H,80H,8DH,10H
          DB      A8H,8DH,93H,84H,91H,8DH,80H,8BH,10H,A2H,8EH,83H,84H
          DB      8FH,8EH,88H,8DH,93H,10H,8CH,84H,92H,92H,80H,86H,84H,32H
;
; THE ABOVE MESSAGE SAYS: THIS IS AN EXAMPLE OF A CODEPOINT MESSAGE
;
;
          DB      27      DUP(10H)
;
; PAD WITH BLANKS TO EQUAL 80 CHARACTERS
;
          DB      10H,AFH,91H,84H,92H,92H,10H,80H,8DH,98H,10H,8AH,84H,98H
          DB      10H,93H,8EH,10H,82H,8EH,8DH,93H,88H,8DH,94H,84H,32H
;
; THE ABOVE MESSAGE SAYS: PRESS ANY KEY TO CONTINUE
;
          DB      53      DUP(10H)
;
```

Coding to Request Informational Messages

```
; PAD WITH BLANKS TO EQUAL 80 CHARACTERS
;
;
;
;
;SAVE ALL REGISTERS
;
    PUSH AX
    PUSH BX
    PUSH CX
    PUSH DX
    PUSH BP
    PUSH SI
    PUSH DI
    PUSH DS
    PUSH ES
; INITIALIZE REGISTERS FOR SYSTEM EXTENSION MESSAGE SERVICE TO REQUEST
; AN INFORMATIONAL MESSAGE.
;
    MOV     AH,91H
    MOV     AL,00H           ; MUST BE X'00'.
    MOV     CX,7FFFH        ; MUST BE X'7FFF'
    MOV     DX,OFFSET MSG   ; DX = OFFSET ADDRESS OF THE MESSAGE
    MOV     ES,SEGMENT MSG   ; ES = SEGMENT ADDRESS OF THE MESSAGE
    MOV     DI,SP
;
; SIGNAL WORKSTATION PROGRAM FOR SYSTEM EXTENSION MESSAGE SERVICE
;
    INT     7AH
;
;
;
;
; RESTORE ALL REGISTERS
;
    POP ES
    POP DS
    POP DI
    POP SI
    POP BP
    POP DX
    POP CX
    POP BX
    POP AX
```

Design Considerations

Managing Resources

If your system extension is going to manage internal resources (for example, control blocks or hardware) for applications in stoppable environments, it must include a resource manager. A resource manager is a component that is called by the environment manager when a stoppable environment to which the system extension has allocated resources is being stopped. This allows the system extension to reclaim resources outstanding in that stoppable environment. In order for your system extension's cleanup component to be notified when a PC environment is stopped (so your system extension can reclaim resources used for that PC environment), it must have added a resource to the PC environment's chain.

When a PC environment is stopped, the environment manager notifies *all* the resource manager's cleanup components that have a resource on the PC environment chain that the environment is being stopped. Thus, it is important for your system extension to add a resource to any PC environment about which it is to be notified. If your system extension that handles requests from a PC is a **component**, then it is running under the PC task that is in the PC environment, and it can simply add a resource to the current active task environment.

If the system extension that handles requests from a PC is a **task**, then it is executing in a different environment from the PC. When it received the request, the DX register contained the task ID of the task that made the request. Your system extension must add a resource to the environment of the task (in the DX register) that made the request.

If a resource is added to the PC environment, the resource managers that added the resource are notified (via the cleanup component) when the PC environment is stopped. Resource managers are described in Chapter 22, "Environments and the Environment Manager."

Design Considerations for System Extensions and the XMA Card

When the XMA card is used and there are multiple PC sessions, each PC session is in its own 1-megabyte address space. This was not the case in Release 2.1, where all PC sessions shared the same 1-megabyte address space; thus PC sessions and system extensions could pass information in any way, and avoiding the Application Program Interface (API) would not cause a problem. With the XMA card, PC sessions cannot communicate directly with one another, since the storage of one PC session is not addressable from the second. System extensions are loaded in the 1-megabyte address space of all PC sessions, so all the PC sessions can communicate with a system extension.

In a Release 4.0 system, all tasks are assigned an address space to run in. Each task can address 1 megabyte of storage. Since each PC is in its own address space, a single task cannot address storage in all the PC sessions at the same time. To interface a system extension to all PC sessions, the following elements of the API should be used.

Components

Components installed by a system extension always run on the PC sessions task in the PC sessions address space. There is no problem passing information to and from all PC sessions using components.

Tasks

Tasks installed by a system extension initially run in the address space of the workstation program, since during initialization of the system extension there are no PC sessions created yet. The workstation program has built into the Get a Request service on the API an automatic switching of address spaces. When a task does a Get a Request service, the task is switched to the address space of the requesting PC session. The task continues to run in this address space until another Get a Request call is done. The Get a Request call is the only one that causes the address space to change; waiting for a signal, timer, semaphore, fixed-length queue, or a completion queue request (RQE) does not change the address space.

Fixed-Length Queues

When a fixed-length queue is created, it is assigned to an address space. The workstation program is aware of which address space the queue is in, and Enqueue Data and Dequeue Data services can be used to pass data between address spaces. Note that the data is passed, but if pointers to data are passed in the queue, the pointer may not be valid in the address space of the task that did the dequeue.

General Notes

System extensions that rely on absolute addresses may have problems handling requests from multiple PC sessions. Since the PC sessions are in different address spaces, the PC sessions are located at the same address within the 1-megabyte address. If you run the same program in three different PC sessions, and that program passes an address to the system extension, the address could be exactly the same for all three PC sessions. If your system extension needs to track which PC made the request, the Query Task's Environment ID call can be used to distinguish between the different PC sessions. Each PC has a unique environment ID.

Part 4. Sample Programs

This part contains five sample programs that use the services of the API.

- Sample Program 1 (Chapter 25) locks all keyboard input until the correct password is typed.
- Sample Program 2 (Chapter 26) prevents a 3270 PC user from using the Work Station Control windowing features, allowing a user to set up the screens using the RESTORE utility but not to change the setup of the windows on the screen.
- Sample Program 3 (Chapter 27) presents a summary of the currently active sessions.
- Sample Program 4 (Chapter 28) accesses data from the host and displays the information about a business in a personal computer window in bar chart form.
- Sample Program 5 (Chapter 29) is the second half of Sample Program 4. It should be linked with Sample Program 4 at link-edit time.

Chapter 25. Sample Program 1

TITLE LOCKKYBD

PAGE 80,132

```

;
; PROGRAM : LOCKKYBD
;
; FUNCTION :
;   THIS PROGRAM LOCKS ALL KEYBOARD INPUT UNTIL THE CORRECT PASSWORD
;   IS TYPED. THIS PREVENTS UNWANTED USERS FROM GAINING ACCESS TO 3270
;   PC FUNCTIONS AND DATA. THE USER INVOKING THE PROGRAM IS PROMPTED
;   TO ENTER A PASSWORD. THIS PASSWORD MUST BE REENTERED IN ORDER TO
;   REGAIN ACCESS TO THE 3270 PC.
;   TO DO THIS, TWO OTHER TASKS ARE CREATED. THE FIRST TASK IN-
;   TERCEPTS THE WSCTRL KEYSTROKES TO KEEP THE USER FROM ACCESSING
;   WSCTRL FUNCTIONS. THIS IS ACCOMPLISHED BY CONNECTING TO THE WSCTRL
;   KEYBOARD, INTERCEPTING THE KEYSTROKES AND IGNORING THEM.
;   THE SECOND TASK PREVENTS THE PC IPL SEQUENCE (CTRL-ALT-DEL) FROM
;   REACHING THE PC SESSION, THUS PREVENTING THE USER FROM RESTARTING THE
;   SYSTEM TO GAIN ACCESS TO ANY FUNCTIONS OR DATA. THIS SECOND TASK IS
;   CONNECTED TO THE PC SESSION'S KEYBOARD. IT INTERCEPTS THE KEYSTROKES
;   GOING TO THE PC AND CHECKS FOR THE CTRL-ALT-DEL SEQUENCE. THIS KEY-
;   STROKE IS IGNORED, WHILE ALL OTHERS ARE PASSED ON TO THE PC SESSION.
;   A SIGNAL MUST BE SENT TO THE TASKS TO TELL THEM WHEN TO STOP
;   READING KEYSTROKES. THE READ KEYSTROKE FUNCTION PUTS A REQUEST QUEUE
;   ELEMENT (RQE) ON THE KEYSTROKING QUEUE TO SIGNAL THAT IT IS WAITING
;   FOR DATA IN THE QUEUE. SINCE THE TASKS ARE ALWAYS LOOPING BACK TO
;   GET KEYSTROKES, THERE ARE ALWAYS RQE'S ON THE KEYSTROKING QUEUES.
;   THIS CAUSES A PROBLEM DURING CLEAN UP SINCE THE WORKSTATION PROGRAM WILL
;   NOT ALLOW A FIXED-LENGTH QUEUE WITH AN RQE ON IT TO BE DELETED. SO
;   AN UNUSED SCAN CODE IS SENT TO THE TASKS TO SIGNAL THAT IT IS TIME TO
;   STOP READING KEYSTROKES, THUS KEEPING RQE'S OFF THE KEYSTROKING
;   QUEUES.
;
; THIS PROGRAM USES THE FOLLOWING API FUNCTIONS:
;
;   CONNECT TO KEYBOARD
;   CREATE A FIXED LENGTH QUEUE ENTRY
;   CREATE A TASK ENTRY
;   DELETE AN ENTRY
;   DISCONNECT FROM KEYBOARD
;   RETURN TO DISPATCHER
;   ENQUEUE DATA
;   NAME RESOLUTION
;   QUERY SESSION ID
;   QUERY BASE WINDOW
;   QUERY ACTIVE TASK
;   READ KEYSTROKE
;   SET A TASK "READY"
;   SET A TASK "UNREADY"
;   WRITE KEYSTROKE
;
;
; SUBTTL MACROS
; PAGE
;
; ;;;;;;;;;;;;;;
; ;; SET UP MACROS ;;
; ;;;;;;;;;;;;;;

```


Sample Program 1

```
;
;      MACRO : DOSFUNCT
;      FUNCTION :
;          ISSUE THE DOS CALL WITH THE FUNCTION NUMBER PASSED IN
;          FUNCTNUM.
;
DOSFUNCT MACRO  FUNCTNUM

    MOV    AH,FUNCTNUM
    INT    21H

    ENDM

;
;      MACRO : PROMPT
;      FUNCTION :
;          DISPLAY THE PROMPT "ENTER PASSWORD : "
;
; ESTABLISH CONSTANTS

DISPSTR EQU    9                ; DOS DISPLAY STRING FUNCTION NUMBER

PROMPT MACRO

    LEA    DX,PASSPRMT    ; POINT DX TO THE START OF THE PROMPT
    DOSFUNCT DISPSTR    ; DISPLAY THE PROMPT STRING

    ENDM

;
;      MACRO : DISPLAY
;      FUNCTION :
;          DISPLAY THE CHARACTER PASSED IN CHAR.
;
; ESTABLISH CONSTANTS

DISPCHAR EQU    2                ; DOS DISPLAY CHARACTER FUNCTION NUMBER

DISPLAY MACRO CHAR

    MOV    DL,CHAR        ; PUT THE CHARACTER IN DL
    DOSFUNCT DISPCHAR    ; DISPLAY THE CHARACTER

    ENDM

;
;      MACRO : CHEK4ERR
;      FUNCTION :
;          SET UP THE REGISTERS FOR THE ERROR CHECKER PROCEDURE.
;
```

Sample Program 1

```
CHEK4ERR  MACRO  RETNCODE

    IFNB  <RETNCODE>      ; IF THERE IS A PARAMETER LIST RETURN CODE
    MOV   BL,RETNCODE     ; SPECIFIED, PASS IT TO THE ERROR CHECKER
    ELSE                        ; IN BL.
    MOV   BL,0            ; OTHERWISE, SEND A 0 IN BL
    ENDIF

    CALL  CHECKERR        ; CALL THE ERROR CHECKER

    ENDM

;
;      MACRO NAME : CONN$KEY
;      PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'KEYBOARD'
;                  SESSID  -- SESSION ID
;                  KEYSTQ   --
;      _ KEYSTROKE QUEUE ID
;                  EVNTQ    --
;      _ EVENT QUEUE ID
;      FUNCTION :
;      CONNECT THE KEYBOARD TO THE SPECIFIED SESSION.
;

CONN$KEY  MACRO  SERVTYPE,SESSID,KEYSTQ,EVNTQ

    ; INITIALIZE PARAMETER LIST FOR CONN$KEY
    MOV   CKRETNCD,00H      ; RETURN CODE MUST = 0 BEFORE REQUEST
    MOV   CKFXNID,00H       ; FUNCTION ID MUST = 0 BEFORE REQUEST
    MOV   AL,SESSID         ; SESSION ID INTO THE LIST
    MOV   CKSESSID,AL

    IFNB  <KEYSTQ>
    MOV   AX,KEYSTQ         ; IF A KEYSTROKE QUEUE WAS SPECIFIED,
    ELSE                        ; PUT IT INTO THE LIST
    MOV   AX,0
    ENDIF
    MOV   CKKEYSTQ,AX

    IFNB  <EVNTQ>
    MOV   AX,EVNTQ          ; IF AN EVENT QUEUE WAS SPECIFIED,
    ELSE                        ; PUT IT INTO THE LIST
    MOV   AX,0
    ENDIF
    MOV   CKEVENTQ,AX

    ; INITIALIZE REGISTERS FOR CONN$KEY
    MOV   AH,09H
    MOV   AL,01H
    MOV   BH,80H
    MOV   BL,20H
    MOV   CX,0000H
    MOV   DX,SERVTYPE       ; RESOLVED VALUE FOR 'KEYBOARD'
    MOV   DI,SEG CKRETNCD    ; SEGMENT ADDRESS OF PARAMETER LIST
    MOV   ES,DI             ; IN ES
    MOV   DI,OFFSET CKRETNCD ; OFFSET OF PARAMETER LIST IN DI

    ; SIGNAL WORKSTATION PROGRAM FOR CONN$KEY SERVICE
    INT   7AH

    ENDM
```

Sample Program 1

```
;
;
;      MACRO NAME : CRT$Q
;      PARAMETERS : QSEGMENT -- SEGMENT OF QUEUE
;                  QOFFSET  -- OFFSET OF QUEUE
;                  QSIZE    -- SIZE OF QUEUE
;
;      FUNCTION :
;          CREATE A FIXED LENGTH QUEUE.
;

CRT$Q      MACRO  QSEGMENT,QOFFSET,QSIZE

; INITIALIZE PARAMETER LIST FOR CRT$Q
MOV  AX,QSEGMENT      ; SEGMENT ADDRESS INTO THE LIST
MOV  CQSEGMENT,AX
MOV  AX,QOFFSET       ; OFFSET INTO THE LIST
MOV  CQOFFSET,AX

; INITIALIZE REGISTERS FOR CRT$Q
MOV  AH,04H           ;
MOV  BL,00H           ; NO NAME SPECIFIED
MOV  CX,QSIZE         ; SIZE OF THE QUEUE
MOV  DX,0000H         ; DX MUST = 0 FOR THE REQUEST
MOV  DI, SEG CQOFFSET ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI             ; IN ES
MOV  DI,OFFSET CQOFFSET ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR CRT$Q SERVICE
INT  7AH

ENDM

;
;
;      MACRO NAME : CRT$TASK
;      PARAMETERS : TASK      -- TASK TO BE CREATED
;                  NAME       -- NAME OF THE TASK
;                  STACK      -- STACK FOR THE TASK
;                  PREEMPT    -- PREEMPTION TYPE
;                  PRIORITY   -- PRIORITY OF THE TASK
;                  DATA      -- VARIABLE IN THE TASK'S DATA SEGMENT
;
;      FUNCTION :
;          CREATE A TASK.
;

CRT$TASK   MACRO  TASK,NAME,STACK,PREEMPT,PRIORITY,DATA

; INITIALIZE PARAMETER LIST FOR CRT$TASK
MOV  AX,OFFSET STACK  ; GET THE OFFSET OF THE TASK'S STACK
ADD  AX,232           ; INCREMENT THE SP TO POINT AT THE
                        ; START OF REGISTER RESTORE AREA
MOV  CTTASKSS,AX      ; PUT TASK'S SP IN PARAMETER LIST
MOV  AX,SEG STACK     ; PUT TASK'S SS IN PARAMETER LIST
MOV  CTTASKSP,AX

MOV  BL,0             ; BL = 0 IF NO NAME SPECIFIED
```

```

IFNB <NAME>                                ; IF THERE IS A NAME SPECIFIED, PUT
                                           ; IT IN THE PARAMETER LIST
MOV     BL,1                                ; BL = 1 IF A NAME IS SPECIFIED
MOV     AX,SEG CTTSKNAME                    ; ES:DI POINT TO DESTINATION IN
MOV     ES,AX                               ; PARAMETER LIST
MOV     DI,OFFSET CTTSKNAME
MOV     SI,OFFSET NAME                      ; DS:SI POINT TO SOURCE OF THE NAME
MOV     CX,8                                ; MOVE 8 BYTES
REP     MOVSB                               ; COPY THE NAME INTO THE PARM LIST

ENDIF

; INITIALIZE THE TASK'S STACK
PUSH    DS                                  ; SAVE DS
MOV     AX,SEG STACK                        ; GET THE TASK'S STACK SEGMENT
MOV     DS,AX
MOV     SI,OFFSET STACK                    ; DS:SI NOW POINT TO THE TASK STACK
MOV     WORD PTR [SI+254],0F242H           ; SET FLAGS IN THE STACK
MOV     AX,SEG TASK
MOV     WORD PTR [SI+252],AX                ; SET SEGMENT OF TASK IN STACK
MOV     AX,OFFSET TASK
MOV     WORD PTR [SI+250],AX                ; SET OFFSET OF TASK IN STACK

IFNB <DATA>
MOV     AX,SEG DATA
MOV     WORD PTR [SI+234],AX                ; SET DATA SEGMENT IN STACK
ENDIF

POP     DS                                  ; RESTORE DS

; INITIALIZE REGISTERS FOR CRT$TASK
MOV     AH,92H                              ;
MOV     BH,PREEMPT                          ; PREEMPTION TYPE IN BH
MOV     CX,PRIORITY                          ; PRIORITY IN CX
MOV     DI,SEG CTTASKSS                      ; SEGMENT ADDRESS OF PARAMETER LIST
MOV     ES,DI                                ; IN ES
MOV     DI,OFFSET CTTASKSS                   ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR CRT$TASK SERVICE
INT     7AH

ENDM

;
;
; MACRO NAME : DEL$ENT
; PARAMETERS : ENTRYID -- ID OF THE ENTRY TO BE DELETED
; FUNCTION :
;           DELETE AN ENTRY FORM THE SYSTEM.
;
;
DEL$ENT    MACRO    ENTRYID

; INITIALIZE REGISTERS FOR DEL$ENT
MOV     AH,06H
MOV     CX,0000H
MOV     DX,ENTRYID                          ; DX = ENTRY ID

; SIGNAL WORKSTATION PROGRAM FOR DEL$ENT SERVICE
INT     7AH

ENDM

```

Sample Program 1

```
;
;
;      MACRO NAME : DISC$KEY
;      PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'KEYBOARD'
;                  SESSID  -- SESSION ID
;
;      FUNCTION :
;          DISCONNECT THE KEY BOARD FROM THE SPECIFIED SESSION.
;
;

DISC$KEY MACRO  SERVTYPE,SESSID,TASKID

; INITIALIZE PARAMETER LIST FOR DISC$KEY
MOV     DKRETNCD,00H          ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV     DKFXNID,00H          ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV     AL,SESSID            ; SESSION ID INTO THE LIST
MOV     DKSESSID,AL

IFNB    <TASKID>
MOV     AX,TASKID            ; IF A TASK ID WAS SPECIFIED, PUT IT
ELSE
;      IN THE LIST
MOV     AX,0000H
ENDIF
MOV     DKTASKID,AX

; INITIALIZE REGISTERS FOR DISC$KEY
MOV     AH,09H
MOV     AL,02H
MOV     BH,80H
MOV     BL,20H
MOV     CX,0000H
MOV     DX,SERVTYPE          ; RESOLVED VALUE FOR 'KEYBOARD'
MOV     DI, SEG DKRETNCD      ; SEGMENT ADDRESS OF PARAMETER LIST
MOV     ES,DI                ; IN ES
MOV     DI,OFFSET DKRETNCD    ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR DISC$KEY SERVICE
INT     7AH

ENDM

;
;
;      MACRO NAME : DISP$RET
;      PARAMETERS : WAITTYPE -- WAIT TYPE
;
;      FUNCTION :
;          RETURN TO THE DISPATCHER.
;
;

DISP$RET MACRO  WAITTYPE

; INITIALIZE REGISTERS FOR DISP$RET
MOV     AH,95H
MOV     BL,WAITTYPE          ; WAIT TYPE IN BL

; SIGNAL WORKSTATION PROGRAM FOR DISP$RET SERVICE
INT     7AH

ENDM
```

```

;
;      MACRO NAME : ENQUEUE
;      PARAMETERS : NUMBYTES -- NUMBER OF BYTES TO BE ENQUEUED
;                  QUEUEID  -- FIXED LENGTH QUEUE ID
;                  DATANAME -- MEMORY LOCATION NAME OF THE DATA
;
;      FUNCTION :
;          USE THIS SERVICE TO ENQUEUE DATA ON THE SPECIFIED FIXED
;          LENGTH QUEUE ID.
;
;

```

```
ENQUEUE MACRO NUMBYTES,QUEUEID,DATANAME
```

```

; INITIALIZE REGISTERS FOR ENQUEUE
MOV  AH,0CH                ;
MOV  CX,NUMBYTES           ; NUMBER OF BYTES
MOV  DX,QUEUEID            ; QUEUE ID
MOV  DI, SEG DATANAME       ; SEGMENT ADDRESS OF DATA
MOV  ES,DI                 ; IN ES
MOV  DI,OFFSET DATANAME    ; OFFSET OF DATA IN DI

; SIGNAL WORKSTATION PROGRAM FOR ENQUEUE SERVICE
INT  7AH

ENDM

```

```

;
;      PARAMETERS : NR$SERVN -- LOCATION OF THE 8-BYTE
;                      SERVICE NAME. I.E. 'SESSMGR '
;                      NR$SERVT -- RETURN CODE FROM PARAMETER LIST
;
;

```

```
NAME$RES MACRO NR$SERVN,NR$SERVT
```

```

; SET UP REGISTERS NAME$RES
MOV  AX,SEG NR$SERVN       ; SEGMENT ADDRESS OF PARM LIST
MOV  ES,AX                 ; ES = SEGM ADDRESS OF PARM LIST
MOV  AH,81H               ; AH = X'81'
MOV  CX,0000H             ; CX = X'0000'
MOV  DI,OFFSET NR$SERVN   ; DI = OFFSET ADDR. OF PARM LIST

; SIGNAL WORKSTATION PROGRAM FOR NAME$RES SERVICE
INT  7AH

; RETURN SERVICE TYPE ID TO CALLER
MOV  NR$SERVT,DX

ENDM

```

```

;
;      MACRO NAME : QUERY$ID
;      PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'SESSMGR '
;                  NAMEARRY  -- NAME ARRAY
;                  OPTION    -- OPTION BYTE
;                  DATA     -- DATA BYTE
;                  LONGNAME  -- SESSION LONG NAME
;
;

```

Sample Program 1

```
;      FUNCTION :
;      GET THE SESSION ID(S) OF THE SESSION(S) SPECIFIED BY
;      THE OPTION AND DSTS BYTES AND RETURNS THEM IN THE NAME
;      ARRAY.
;      NOTE - THE NAME ARRAY IS SET UP BY THE USER AND MUST HAVE
;              THE LENGTH OF THE ARRAY CONTAINED IN THE 1ST BYTE.
;

QUERY$ID MACRO  SERVTYPE,NAMEARRY,OPTION,DATA,LONGNAME

; INITIALIZE PARAMETER LIST FOR QUERY$ID
MOV  QDRETNCD,00H      ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV  QDFXNID,00H      ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV  AL,OPTION        ; OPTION BYTE INTO THE LIST
MOV  QDOPTION,AL
MOV  AL,DATA          ; DATA BYTE INTO THE LIST
MOV  QDDATA,AL
MOV  AX,OFFSET NAMEARRY ; NAME ARRAY OFFSET INTO THE LIST
MOV  QDNAMOFF,AX
MOV  AX,SEG NAMEARRY   ; NAME ARRAY SEGMENT INTO THE LIST
MOV  QDNAMSEG,AX

IFNB <LONGNAME>      ; CHECK IF A LONG NAME WAS SPECIFIED

CLD                  ; COPY DIRECTION = FORWARD
MOV  AX,SEG QDLNGNAM
MOV  ES,AX            ; ES:DI POINTS TO DESTINATION IN PARM
MOV  DI,OFFSET QDLNGNAM ; LIST
MOV  SI,OFFSET LONGNAME ; DS:SI POINTS TO SOURCE OF LONG NAME
MOV  CX,8              ; MOVE 8 BYTES
REP  MOVSB            ; COPY LONG NAME INTO THE PARM LIST

ENDIF

; INITIALIZE REGISTERS FOR QUERY$ID
MOV  AH,09H
MOV  AL,01H
MOV  BH,80H
MOV  BL,20H
MOV  CX,0000H
MOV  DX,SERVTYPE      ; RESOLVED VALUE FOR 'SESSMGR '
MOV  DI, SEG QDRETNCD  ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI             ; IN ES
MOV  DI,OFFSET QDRETNCD ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR QUERY$ID SERVICE
INT  7AH

ENDM

;
;      MACRO NAME : Q$BAS$W
;      PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'SESSMGR '
;      FUNCTION :
;      FIND THE SESSION ID AND SHORT NAME FOR THE BASE WINDOW
;      OF AN ENVIRONMENT.
;
```

```

Q$BAS$W  MACRO  SERVTYPE

; INITIALIZE PARAMETER LIST FOR Q$BAS$W
MOV  QSRETNCD,00H      ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV  QSFYNID,00H      ; FUNCTION ID MUST = 0 BEFORE REQUEST

; INITIALIZE REGISTERS FOR Q$BAS$W
MOV  AH,09H
MOV  AL,0AH
MOV  BH,80H
MOV  BL,20H
MOV  CX,0FFH
MOV  DX,SERVTYPE      ; RESOLVED VALUE FOR 'SESSMGR '
MOV  DI, SEG QSRETNCD  ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI             ; IN ES
MOV  DI,OFFSET QSRETNCD ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR Q$BAS$W SERVICE
INT  7AH

ENDM

;
;
; MACRO NAME : Q$TASK
; FUNCTION :
; GET THE ID OF THE CURRENT ACTIVE TASK.
;

Q$TASK  MACRO

; INITIALIZE REGISTERS FOR Q$TASK
MOV  AH,9CH

; SIGNAL WORKSTATION PROGRAM FOR Q$TASK SERVICE
INT  7AH

ENDM

;
;
; MACRO NAME : READ$KEY
; PARAMETERS : SERVTYPE -- SERVICE TYPE
;              SESSID  -- SESSION ID
; FUNCTION :
; READ KEYSTROKE DATA.
;

READ$KEY  MACRO  SERVTYPE,SESSID,TASKID

; INITIALIZE PARAMETER LIST FOR READ$KEY
MOV  RKRETNCD,00H      ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV  RKFXNID,00H      ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV  AL,SESSID        ; SESSION ID INTO THE LIST
MOV  RKSESSID,AL

IFNB <TASKID>
MOV  AX,TASKID        ; IF A TASK ID WAS SPECIFIED PUT IT
ELSE
; IN THE LIST, ELSE PUT IN A 0
MOV  AX,0
ENDIF
MOV  RKTASKID,AX

```


Sample Program 1

```
; INITIALIZE REGISTERS FOR READ$KEY
MOV  AH,09H
MOV  AL,03H
MOV  BH,80H
MOV  BL,20H
MOV  CX,0FFH
MOV  DX,SERVTYPE      ; SERVICE TYPE IN DX
MOV  DI,SEG RKRETNCD   ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI             ; IN ES
MOV  DI,OFFSET RKRETNCD ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR READ$KEY SERVICE
INT  7AH

ENDM
```

```
;
;
; MACRO NAME : SET$RDY
; PARAMETERS : TASKID  -- TASK ID
;              WAITTYPE -- WAIT TYPE
;
; FUNCTION :
;           SET THE SPECIFIED TASK TO THE "READY" STATE.
;
```

SET\$RDY MACRO TASKID,WAITTYPE

```
; INITIALIZE REGISTERS FOR SET$RDY
MOV  AH,02H
MOV  BL,WAITTYPE      ; WAIT TYPE IN BL
MOV  DX,TASKID        ; TASK ID IN DX

; SIGNAL WORKSTATION PROGRAM FOR SET$RDY SERVICE
INT  7AH

ENDM
```

```
;
;
; MACRO NAME : SETUNRDY
; PARAMETERS : TASKID  -- TASK ID
;              WAITTYPE -- WAIT TYPE
;
; FUNCTION :
;           SET THE SPECIFIED TASK TO THE "UNREADY" STATE.
;
```

SETUNRDY MACRO TASKID,WAITTYPE

```
; INITIALIZE REGISTERS FOR SETUNRDY
MOV  AH,05H
MOV  BL,WAITTYPE      ; WAIT TYPE IN BL
MOV  DX,TASKID        ; TASK ID IN DX

; SIGNAL WORKSTATION PROGRAM FOR SETUNRDY SERVICE
INT  7AH

ENDM
```

```

;
;
;      MACRO : WRIT$KEY
;      PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'KEYBOARD'
;                   SESSID  -- SESSION ID
;                   SCANCD  -- SCAN CODE
;                   SHIFST  -- SHIFT STATE
;                   LISTOFF -- LIST OFFSET
;                   LISTSEG -- LIST SEGMENT
;                   TASKID  -- CONNECTOR'S TASK ID
;
;      FUNCTION :
;      SEND A KEYSTROKE OR A LIST OF KEYSTROKES TO THE
;      SPECIFIED SESSION.
;
;
WRIT$KEY MACRO  SERVTYPE,SESSID,SCANCD,SHIFST,LISTOFF,LISTSEG,TASKID
LOCAL  WKEND

MOV  WKPARLST.WKRETNCD,0H ; WKRETCD MUST BE 0 FOR THE CALL
MOV  WKPARLST.WKFXNID,0H ; WKFXNID MUST BE 0 FOR THE CALL
MOV  AL,SESSID           ; PUT THE SESSION ID IN PARM LIST
MOV  WKPARLST.WKSESSID,AL

IFNB <SCANCD>             ; CHECK IF SENDING ONE KEYSTROKE
                           ; OR A LIST OF KEYSTROKES

; SENDING ONE KEYSTROKE

MOV  AL,SCANCD            ; PUT THE SCAN CODE IN THE PARM LIST
MOV  WKPARLST.WKSCNCOD,AL
MOV  AL,SHIFST            ; PUT SHIFT STATE IN THE PARM LIST
MOV  WKPARLST.WKSHFST,AL
MOV  AL,20H              ; PUT THE OPTION BYTE FOR SENDING
MOV  WKPARLST.WKOPTION,AL ; ONE CHARACTER IN THE PARM LIST

ELSE
; SENDING A LIST OF KEYSTROKES

MOV  AX,LISTOFF           ; PUT THE OFFSET ADDRESS OF THE LIST
MOV  WKPARLST.WKLSTOFF,AX ; INTO THE PARAMETER LIST
MOV  AX,LISTSEG           ; PUT THE SEGMENT ADDRESS OF THE
MOV  WKPARLST.WKLSTSEG,AX ; LIST INTO THE PARAMETER LIST

MOV  AL,30H              ; PUT OPTION BYTE FOR SENDING LIST
MOV  WKPARLST.WKOPTION,AL ; OF CHARACTERS IN THE PARM LIST

ENDIF

IFNB <TASKID>             ; IF A CONNECTOR'S TASK ID WAS
MOV  AX,TASKID            ; SPECIFIED, PUT IT IN THE LIST
ELSE
MOV  AX,0                 ; OTHERWISE PUT A 0 IN THE LIST
ENDIF
MOV  WKPARLST.WKTASKID,AX

; INITIALIZE THE REGISTERS FOR WRIT$KEY
MOV  AH,09H
MOV  AL,04H
MOV  BH,80H
MOV  BL,20H
MOV  CX,0000H

```

Sample Program 1

```
        MOV  DX,SERVTYPE          ; RESOLVED VALUE FOR 'KEYBOARD'
        MOV  DI, SEG WKPARLST     ; GET SEGMENT ADDRESS OF PARM LIST
        MOV  ES,DI                ; AND PUT IT IN ES
        MOV  DI, OFFSET WKPARLST  ; SET DI TO OFFSET OF PARM LIST

        INT  7AH                  ; PASS THE REQUEST TO THE API

        ENDM

        SUBTTL DATASEG
        PAGE

DATASEG  SEGMENT 'DATA'

; PARAMETER LIST FOR CONN$KEY

CKRETNCD DB 0                    ; RETURN CODE
CKFXNID  DB 0                    ; FUNCTION NUMBER
CKSESSID DB 0                    ; SESSION ID
CKRESRV1 DB 0                    ; RESERVED
CKEVENTQ DW 0                    ; EVENT QUEUE ID
CKKEYSTQ DW 0                    ; KEYSTROKE QUEUE ID
CKOPTION DB 40H                  ; OPTION BYTE (INTERCEPT ALL)
CKRESRV2 DB 0                    ; RESERVED

; PARAMETER LIST FOR CRT$Q

CQOFFSET DW 0                    ; OFFSET OF THE QUEUE
CQSEGMENT DW 0                    ; SEGMENT ADDRESS OF THE QUEUE
CQNAME    DB 8 DUP(0)            ; QUEUE NAME

; PARAMETER LIST FOR CRT$TASK

CTTASKSS DW 0                    ; TASK'S STARTING SS
CTTASKSP DW 0                    ; TASK'S STARTING SP
CTTSKNAM DB 8 DUP(?)            ; TASK NAME

; PARAMETER LIST FOR DISC$KEY

DKRETNCD DB 0                    ; RETURN CODE
DKFXNID  DB 0                    ; FUNCTION NUMBER
DKSESSID DB 0                    ; SESSION ID
DKRESRV1 DB 0                    ; RESERVED
DKTASKID DW 0                    ; CONNECTOR'S TASK ID
DKRESRV2 DB 0                    ; RESERVED

; PARAMETER LIST FOR QUERY$ID

QDRETNCD DB 0                    ; RETURN CODE
QDFXNID  DB 0                    ; FUNCTION NUMBER
QDOPTION DB 0                    ; OPTION BYTE
QDDATA   DB 0                    ; DATA BYTE
QDNAMOFF DW 0                    ; OFFSET OF NAME TABLE
QDNAMSEG DW 0                    ; SEGMENT OF NAME TABLE
QDLNGNAM DB 8 DUP(?)            ; SESSION LONG NAME
```

; PARAMETER LIST FOR Q\$BAS\$W

QSRETNC	DB	0	; RETURN CODE
QSFNID	DB	0	; FUNCTION NUMBER
QSENV	DB	0	; ENVIRONMENT ID
QSSID	DB	0	; SESSION ID
QSWIN	DB	0	; WINDOW SHORT NAME
QSRV	DB	0	; RESERVED

; PARAMETER LIST FOR READ\$KEY

RKRETNC	DB	0	; RETURN CODE
RKFNID	DB	0	; FUNCTION NUMBER
RKSID	DB	0	; SESSION ID
RKSRV1	DB	0	; RESERVED
RKTASKID	DW	0	; CONNECTOR'S TASK ID
	DB	20H	; MUST BE 20H
RKSRV2	DB	0	; RESERVED
RKSCANCD	DB	0	; SCAN CODE
RKSHIFST	DB	0	; SHIFT STATE
	DB	0	; NOT USED = 01H ON RETURN
	DB	0	; NOT USED = 00H ON RETURN

; PARAMETER LIST STRUCTURE FOR WRIT\$KEY

WRKYPARM	STRUC		
WKRETNC	DB	0	; RETURN CODE
WKFNID	DB	0	; FUNCTION NUMBER
WKSID	DB	0	; SESSION ID
WKSRV1	DB	0	; RESERVED
WKTASKID	DW	0	; CONNECTOR'S TASK ID
WKOPTION	DB	0	; OPTIONS BYTE
WKNUMKEY	DB	0	; KEYS SENT SUCCESSFULLY
WKSCNCOD	DB	0	; SCAN CODE OF THE KEY
WKSHFST	DB	0	; SHIFT STATE OF THE KEY
WRKYPARM	ENDS		
WRKYPAR2	STRUC		
	DB	8 DUP(00)	
WKLSTOFF	DW	0	; OFFSET OF LIST OF KEYSTROKES
WKLSTSEG	DW	0	; SEGMENT OF LIST OF KEYSTROKES
WRKYPAR2	ENDS		

; ALLOCATE STORAGE FOR THE PARAMETER LIST

WKPARLST WRKYPARM <>

PASSPRMT	DB	'ENTER PASSWORD : \$'	
PASSWORD	DB	21 DUP(?)	; PASSWORD (IN ASCII) 20 CHARACTERS + ; A 1 CHARACTER OVERFLOW
PCTASKID	DW	0	; THIS PROGRAM'S TASK ID
NAMARRAY	DB	14	; NAME ARRAY FOR QUERY\$ID FUNCTION
NUMSESS	DB	0	; NUMBER OF MATCHING SESSIONS
SHRTNAME	DB	0	; SESSION SHORT NAME
SESTYPE	DB	0	; SESSION TYPE
WSCSID	DB	0	; SESSION ID (WSCtrl)
SPARE	DB	0	
LONGNAME	DB	8 DUP(0)	; LONG NAME OF SESSION

Sample Program 1

```
WSCSTACK DB 256 DUP(0) ; STACK FOR THE LOCKWSC TASK
IPLSTACK DB 256 DUP(0) ; STACK FOR THE LOCKIPL TASK
LOCKWSID DW 0 ; LOCKWSC TASK ID
LOCKIPLID DW 0 ; LOCKIPL TASK ID
KYBDNAME DB 'KEYBOARD' ; PARM LIST FOR NAME$RES ON KEYBOARD
SMGRNAME DB 'SESSMGR ' ; PARM LIST FOR NAME$RES ON SESSMGR
KEYBOARD DW 0 ; RESOLVED ID FOR 'KEYBOARD'
SESSMGR DW 0 ; RESOLVED ID FOR 'SESSMGR '
WSQUEID DW 0 ; SVC ID OF THE KEYSTROKE QUEUE
WSQSIZE DW 64 ; SIZE OF THE KEYSTROKE QUEUE
WSQSEG DW 0 ; SEGMENT OF THE KEYSTROKE QUEUE
WSQOFF DW 0 ; OFFSET OF THE KEYSTROKE QUEUE
WSQUEUE DB 64 DUP(?) ; THE KEYSTROKE QUEUE
IPQUEID DW 0 ; SVC ID OF THE KEYSTROKE QUEUE
IPQSIZE DW 64 ; SIZE OF THE KEYSTROKE QUEUE
IPQSEG DW 0 ; SEGMENT OF THE KEYSTROKE QUEUE
IPQOFF DW 0 ; OFFSET OF THE KEYSTROKE QUEUE
IPQUEUE DB 64 DUP(?) ; THE KEYSTROKE QUEUE
ENDDATA DB 90H,00H,01H,00H
; BYTES TO ENQUEUE TO SIGNAL THE TASKS TO
; STOP
ERRMSG DB 'ERROR. PROGRAM ABORTED.$'
DATASEG ENDS
STACKSEG SEGMENT STACK
DB 256 DUP(?)
STACKSEG ENDS
SUBTTL MAIN
PAGE
CODESEG SEGMENT 'CODE'
; ESTABLISH CONSTANTS
BAKSPACE EQU 08H ; ASCII FOR A BACKSPACE
BOX EQU 0B1H ; ASCII FOR A BOX
CR EQU 0DH ; ASCII FOR A CARRIAGE RETURN
LF EQU 0AH ; ASCII FOR A LINE FEED
SPACE EQU 20H ; ASCII FOR A SPACE
INNOECHO EQU 8 ; FUNCTION NUMBER FOR DOS INPUT WITH NO ECHO
MAIN PROC FAR
ASSUME CS:CODESEG,DS:DATASEG,SS:STACKSEG,ES:DATASEG
MOV AX,DATASEG ; ESTABLISH ADDRESSABILITY TO THE DATA
MOV DS,AX
```

```

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; PROMPT THE USER TO ENTER THE PASSWORD ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

PROMPT

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; READ THE PASSWORD INTO MEMORY.  USE SI TO INDEX INTO THE ;;
;; PASSWORD AREA.  AFTER EACH CHARACTER IS READ STORE IT AT ;;
;; LOCATION [SI], INCREMENT SI, AND DISPLAY A BOX TO CONFIRM ;;
;; THAT A CHARACTER WAS ENTERED.  IF THE CHARACTER WAS A ;;
;; BACKSPACE, THEN ERASE THE LAST BOX DISPLAYED AND DECREMENT;;
;; SI.  IF THE CHARACTER WAS A CARRIAGE RETURN, THEN THE ;;
;; PASSWORD IS FINISHED.  SAVE THE CARRIAGE RETURN AND GO ON ;;
;; TO LOCK THE KEYBOARD. ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

LEA    SI,PASSWORD    ; POINT SI TO THE START OF PASSWORD AREA

NEXTCHAR: DOSFUNCT INNOECHO    ; GET A CHARACTER FROM THE KEYBOARD, NO ECHO

CMP    AL,BAKSPACE    ; CHECK IF IT IS A BACK SPACE
JE     BACKUP         ; IF SO, GO TO BACKUP ROUTINE

MOV    [SI],AL        ; SAVE THE CHARACTER IN PASSWORD AREA
INC    SI

CMP    AL,CR          ; CHECK IF THIS IS THE END OF THE PASSWORD
JE     LOCK           ; IF SO, PROCEED TO LOCK THE KEYBOARD

DISPLAY BOX          ; DISPLAY A BOX TO CONFIRM CHAR. ENTERED

JMP    NEXTCHAR       ; GET THE NEXT PASSWORD CHARACTER

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; THIS ROUTINE IS JUMPED TO WHEN THE BACKSPACE KEY IS ;;
;; PRESSED WHILE ENTERING THE PASSWORD.  IT ALLOWS THE ;;
;; USER TO MAKE CORRECTIONS WHILE ENTERING THE PASS- ;;
;; WORD.  THE ROUTINE BACKS UP THE POINTER TO THE PASS- ;;
;; WORD BY ONE CHARACTER AND ERASES THE LAST BOX DIS- ;;
;; PLAYED TO THE SCREEN.  A CHECK IS MADE TO SEE IF THE ;;
;; PASSWORD POINTER IS ALREADY AT THE START OF THE ;;
;; PASSWORD BUFFER TO PREVENT BACKING THE POINTER PAST ;;
;; THE START OF THE BUFFER. ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

BACKUP:  CMP    SI,OFFSET PASSWORD ; CHECK IF WE'RE ALREADY AT THE START
JLE     NEXTCHAR    ; IF SO, SKIP BACKING UP

DEC     SI          ; POINT TO THE PREVIOUS CHARACTER

DISPLAY BAKSPACE    ; ECHO A BACKSPACE TO THE SCREEN
DISPLAY SPACE       ; ERASE THE LAST CHARACTER
DISPLAY BAKSPACE    ; ECHO A BACKSPACE TO THE SCREEN

JMP     NEXTCHAR

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; LOCK THE WSCtrl KEYBOARD SO THE USER CAN'T JUMP TO ;;
;; ANOTHER SESSION.  LOCK OUT THE PC IPL SEQUENCE TO ;;
;; PREVENT THE USER FROM RESTARTING THIS PC SESSION. ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

```

Sample Program 1

```
LOCK:      CALL  CLEAR                ; CLEAR THE SCREEN

Q$TASK     ; FIND THE TASK ID OF THIS PC PROGRAM
MOV  PCTASKID,DX    ; SAVE THE TASK ID IN PCTASKID
CHK4ERR

NAME$RES   SMGRNAME,SESSMGR
                ; FIND THE RESOLVED ID FOR 'SESSMGR '
CHK4ERR

NAME$RES   KYBDNAME,KEYBOARD
                ; FIND THE RESOLVED ID FOR 'KEYBOARD'
CHK4ERR

;;;;;;;;;;;;;
;; SET UP THE KEYSTROKE QUEUE AND CONNECT TO THE WSCTRL ;;
;; KEYBOARD FOR THE LOCKWSC TASK.                        ;;
;;;;;;;;;;;;;

QUERY$ID   SESSMGR,NAMARRAY,00H,01H
                ; GET THE SESSION ID FOR WSCTRL
CHK4ERR     QDRETNCD

MOV  WSQSEG,SEG WSQUEUE    ; INITIALIZE VARIABLES FOR KEY-
MOV  WSQOFF,OFFSET WSQUEUE ; BOARD QUEUE OFFSET AND SEGMENT

CRT$Q     WSQSEG,WSQOFF,WSQSIZE
                ; CREATE A QUEUE TO RECEIVE KEYSTROKES
MOV  WSQUEID,DX    ; SAVE THE KEYSTROKE QUEUE ID
CHK4ERR

CONN$KEY   KEYBOARD,WSCSEID,WSQUEID
                ; CONNECT TO WSCTRL KEYBOARD
CHK4ERR     CKRETNCD

;;;;;;;;;;;;;
;; SET UP THE KEYSTROKE QUEUE AND CONNECT TO THE PC ;;
;; KEYBOARD FOR THE LOCKIPL TASK.                    ;;
;;;;;;;;;;;;;

Q$BAS$W    SESSMGR    ; GET THE SESSION ID FOR THIS PC SESSION
CHK4ERR     QSRETNCD

MOV  IPQSEG,SEG IPQUEUE    ; INITIALIZE VARIABLES FOR KEY-
MOV  IPQOFF,OFFSET IPQUEUE ; BOARD QUEUE OFFSET AND SEGMENT

CRT$Q     IPQSEG,IPQOFF,IPQSIZE
                ; CREATE A QUEUE TO RECEIVE KEYSTROKES
MOV  IPQUEID,DX    ; SAVE THE KEYSTROKE QUEUE ID
CHK4ERR

CONN$KEY   KEYBOARD,QSSEID,IPQUEID
                ; CONNECT TO PC KEYBOARD
CHK4ERR     CKRETNCD

;;;;;;;;;;;;;
;; CREATE THE TASKS TO LOCK OUT WSCTRL AND THE PC IPL KEYS ;;
;; SET THE TASKS AT A HIGHER PRIORITY (59) THAN THIS MAIN ;;
;; TASK (60) SO THAT THEY CAN BE GIVEN CONTROL WHEN IT IS ;;
;; TIME TO CLEAN UP.                                     ;;
;;;;;;;;;;;;;
```

```

CRT$TASK LOCKWSC,,WSCSTACK,00H,59,LOCKWSID
                                ; CREATE THE TASK TO INTERCEPT WSCTRL KEYS
MOV  LOCKWSID,DX                ; SAVE THE TASK ID IN LOCKWSID
CHK4ERR

CRT$TASK LOCKIPL,,IPLSTACK,00H,59,LOCKIPID
                                ; CREATE THE TASK TO INTERCEPT PC IPL KEYS
MOV  LOCKIPID,DX                ; SAVE THE TASK ID IN LOCKIPID
CHK4ERR

;;;;;;;;;;;;;
;; START THE TASKS RUNNING ;;
;;;;;;;;;;;;;

SET$RDY  LOCKWSID,00H
                                ; SET THE TASK RUNNING (NO WAIT)
CHK4ERR

SET$RDY  LOCKIPID,00H
                                ; SET THE TASK RUNNING (NO WAIT)
CHK4ERR

;;;;;;;;;;;;;
;; READ IN KEYSTROKES AND CHECK FOR THE PASSWORD SEQUENCE ;;
;;;;;;;;;;;;;

PROMPT                                ; PROMPT THE USER FOR THE PASSWORD

TRYAGAIN: LEA  SI,PASSWORD          ; POINT SI TO THE START OF THE PASSWORD

NXTPCCHAR: DOSFUNCT  INNOECHO        ; READ A KEY WITHOUT ECHOING IT

CMP  AL,[SI]                      ; CHECK IF IT MATCHES THE PASSWORD CHAR.
JNE  TRYAGAIN                      ; IF NOT, START OVER

INC  SI                            ; POINT TO THE NEXT CHARACTER
CMP  BYTE PTR [SI],CR              ; CHECK IF WE'RE AT END OF THE PASSWORD
JNE  NXTPCCHAR                      ; IF NOT, CHECK NEXT PASSWORD CHARACTER

;;;;;;;;;;;;;
;; THE CORRECT PASSWORD WAS ENTERED.  START THE CLEAN UP. ;;
;; SEND A 90H SCAN CODE TO THE TASKS TO TELL THEM TO QUIT. ;;
;; RETURN TO THE DISPATCHER AFTER EACH 90H SCAN CODE IS    ;;
;; SENT TO A TASK.  THE TASKS WILL GET CONTROL AFTER A    ;;
;; RETURN TO THE DISPATCHER SINCE THEY ARE RUNNING AT A    ;;
;; HIGHER PRIORITY THAN THIS MAIN TASK.                    ;;
;; DISCONNECT THE TASKS FROM THEIR RESPECTIVE KEYBOARDS,  ;;
;; DELETE THE KEYSTROKING QUEUES THAT WERE CREATED FOR    ;;
;; EACH TASK, DELETE THE TASKS FROM THE SYSTEM, AND CLEAR ;;
;; THE SCREEN.                                             ;;
;;;;;;;;;;;;;

ENQUEUE  4,WSQUEID,ENDDATA
                                ; SEND THE DATA TO SIGNAL THE END TO THE
                                ; LOCKWSC TASK
CHK4ERR

DISP$RET  00H                    ; RETURN TO THE DISPATCHER TO GIVE THE TASK
                                ; CONTROL SO IT CAN PROCESS THE DATA

ENQUEUE  4,IPQUEID,ENDDATA
                                ; SEND THE DATA TO SIGNAL THE END TO THE
                                ; LOCKIPL TASK
CHK4ERR

```


Sample Program 1

```
DISP$RET 00H          ; RETURN TO THE DISPATCHER TO GIVE THE TASK
                   ; CONTROL SO IT CAN PROCESS THE DATA

DISC$KEY  KEYBOARD,WSCSID
                   ; DISCONNECT THE LOCKWSC TASK FROM WSCTRL
CHEK4ERR  DKRETNCD

DISC$KEY  KEYBOARD,QSSESSID
                   ; DISCONNECT THE LOCKIPL TASK FROM THIS PC
                   ; SESSION
CHEK4ERR  DKRETNCD

DEL$ENT   WSQUEID     ; DELETE THE LOCKWSC KEYSTROKE QUEUE
CHEK4ERR

DEL$ENT   IPQUEID     ; DELETE THE LOCKIPL KEYSTROKE QUEUE
CHEK4ERR

DEL$ENT   LOCKWSID    ; DELETE THE LOCKWSC TASK
CHEK4ERR

DEL$ENT   LOCKIPID    ; DELETE THE LOCKIPL TASK
CHEK4ERR

FINISH:  CALL  CLEAR          ; CLEAR THE SCREEN BEFORE EXITING
        MOV  AX,4C00H        ; RETURN TO DOS
        INT  21H

;
; PROCEDURE : CLEAR
; FUNCTION :
; CLEAR THE PC SCREEN.
;

; ESTABLISH CONSTANTS

SETCURS   EQU    02H          ; BIOS SET CURSOR FUNCTION NUMBER
VIDSTAT   EQU    0FH          ; BIOS GET VIDEO STATE FUNCTION NUMBER
WRITCHAR   EQU    0AH          ; BIOS WRITE CHARACTER WITHOUT ATTRIBUTE

CLEAR     PROC  NEAR

        MOV  AH,VIDSTAT      ; GET THE STATE OF THE SCREEN
        INT  10H

        MOV  DX,0000H        ; MOVE CURSOR TO HOME POSITION
        MOV  AH,SETCURS
        INT  10H

        MOV  AL,SPACE        ; DISPLAY A SCREENFUL OF BLANKS
        MOV  CX,2000          ; 25 X 80 = 2000 BLANKS
        MOV  AH,WRITCHAR
        INT  10H

        RET

CLEAR     ENDP
```

```

;
;   PROCEDURE : CHECKERR
;   FUNCTION :
;       THIS PROCEDURE IS PASSED TWO RETURN CODES IN CL AND BL.
;       BL CONTAINS A RETURN CODE FROM THE FIRST BYTE IN A PARAMETER
;       LIST. BOTH CL AND BL ARE CHECKED FOR 0'S. IF EITHER CONTAINS
;       A NON-ZERO RETURN CODE, AN ERROR MESSAGE IS DISPLAYED AND THE
;       PROGRAM IS TERMINATED.
;
;   NOTE : THIS IS A VERY SIMPLE ERROR HANDLER USED TO PRESERVE
;           PROGRAM FLOW AND IS NOT LISTED AS AN EXAMPLE OF AN
;           APPROPRIATE ERROR HANDLER. THIS ERROR HANDLER SIMPLY
;           TERMINATES THE PROGRAM WHEN AN ERROR IS ENCOUNTERED
;           LEAVING ANY RESOURCES, SUCH AS FIXED LENGTH QUEUES,
;           PRESENTATION SPACES, AND A CONNECTION TO THE WINDOW
;           SERVICES, STILL ALLOCATED. A MORE APPROPRIATE ERROR
;           HANDLER WOULD DELETE ALL RESOURCES BEFORE TERMINATING.
;
CHECKERR PROC NEAR

    CMP     CL,0           ; CHECK THE RETURN CODE IN CL
    JNE     ERROR

    CMP     BL,0           ; CHECK THE RETURN CODE PASSED IN BL
    JNE     ERROR

    RET                     ; RETURN TO THE CALLER

ERROR:    LEA     DX,ERRMSG   ; DISPLAY THE ERROR MESSAGE
          DOSFUNC DISPSTR

          JMP     FINISH      ; TERMINATE THE PROGRAM AND EXIT TO DOS

CHECKERR ENDP

MAIN      ENDP

          SUBTTL LOCKWSC
          PAGE

;
;   PROCEDURE : LOCKWSC
;   FUNCTION :
;       THIS PROCEDURE IS KICKED OFF AS A SEPARATE TASK. ITS JOB
;       IS TO INTERCEPT AND DISCARD ALL WSCTRL KEYSTROKES. THIS
;       PREVENTS THE USER FROM ACCESSING THE WSCTRL FUNCTIONS.
;       THIS TASK WILL STOP INTERCEPTING KEYS AND SET ITSELF
;       "UNREADY" WHEN IT READS A 90H SCAN CODE. A 90H SCAN CODE IS A
;       SIGNAL FROM THE MAIN TASK TO STOP PROCESSING.
;
LOCKWSC   PROC

GETWSKEY: READ$KEY  KEYBOARD,WSCSEID,PCTASKID
          ; GRAB A KEYSTROKE OUT OF THE QUEUE WHEN
          ; ONE IS READY

    CMP     RKSCANCD,90H   ; CHECK FOR THE SCAN CODE THAT SIGNALS THE
          ; END
    JE      WSFINISH       ; IF SO, QUIT READING KEYSTROKES AND SET
          ; THIS TASK UNREADY
    JMP     GETWSKEY       ; TRY FOR ANOTHER KEYSTROKE.

```

```
WSFINISH: SETUNRDY  LOCKWSID,00H
                ; SET THIS TASK UNREADY (NO WAIT)
```

SUBTTL LOCKIPL
PAGE

```

;
;      PROCEDURE : LOCKIPL
;      FUNCTION :
;
;          THIS PROCEDURE IS KICKED OFF AS A SEPARATE TASK.  ITS JOB IS TO
;          INTERCEPT AND DISCARD THE PC IPL KEY SEQUENCE (CTRL - ALT - DEL).
;          THIS PREVENTS THE USER FROM RE-IPLING THE PC SESSION.
;
;          THIS IS ACCOMPLISHED BY READING THE PC KEYSTROKES AND CHECKING
;          FOR THE CTRL-ALT-DEL KEYSTROKE.  THIS KEYSTROKE IS DROPPED WHILE ALL
;          OTHERS ARE WRITTEN BACK TO THE PC SESSION.
;
;          THE BREAK KEYSTROKES ARE NOT SENT BACK TO THE PC SESSION SINCE
;          THE WRIT_KEY API SERVICE WILL TAKE CARE OF SENDING BREAK KEYSTROKES
;          TO THE PC SESSION.  HOWEVER, ANY BREAK KEYSTROKES SENT THROUGH THE
;          API SERVICE WILL NOT AFFECT THE KEYSTROKING IN THE PC SESSION.  BY
;          DROPPING THE BREAK KEYSTROKES THE SPEED OF THIS ROUTINE IS IMPROVED.
;
;          THIS TASK WILL STOP INTERCEPTING KEYS AND SET ITSELF "UNREADY"
;          WHEN IT READS A 90H SCAN CODE.  A 90H SCAN CODE IS A SIGNAL FROM THE
;          MAIN TASK TO STOP PROCESSING.
;
;

```

```
DEL_SCAN EQU 71H ; 3270 PC SCAN CODE FOR THE DELETE KEY
CTRL EQU 00001000B ; SHIFT STATE FOR CTRL KEY PRESSED
ALT EQU 00000100B ; SHIFT STATE FOR ALT KEY PRESSED
```

```
GETPCKEY: READ_KEY    KEYBOARD,QSSESSID,PCTASKID
                                ; GRAB A KEYSTROKE OUT OF THE QUEUE WHEN ONE IS
                                ;   READY
```

```
CMP     RKSCANCD,0F0H ; CHECK FOR THE "BREAK" KEYSTROKE
JNE     MAKEKEY       ; IF NOT THE BREAK THEN IT IS A MAKE KEYSTROKE
```

```

READ_KEY    KEYBOARD,QSSESSID,PCTASKID
            ; GET THE SCAN CODE FOR THE BREAKING KEYSTROKE
            ; FROM THE QUEUE.

```

```
JMP    GETPCKEY      ; GET ANOTHER KEYSTROKE
```

```
MAKEKEY:  CMP    RKSCANCD,90H    ; CHECK FOR THE SCAN CODE THAT SIGNALS THE END
          JNE    VALIDKEY        ; IF NOT, CONTINUE PROCESSING
          JMP     IPFINISH        ; OTHERWISE, QUIT READING KEYSTROKES AND SET THIS
                                   ; TASK UNREADY
```

[illegible]

```
VALIDKEY:  CMP    RKSCANCD,DEL_SCAN ; CHECK FOR THE DEL KEY
           JNE     OKKEY             ; IF NOT, SEND THE KEYSTROKE TO THE PC

           TEST    RKSHIFST,CTRL ; CHECK IF THE CTRL KEY IS PRESSED
           JZ      OKKEY             ; IF NOT, SEND THE KEYSTROKE TO THE PC

           TEST    RKSHIFST,ALT  ; CHECK IF THE ALT KEY IS ALSO PRESSED
           JZ      OKKEY             ; IF NOT, SEND THE KEYSTROKE TO THE PC

           JMP     GETPCKEY        ; GET ANOTHER KEYSTROKE

OKKEY:     WRIT_KEY  KEYBOARD,QSSESSID,RKSCANCD,RKSHIFST,,,PCTASKID
           ; SEND THE KEYSTROKE TO THE PC SESSION

           JMP     GETPCKEY        ; TRY FOR ANOTHER KEYSTROKE.

IPFINISH:  SETUNRDY  LOCKIPID,00H
           ; SET THIS TASK UNREADY (NO WAIT)

LOCKIPL    ENDP

CODESEG    ENDS

           END
```


Chapter 26. Sample Program 2

PAGE 80,132

```

;
;      PROGRAM      : LOCK$WSC
;
;      FUNCTION :
;      THIS PROGRAM WILL STOP A USER FROM USING THE WORK STATION
;      CONTROL WINDOWING FEATURES.  NOTHING WILL HAPPEN IF THE WSCTRL
;      KEY IS PRESSED.  THIS ALLOWS A USER TO SET UP THE SCREENS VIA
;      THE RESTORE UTILITY BUT NOT TO CHANGE THE SET-UP OF THE WINDOWS
;      ON THE SCREENS.
;
;      THE USER WILL ONLY BE ALLOWED TO JUMP BETWEEN WINDOWS AND
;      SCREENS, AND ENLARGE THE WINDOWS.
;
;      THIS WILL BE IMPLEMENTED BY CREATING A TASK THAT WILL INTERCEPT
;      ALL WORK STATION CONTROL KEYSTROKES AND ACCEPT ONLY THE JUMP
;      AND ENLARGE KEYS.  THIS PROGRAM CAN BE USED AS A SYSTEM
;      EXTENSION.  WHICH MEANS THAT THE WORK STATION CONTROL KEY IS
;      DISABLED WHEN THE 3270 PERSONAL COMPUTER CONTROL PROGRAM IS
;      LOADED.
;
;      IF THIS PROGRAM IS USED AS A PC APPLICATION, AND IT IS RUN
;      IN A MULTI-DOS SYSTEM, THEN IT MUST HAVE A PIF INDICATING THIS
;      PROGRAM IS ALL GOOD.
;      THIS PROGRAM WILL DEMONSTRATE THE FOLLOWING
;      API FUNCTIONS:
;
;      NAME RESOLUTION
;      QUERY SESSION ID
;      CONNECT TO THE KEYBOARD
;      CREATE A FIXED LENGTH QUEUE
;      CREATE A TASK
;      SET READY
;      READ INPUT
;      WRITE KEYSTROKE
;      GET REQUEST COMPLETION
;      QUERY ACTIVE TASK
;
;  DOS FUNCTION CALLS
DISPSTRG EQU      09H      ; PRINTING A STRING
DISPCHAR EQU      02H      ; PRINTING A CHARACTER
;
;  CONSTANTS
QRY$TYPE EQU      00H      ; OPTION BYTE FOR QUERY ID, QUERY BY
;      SESSION TYPE
WSCTRL EQU        01H      ; DATA BYTE FOR QUERY ID,WORK STATION
;      CONTROL
WSCTRL$K EQU      04H      ; SCAN CODE FOR WORK STATION CONTROL KEY
NON$PRE EQU       01H      ; A TASK IS NON-PREEMPTABLE
PRIO EQU          60      ; PRIORITY OF A TASK

```



```

QUERY$ID MACRO QIDARRYN,QIDOPT,QIDDATA,LSESSNAM,SERVTYPE,QID$RETC

    MOV AX,SEG QDRCODE ; ADDRESSABILITY TO
    MOV ES,AX ; PARAMETER LIST
    MOV DI,OFFSET QDRCODE
                                ; USING ES:DI

    ; INITIALIZE PARAMETER LIST FOR QUERY$ID
    MOV QDRCODE,00H ; RETURN CODE = 0 ON REQUEST
    MOV QDFXNID,00H ; FUNCTION CODE=0 ON REQUEST
    MOV AL,QIDOPT
    MOV QDOPT,AL ; OPTION BYTE
    MOV AL,QIDDATA
    MOV QDDATA,AL ; DATA BYTE
    MOV QDAOFF,OFFSET QIDARRYN
                                ; ARRAY OFFSET
    MOV QDASEG,SEG QIDARRYN
                                ; ARRAY SEGMENT

    ; IS THERE A LONG SESSION NAME ?
    IFNB <LSESSNAM> ; IF NAME IS SPECIFIED,
    CLD ; THEN BEGIN MOVING NAME
    PUSH DS ; INTO PARAMETER LIST
    MOV CX,4 ; NAME IS FOUR WORDS LONG
    MOV SI,OFFSET LSESSNAM
                                ; SOURCE OFFSET IN SI
    MOV AX,SEG LSESSNAM
    MOV DS,AX ; SOURCE SEGMENT IN DS
    MOV DI,OFFSET QDLNAM
                                ; DESTINATION OFFSET IN DI
    REP MOVSW ; MOVE SESSION NAME TO PARAMETER LIST
    POP DS
    ENDIF

    ; SET UP REGISTERS FOR QUERY$ID
    MOV AX,0901H ; AH = 09H, AL = 01H
    MOV BX,8020H ; BH = 80H, BL = 20H
    MOV CX,0000H ; CX = 0000H
    MOV DX,SERVTYPE ; DX = SESSION SERVICE TYPE
    MOV DI,OFFSET QDRCODE
                                ; OFFSET ADDRESS OF PARAM LIST

    ; REQUEST QUERY$ID SERVICE
    INT 7AH

    ; SEND RETURN CODE BACK TO CALLER
    MOV BL,QDRCODE
    MOV QID$RETC,BL

    ENDM

;
;
; MACRO NAME : CONN$KEY
; PARAMETERS : SERVTYPE -- SERVICE TYPE
;              SESSID -- SESSION ID
;              KEYSTQ -- KEYSTROKE QUEUE ID (OPTIONAL)
;              EVNTQ -- EVENT QUEUE ID (OPTIONAL)
;              OPT -- RECEIVE ALL, SOME OR NO KEYSTROKES
;
; FUNCTION :
;           CONNECT THE KEYBOARD TO THE SPECIFIED SESSION.
;

```


Sample Program 2

```
CONN$KEY MACRO  SERVTYPE,SESSID,KEYSTQ,EVNTQ,OPT

; INITIALIZE PARAMETER LIST FOR CONN$KEY
MOV  CKRETNCD,00H      ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV  CKFXNID,00H       ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV  AL,SESSID        ; SESSION ID INTO THE LIST
MOV  CKSESSID,AL
MOV  AL,OPT           ; OPTION BYTE INTO LIST
MOV  CKOPTION,AL

IFNB <KEYSTQ>
MOV  AX,KEYSTQ        ; IF A KEYSTROKE QUEUE WAS SPECIFIED,
ELSE                                ; PUT IT INTO THE LIST
MOV  AX,0
ENDIF
MOV  CKKEYSTQ,AX

IFNB <EVNTQ>
MOV  AX,EVNTQ         ; IF AN EVENT QUEUE WAS SPECIFIED,
ELSE                                ; PUT IT INTO THE LIST
MOV  AX,0
ENDIF
MOV  CKEVENTQ,AX

; INITIALIZE REGISTERS FOR CONN$KEY
MOV  AH,09H
MOV  AL,01H
MOV  BH,80H
MOV  BL,20H
MOV  CX,0000H
MOV  DX,SERVTYPE      ; SERVICE TYPE IN DX
MOV  DI,SEG CKRETNCD  ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI            ; IN ES
MOV  DI,OFFSET CKRETNCD ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR CONN$KEY SERVICE
INT  7AH

ENDM

;
;
; MACRO NAME : CRT$Q
; PARAMETERS : Q      -- Q IS THE MEMORY LOCATION NAME
;                                FOR THE QUEUE
;                QNAME -- QNAME IS THE USERS NAME FOR THE
;                                NEWLY CREATED QUEUE. QNAME IS
;                                OPTIONAL. CRT$Q EXPECTS QNAME TO BE
;                                REFERENCED BY THE DS REGISTER
;                NUMBYTES -- NUMBER OF BYTES IN THE QUEUE
;                RET$ID  -- CRT$Q RETURNS THE QUEUES ID
;
; FUNCTION :
;   USE THE CRT$Q SERVICE TO CREATE A FIXED LENGTH QUEUE
; ENTRY.
;
;
CRT$Q MACRO Q,QNAME,NUMBYTES,RET$ID

MOV  AX,SEG CQQOFFS    ; ADDRESSABILITY TO
MOV  ES,AX             ; PARAMETER LIST
MOV  DI,OFFSET CQQOFFS ; USING ES:DI
```

```

; INITIALIZE FIRST 2 ENTRIES OF PARAMETER LIST
MOV  CQQOFFS,OFFSET Q    ; OFFSET OF QUEUE
MOV  CQSEGM,SEG Q        ; SEGMENT OF QUEUE

; USER SPECIFY A QUEUE NAME?
IFNB <QNAME>              ; IF THERE IS A QUEUE NAME THEN
MOV  BL,01H               ; INDICATE A QNAME IS SPECIFIED
CLD                        ; BEGIN MOVING QNAME TO THE PARAM LIST
MOV  CX,4                 ; QNAME IS FOUR WORDS LONG
MOV  SI,OFFSET QNAME      ; SOURCE OFFSET OF QUEUE
MOV  DI,OFFSET CQQNAME; DESTINATION OFFSET IS CQQNAME
REP  MOVSW                ; MOVE QNAME TO PARAMETER LIST
ELSE
MOV  BL,00H               ; ELSE INDICATE QNAME IS NOT SPECIFIED
ENDIF

; INITIALIZE REGISTERS FOR CRT$Q
MOV  CX,NUMBYTES          ; CX = NUMBER OF BYTES FOR QUEUE
MOV  AH,04H
MOV  DI,OFFSET CQQOFFS;DI = OFFSET OF PARAM LIST

; SIGNAL WORKSTATION PROGRAM FOR CRT$Q SERVICE
INT  7AH

MOV  RET$ID,DX            ; RETURN THE QUEUE ID TO CALLER

ENDM

;
; MACRO NAME : CRT$TASK
; PARAMETERS : TASK      -- TASK TO BE CREATED
;              NAME      -- NAME OF THE TASK
;              STACK     -- STACK FOR THE TASK
;              PREEMPT   -- PREEMPTION TYPE
;              PRIORITY  -- PRIORITY OF THE TASK
;              DATA     -- A DATA ITEM IN DATASEG ACCESSED BY THE
;                          TASK IN ORDER TO PUT THE DATA SEG IN
;                          TASKS STACK
;
; FUNCTION :
;   CREATES A TASK .
;
;
CRT$TASK MACRO  TASK,NAME,STACK,PREEMPT,PRIORITY,DATA

; INITIALIZE PARAMETER LIST FOR CRT$TASK
MOV  AX,OFFSET STACK      ; GET THE OFFSET OF THE TASK'S STACK
ADD  AX,232               ; INCREMENT THE SP TO POINT AT THE
                          ; START OF REGISTER RESTORE AREA
MOV  CTTSKOFF,AX          ; PUT TASK'S SP IN PARAMETER LIST
MOV  AX,SEG STACK         ; PUT TASK'S SS IN PARAMETER LIST
MOV  CTTSKSEG,AX

MOV  BL,0                 ; BL = 0 IF NO NAME SPECIFIED
IFNB <NAME>                ; IF THERE IS A NAME SPECIFIED, PUT
                          ; IT IN THE PARAMETER LIST
MOV  BL,1                 ; BL = 1 IF A NAME IS SPECIFIED
MOV  AX,SEG CTTSKNAM      ; ES:DI POINT TO DESTINATION IN PARM
MOV  ES,AX                ; LIST
MOV  DI,OFFSET CTTSKNAM
MOV  SI,OFFSET NAME       ; DS:SI POINT TO SOURCE OF THE NAME
MOV  CX,8                 ; MOVE 8 BYTES
REP  MOVSB                ; COPY THE NAME INTO THE PARM LIST

ENDIF

```

Sample Program 2

```
; INITIALIZE THE TASK'S STACK
PUSH DS                ; SAVE DS
MOV AX,SEG STACK       ; GET THE TASK'S STACK SEGMENT
MOV DS,AX
MOV SI,OFFSET STACK    ; DS:SI NOW POINT TO THE TASK STACK
MOV WORD PTR [SI+254],0F242H
                        ; SET FLAGS IN THE STACK

MOV AX,SEG TASK
MOV WORD PTR [SI+252],AX
                        ; SET SEGMENT OF TASK IN STACK

MOV AX,OFFSET TASK
MOV WORD PTR [SI+250],AX
                        ; SET OFFSET OF TASK IN STACK

MOV AX,SEG DATA
                        ; YOU NEED THE DATA SEGMENT
MOV WORD PTR [SI+234],AX
                        ; TO GET TO THE VARIABLES
POP DS                 ; RESTORE DS

; INITIALIZE REGISTERS FOR CRT$TASK
MOV AH,92H             ;
MOV BH,PREEMPT         ; PREEMPTION TYPE IN BH
MOV CX,PRIORITY        ; PRIORITY IN CX
MOV DI,SEG CTTSKOFF    ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI              ; IN ES
MOV DI,OFFSET CTTSKOFF ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR CRT$TASK SERVICE
INT 7AH

ENDM
```

```
;
;
; MACRO NAME : SET$RDY
; PARAMETERS : REPLYTYP -- REPLY TYPE
;              WAITTYPE -- WAIT TYPE
;              PRIORITY -- PRIORITY
;              SERVTYPE -- SERVICE TYPE
;              SESSID -- SESSION ID
; FUNCTION :
;           SET THE SPECIFIED TASK TO THE "READY" STATE.
;
```

SET\$RDY MACRO TASKID,WAITTYPE

```
; INITIALIZE REGISTERS FOR SET$RDY
MOV AH,02H             ;
MOV BL,WAITTYPE        ; WAIT TYPE IN BL
MOV DX,TASKID          ; TASK ID IN DX

; SIGNAL WORKSTATION PROGRAM FOR SET$RDY SERVICE
INT 7AH

ENDM
```

```

;
;
;   MACRO NAME : READ$KEY
;   PARAMETERS : SERVTYPE -- SERVICE TYPE
;                 SESSID  -- SESSION ID
;                 TASKID   -- ID OF TASK (OPTIONAL)
;
;   FUNCTION :
;       THIS SERVICE READS KEYSTROKE DATA FROM A SESSION.
;

READ$KEY MACRO  SERVTYPE,SESSID,TASKID

; INITIALIZE PARAMETER LIST FOR READ$KEY
MOV     RKRETNCD,00H      ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV     RKFXNID,00H      ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV     AL,SESSID        ; SESSION ID INTO THE LIST
MOV     RKSESSID,AL
IFNB    <RKTASKID>
MOV     AX,TASKID        ; IF A TASK ID WAS SPECIFIED, PUT IT
ELSE
MOV     AX,0             ; IN THE LIST, ELSE PUT IN A 0
ENDIF
MOV     RKTASKID,AX

; INITIALIZE REGISTERS FOR READ$KEY
MOV     AH,09H
MOV     AL,03H
MOV     BH,80H
MOV     BL,20H
MOV     CX,0FFH
MOV     DX,SERVTYPE      ; SERVICE TYPE IN DX
MOV     DI,SEG RKRETNCD  ; SEGMENT ADDRESS OF PARAMETER LIST
MOV     ES,DI            ; IN ES
MOV     DI,OFFSET RKRETNCD ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR READ$KEY SERVICE
INT     7AH

ENDM

;
;
;   MACRO : WRIT$KEY
;   PARAMETERS : SERVTYPE -- SERVICE TYPE
;                 SESSID  -- SESSION ID
;                 SCANCD  -- SCAN CODE
;                 SHIFST  -- SHIFT STATE
;                 LISTOFF -- LIST OFFSET (OPTIONAL)
;                 LISTSEG -- LIST SEGMENT (OPTIONAL)
;                 TASKID  -- CONNECTOR'S TASK ID (OPTIONAL)
;
;   FUNCTION :
;       SEND A KEYSTROKE OR A LIST OF KEYSTROKES TO THE
;       SPECIFIED SESSION.
;

WRIT$KEY MACRO  SERVTYPE,SESSID,SCANCD,SHIFST,LISTOFF,LISTSEG,TASKID
LOCAL  WKEND

MOV     WKPARLST.WKRETNCD,0H ; WKRETNCD MUST BE 0 FOR THE CALL
MOV     WKPARLST.WKFXNID,0H  ; WKFXNID MUST BE 0 FOR THE CALL
MOV     AL,SESSID            ; PUT THE SESSION ID IN PARM LIST
MOV     WKPARLST.WKSESSID,AL

IFNB    <SCANCD>
; CHECK IF SENDING ONE KEYSTROKE
; OR A LIST OF KEYSTROKES

```

Sample Program 2

```
; SENDING ONE KEYSTROKE

MOV  AL,SCANCD          ; PUT THE SCAN CODE IN THE PARM LIST
MOV  WKPARLST.WKSCNCOD,AL
MOV  AL,SHIFST          ; PUT SHIFT STATE IN THE PARM LIST
MOV  WKPARLST.WKSHFST,AL

MOV  AL,20H             ; PUT THE OPTION BYTE FOR SENDING
MOV  WKPARLST.WKOPTION,AL ; ONE CHARACTER IN THE PARM LIST

ELSE
; SENDING A LIST OF KEYSTROKES

MOV  AX,LISTOFF          ; PUT THE OFFSET ADDRESS OF THE LIST
MOV  WKPARLST.WKLSTOFF,AX ; INTO THE PARAMETER LIST
MOV  AX,LISTSEG          ; PUT THE SEGMENT ADDRESS OF THE LIST
MOV  WKPARLST.WKLSTSEG,AX ; INTO THE PARAMETER LIST
MOV  AL,30H             ; PUT THE OPTION BYTE FOR SENDING A
MOV  WKPARLST.WKOPTION,AL ; LIST OF CHARS. IN THE PARM LIST

ENDIF

IFNB <TASKID>           ; IF A CONNECTOR'S TASK ID WAS
MOV  AX,TASKID          ; SPECIFIED, PUT IT IN THE LIST
ELSE
MOV  AX,0               ; OTHERWISE PUT A 0 IN THE LIST
ENDIF
MOV  WKPARLST.WKTASKID,AX

; INITIALIZE THE REGISTERS FOR WRIT$KEY
MOV  AH,09H
MOV  AL,04H
MOV  BH,40H
MOV  BL,40H
MOV  CX,0000H
MOV  DX,SERVTYPE        ; SERVICE TYPE IN DX
MOV  DI, SEG WKPARLST    ; GET SEGMENT ADDRESS OF PARM LIST
MOV  ES,DI              ; AND PUT IT IN ES
MOV  DI, OFFSET WKPARLST ; SET DI TO OFFSET OF PARM LIST

INT  7AH                ; PASS THE REQUEST TO THE API

CMP  CL,00H             ; CHECK FOR AN OK RETURN CODE
JNE  WKEND              ; IF NOT OK, SKIP GETTING RESULTS

GET$COMP 00H            ; GET THE RESULTS. DON'T WAIT.

WKEND:  NOP

ENDM

;
;  MACRO : GET$COMP
;  FUNCTION:
;      USE THIS SERVICE TO OBTAIN THE CONTENTS OF A SPECIFIED
;      REQUEST QUEUE ELEMENT.
;      ONE PARAMETER IS PASSED THAT INDICATES WHETHER THE USER
;      WANTS TO WAIT UNTIL RESULTS ARE READY (40H) OR TO CHECK IF
;      RESULTS ARE AVAILABLE AND GET THEM IF THEY ARE READY (00H).
;
;
```

```

GET$COMP  MACRO  WAIT

            MOV  BL,WAIT
            MOV  AH,83H                ; SET UP THE REGS FOR A GET$COMP CALL
            MOV  CX,0000H
            INT  7AH

            ENDM

STACKSEG  SEGMENT STACK 'STACK'
            DB  STAK$SIZ DUP(?)        ; STACK SPACE
STACKSEG  ENDS

DATASEG  SEGMENT 'DATASEG'

; NAME$RES PARAMETER LIST ELEMENT, SERVICE NAME
SESSMGR   DB 'SESSMGR '
KEYBOARD  DB 'KEYBOARD'

; RETURNED SERVICE TYPES FROM NAME$RES
SMGR      DW 0
KEYB      DW 0

; NAME ARRAY FORMAT FOR Q$ID
LENARRAY  DB 22                      ; NUMBER OF BYTES IN NAME ARRAY
MATCHES   DB 0                      ; NUMBER OF MATCHING SESSIONS
SNAME$1   DB 0                      ; SHORT NAME OF MATCHING SESSION
STYPE$1   DB 0                      ; TYPE OF MATCHING SESSION
SESSID$1  DB 0                      ; SESSION ID OF MATCHING SESSION
SPARE$1   DB 0
LNAME$1   DB 0                      ; LONG NAME OF MATCHING SESSION
            DB 15 DUP(0)

; DEFINE PARAMETER LIST FOR QUERY$ID
QDRCODE   DB 0                      ; RETURN CODE
QDFXNID   DB 0                      ; FUNCTION ID
QDOPT     DB 0                      ; OPTION BYTE
QDDATA    DB 0                      ; DATA BYTE
QDAOFF    DW 0                      ; NAMES ARRAY OFFSET
QDASEG    DW 0                      ; NAMES ARRAY SEGMENT ADDRESS
QDLNAM    DB 8 DUP(' ')            ; WINDOW LONG NAME

; PARAMETER LIST FOR CRT$TASK
CTTSKOFF  DW 0                      ; OFFSET OF TASK TO BE CREATED
CTTSKSEG  DW 0                      ; SEGMENT OF TASK TO BE CREATED
CTTSKNAM  DB 8 DUP(?)              ; TASK NAME

```

Sample Program 2

; PARAMETER LIST FOR CONN\$KEY

CKRETNCD	DB	0	; RETURN CODE
CKFXNID	DB	0	; FUNCTION NUMBER
CKSESSID	DB	0	; SESSION ID
CKRESRV1	DB	0	; RESERVED
CKEVENTQ	DW	0	; EVENT QUEUE ID
CKKEYSTQ	DW	0	; KEYSTROKE QUEUE ID
CKOPTION	DB	0	; OPTION BYTE
CKRESRV2	DB	0	; RESERVED

; PARAMETER LIST STRUCTURE FOR WRIT\$KEY

```
WRKYPARM  STRUC
WKRETNCD  DB  0
WKFXNID   DB  0
WKSESSID  DB  0
WKSPARE   DB  0
WKTASKID  DW  0
WKOPTION  DB  0
WKNUMKEY  DB  0
WKSCNCOD  DB  0
WKSHFST   DB  0
WKRESRV2  DW  0
WRKYPARM  ENDS
WRKYPAR2   STRUC
           DB  8  DUP(00)
WKLSTOFF  DW  0
WKLSTSEG  DW  0
WRKYPAR2  ENDS
```

; ALLOCATE STORAGE FOR THE PARAMETER LIST

WKPARLST WRKYPARM <>

; PARAMETER LIST FOR READ\$KEY

RKRETNCD	DB	0	; RETURN CODE
RKFXNID	DB	0	; FUNCTION NUMBER
RKSESSID	DB	0	; SESSION ID
RKRESRV1	DB	0	; RESERVED
RKTASKID	DW	0	; CONNECTOR'S TASK ID
RKOPTION	DB	20H	; OPTION BYTE (INTERCEPT ALL)
RKRESRV2	DB	0	; RESERVED
RKSCANCD	DB	0	; SCAN CODE
RKSHIFST	DB	0	; SHIFT STATE
RKDEVID	DB	0	; DEVICE ID
RKRESRV3	DB	0	; RESERVED

; DEFINE PARAMETER LIST FOR CRT\$Q

```
CQQOFFS  DW  0
CQSEGM   DW  0
CQQNAME  DB  8  DUP(' ')
```

; DECLARE A STACK FOR TASK, USED BY CRT\$TAST API FUNCTION

```
TASK$STK  DB  256  DUP(?)
TASK$ID   DW  0                ; TASK ID RETURNED FROM CRT$TASK
```

```

; CREATE AREA FOR A QUEUE, USED BY CRT$Q API FUNCTION
QUE      DB 100 DUP(?)
LEN$Q    DW 100          ; LENGTH OF QUEUE
Q$ID     DW 0            ; QUEUE ID RETURNED FROM CRT$Q

CONNID    DW 0          ; THE ID OF THE TASK THAT CONNECTS TO
;                               WSCTRL
RCODE     DB 0          ; RETURN CODE FORM Q$ID PARAM LIST

DLAST     DB 0          ; LAST BYTE IN THE DATA SEGMENT
DATASEG   ENDS

;
;
;                               *** MAIN BODY ***
;

ZCODE SEGMENT 'CODE'

; ESTABLISH ADDRESSABILITY OF CODE
ASSUME    CS:ZCODE
; ESTABLISH ADDRESSABILITY OF DATA

START:    MOV     AX,DATASEG
          MOV     DS,AX
          ASSUME   DS:DATASEG

; THE CODE THAT WILL CREATE THE TASK WILL FOLLOW THE ACTUAL
; DEFINITION OF THE TASK
JMP       INITTSK

;
;
;                               *** TASK DEFINITION ***
;

TASK      PROC FAR
; READ A KEYSTROKE THAT IS DIRECTED TO WORK STATION CONTROL
; NOTE: THE CONNECTOR'S ID (CONNID) IS USED BY THE READ$KEY
;       FUNCTION BECAUSE THE TASK THAT REQUESTED THE CONNECT
;       TO WORK STATION CONTROL IS DIFFERENT FROM THE TASK THAT
;       IS REQUESTING THE READ KEYSTROKE.
READKEYS: READ$KEY KEYB,SESSID$1,CONNID

; IF THE KEYSTROKE IS FOR WORK STATION CONTROL THEN IGNORE
; IT AND READ THE NEXT KEYSTROKE.
CMP       RKSCANCD,WSCTRL$K
JE        READKEYS

; THE KEYSTROKE MUST BE THE ENLARGE WINDOW OR JUMP KEY.
; SEND THE KEYSTROKE TO WORK STATION CONTROL TO BE PROCESSED.
; NOTE: THE CONNECTOR'S ID (CONNID) IS USED BY THE WRIT$KEY
;       FUNCTION BECAUSE THE TASK THAT REQUESTED THE CONNECT
;       TO WORK STATION CONTROL IS DIFFERENT FROM THE TASK THAT
;       IS REQUESTING THE WRITE KEYSTROKE.
WRIT$KEY KEYB,SESSID$1,RKSCANCD,RKSHIFST,,CONNID

; CONTINUE READING KEYSTROKES
JMP READKEYS

TASK      ENDP

```



```
EXIT$RES  PROC NEAR

            ; CALCULATE THE LENGTH OF THIS PROGRAM
MOV     AX,OFFSET DLAST      ; LENGTH OF DATA
ADD     AX,STAK$SIZ          ; + LENGTH OF STACK
ADD     AX,OFFSET INITTSK    ; + LENGTH OF TASK
ADD     AX,100H              ; + LENGTH OF THE PSP
MOV     CL,4                 ; CALCULATE THE LENGTH OF THIS PROGRAM
SAR     AX,CL                ; IN PARAGRAPHS (DIVIDE BY SIXTEEN)
INC     AX                   ; ROUND UP

MOV     DX,AX                ; STORE IN DX FOR THE DOS INTERRUPT
MOV     AH,31H               ; DOS FUNCTION CALL TO
INT     21H                  ; EXIT & REMAIN
                                ; RESIDENT

EXIT$RES  ENDP

ZCODE     ENDS                ; END CODE SEGMENT
END       START
```


Chapter 27. Sample Program 3

TITLE WNDWSMRY
PAGE 60,132

PROGRAM : WNDWSMRY

FUNCTION :

THIS PROGRAM PRESENTS A SUMMARY OF THE USER'S CURRENTLY ACTIVE SESSIONS. A SAMPLE WINDOW IS DISPLAYED FOR EACH SESSION. THE WINDOWS ARE DISPLAYED ACCORDING TO SESSION TYPE, FIRST THE PC WINDOWS, THEN THE HOST WINDOWS, THEN THE NOTEPAD WINDOWS. A MESSAGE IN THE LOWER LEFT CORNER TELLS THE NUMBER OF WINDOWS FOR THAT SESSION TYPE. THE USER IS THEN PROMPTED TO PRESS A KEY TO INITIATE THE PROGRAM EXIT.

A PROBLEM ARISES IN DISPLAYING THE MESSAGE IN THE LOWER LEFT CORNER. ANY MESSAGE MUST BE DISPLAYED IN A WINDOW. BUT THE WINDOW THIS PROGRAM WILL RUN IN IS ONE OF THE SAMPLE WINDOWS TO BE DISPLAYED. TO SOLVE THIS, AN ALTERNATE PRESENTATION SPACE IS CREATED. THE MESSAGE IS DISPLAYED IN THE LOWER LEFT CORNER OF THE ALTERNATE PRESENTATION SPACE AND A SIZED WINDOW OF EACH OF THE OTHER SESSIONS IS DISPLAYED ON TOP OF THE ALTERNATE PRESENTATION SPACE.

EACH WINDOW IS SIZED TO 4 ROWS BY 14 COLUMNS. THIS SIZING PERMITS 20 WINDOWS (THE MAXIMUM NUMBER OF WINDOWS ON THE 3270 PC) TO BE DISPLAYED IN A 5 BY 4 ARRAY ON THE SCREEN.

IN ORDER TO PRESERVE THE USER'S CURRENT WINDOW AND SCREEN SETUP, THE PROGRAM DOES ALL THE WORK ON AN UNUSED SCREEN PROFILE. A SEARCH IS MADE STARTING AT SCREEN 9 AND COUNTING DOWN FOR A SCREEN PROFILE THAT HAS NO WINDOWS. IF NO BLANK SCREEN IS FOUND, THEN A MESSAGE IS DISPLAYED INDICATING A BLANK SCREEN IS NEEDED AND THE PROGRAM ENDS. ONCE AN UNUSED SCREEN PROFILE IS FOUND, THE ALTERNATE PRESENTATION SPACE IS BROUGHT TO THAT SCREEN. THEN A SIZED AND COLORED WINDOW OF EACH OF THE SESSIONS IS BROUGHT TO THE SCREEN WITH THE MESSAGE IN THE ALTERNATE PRESENTATION SPACE SHOWING THE NUMBER AND TYPE OF THE WINDOWS BEING DISPLAYED.

THIS PROGRAM USES THE FOLLOWING API FUNCTIONS:

SELECT ACTIVE SCREEN
SELECT ACTIVE WINDOW
ADD WINDOW
CLEAR SCREEN
CONNECT TO WORK STATION CONTROL
CHANGE ENLARGE STATE
CHANGE WINDOW POSITION ON SCREEN
CHANGE WINDOW COLORS
CHANGE WINDOW SIZE
DEFINE PRESENTATION SPACE
DISCONNECT FROM WORK STATION CONTROL
DELETE PRESENTATION SPACE
NAME RESOLUTION
QUERY SESSION ID
QUERY ACTIVE SCREEN
QUERY ACTIVE WINDOW
QUERY ENLARGE STATE
QUERY BASE WINDOW
QUERY WINDOW NAMES

Sample Program 3

```
SUBTTL CONSTANTS
PAGE

; ESTABLISH CONSTANTS

CR      EQU    13          ; ASCII FOR A CARRIAGE RETURN
LF      EQU    10          ; ASCII FOR A LINE FEED
DISPSTR EQU    9           ; DOS DISPLAY CHARACTER FUNCTION NUMBER
INKEY   EQU    8           ; DOS INPUT CHARACTER FUNCTION NUMBER
RED     EQU    00000100B   ; ATTRIBUTE BYTE FOR RED ON BLACK
GREEN   EQU    00000010B   ; ATTRIBUTE BYTE FOR GREEN ON BLACK
BLUE    EQU    00000001B   ; ATTRIBUTE BYTE FOR BLUE ON BLACK
WHITE   EQU    00000111B   ; ATTRIBUTE BYTE FOR WHITE ON BLACK

SUBTTL MACROS

.SALL           ; SUPPRESS LISTING ALL MACROS

;
; MACRO NAME : ACT$SCR
; PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'WSCTRL '
;              SESSID  -- SESSION ID
;              SCREEN  -- SCREEN PROFILE NUMBER
; FUNCTION :
;           MAKE THE SPECIFIED SCREEN PROFILE THE ACTIVE SCREEN.
;
ACT$SCR MACRO SERVTYPE,SESSID,SCREEN

; INITIALIZE PARAMETER LIST FOR ACT$SCR
MOV  ASRETNCD,00H          ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV  ASFXNID,00H           ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV  AL,SESSID             ; SESSION ID INTO THE LIST
MOV  ASSESSID,AL
MOV  AL,SCREEN              ; SCREEN NUMBER
MOV  ASSCREEN,AL

; INITIALIZE REGISTERS FOR ACT$SCR
MOV  AH,09H
MOV  AL,1CH
MOV  BH,80H
MOV  BL,20H
MOV  CX,0FFH
MOV  DX,SERVTYPE           ; RESOLVED VALUE FOR 'WSCTRL '
MOV  DI, SEG ASRETNCD      ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI                 ; IN ES
MOV  DI,OFFSET ASRETNCD    ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR ACT$SCR SERVICE
INT  7AH

ENDM
```

```

;
; MACRO NAME : ACT$WNDW
; PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'WSCTRL '
;              SESSID  -- SESSION ID
;              SCREEN   -- SCREEN NUMBER
;              WINDOW    -- WINDOW SHORT NAME
;
; FUNCTION :
;           MAKE A WINDOW ON THE SPECIFIED SCREEN PROFILE THE
;           ACTIVE WINDOW.
;
ACT$WNDW MACRO SERVTYPE,SESSID,SCREEN,WINDW

; INITIALIZE PARAMETER LIST FOR ACT$WNDW
MOV  ACRETNCD,00H      ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV  ACFXNID,00H      ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV  AL,SESSID        ; SESSION ID INTO THE LIST
MOV  ACSESSID,AL
MOV  AL,SCREEN         ; SCREEN NUMBER
MOV  ACSCREEN,AL
MOV  AL,WINDW          ; WINDOW SHORT NAME
MOV  ACWINDW,AL

; INITIALIZE REGISTERS FOR ACT$WNDW
MOV  AH,09H
MOV  AL,14H
MOV  BH,80H
MOV  BL,20H
MOV  CX,0FFH
MOV  DX,SERVTYPE      ; RESOLVED VALUE FOR 'WSCTRL '
MOV  DI, SEG ACRETNCD  ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI             ; IN ES
MOV  DI,OFFSET ACRETNCD ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR ACT$WNDW SERVICE
INT  7AH

ENDM

;
; MACRO NAME : ADD$WNDW
; PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'WSCTRL '
;              SESSID  -- SESSION ID
;              SCRPRO   -- SCREEN PROFILE NUMBER IN ASCII
;              WINDN    -- WINDOW SHORT NAME IN ASCII
;
; FUNCTION :
;           ADD A WINDOW FROM SCREEN PROFILE 0 TO THE SPECIFIED
;           SCREEN PROFILE. THE ADDED WINDOW BECOMES THE ACTIVE
;           WINDOW. WINDOWS CANNOT BE ADDED TO SCREEN PROFILE 0.
;
ADD$WNDW MACRO SERVTYPE,SESSID,SCRPRO,WINDN

; INITIALIZE PARAMETER LIST FOR ADD$WNDW
MOV  AWRETNCD,00H      ; AWRETNCD MUST BE 0 BEFORE REQUEST
MOV  AL,SESSID        ; SESSION ID
MOV  AWSESSID,AL       ; IN LIST
MOV  AL,SCRPRO         ; SCREEN PROFILE NUMBER
MOV  AWSCRPRO,AL        ; IN LIST
MOV  AL,WINDN          ; WINDOW SHORT NAME
MOV  AWWINDN,AL        ; IN LIST

; INITIALIZE REGISTERS FOR ADD$WNDW
MOV  AH,09H
MOV  AL,03H
MOV  BH,80H
MOV  BL,20H

```

Sample Program 3

```
MOV    CX,0FFH
MOV    DX,SERVTYPE      ; RESOLVED VALUE FOR 'WSCTRL '
MOV    DI, SEG AWRETNCD  ; SEGMENT ADDRESS OF PARAMETER LIST
MOV    ES,DI             ; IN ES
MOV    DI,OFFSET AWRETNCD ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR ADD$WNDW SERVICE
INT     7AH

ENDM

;
;
; MACRO NAME : CLEAR$SC
; PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'WSCTRL '
;              SESSID  -- SESSION ID
;              SCRPRO   -- SCREEN PROFILE NUMBER IN ASCII
; FUNCTION :
;           DELETE ALL WINDOWS FROM THE SPECIFIED SCREEN PROFILE.
;           WINDOWS CAN NOT BE DELETED FROM SCREEN PROFILE 0.
;
;
CLEAR$SC MACRO  SERVTYPE,SESSID,SCRPRO

; INITIALIZE PARAMETER LIST FOR CLEAR$SC
MOV    CLRETNCD,00H      ; CLRETNCD MUST BE 0 BEFORE REQUEST
MOV    CLFXNID,00H       ; CLFXNID MUST BE 0 BEFORE REQUEST
MOV    AL,SESSID         ; SESSION ID OBTAINED FROM REQUEST
MOV    CLSESSID,AL       ; TO QUERY$ID
MOV    AL,SCRPRO         ; SCREEN PROFILE NUMBER
MOV    CLSCRPRO,AL       ; IN LIST

; INITIALIZE REGISTERS FOR CLEAR$SC
MOV    AH,09H
MOV    AL,13H
MOV    BH,80H
MOV    BL,20H
MOV    CX,0FFH
MOV    DX,SERVTYPE      ; RESOLVED VALUE FOR 'WSCTRL '
MOV    DI, SEG CLRETNCD  ; SEGMENT ADDRESS OF PARAMETER LIST
MOV    ES,DI             ; IN ES
MOV    DI,OFFSET CLRETNCD ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR CLEAR$SC SERVICE
INT     7AH

ENDM

;
;
; MACRO NAME : CONN$WSC
; PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'WSCTRL '
;              SESSID  -- SESSION ID
; FUNCTION :
;           CONNECT TO THE WORK STATION CONTROL SESSION FOR THE WIN-
;           DOW MANAGEMENT SERVICES. ONLY ONE SESSION CAN BE CONNECTED
;           FOR WINDOW MANAGEMENT SERVICES AT A TIME.
;
;
CONN$WSC MACRO  SERVTYPE,SESSID

; INITIALIZE PARAMETER LIST FOR CONN$WSC
MOV    CWRETNCD,00H      ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV    CWFNID,00H       ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV    AL,SESSID         ; SESSION ID INTO PARAMETER LIST
MOV    CWSESSID,AL
```

```

; INITIALIZE REGISTERS FOR CONN$WSC
MOV  AH,09H
MOV  AL,01H
MOV  BH,80H
MOV  BL,20H
MOV  CX,0FFH
MOV  DX,SERVTYPE      ; RESOLVED VALUE FOR 'WSCTRL '
MOV  DI, SEG CWRETNCD  ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI             ; IN ES
MOV  DI,OFFSET CWRETNCD ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR CONN$WSC SERVICE
INT  7AH

ENDM

;
;
; MACRO NAME : C$ENLUN
;
; PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'WSCTRL '
;              SESSID  -- SESSION ID
;
; FUNCTION :
;           TOGGLE THE "ENLARGE STATE" OF THE DISPLAY. AN ENLARGED
;           DISPLAY BECOMES NORMAL, OR A NORMAL DISPLAY BECOMES
;           ENLARGED.
;
;
C$ENL$UN MACRO  SERVTYPE,SESSID

; INITIALIZE PARAMETER LIST FOR C$ENLUN
MOV  CERETNCD,00H      ; CERETNCD MUST BE 0 BEFORE REQUEST
MOV  CEFXNID,00H       ; CEFXNID MUST BE 0 BEFORE REQUEST
MOV  AL,SESSID         ; SESSION ID OBTAINED FROM REQUEST
MOV  CESESSID,AL       ; TO QUERY$ID

; INITIALIZE REGISTERS FOR C$ENLUN
MOV  AH,09H
MOV  AL,09H
MOV  BH,80H
MOV  BL,20H
MOV  CX,0FFH
MOV  DX,SERVTYPE      ; RESOLVED VALUE FOR 'WSCTRL '
MOV  DI, SEG CERETNCD  ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI             ; IN ES
MOV  DI,OFFSET CERETNCD ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR C$ENLUN SERVICE
INT  7AH

ENDM

;
;
; MACRO NAME : C$SCRPOS
;
; PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'WSCTRL '
;              SESSID  -- SESSION ID
;              SCRNUM   -- SCREEN NUMBER
;              WNDWNAME -- WINDOW SHORT NAME
;              ROW      -- ROW OF UPPER LEFT CORNER
;              COL      -- COLUMN OF UPPER LEFT CORNER
;
; FUNCTION :
;           CHANGE THE POSITION OF A WINDOW ON THE SPECIFIED SCREEN
;           PROFILE. THE NEW WINDOW POSITION IS DETERMINED BY PLACING
;           THE UPPER LEFT CORNER OF THE WINDOW AT THE ROW AND COLUMN
;           NUMBERS SPECIFIED IN THE PARAMETER LIST.
;
;

```


Sample Program 3

C\$SCRPOS MACRO SERVTYPE,SESSID,SCRNUM,WNDWNAME,ROW,COL

```
; INITIALIZE PARAMETER LIST FOR C$SCRPOS
MOV CSRETNC,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV CSFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID INTO PARAMETER LIST
MOV CSSESSID,AL
MOV AL,SCRNUM ; SCREEN NUMBER INTO PARAMETER LIST
MOV CSSCREEN,AL
MOV AL,WNDWNAME ; WINDOW SHORT NAME INTO LIST
MOV CSWINDOW,AL
MOV AL,ROW ; ROW NUMBER INTO THE LIST
MOV CSROW,AL
MOV AL,COL ; COLUMN NUMBER INTO THE LIST
MOV CSCOLUMN,AL

; INITIALIZE REGISTERS FOR C$SCRPOS
MOV AH,09H
MOV AL,04H
MOV BH,80H
MOV BL,20H
MOV CX,0FFH
MOV DX,SERVTYPE ; RESOLVED VALUE FOR 'WSCTRL '
MOV DI,SEG CSRETNC ; SEGMENT ADDRESS OF PARAMETER LIST
MOV ES,DI ; IN ES
MOV DI,OFFSET CSRETNC ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR C$SCRPOS SERVICE
INT 7AH

ENDM
```

```
;
;
; MACRO NAME : C$WNDL
;
; PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'WSCTRL '
;               SESSID -- SESSION ID
;               SCREEN -- SCREEN NUMBER
;               WINDOW -- WINDOW SHORT NAME
;               FOREGND -- FOREGROUND COLOR
;               BACKGND -- BACKGROUND COLOR
;               BASE -- BASE COLOR
;
; FUNCTION :
;           CHANGE THE FOREGROUND AND BACKGROUND COLORS OF A WINDOW
;           ON THE SPECIFIED SCREEN PROFILE.
;
```

C\$WNDL MACRO SERVTYPE,SESSID,SCREEN,WINDOW,FOREGND,BACKGND,BASE

```
; INITIALIZE PARAMETER LIST FOR C$WNDL
MOV CCRETNC,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV CCFXNID,00H ; FUNCTION DI MUST = 0 BEFORE REQUEST
MOV AL,SESSID ; SESSION ID INTO LIST
MOV CCSSESSID,AL
MOV AL,SCREEN ; SCREEN NUMBER INTO LIST
MOV CCSCREEN,AL
MOV AL,WINDOW ; WINDOW SHORT NAME INTO LIST
MOV CCWINDOW,AL
MOV AL,FOREGND ; FOREGROUND COLOR INTO LIST
MOV CCFORGND,AL
```

```

MOV     AL,BACKGND           ; BACKGROUND COLOR INTO LIST
MOV     CCBAGND,AL
MOV     AL,BASE              ; BASE COLOR INTO LIST
MOV     CCBASE,AL

; INITIALIZE REGISTERS FOR C$WNDCOL
MOV     AH,09H
MOV     AL,06H
MOV     BH,80H
MOV     BL,20H
MOV     CX,0FFH
MOV     DX,SERVTYPE          ; RESOLVED VALUE FOR 'WSCTRL '
MOV     DI, SEG CCRETNCD      ; SEGMENT ADDRESS OF PARAMETER LIST
MOV     ES,DI                ; IN ES
MOV     DI,OFFSET CCRETNCD    ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR C$WNDCOL SERVICE
INT     7AH

ENDM

```

```

;
;
; MACRO NAME : C$WNDWSZ
; PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'WSCTRL '
;              SESSID  -- SESSION ID
;              SCRPRO   -- SCREEN PROFILE NUMBER IN ASCII
;              WINDOW   -- WINDOW SHORT NAME IN ASCII
;              ROWS     -- NUMBER OF ROWS IN NEW WINDOW SIZE
;              COLS     -- NUMBER OF COLUMNS IN NEW WINDOW SIZE
;
; FUNCTION :
; CHANGE THE SIZE OF A WINDOW ON A SPECIFIED SCREEN
; PROFILE. THE WINDOW'S NEW SIZE IS DETERMINED BY THE
; NUMBER OF ROWS AND COLUMNS IN THE PARAMETER LIST.
; A VALUE OF ZERO FOR EITHER THE NUMBER OF ROWS OR
; NUMBER OF COLUMNS IN THE WINDOW SIZE IS CHANGED BY THE
; WORKSTATION PROGRAM TO BE A VALUE OF ONE.
;
;

```

C\$WNDWSZ MACRO SERVTYPE,SESSID,SCRPRO,WINDOW,ROWS,COLS

```

; INITIALIZE PARAMETER LIST FOR C$WNDWSZ
MOV     CZRETNCD,00H          ; CZRETNCD MUST BE 0 BEFORE REQUEST
MOV     CZFXNID,00H           ; CZFXNID MUST BE 0 BEFORE REQUEST
MOV     AL,SESSID             ; SESSION ID OBTAINED FROM REQUEST
MOV     CZSESSID,AL           ; TO QUERY$ID
MOV     AL,SCRPRO              ; SCREEN PROFILE NUMBER
MOV     CZSCRPRO,AL            ; IN LIST
MOV     AL,WINDOW              ; WINDOW SHORT NAME OBTAINED FROM
MOV     CZWINDN,AL             ; REQUEST TO QUERY$ID
MOV     AL,ROWS                ; NUMBER OF ROWS IN THE NEW
MOV     CZNUMROW,AL            ; WINDOW SIZE
MOV     AL,COLS                ; NUMBER OF COLUMNS IN THE
MOV     CZNUMCOL,AL            ; WINDOW SIZE

; INITIALIZE REGISTERS FOR C$WNDWSZ
MOV     AH,09H
MOV     AL,05H
MOV     BH,80H
MOV     BL,20H
MOV     CX,0FFH

```

Sample Program 3

```
MOV  DX,SERVTYPE      ; RESOLVED VALUE FOR 'WSCTRL  '
MOV  DI, SEG CZRETNCD  ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI             ;   IN ES
MOV  DI,OFFSET CZRETNCD ; OFFSET OF PARAMETER LIST IN DI
```

```
; SIGNAL WORKSTATION PROGRAM FOR C$WNDWSZ SERVICE
INT  7AH
```

```
ENDM
```

```
;
;
;   MACRO NAME : DEFIN$PS
;   PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'PCPSM  '
;               BUFFSEG  -- WORK BUFFER
;               PSDS     -- POINTER TO PS DATA STREAM
;
;   FUNCTION :
;       CREATE A NEW PRESENTATION SPACE.
;
```

```
DEFIN$PS MACRO  SERVTYPE,BUFFER,PSDS
```

```
; INITIALIZE PARAMETER LIST FOR DEFIN$PS
MOV  DPRETNCD,00H      ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV  DPFKNID,00H       ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV  AX,OFFSET BUFFER  ; BUFFER OFFSET INTO THE LIST
MOV  DPBUFOFF,AX
MOV  AX,SEG BUFFER     ; BUFFER SEGMENT INTO THE LIST
MOV  DPBUFSEG,AX
MOV  AX,OFFSET PSDS    ; PSDS OFFSET INTO THE LIST
MOV  DPDSOFF,AX
MOV  AX,SEG PSDS       ; PSDS SEGMENT INTO THE LIST
MOV  DPDSSEG,AX
```

```
; INITIALIZE REGISTERS FOR DEFIN$PS
MOV  AH,09H
MOV  AL,01H
MOV  BH,80H
MOV  BL,20H
MOV  CX,0FFH
MOV  DX,SERVTYPE      ; SERVICE TYPE IN DX
MOV  DI, SEG DPRETNCD  ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI             ;   IN ES
MOV  DI,OFFSET DPRETNCD ; OFFSET OF PARAMETER LIST IN DI
```

```
; SIGNAL WORKSTATION PROGRAM FOR DEFIN$PS SERVICE
INT  7AH
```

```
ENDM
```

```
;
;
;   MACRO NAME : DISC$WSC
;   PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'WSCTRL  '
;               SESSID   -- SESSION ID
;
;   FUNCTION :
;       DISCONNECT FROM THE WORK STATION CONTROL SESSION.
;
```

```

DISC$WSC MACRO  SERVTYPE,SESSID

; INITIALIZE PARAMETER LIST FOR DISC$WSC
MOV  DWRETNCD,00H      ; DWRETNCD MUST BE 0 BEFORE REQUEST
MOV  DWFXNID,00H       ; DWFXNID MUST BE 0 BEFORE REQUEST
MOV  AL,SESSID         ; SESSION ID OBTAINED FROM REQUEST
MOV  DWSESSID,AL       ; TO QUERY$ID

; INITIALIZE REGISTERS FOR DISC$WSC
MOV  AH,09H
MOV  AL,02H
MOV  BH,80H
MOV  BL,20H
MOV  CX,0FFH
MOV  DX,SERVTYPE      ; RESOLVED VALUE FOR 'WSCtrl '
MOV  DI, SEG DWRETNCD  ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI             ; IN ES
MOV  DI,OFFSET DWRETNCD ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR DISC$WSC SERVICE
INT  7AH

ENDM

;
;
; MACRO NAME : DSTRY$PS
;
; PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'PCPSM '
;              SESSID  -- SESSION ID
;
; FUNCTION :
;           DELETE A PRESENTATION SPACE.
;
;

DSTRY$PS MACRO  SERVTYPE,SESSID

; INITIALIZE PARAMETER LIST FOR DSTRY$PS
MOV  DYRETNCD,00H      ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV  DYFXNID,00H       ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV  AL,SESSID         ; HANDLE ID INTO THE LIST
MOV  DYSESSID,AL

; INITIALIZE REGISTERS FOR DSTRY$PS
MOV  AH,09H
MOV  AL,02H
MOV  BH,80H
MOV  BL,20H
MOV  CX,0FFH
MOV  DX,SERVTYPE      ; RESOLVED VALUE FOR 'PCPSM '
MOV  DI, SEG DYRETNCD  ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI             ; IN ES
MOV  DI,OFFSET DYRETNCD ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR DSTRY$PS SERVICE
INT  7AH

ENDM

;
;
; MACRO      - NAME$RES
;
; PARAMETERS - NR$SERVN - LOCATION OF THE 8-BYTE
;                   SERVICE NAME, I.E.'SESSMGR '
;
;           NR$SERVT - RETURN CODE FROM PARAMETER LIST
;
;

```

Sample Program 3

```
NAME$RES  MACRO      NR$SERVN,NR$SERVT

; SET UP REGISTERS NAME$RES
MOV      AX,SEG NR$SERVN      ; SEGMENT ADDRESS OF PARM LIST
MOV      ES,AX                ; ES = SEGM ADDRESS OF PARM LIST
MOV      AH,81H                ; AH = X'81'
MOV      CX,0000H              ; CX = X'0000'
MOV      DI,OFFSET NR$SERVN    ; DI = OFFSET ADDR. OF PARM LIST

; REQUEST SERVICE TYPE FROM WORKSTATION PROGRAM
INT      7AH

; RETURN SERVICE TYPE ID TO CALLER
MOV      NR$SERVT,DX

ENDM

;
;
; MACRO NAME : QUERY$ID
; PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'SESSMGR '
;              NAMEARRY -- NAME ARRAY
;              OPTION    -- OPTION BYTE
;              DATA     -- DATA BYTE
;              LONGNAME  -- SESSION LONG NAME
;
; FUNCTION :
;           GET THE SESSION ID(S) OF THE SESSION(S) SPECIFIED BY
;           THE OPTION AND DATA BYTES AND RETURNS THEM IN THE NAME
;           ARRAY.
;
; NOTE - THE NAME ARRAY IS SET UP BY THE USER AND MUST HAVE
;         THE LENGTH OF THE ARRAY CONTAINED IN THE 1ST BYTE.
;
;
QUERY$ID  MACRO      SERVTYPE,NAMEARRY,OPTION,DATA,LONGNAME

; INITIALIZE PARAMETER LIST FOR QUERY$ID
MOV      QDRETNCD,00H          ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV      QDFXNID,00H           ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV      AL,OPTION              ; OPTION BYTE INTO THE LIST
MOV      QDOPTION,AL
MOV      AL,DATA                ; DATA BYTE INTO THE LIST
MOV      QDDATA,AL
MOV      AX,OFFSET NAMEARRY     ; NAME ARRAY OFFSET INTO THE LIST
MOV      QDNAMOFF,AX
MOV      AX,SEG NAMEARRY        ; NAME ARRAY SEGMENT INTO THE LIST
MOV      QDNAMSEG,AX

IFNB     <LONGNAME>             ; CHECK IF A LONG NAME WAS SPECIFIED

CLD                                           ; COPY DIRECTION = FORWARD
MOV      AX,SEG QDLNGNAM
MOV      ES,AX                    ; ES:DI POINTS TO DESTINATION IN PARM
MOV      DI,OFFSET QDLNGNAM        ; LIST
MOV      SI,OFFSET LONGNAME        ; DS:SI POINTS TO SOURCE OF LONG NAME
MOV      CX,8                      ; MOVE 8 BYTES
REP      MOVSB                     ; COPY LONG NAME INTO THE PARM LIST

ENDIF
```

```

; INITIALIZE REGISTERS FOR QUERY$ID
MOV  AH,09H
MOV  AL,01H
MOV  BH,80H
MOV  BL,20H
MOV  CX,0000H
MOV  DX,SERVTYPE      ; RESOLVED VALUE FOR 'SESSMGR '
MOV  DI, SEG QDRETNCNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI             ; IN ES
MOV  DI,OFFSET QDRETNCNCD ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR QUERY$ID SERVICE
INT  7AH

ENDM

```

```

;
; MACRO NAME : Q$ASCRID
; PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'WSCTRL '
;              SESSID  -- SESSION ID
;
; FUNCTION :
;           OBTAIN THE ID OF THE ACTIVE SCREEN PROFILE.
;

```

Q\$ASCRID MACRO SERVTYPE,SESSID

```

; INITIALIZE PARAMETER LIST FOR Q$ASCRID
MOV  QARETNCD,00H      ; QARETNCD MUST BE 0 BEFORE REQUEST
MOV  AL,SESSID         ; SESSION ID OBTAINED FROM REQUEST
MOV  QASESSID,AL       ; TO QUERY$ID

; INITIALIZE REGISTERS FOR Q$ASCRID
MOV  AH,09H
MOV  AL,19H
MOV  BH,80H
MOV  BL,20H
MOV  CX,0FFH
MOV  DX,SERVTYPE      ; RESOLVED VALUE FOR 'WSCTRL '
MOV  DI, SEG QARETNCD  ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI             ; IN ES
MOV  DI,OFFSET QARETNCD ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR Q$ASCRID SERVICE
INT  7AH

ENDM

```

```

;
; MACRO NAME : Q$AWNDSN
; PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'WSCTRL '
;              SESSID  -- SESSION ID
;              SCREEN  -- SCREEN PROFILE NUMBER
;
; FUNCTION :
;           OBTAIN THE SHORT NAME OF THE ACTIVE WINDOW IN THE
;           SPECIFIED SCREEN PROFILE.
;

```

Sample Program 3

Q\$AWNDSN MACRO SERVTYPE,SESSID,SCREEN

```
; INITIALIZE PARAMETER LIST FOR Q$AWNDSN
MOV  QNRETCD,00H      ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV  QNFXNID,00H      ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV  AL,SESSID        ; SESSION ID INTO THE LIST
MOV  QNSESSID,AL
MOV  AL,SCREEN         ; SCREEN NUMBER INTO THE LIST
MOV  QNSCREEN,AL
```

```
; INITIALIZE REGISTERS FOR Q$AWNDSN
MOV  AH,09H
MOV  AL,18H
MOV  BH,80H
MOV  BL,20H
MOV  CX,0FFH
MOV  DX,SERVTYPE      ; RESOLVED VALUE FOR 'WSCTRL '
MOV  DI, SEG QNRETCD   ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI             ; IN ES
MOV  DI,OFFSET QNRETCD ; OFFSET OF PARAMETER LIST IN DI
```

```
; SIGNAL WORKSTATION PROGRAM FOR Q$AWNDSN SERVICE
INT  7AH
```

ENDM

```
;
;
; MACRO NAME : Q$ENL$UN
; PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'WSCTRL '
;              SESSID  -- SESSION ID
; FUNCTION :
;           OBTAIN THE "ENLARGE STATE" OF THE DISPLAY.
;
```

Q\$ENL\$UN MACRO SERVTYPE,SESSID

```
; INITIALIZE PARAMETER LIST FOR Q$ENL/UN
MOV  QERETCD,00H      ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV  QEFXNID,00H      ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV  AL,SESSID        ; SESSION ID INTO THE LIST
MOV  QESESSID,AL
```

```
; INITIALIZE REGISTERS FOR Q$ENL/UN
MOV  AH,09H
MOV  AL,10H
MOV  BH,80H
MOV  BL,20H
MOV  CX,0FFH
MOV  DX,SERVTYPE      ; RESOLVED VALUE FOR 'WSCTRL '
MOV  DI, SEG QERETCD   ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI             ; IN ES
MOV  DI,OFFSET QERETCD ; OFFSET OF PARAMETER LIST IN DI
```

```
; SIGNAL WORKSTATION PROGRAM FOR Q$ENL/UN SERVICE
INT  7AH
```

ENDM

```

;
;   MACRO NAME : Q$BAS$W
;   PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'SESSMGR '
;   FUNCTION :
;       FIND THE SESSION ID AND SHORT NAME FOR THE BASE WINDOW
;       OF AN ENVIRONMENT.
;
Q$BAS$W MACRO SERVTYPE

; INITIALIZE PARAMETER LIST FOR Q$BAS$W
MOV  QSRETNCD,00H      ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV  QSFXNID,00H       ; FUNCTION ID MUST = 0 BEFORE REQUEST

; INITIALIZE REGISTERS FOR Q$BAS$W
MOV  AH,09H
MOV  AL,0AH
MOV  BH,80H
MOV  BL,20H
MOV  CX,0FFH
MOV  DX,SERVTYPE      ; RESOLVED VALUE FOR 'SESSMGR '
MOV  DI, SEG QSRETNCD  ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI             ; IN ES
MOV  DI,OFFSET QSRETNCD ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR Q$BAS$W SERVICE
INT  7AH

ENDM

;
;   MACRO NAME : Q$WINDWS
;   PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'WSCTRL '
;               SESSID  -- SESSION ID
;               SCREEN  -- SCREEN PROFILE NUMBER
;   FUNCTION :
;       OBTAIN THE SHORT NAMES OF ALL WINDOWS IN THE SPECIFIED
;       SCREEN PROFILE.
;
Q$WINDWS MACRO SERVTYPE,SESSID,SCREEN

; INITIALIZE PARAMETER LIST FOR Q$WINDWS
MOV  QWRETNCD,00H      ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV  QWFXNID,00H       ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV  AL,SESSID         ; SESSION ID INTO THE LIST
MOV  QWSESSID,AL
MOV  AL,SCREEN          ; SCREEN NUMBER INTO THE LIST
MOV  QWSCREEN,AL

; INITIALIZE REGISTERS FOR Q$WINDWS
MOV  AH,09H
MOV  AL,12H
MOV  BH,80H
MOV  BL,20H
MOV  CX,0FFH

```


Sample Program 3

```
MOV    DX,SERVTYPE      ; RESOLVED VALUE FOR 'WSCTRL  '
MOV    DI, SEG QWRETNCD  ; SEGMENT ADDRESS OF PARAMETER LIST
MOV    ES,DI             ;   IN ES
MOV    DI,OFFSET QWRETNCD ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR Q$WINDWS SERVICE
INT     7AH

ENDM

;
;
; MACRO : DOSFUNCT
; FUNCTION :
;         ISSUE THE DOS CALL WITH THE FUNCTION NUMBER PASSED IN
;         FUNCTNUM
;
DOSFUNCT MACRO  FUNCTNUM

MOV     AH,FUNCTNUM
INT     21H

ENDM

;
;
; MACRO : DISPLAY
; FUNCTION :
;         DISPLAY THE STRING STARTING AT STRING ON THE SCREEN.
;
DISPLAY MACRO  STRING
LEA     DX,STRING
DOSFUNCT 09H

ENDM

;
;
; MACRO : CHEK4ERR
; FUNCTION :
;         SET UP THE REGISTERS FOR THE ERROR CHECKER.
;         THE RETURN CODES FROM AN API FUNCTION CALL ARE IN THE CL
;         REGISTER AND IN THE FIRST BYTE OF THE PARAMETER LIST.
;         THE PARAMETER LIST RETURN CODE (IF IT EXISTS) IS PASSED IN
;         THE PARAMETER RETCODE AND IS PASSED TO THE ERROR CHECKER
;         IN BL.  THE ERROR CHECKER WILL CHECK CL AND BL FOR 0'S
;         AND RETURN IF THEY BOTH ARE 0.
;
CHEK4ERR MACRO  RETCODE

IFNB    <RETCODE>
MOV     BL,RETCODE      ; IF A RETURN CODE FROM A PARAMETER LIST
ELSE     ; WAS SPECIFIED, PUT IT IN BL
MOV     BL,0            ; OTHERWISE, SET BL TO 0 SO IT LOOKS LIKE
ENDIF    ; A GOOD RETURN CODE TO THE ERROR CHECKER

CALL    CHECKERR

ENDM
```

```

;
;      MACRO : WRITBUFF
;      FUNCTION :
;          COPIES N CHARACTERS STARTING AT TEXT TO THE PS BUFFER WITH
;          THE SPECIFIED ATTRIBUT (ATTRIBUTE).
;
WRITBUFF MACRO TEXT,N,ATTRIBUT
LOCAL NEXTCHAR

        XOR    CX,CX          ; CLEAR CX TO RECEIVE THE # OF CHARACTERS
        MOV    CL,N
        LEA    SI,TEXT        ; POINT SI TO THE TEXT TO COPY
        LEA    DI,BUFFMSG     ; POINT DI TO THE PS BUFFER MESSAGE AREA

NEXTCHAR: MOVSB                ; COPY A CHARACTER TO THE BUFFER
        MOV    BYTE PTR [DI],ATTRIBUT ; COPY THE ATTRIBUTE INTO THE BUFFER
        INC    DI              ; POINT DI TO THE NEXT CHARACTER
        LOOP   NEXTCHAR        ; COPY THE NEXT CHARACTER AND ATTRIBUTE

        ENDM

;
;      MACRO : BI2ASCII
;      FUNCTION :
;          CONVERTS THE VALUE IN VARIABLE - ASSUMING IT'S BETWEEN 0
;          AND 9 - TO ITS ASCII EQUIVALENT BY ADDING 30H.
;          THIS IS USED TO CONVERT SCREEN PROFILE NUMBERS, WHICH RANGE
;          FROM 0 TO 9, TO THEIR ASCII EQUIVALENT WHICH IS NEEDED FOR
;          THE API FUNCTION CALLS.
;
BI2ASCII MACRO VARIABLE

        MOV    AL,VARIABLE     ; GET THE VARIABLE BINARY VALUE
        ADD    AL,30H          ; CONVERT TO ASCII
        MOV    VARIABLE,AL     ; REPLACE VARIABLE WITH ITS ASCII VALUE

        ENDM

        SUBTTL DATA
        PAGE

DATASEG SEGMENT

WORKAREA DB 1552 DUP(?)        ; WORK AREA FOR DEFIN$PS
; PARAMETER LIST FOR ACT$SCR

ASRETNCD DB 0                  ; RETURN CODE
ASFXNID DB 0                   ; FUNCTION NUMBER
ASSESSID DB 0                  ; SESSION ID
ASSCREEN DB 0                  ; SCREEN NUMBER

; PARAMETER LIST FOR ACT$WNDW

ACRETNCD DB 0                  ; RETURN CODE
ACFXNID DB 0                   ; FUNCTION NUMBER
ACSESSID DB 0                  ; SESSION ID
ACSCREEN DB 0                   ; SCREEN NUMBER
ACWINDOW DB 0                  ; WINDOW SHORT NAME

```

Sample Program 3

; PARAMETER LIST FOR ADD\$WNDW

AWRETNCD	DB	0	; RETURN CODE
AWFXNID	DB	0	; FUNCTION ID
AWSESSID	DB	0	; SESSION ID
AWSCRPRO	DB	0	; SCREEN PROFILE NUMBER IN ASCII
AWWINDN	DB	0	; WINDOW SHORT NAME IN ASCII

; PARAMETER LIST FOR CLEAR\$SC

CLRETNCD	DB	0	; RETURN CODE
CLFXNID	DB	0	; FUNCTION NUMBER
CLSESSID	DB	0	; SESSION ID
CLSCRPRO	DB	0	; SCREEN PROFILE NUMBER IN ASCII

; PARAMETER LIST FOR CONN\$WSC

CWRETNCD	DB	0	; RETURN CODE
CWFXNID	DB	0	; FUNCTION NUMBER
CWSESSID	DB	0	; SESSION ID

; PARAMETER LIST FOR C\$ENLUN

CERETNCD	DB	0	; RETURN CODE
CEFXNID	DB	0	; FUNCTION NUMBER
CESESSID	DB	0	; SESSION ID

; PARAMETER LIST FOR C\$SCRPOS

CSRETNCD	DB	0	; RETURN CODE
CSFXNID	DB	0	; FUNCTION NUMBER
CSSESSID	DB	0	; SESSION ID
CSSCREEN	DB	0	; SCREEN PROFILE NUMBER (IN ASCII)
CSWINDOW	DB	0	; WINDOW SHORT NAME (IN ASCII)
CSROW	DB	0	; ROW NUMBER FOR POSITION OF UPPER LEFT CORNER OF WINDOW
CSCOLUMN	DB	0	; COLUMN NUMBER FOR POSITION OF UPPER LEFT CORNER OF WINDOW

; PARAMETER LIST FOR C\$WNDCOL

CCRETNCD	DB	0	; RETURN CODE
CCFXNID	DB	0	; FUNCTION NUMBER
CCSESSID	DB	0	; SESSION ID
CCSCREEN	DB	0	; SCREEN NUMBER
CCWINDOW	DB	0	; WINDOW SHORT NAME
CCFORGND	DB	0	; FOREGROUND COLOR
CCBAKGND	DB	0	; BACKGROUND COLOR
CCBASE	DB	0	; BASE COLOR

; PARAMETER LIST FOR C\$WNDWSZ

CZRETNCD	DB	0	; RETURN CODE
CZFXNID	DB	0	; FUNCTION NUMBER
CZSESSID	DB	0	; SESSION ID
CZSCRPRO	DB	0	; SCREEN PROFILE NUMBER IN ASCII
CZWINDN	DB	0	; WINDOW SHORT NAME IN ASCII
CZNUMROW	DB	0	; NUMBER OF ROWS IN NEW WINDOW SIZE
CZNUMCOL	DB	0	; NUMBER OF COLUMNS IN NEW WINDOW SIZE

; PARAMETER LIST FOR DEFIN\$PS

DPRETNCD	DB	0	; RETURN CODE
DPFXNID	DB	0	; FUNCTION NUMBER
DPSESSID	DB	0	; SESSION ID
DPRESERV	DB	0	; RESERVED
DPBUFOFF	DW	0	; OFFSET ADDRESS OF THE 1K BUFFER
DPBUFSEG	DW	0	; SEGMENT ADDRESS OF THE 1K BUFFER
DPDSOFF	DW	0	; OFFSET OF DATA STREAM
DPDSSEG	DW	0	; SEGMENT OF DATA STREAM
	DB	0	; MUST BE 0
DPWINDOW	DB	0	; RETURNED WINDOW SHORT NAME

; PARAMETER LIST FOR DISC\$WSC

DWRETNCD	DB	0	; RETURN CODE
DWFXNID	DB	0	;
DWSESSID	DB	0	; SESSION ID

; PARAMETER LIST FOR DSTRY\$PS

DYRETNCD	DB	0	; RETURN CODE
DYFXNID	DB	0	; FUNCTION NUMBER
DYSESSID	DB	0	; SESSION ID
DYRESERV	DB	0	; RESERVED

; PARAMETER LIST FOR QUERY\$ID

QDRETNCD	DB	0	; RETURN CODE
QDFXNID	DB	0	; FUNCTION NUMBER
QDOPTION	DB	0	; OPTION BYTE
QDDATA	DB	0	; DATA BYTE
QDNAMOFF	DW	0	; OFFSET OF NAME TABLE
QDNAMSEG	DW	0	; SEGMENT OF NAME TABLE
QDLNGNAM	DB	8 DUP(?)	; SESSION LONG NAME

; PARAMETER LIST FOR Q\$ASCRID

QARETNCD	DB	0	; RETURN CODE
QAFXNID	DB	0	; FUNCTION ID
QASESSID	DB	0	; SESSION ID
QASCRPRO	DB	0	; SCREEN PROFILE NUMBER IN ASCII

; PARAMETER LIST FOR Q\$AWNDSN

QNRETNCD	DB	0	; RETURN CODE
QNFXNID	DB	0	; FUNCTION NUMBER
QNSESSID	DB	0	; SESSION ID
QNSCREEN	DB	0	; SCREEN NUMBER
QNWINDOW	DB	0	; SHORT NAME OF ACTIVE WINDOW

; PARAMETER LIST FOR Q\$BAS\$W

QSRETNCD	DB	0	; RETURN CODE
QSFXNID	DB	0	; FUNCTION NUMBER
QSENVID	DB	0	; ENVIRONMENT ID
QSSESSID	DB	0	; SESSION ID
QSWINDOW	DB	0	; WINDOW SHORT NAME
QSRESERV	DB	0	; RESERVED

Sample Program 3

; PARAMETER LIST FOR Q\$ENL/UN

QERETNCD	DB	0	; RETURN CODE
QEFXNID	DB	0	; FUNCTION NUMBER
QESESSID	DB	0	; SESSION ID
QEENLFLG	DB	0	; ENLARGE FLAG

; PARAMETER LIST FOR Q\$WINDWS

QWRETNCD	DB	0	; RETURN CODE
QWFXNID	DB	0	; FUNCTION NUMBER
QWSESSID	DB	0	; SESSION ID
QWSCREEN	DB	0	; SCREEN NUMBER
QWWNDLST	DB	20 DUP(0)	; LIST OF WINDOW SHORT NAMES

SMGRNAME	DB	'SESSMGR '	; PARAMETER LIST FOR NAME\$RES ON "SESSMGR"
WSCTNAME	DB	'WSCTRL '	; PARAMETER LIST FOR NAME\$RES ON "WSCTRL"
PCPSNAME	DB	'PCPSM '	; PARAMETER LIST FOR NAME\$RES ON "PCPSM"

SESSMGR	DW	0	; SESSION MANAGER SERVICE TYPE
WSCTRL	DW	0	; WINDOW MANAGER SERVICE TYPE
PCPSM	DW	0	; PRESENTATION SPACE MANAGER SERVICE TYPE

SCREEN	DB	0	; THE SCREEN WE ARE WORKING IN
--------	----	---	--------------------------------

ROWSIZE	DB	4	; NUMBER OF ROWS IN SUMMARY WINDOWS
COLSIZE	DB	14	; NUMBER OF COLUMNS IN SUMMARY WINDOWS
ROWPOS	DB	1	; ROW POSITION OF WINDOW ON THE SCREEN
COLPOS	DB	1	; COLUMN POSITION OF WINDOW ON THE SCREEN

NAMEARRY	DB	170	; NAME ARRAY FOR QUERY\$ID FUNCTION
NUMSESS	DB	0	; NUMBER OF MATCHING SESSIONS
SHRTNAME	DB	0	; SHORT NAME OF THE FIRST MATCHING WINDOW
SESSTYPE	DB	0	; SESSION TYPE
SESSID	DB	0	; SESSION ID
SPARE	DB	0	; SPARE BYTE (UNUSED)
LONGNAME	DB	156 DUP(0)	; LONG NAME OF THE WINDOW
	DB	228 DUP(0)	; REMAINDER OF THE NAME ARRAY
PSDS	DB	3	; PRESENTATION SPACE DATA STREAM FOR
			; DEFIN\$PS FUNCTION (3 COMMANDS)
PSSIZE	DB	01	; COMMAND FOR SIZE OF PRESENTATION SPACE
	DB	25	; 25 ROWS
	DB	80	; 80 COLUMNS
PSTYPE	DB	02	; COMMAND TO SET PRESENTATION SPACE TYPE
	DB	0	; PC TEXT INDIRECT (WELL BEHAVED)
SETBUFFER	DB	03	; COMMAND TO SET THE PS BUFFER
BUFFOFF	DW	0	; OFFSET OF THE BUFFER
BUFFSEG	DW	0	; SEGMENT OF THE BUFFER

PSBUFFER	DW	1920 DUP(0)	; THE PRESENTATION SPACE BUFFER
BUFFMSG	DW	80 DUP(0)	; SPACE TO DISPLAY MESSAGES (25TH LINE)

NOSCRMSG	DB	'PROGRAM NEEDS A BLANK SCREEN PROFILE TO RUN.',CR,LF	
	DB	'DELETE ALL THE WINDOWS FROM ONE SCREEN AND TRY AGAIN.'	
	DB	CR,LF,'\$'	

ISPC	DB	'THERE IS 1 PC SESSION'	
AREPCS	DB	'THERE ARE '	
PCSESNUM	DW	0	
	DB	' PC SESSIONS'	

```

NOHOST    DB      'THERE ARE NO HOST SESSIONS'
ISHOST    DB      'THERE IS 1 HOST SESSION '
AREHOST   DB      'THERE ARE '
HOSTNUM   DW      0
          DB      ' HOST SESSIONS'

NONOTE    DB      'THERE ARE NO NOTEPAD SESSIONS'
ISNOTE    DB      'THERE IS 1 NOTEPAD SESSION'
ARENOTE   DB      'THERE ARE '
NOTENUM   DW      0
          DB      ' NOTEPAD SESSIONS'

NUMBRTAB  DB      ' 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20'

EXITMSG   DB      'PRESS ANY KEY TO EXIT      '

ERRORMSG  DB      'ERROR.  PROGRAM TERMINATED.',CR,LF,'$'

DATASEG   ENDS

STACKSEG  SEGMENT STACK
          DB      20 DUP('STACK  ')

STACKSEG  ENDS

          SUBTTL MAIN
          PAGE

CODESEG   SEGMENT
          ASSUME CS:CODESEG,SS:STACKSEG,DS:DATASEG,ES:DATASEG

MAIN      PROC   FAR

          MOV     AX,DATASEG      ; ESTABLISH ADDRESSABILITY TO THE DATA
          MOV     DS,AX
          MOV     ES,AX

          ;;;;;;;;;;;;;;
          ;; FIND THE RESOLVED VALUES FOR SESSMGR, WSCTRL, AND PCPSM ;;
          ;;;;;;;;;;;;;;

          NAME$RES SMGRNAME,SESSMGR
          CHEK4ERR
          NAME$RES WSCTNAME,WSCTRL
          CHEK4ERR
          NAME$RES PCPSNAME,PCPSM
          CHEK4ERR

          ;;;;;;;;;;;;;;
          ;; FIND THE SESSION ID FOR THIS PC SESSION ;;
          ;;;;;;;;;;;;;;

          Q$BAS$W  SESSMGR
          CHEK4ERR QSRETNCD

```

Sample Program 3

```
;;;;;;;;;;;;;
;; CREATE A PRESENTATION SPACE TO WORK IN ;;
;;;;;;;;;;;;;

MOV     AX,OFFSET PSBUFFER
MOV     BUFOFF,AX      ; PUT THE OFFSET AND SEGMENT OF THE PS
MOV     AX,SEG PSBUFFER ; IN THE PSDS
MOV     BUFFSEG,AX
DEFIN$PS PCPSM,WORKAREA,PSDS
                        ; CREATE A PRESENTATION SPACE
CHEK4ERR DPRETNCD

;;;;;;;;;;;;;
;; CONNECT TO WINDOW MANAGER SERVICES IN ORDER TO DO ;;
;; WINDOW MANAGER API FUNCTIONS.                      ;;
;;;;;;;;;;;;;

CONN$WSC WSCtrl,QSSESSID
CHEK4ERR CWRETNCD

;;;;;;;;;;;;;
;; FIND AN UNUSED SCREEN PROFILE.  START WITH SCREEN 9 AND ;;
;; COUNT DOWN.                                           ;;
;;;;;;;;;;;;;

MOV     CX,9           ; LOAD THE COUNT INTO CX

FINDSCR: MOV     SCREEN,CL      ; CHECK SCREEN(COUNT)
          PUSH    CX           ; SAVE THE COUNT
          BI2ASCII SCREEN      ; CONVERT BINARY SCREEN NUMBER TO ASCII
          Q$WINDWS WSCtrl,QSSESSID,SCREEN
                        ; GET THE LIST OF WINDOWS FOR THIS SCREEN

          CMP     QWRETNCD,0EH  ; CHECK FOR RETURN CODE INDICATING NO
                        ; WINDOWS
          JE      FOUNDSCR      ; IF NO WINDOWS, A BLANK SCREEN WAS FOUND
          CHEK4ERR QWRETNCD

          POP     CX           ; RESTORE THE COUNT
          LOOP    FINDSCR      ; CHECK THE NEXT SCREEN PROFILE

;;;;;;;;;;;;;
;; IF NO UNUSED SCREEN WAS FOUND, CLEAN UP BY DISCONNECTING ;;
;; FROM WINDOW MANAGER SERVICES AND DELETING THE NEW PRE- ;;
;; SENTATION SPACE.  THEN PRINT AN ERROR MESSAGE INDICATING ;;
;; A BLANK SCREEN PROFILE IS NEEDED AND EXIT TO DOS.      ;;
;;;;;;;;;;;;;

DISC$WSC WSCtrl,QSSESSID
                        ; DISCONNECT FROM WINDOW SERVICES
CHEK4ERR DWRETNCD
DSTRY$PS PCPSM,DPSSESSID
                        ; DESTROY THE PRESENTATION SPACE
CHEK4ERR DYRETNCD
DISPLAY  NOSCRMSG      ; DISPLAY THE ERROR MESSAGE

JMP     FINISH         ; EXIT TO DOS
```

```

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; HAVING FOUND AN UNUSED SCREEN PROFILE, SAVE THE CURRENT ;;
;; ACTIVE SCREEN PROFILE AND WINDOW. QUERY THE ENLARGE ;;
;; STATE OF THE DISPLAY AND SET IT TO "UNENLARGED" IF IT IS ;;
;; NOT ALREADY UNENLARGED. (THE DISPLAY MUST BE UNENLARGED ;;
;; IN ORDER FOR THE SAMPLE WINDOWS TO SHOW) ACTIVATE THE ;;
;; UNUSED SCREEN PROFILE AND ADD THE WORK WINDOW TO THE ;;
;; SCREEN. ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

FOUNDSCR: Q$ASCRID WCTRL,Q$SESSID
                ; SAVE THE CURRENT ACTIVE SCREEN
CHEK4ERR QARETNCD
Q$AWNDSN WCTRL,Q$SESSID,Q$ASCRPRO
                ; SAVE THE CURRENT ACTIVE WINDOW
CHEK4ERR QNRETNCD

Q$ENL$UN WCTRL,Q$SESSID,Q$ASCRPRO
                ; CHECK THE "ENLARGE STATE" OF THE DISPLAY
CHEK4ERR QERETNCD

CMP Q$ENLFLG,0 ; IF THE DISPLAY IS NOT ENLARGED, THEN
JE ISNTENL ; DON'T TOGGLE THE ENLARGED STATE.

C$ENL$UN WCTRL,Q$SESSID,Q$ASCRPRO
                ; TOGGLE THE ENLARGE STATE OF THE DISPLAY
                ; TO UNENLARGE IT.
CHEK4ERR CERETNCD

ISNTENL: ACT$SCR WCTRL,Q$SESSID,SCREEN
                ; MAKE THE BLANK SCREEN ACTIVE
CHEK4ERR ASRETNCD

ADD$WNDW WCTRL,Q$SESSID,SCREEN,DPWINDOW
                ; ADD THE WORK WINDOW TO THE SCREEN
CHEK4ERR AWRETNCD

CALL PCWNDWS ; DISPLAY A SIZED WINDOW OF EACH PC SESSION
CALL HSTWNDWS ; DISPLAY A SIZED WINDOW OF EACH HOST
                ; SESSION
CALL NOTWNDWS ; DISPLAY A SIZED WINDOW OF EACH NOTEPAD
                ; SESSION

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; PROMPT THE USER TO INITIATE THE CLEANUP AND EXIT ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

DONE: WRITBUFF EXITMSG,30,WHITE
                ; WRITE THE MESSAGE TO THE PS BUFFER
ACT$WNDW WCTRL,Q$SESSID,SCREEN,Q$SWINDOW
                ; MAKE THIS PC WINDOW THE ACTIVE WINDOW SO
                ; THAT THE DOS KEY INPUT WILL COME TO
                ; THIS PC SESSION
CHEK4ERR ACRETNCD
DISC$WSC WCTRL,Q$SESSID
                ; DISCONNECT FROM WINDOW SERVICES TO SHOW
                ; THE CHANGES TO THE SCREEN
CHEK4ERR DWRETNCD

MOV AH,INKEY ; WAIT FOR THE USER TO HIT A KEY
INT 21H

```


Sample Program 3

```
;;;;;;;;;;;;;
;; CLEAN UP BY DESTROYING THE PRESENTATION SPACE THAT WAS ;;
;; CREATED, DELETING ALL WINDOWS FROM THE SCREEN, MAKING THE;;
;; ORIGINAL WINDOW AND SCREEN ACTIVE AGAIN, AND RESTORING ;;
;; THE ENLARGE STATE OF THE DISPLAY. ;;
;;;;;;;;;;;;;

DSTRY$PS PCPSM,DPSSESSID
; DESTROY THE PRESENTATION SPACE THAT WAS
; CREATED FOR A WORK SPACE

CHEK4ERR DYRETNCD

CONN$WSC WSCtrl,QSSESSID
; CONNECT FOR WINDOW MANAGER SERVICES TO
; THE WINDOW AND SCREEN THAT WERE ACTIVE
; ON ENTRY TO THIS PROGRAM

CHEK4ERR CWRETNCD
CLEAR$SC WSCtrl,QSSESSID,SCREEN
; DELETE THE WINDOWS FROM THE SCREEN

CHEK4ERR CLRETNCD
ACT$SCR WSCtrl,QSSESSID,QASCRPRO
; MAKE THE ORIGINAL SCREEN PROFILE ACTIVE

CHEK4ERR ASRETNCD
ACT$WNDW WSCtrl,QSSESSID,QASCRPRO,QNWINDOW
; MAKE THE ORIGINAL WINDOW ACTIVE

CMP QEENLFLG,0 ; IF THE DISPLAY WAS NOT ENLARGED, SKIP
JE WASNTENL ; TOGGLING IT BACK TO ENLARGED STATE

C$ENL$UN WSCtrl,QSSESSID
; TOGGLE THE DISPLAY BACK TO THE ENLARGED
; STATE

CHEK4ERR CERETNCD

WASNTENL: DISC$WSC WSCtrl,QSSESSID
; DISCONNECT FROM WINDOW MANAGER SERVICES

CHEK4ERR DWRETNCD

FINISH: MOV AX,4C00H ; RETURN TO DOS
INT 21H

;
; PROCEDURE : PCWNDWS
; CALLED BY : MAIN
; FUNCTION :
; THIS PROCEDURE DISPLAYS A MESSAGE TELLING THE NUMBER OF PC
; SESSIONS AND DISPLAYS A SIZED WINDOW OF EACH PC SESSION.
; FIRST IT GETS A LIST OF THE CURRENT PC SESSIONS BY USING THE
; QUERY SESSION ID API FUNCTION ON THE SESSION TYPE FOR PC'S
; (X'05'). IT THEN DISPLAYS A MESSAGE IN THE CREATED WORK
; PRESENTATION SPACE INDICATING THE NUMBER OF PC SESSIONS. EACH
; WINDOW IN THE LIST IS ADDED TO THE SCREEN PROFILE THEN SIZED
; AND MOVED TO THE PROPER POSITION. SINCE THE CREATED WORK PS
; IS ALSO RETURNED IN THIS LIST, A CHECK IS MADE FOR THE WORK
; WINDOW. WHEN IT IS ENCOUNTERED, THE WORK WINDOW IS NOT SIZED
; OR MOVED SINCE IT WAS NOT A CURRENT PC SESSION WHEN THE PROGRAM
; WAS STARTED.
;
```


Sample Program 3

```
SKIPWNDX: ADD    SI,12          ; POINT TO THE NEXT SHORT NAME
           LOOP   NXTPCWND      ; DISPLAY THE NEXT PC WINDOW
```

```
           RET
```

```
PCWNDWS   ENDP
```

```
;
;
;   PROCEDURE : HSTWNDWS
;   CALLED BY : MAIN
;   FUNCTION :
;       THIS PROCEDURE DISPLAYS A MESSAGE TELLING THE NUMBER OF
;       HOST WINDOWS AND DISPLAYS A SIZED AND COLORED SAMPLE WINDOW OF
;       EACH HOST SESSION.
;       HOST SESSIONS CAN BE EITHER CUT OR DFT; BOTH TYPES OF SESSIONS
;       MUST BE QUERIED.  FIRST A QUERY IS MADE FOR A CUT TYPE SESSION.
;       IF THERE IS A CUT SESSION, THEN A MESSAGE IS DISPLAYED TELLING
;       THERE IS ONE HOST SESSION.  IF THERE IS NOT A CUT SESSION, THEN
;       A QUERY IS MADE FOR ANY DFT SESSIONS AND THE NUMBER OF SESSIONS
;       IS DISPLAYED.  A SAMPLE WINDOW OF EACH HOST SESSION (IF ANY) IS
;       ADDED TO THE SCREEN PROFILE AND THEN COLORED RED AND SIZED AND
;       MOVED TO ITS PROPER POSITION.
;
```

```
HSTWNDWS  PROC
```

```
;;;;;;;;;;;;;
;; SET THE ROWPOS AND COLPOS VARIABLES SO THAT THE HOST ;;
;; SESSION WINDOWS WILL BE DISPLAYED ON THE NEXT ROW.  ;;
;;;;;;;;;;;;;
```

```
MOV    COLPOS,1      ; START AT THE FIRST COLUMN
ADD     ROWPOS,6      ; START A NEW ROW
```

```
;;;;;;;;;;;;;
;; GET A LIST OF THE HOST SESSIONS.  FIRST CHECK FOR A CUT ;;
;; HOST SESSION.  IF THERE IS NO CUT SESSION, THEN CHECK FOR ;;
;; ANY DFT HOST SESSIONS.                                     ;;
;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;
;; CHECK FOR A CUT HOST SESSION. ;;
;;;;;;;;;;;;;
```

```
QUERY$ID  SESSMGR,NAMEARRY,00H,03H
           ; GET THE LIST FOR A CUT HOST SESSION
CMP    QDRETNCD,11H  ; CHECK IF THERE IS NOT A CUT SESSION
JE     CHECKDFT      ; IF SO, CHECK FOR DFT HOST SESSIONS
```

```
CHEK4ERR  QDRETNCD
```

```
JMP     ONEHOST      ; IF THERE IS A CUT SESSION, DISPLAY THE
                     ; MESSAGE FOR ONE HOST SESSION.
```

```
;;;;;;;;;;;;;
;; CHECK FOR ANY DFT HOST SESSIONS. ;;
;;;;;;;;;;;;;
```

```

CHECKDFT: QUERY$ID  SESSMGR,NAMEARRY,00H,02H
          ; GET A LIST OF THE HOST DFT SESSIONS
CMP      QDRETNCD,11H ; CHECK IF THERE ARE NO DFT SESSIONS
JNE      DISPHOST

          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
          ;; DISPLAY THE MESSAGE FOR NO HOST SESSIONS ;;
          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

WRITBUFF  NOHOST,26,RED
          ; WRITE THE MESSAGE TO THE PS BUFFER
CALL     DISPWNDX    ; DISPLAY THE MESSAGE
JMP      NOTEPAD

DISPHOST: CHEK4ERR  QDRETNCD

          ; CHECK IF THERE IS ONLY ONE SESSION
CMP      NUMSESS,1
JNE      MANYHOST

          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
          ;; DISPLAY THE MESSAGE FOR ONE HOST SESSION ;;
          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

ONEHOST:  WRITBUFF  ISHOST,24,RED
          ; WRITE THE MESSAGE TO THE PS BUFFER
JMP      DISHSTWN

          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
          ;; DISPLAY THE MESSAGE FOR MULTIPLE HOST SESSIONS ;;
          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

MANYHOST: XOR      BX,BX          ; CLEAR BX TO READ THE NUMBER OF SESSIONS
MOV      BL,NUMSESS
ADD      BX,BX
          ; CALCULATE THE INDEX INTO THE ASCII
          ; TRANSLATION TABLE
MOV      AX,WORD PTR NUMBRTAB[BX]
          ; GET THE ASCII VALUE OF THE NUMBER
MOV      HOSTNUM,AX
          ; PUT ASCII FOR THE NUMBER IN THE MESSAGE
WRITBUFF  AREHOST,26,RED
          ; WRITE THE MESSAGE TO THE PS BUFFER

DISHSTWN: CALL     DISPWNDX    ; DISPLAY THE MESSAGE

          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
          ;; DISPLAY THE HOST WINDOWS ON THIS SCREEN PROFILE ;;
          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

          XOR      CX,CX          ; CLEAR CX BEFORE LOADING ONE BYTE
          MOV      CL,NUMSESS    ; LOAD CX WITH THE NUMBER OF HOST SESSIONS
          LEA      SI,SHRTNAME   ; POINT SI TO THE FIRST SESSION SHORT NAME

NXTHSTWN: PUSH     CX            ; SAVE THE COUNT OF THE WINDOWS
          CALL     ADDNMOV        ; SIZE AND MOVE THE WINDOW
          C$WND$COL  WSCTRL,Q$SESSID,SCREEN,[SI],2,0,0
          ; SET WINDOW COLOR - RED FORE, BLACK BACK
          CHEK4ERR  CCRETNCD
          POP      CX            ; RESTORE THE COUNT
          ADD      SI,12         ; POINT TO THE NEXT SHORT NAME
          LOOP     NXTHSTWN      ; DISPLAY THE NEXT HOST WINDOW

          RET

HSTWNDWS  ENDP

```

27-26

```

MANYNOTE: XOR    BX,BX          ; CLEAR BX TO READ THE NUMBER OF SESSIONS
           MOV    BL,NUMSESS
           ADD    BX,BX          ; CALCULATE THE INDEX INTO THE ASCII
                                   ; TRANSLATION TABLE
           MOV    AX,WORD PTR NUMBRTAB[BX]
                                   ; GET THE ASCII VALUE OF THE NUMBER
           MOV    NOTENUM,AX      ; PUT ASCII FOR THE NUMBER IN THE MESSAGE
           WRITBUFF ARENOTE,29,GREEN
                                   ; WRITE THE MESSAGE TO THE PS BUFFER
DISNOTWN: ACT$WNDW  WSCTRL,QSSESSID,SCREEN,QSWINDOW
                                   ; MAKE THIS PC WINDOW THE ACTIVE WINDOW
           CHEK4ERR  ACRETNCD
           CALL  DISPWNDX

           ;;;;;;;;;;;;;;
           ;; DISPLAY THE NOTEPAD WINDOWS ON THIS SCREEN PROFILE ;;
           ;;;;;;;;;;;;;;

           XOR    CX,CX          ; CLEAR CX BEFORE LOADING ONE BYTE
           MOV    CL,NUMSESS      ; LOAD CX WITH NUMBER OF NOTEPAD SESSIONS
           LEA    SI,SHRTNAME     ; POINT SI TO THE FIRST SESSION SHORT NAME

NXTNOTWN: PUSH    CX            ; SAVE THE COUNT OF THE WINDOWS
           CALL  ADDNMOV          ; SIZE AND MOVE THE WINDOW
           C$WNDWCOL  WSCTRL,QSSESSID,SCREEN,[SI],4,0,0
                                   ; SET WINDOW COLOR - GREEN FORE,BLACK BACK
           CHEK4ERR  CCRETNCD
           POP     CX            ; RESTORE THE COUNT
           ADD     SI,12          ; POINT TO THE NEXT SHORT NAME
           LOOP   NXTNOTWN       ; DISPLAY THE NEXT NOTEPAD WINDOW

           RET

NOTWNDWS  ENDP

;
;   PROCEDURE : ADDNMOV
;   FUNCTION  :
;       GIVEN A SHORT WINDOW NAME POINTED TO BY SI, ADD THE WINDOW
;       TO THE CURRENT WORK SCREEN.  NEXT, SIZE THE WINDOW TO ROWSIZE
;       BY COLSIZE.  THEN MOVE THE WINDOW TO ITS POSITION ON THE SCREEN
;       SPECIFIED BY ROWPOS AND COLPOS.  REDRAW THE SCREEN TO SHOW
;       THE CHANGES.  UPDATE ROWPOS AND COLPOS FOR THE NEXT WINDOW.
;       THIS PROCEDURE ASSUMES THAT A CONN$WSC HAS ALREADY BEEN
;       ISSUED BY THE CALLING PROCEDURE.
;
;
ADDNMOV  PROC  NEAR

           ADD$WNDW  WSCTRL,QSSESSID,SCREEN,[SI]
                                   ; ADD THIS NOTEPAD WINDOW TO THE SCREEN
           CHEK4ERR  AWRETNCD
           C$WNDWSZ  WSCTRL,QSSESSID,SCREEN,[SI],ROWSIZE,COLSIZE
                                   ; SIZE THE WINDOW TO ROWSIZE X COLSIZE
           CHEK4ERR  CZRETNCD
           C$SCRPOS  WSCTRL,QSSESSID,SCREEN,[SI],ROWPOS,COLPOS
                                   ; MOVE THE WINDOW TO ROWPOS,COLPOS
           CHEK4ERR  CSRETNCD

```



```

;
;      PROCEDURE : CHECKERR
;      FUNCTION :
;          CHECK THE RETURN CODES FROM THE API CALLS.  BL CONTAINS
;          THE RETURN CODE IN THE PARAMETER LIST.  BL AND CL ARE CHECKED
;          FOR OS.  IF EITHER DOES NOT CONTAIN A 0, AN ERROR MESSAGE IS
;          DISPLAYED AND THE PROGRAM IS TERMINATED.
;
;      NOTE :  THIS IS A VERY SIMPLE ERROR HANDLER USED TO PRE-
;              SERVE PROGRAM FLOW AND IS NOT LISTED AS AN EXAMPLE OF
;              AN APPROPRIATE ERROR HANDLER.  THIS ERROR HANDLER SIMPLY
;              TERMINATES THE PROGRAM WHEN AN ERROR IS ENCOUNTERED
;              LEAVING ANY RESOURCES, SUCH AS FIXED LENGTH QUEUES,
;              PRESENTATION SPACES, AND A CONNECTION TO THE WINDOW
;              SERVICES, STILL ALLOCATED.  A MORE APPROPRIATE ERROR
;              HANDLER WOULD DELETE ALL RESOURCES BEFORE TERMINATING.
;
CHECKERR  PROC  NEAR

            CMP    CL,0          ; CHECK THE RETURN CODE IN CL
            JNE    ERROR

            CMP    BL,0          ; CHECK THE PARAMETER LIST RETURN CODE
            JNE    ERROR

            RET                  ; RETURN CODES OK.  RETURN TO CALLER.

ERROR:     DISPLAY  ERRORMSG      ; AN ERROR OCCURRED.  DISPLAY THE ERROR
            JMP     FINISH        ; MESSAGE AND EXIT TO DOS.

CHECKERR  ENDP

MAIN      ENDP

CODESEG   ENDS

            END

```


Sample Program 3

Chapter 28. Sample Program 4

PAGE 80,132

```

;
;      PROGRAM   : STEMIR
;
;      FUNCTION  :
;          THIS PROGRAM GRAPHICALLY DISPLAYS (IN BAR CHART FORM)
;          INFORMATION ABOUT A BUSINESS. THERE ARE FIVE CATEGORIES OF
;          INFORMATION RELATING TO THIS BUSINESS.
;          THE CATEGORIES ARE: 1. GROSS REVENUE
;                               2. NET REVENUE
;                               3. NUMBER OF PRODUCTS SOLD
;                               4. EMPLOYEE OVERTIME
;                               5. EMPLOYEE ILLNESS
;
;          THE DATA CORRESPONDING TO EACH CATEGORY IS LOCATED IN
;          A FILE LOCATED ON A VM HOST SYSTEM.
;
;          A MENU IS DISPLAYED AND THE USER CHOOSES ONE OF THE ABOVE
;          CATEGORIES. THIS PROGRAM GETS THE DATA THAT CORRESPONDS TO
;          THE USER'S CHOICE FROM THE HOST AND GRAPHS THE DATA IN A BAR
;          CHART FORM. THIS CONTINUES UNTIL THE USER HITS THE ESCAPE KEY
;          FROM THE MENU.
;
;          DATA IS TRANSFERRED FROM THE HOST USING DESTINATION/ORIGIN
;          STRUCTURED FIELDS AND THE MFIC API.
;
;          THIS PROGRAM IS NOT WELL BEHAVED SINCE IT USES GRAPHICS
;          AND WRITES DIRECTLY TO THE DISPLAY BUFFER. THE INDSPIF UTILITY
;          WAS USED TO CREATE A PIF FILE THAT REFLECTS THIS BEHAVIOR.
;
;          THIS PROGRAM DEMONSTRATES THE FOLLOWING
;          API FUNCTIONS:
;              NAME RESOLUTION
;              QUERY ID
;              CONNECT TO KEYBOARD/DISCONNECT FROM KEYBOARD
;              DISABLE INPUT/ENABLE INPUT
;              CREATE A QUEUE/DELETE AN ENTRY
;              DEFINE RECEIVE BUFFER
;              CONNECT/DISCONNECT TO HOST SESSION
;              READ STRUCTURED FIELD
;              WRITE STRUCTURED FIELD
;              DEQUEUE
;              QUERY BASE WINDOW
;              WRITE KEYSTROKE
;              GET REQUEST COMPLETION
;              QUERY ACTIVE TASK
;              TRANSLATE
;
;          NOTE: THIS PROGRAM IS SEPARATED INTO TWO LOAD MODULES.
;          THIS MODULE CONTAINS THE MAIN BODY OF THE PROGRAM. THE MAIN
;          BODY CALLS PROCEDURES IN BOTH THIS AND THE SECOND MODULE.
;
;          NOTE: DESTINATION/ORIGIN STRUCTURED FIELDS CAN ONLY BE
;          USED WHEN THE 3270 PERSONAL COMPUTER HAS BEEN CUSTOMIZED FOR
;          DFT COMMUNICATIONS. ALSO, THERE IS A HOST PROGRAM THAT RUNS
;          ON VM THAT RETRIEVES THE DATA AND COMMUNICATES WITH THIS
;          PERSONAL COMPUTER PROGRAM.
;
;
;

```

Sample Program 4

```
; DOS FUNCTION CALLS
DISPSTRG EQU 09H      ; PRINTING A STRING
NO$ECHO EQU 7          ; DOS FUNCTION TO READ A CHARACTER WITH NO
                        ; ECHO

; BIOS VIDEO FUNCTION CALLS
SET$MODE EQU 0         ; BIOS VIDEO FUNCTION TO SET THE MODE
SET$PALL EQU 11        ; BIOS VIDEO FUNCTION TO SET PALETTE
WRITEDOT EQU 12        ; BIOS VIDEO FUNCTION TO WRITE A DOT

; CONSTANTS
MSG$AVAIL EQU 8004H    ; COMMUNICATION STATUS, MESSAGE AVAILABLE
                        ; FROM HOST
UPCASE EQU 01H        ; UPPERCASE SHIFTSTATE
AIDKEY EQU 12H        ; AID KEY GENERATES A 12H RETURN CODE FROM
                        ; WRIT$KEY
ESC EQU 1BH           ; THE ASCII VALUE FOR THE ESCAPE KEY
STAKSIZE EQU 512      ; SIZE OF THE STACK

; THE FOLLOWING CONSTANTS ARE USED TO SWITCH TO GRAPHICS MODE AND SET THE COLORS
MEDRES$G EQU 04H      ; MEDIUM RESOLUTION IN GRAPHICS MODE
COL$8025 EQU 3         ; 80 COLUMNS BY 25 ROWS IN ASCII MODE
GRY EQU 0              ; THE PALETTE EQUALS GREEN, RED AND YELLOW
COL$AXIS EQU 01H      ; THE COLOR OF THE AXIS IS GREEN

; THE FOLLOWING ARE REPLY AND WAIT STATES USED BY SOME OF THE API FUNCTIONS
WAITCMPS EQU 20H      ; WAIT FOR COMPLETION SIGNAL
RPLYCMPS EQU 80H      ; REPLY COMPLETION SIGNAL
RPLYCMPQ EQU 40H      ; REPLY COMPLETION QUEUE
WAITCMPQ EQU 40H      ; WAIT FOR RQE ON COMPLETION QUEUE

; THE FOLLOWING CONSTANTS ARE USED TO PRINT THE GRAPH (E.G. BAR CHARTS)
; AND COMMENTS AROUND THE GRAPH
LEN$DASH EQU 5         ; LENGTH OF THE DASH ON THE VERTICAL AXIS
X$VERTEX EQU 50        ; X COORDINATE FOR THE AXIS VERTEX
Y$VERTEX EQU 167       ; Y COORDINATE FOR THE AXIS VERTEX
LENXAXIS EQU 260       ; LENGTH OF THE Y AXIS
LENYAXIS EQU 160       ; LENGTH OF THE X AXIS
ENTRYSIZ EQU 20        ; THERE ARE 20 BYTES PER ENTRY IN THE COMMENT
                        ; TABLES
X$COMNT EQU 7          ; THE COMMENTS DESCRIBING THE INTERVALS ALONG
Y$COMNT EQU 22         ; THE X AXIS AT (X$COMNT,Y$COMNT)
                        ; I.E. COLUMN,ROW
X$DASHC EQU 1          ; THE COMMENTS DESCRIBING THE INTERVALS ALONG
Y$DASHC EQU 17         ; THE Y AXIS BEGIN AT (X$DASHC,Y$DASHC)
X$TMSG EQU 10          ; THE MESSAGE THAT DESCRIBES THE TYPE OF DATA
Y$TMSG EQU 23          ; THAT IS BEING DISPLAYED IS LOCATED AT
                        ; (X$TMSG,Y$TMSG)
X$CMMSG EQU 15         ; THE MESSAGE THAT DISPLAYS THE CORPORATION
Y$CMMSG EQU 0          ; NAME IS LOCATED AT (X$CMMSG,Y$CMMSG)
X$HKMSG EQU 8          ; THE MESSAGE THAT DISPLAYS THE "HIT ANY
Y$HKMSG EQU 24         ; KEY..." IS LOCATED AT (X$HKMSG,Y$HKMSG)

;
; *** MACRO DEFINITIONS ***
;
;
;
;
; MACRO NAME : DOSFXN
; PARAMETERS : FXNNUM -- DOS FUNCTION NUMBER
```

```

;      FUNCTION :
;      THIS WILL CALL A DOS FUNCTION SPECIFIED BY THE PARAMETER
;      FXNNUM.
;
;

DOSFXN  MACRO FXNNUM

        MOV     AH,FXNNUM      ; THE FUNCTION NUMBER BELONGS IN AL
        INT     21H           ; CALL DOS

        ENDM

;
;      MACRO NAME : MOV$CURS
;      FUNCTION :
;      THIS WILL CALL THE BIOS VIDEO FUNCTION TO MOVE THE
;      CURSOR TO POSITION (X$COLUMN,Y$ROW).
;
;

MOV$CURS  MACRO

        MOV     AH,2          ; BIOS VIDEO FUNCTION NUMBER TWO
        MOV     BH,0          ; THE PAGE NUMBER IS ZERO IN GRAPHICS MODE
        INT     10H           ; CALL DOS

        ENDM

;
;      MACRO NAME : DISPMENU
;      PARAMETERS : SESSMGR -- RESOLVED VALUE FOR WCTRL
;      PARAMETERS : PCSESSID -- SESSION ID FOR WINDOW WITH THE MENU
;      MENU$PS -- SCREEN NUMBER
;      FUNCTION :
;      THIS WILL MAKE THE MENU APPEAR.
;
;

DISPMENU  MACRO      PCSESSID,MENU$PS

        CLD                  ; INCREMENT SI AND DI ON THE REPEAT
        MOV     CX,4000      ; MOVE 4000 BYTES OF CHARACTER-ATTRIBUTE
        MOV     SI,OFFSET MENU$PS
                                ; THE SOURCE DATA BEGINS AT MENU$PS
        MOV     DI,0         ; THE DESTINATION IS THE DISPLAY BUFFER
        MOV     AX,0B000H
        MOV     ES,AX
REP      MOVSB                ; MOVE THE MENU DATA INTO THE DISPLAY BUFFER

        MOV     AX,DATASEG   ; ASSIGN ES TO BACK TO THE DATASEG
        MOV     ES,AX

        ENDM

;
;      MACRO NAME : DRAWVERT
;      PARAMETERS : X -- VALUE ON THE X AXIS,(COLUMN)
;      YBEGIN -- VALUE ON THE Y AXIS,(ROW)
;      LEN -- LENGTH OF THE LINE
;      COLOR -- COLOR OF THE LINE
;      FUNCTION :

```

Sample Program 4

```
;          THIS WILL DRAW A VERTICAL LINE.  THE LINE WILL BE DRAWN
;          FROM POINT (X,YBEGIN) TO (X,YBEGIN+LEN).
;
;
DRAWVERT MACRO    X,YBEGIN,LEN,COLOR
LOCAL          LP,EXIT

; SAVE THE CURRENT Y VALUE
MOV           AX,YBEGIN
MOV           CURR$Y,AX

; HAVE WE REACHED THE LAST PEL YET?
MOV           BX,LEN          ; THE LAST PEL = THE FIRST PEL PLUS
ADD           BX,CURR$Y       ; THE LENGTH OF THE LINE
LP:           CMP           CURR$Y,BX

; YES, WE ARE DONE
JE            EXIT

; NO, TURN THE PEL ON
; INITIALIZE REGISTERS FOR THE BIOS WRITE DOT FUNCTION
MOV           CX,X            ; COLUMN NUMBER FOR VERTICAL LINE
MOV           AL,COLOR        ; COLOR FOR VERTICAL
MOV           AH,WRITEDOT     ;
BIOS FUNCTION NUMBER
MOV           DX,CURR$Y        ; ROW NUMBER OF PEL
INT           10H             ; CALL THE BIOS VIDEO FUNCTION

; INCREMENT THE ROW NUMBER AND CHECK THE PEL
INC           CURR$Y
JMP           LP

EXIT:         NOP

ENDM

;
;          MACRO NAME : DRAWHORZ
;          PARAMETERS : Y      -- VALUE ON THE Y AXIS,(ROW)
;                      XBEGIN  -- VALUE ON THE X AXIS,(COLUMN)
;                      LEN      -- LENGTH OF THE LINE
;                      COLOR    -- COLOR OF THE LINE
;          FUNCTION :
;          THIS WILL DRAW A HORIZONTAL LINE. THE LINE WILL BE
;          DRAWN FROM POINT (XBEGIN,Y) TO POINT (XBEGIN+LEN,Y).
;
;
DRAWHORZ MACRO    Y,XBEGIN,LEN,COLOR
LOCAL          LP,EXIT

; SAVE THE CURRENT X VALUE
MOV           AX,XBEGIN
MOV           CURR$X,AX
```

```

; HAVE WE REACHED THE LAST PEL YET?
MOV     BX,LEN          ; THE LAST PEL = THE FIRST PEL PLUS
ADD     BX,CURR$X       ; THE LENGTH OF THE LINE
LP:     CMP             CURR$X,BX

; YES, WE ARE DONE
JE      EXIT

; NO, TURN THE PEL ON
; INITIALIZE REGISTERS FOR THE BIOS WRITE DOT FUNCTION
MOV     DX,Y            ; ROW NUMBER FOR VERTICAL LINE
MOV     AL,COLOR        ; COLOR FOR VERTICAL
MOV     AH,WRITEDOT     ; BIOS FUNCTION NUMBER
MOV     CX,CURR$X       ; ROW NUMBER OF PEL
INT     10H            ; CALL THE BIOS VIDEO FUNCTION

; INCREMENT THE COLUMN NUMBER AND CHECK THE PEL
INC     CURR$X
JMP     LP

EXIT:   NOP

ENDM

;
;
; MACRO NAME : DRAWAXIS
; PARAMETERS : X$VERTEX-- X COORDINATE OF THE AXIS VERTEX
;             Y$VERTEX-- Y COORDINATE OF THE AXIS VERTEX
;             LENXAXIS-- LENGTH OF THE X AXIS
;             LENYAXIS-- LENGTH OF THE Y AXIS
;
; FUNCTION :
; THIS WILL DRAW THE VERTICAL AND HORIZONTAL AXIS FOR THE
; BAR CHARTS.
;
;
DRAWAXIS MACRO X$VERTEX,Y$VERTEX,LENXAXIS,LENYAXIS

; THE BEGINNING POINT OF THE VERTICAL AXIS = (Y$VERTEX - LENYAXIS)
MOV     AX,Y$VERTEX
SUB     AX,LENYAXIS

; DRAW THE VERTICAL LINE
DRAWVERT X$VERTEX,AX,LENYAXIS,COL$AXIS

; DRAW THE HORIZONTAL LINE
DRAWHORZ Y$VERTEX,X$VERTEX,LENXAXIS,COL$AXIS

ENDM

;
;
; MACRO NAME : CMNT$DSH
; PARAMETERS : FIRSTINP-- THE USERS FIRST INPUT, WHICH DATA
;                 THE USER WANTED TO DISPLAY
;                 INTVTABL-- THE TABLE NAME OF THE COMMENTS FOR
;                 THE Y AXIS
;
; FUNCTION :
; THIS WILL DISPLAY THE COMMENTS ALONG SIDE THE DASHES ON
; THE Y AXIS.
;
;

```

Sample Program 4

```
CMNT$DSH  MACRO      FIRSTINP,INTVTABL
          LOCAL      NXTCMNT

          ; CONVERT THE USERS FIRST INPUT FROM ASCII TO BINARY
          MOV        BL,FIRSTINP
          XOR        BH,BH
          SUB        BX,'0'

          ; FIND THE BEGINNING OF THE ENTRY THAT CORRESPONDS TO THE USERS CHOICE
          SUB        BX,1          ; THE TABLES BEGIN AT OFFSET ZERO
          MOV        AL,ENTRYSIZ
          MUL        BL
          MOV        SI,AX

          ; FOR EACH DASH
          MOV        CX,5          ; INITIALIZE THE LOOP COUNTER
          MOV        DL,X$DASHC    ; THE FIRST COMMENTS STARTS HERE
          MOV        DH,Y$DASHC    ; AT (X$DASHC,Y$DASHC)

          ; MOV THE CURSOR TO THE CORRECT POSITION
NXTCMNT:  MOV$CURS
          PUSH       DX          ; SAVE THE CURRENT POSITION OF THE CURSOR

          ; CALCULATE THE POSITION IN THE TABLE OF THE COMMENT TO BE PRINTED
          MOV        DX,OFFSET INTVTABL
          ADD        DX,SI

          ; PRINT THE STRING
          DOSFXN     DISPSTRG

          ; INCREMENT THE INDEX INTO THE TABLE
          ADD        SI,4

          ; INCREMENT THE POSITION OF THE NEXT DASH
          POP        DX
          SUB        DH,4

          LOOP       NXTCMNT      ; CONTINUE PROCESSING

          ENDM

;
;
; MACRO NAME : CMNT$GRP
; PARAMETERS : FIRSTINP-- THE USERS FIRST INPUT, WHICH DATA
;                  THE USER WANTED TO DISPLAY
;                  SECNDINP-- THE USERS SECOND INPUT, WHETHER THE
;                  DATA IS DISPLAYED BY YEARS OR MONTHS
;
; FUNCTION :
;   THIS WILL DISPLAY THE COMMENTS ON THE X AND Y AXIS.
;
;
CMNT$GRP  MACRO FIRSTINP,SECNDINP

          ; PUT FIVE HORIZONTAL DASHES ON THE VERTICAL AXIS
          MOV        AX,X$VERTEX    ; BEGINNING OF A DASH = COLUMN-LENGTH
          SUB        AX,LEN$DASH    ; OF A DASH
          MOV        DASH$BEG,AX    ; STORE THE VALUE
          MOV        AX,LENYAXIS    ; FIGURE OUT THE NUMBER OF ROWS BETWEEN
          DIV        FIVE           ; DASHES = LENYAXIS / 5
          MOV        AH,0
          MOV        DASHINTV,AX    ; STORE THE INTERVAL BETWEEN DASHES
```

```

; CALCULATE THE FIRST DASH ROW
MOV     CURR$Y,Y$VERTEX ; THE Y VERTEX
SUB     CURR$Y,LENYAXIS ; MINUS THE LENGTH OF THE Y AXIS

; FOR EACH FIVE DASHES DRAW A HORIZONTAL LINE
MOV     CX,5
DRAWDASH: MOV     SAVECX,CX          ; SAVE THE LOOP COUNTER
DRAWHORZ CURR$Y,DASH$BEG,LEN$DASH,COL$AXIS
MOV     AX,DASHINTV          ; UPDATE THE ROW VALUE
ADD     CURR$Y,AX

; GET THE LOOP COUNTER
MOV     CX,SAVECX
LOOP    DRAWDASH

; MOVE THE CURSOR TO PRINT THE COMMENT UNDER THE X AXIS
MOV     DL,X$COMNT          ; COLUMN
MOV     DH,Y$COMNT          ; ROW
MOV$CURS

; FIND OUT WHICH INTERVAL WE ARE WORKING WITH
CMP     SECNDINP,'Y'
JE      PRINTYRS

; PRINT MONTHS
MOV     DX,OFFSET MONTHS
DOSFXN  DISPSTRG

; PRINT THE COMMENTS BY THE DASHES ON THE Y AXIS
CMNT$DSH FIRSTINP,M$INTVL
JMP     NXTCMNT

; PRINT YEARS
PRINTYRS: MOV     DX,OFFSET YEARS
DOSFXN  DISPSTRG

; PRINT THE COMMENTS BY THE DASHES ON THE Y AXIS
CMNT$DSH FIRSTINP,Y$INTVL

; MOVE THE CURSOR IN ORDER TO
; PRINT A MESSAGE THAT DESCRIBES THE TYPE OF DATA BEING DISPLAYED
NXTCMNT: MOV     DL,X$TMSG
MOV     DH,Y$TMSG
MOV$CURS

; CALCULATE THE ADDRESS OF THE APPROPRIATE MESSAGE TO PRINT UNDER
; THE AXIS
MOV     AL,FIRSTINP          ; CONVERT THE USERS FIRST INPUT FROM ASCII
SUB     AL,'1'              ; TO HEXADECIMAL
MUL     LENCMNT              ; THE OFFSET INTO THE TABLE IS THE USERS
                                ; CHOICE * THE LENGTH OF A MESSAGE
MOV     DX,OFFSET DATATTAB
ADD     DX,AX                ; ADD THE OFFSET TO THE BEGINNING OF THE
                                ; TABLE

; PRINT THE MESSAGE
DOSFXN  DISPSTRG

; MOVE THE CURSOR IN ORDER TO PRINT "HIT ANY KEY TO RETURN TO MENU"
MOV     DL,X$HKMSG
MOV     DH,Y$HKMSG
MOV$CURS

```


Sample Program 4

```
    ; PRINT THE MESSAGE UNDER THE GRAPH
    MOV     DX,OFFSET HITKEYM
    DOSFXN  DISPSTRG

    ; MOVE THE CURSOR IN ORDER TO PRINT THE CORPORATION NAME
    MOV     DL,X$CMMSG
    MOV     DH,Y$CMMSG
    MOV$CURS

    ; PRINT THE CORPORATION NAME UNDER THE GRAPH
    MOV     DX,OFFSET CORPNAME
    DOSFXN  DISPSTRG

    ENDM

;
;
;   MACRO NAME : DISPDATA
;   PARAMETERS : FIRSTINP-- THE USER'S CHOICE
;               SECNDINP-- INTERVAL OF DATA (MONTH OR YEAR)
;   FUNCTION :
;   THIS WILL DISPLAY DATA ON A GRAPHICS SCREEN IN A BAR
;   CHART FORM.
;
;
DISPDATA MACRO FIRSTINP,SECNDINP

    ; SET THIS WINDOW TO GRAPHICS MODE
DISPD:  MOV     AL,MEDRES$G      ; MEDIUM RESOLUTION GRAPHICS
    MOV     AH,SET$MODE        ; BIOS FUNCTION TO SET THE MODE
    INT     10H

    ; SET THE COLOR PALETTE
    MOV     BH,COL$AXIS
    MOV     BL,GRY
    MOV     AH,SET$PALL
    INT     10H

    ; DRAW THE AXIS FOR THE BAR CHARTS
    DRAWAXIS X$VERTEX,Y$VERTEX,LENXAXIS,LENYAXIS

    ; DRAW BARS ON THE BAR CHART
    CALL    GRAFDATA

    ; COMMENT THE GRAPH
    CMNT$GRP FIRSTINP,SECNDINP

    ; WAIT FOR THE USER TO HIT A KEY
    DOSFXN  NO$ECHO             ; DOS FUNCTION TO INPUT A CHARACTER WITHOUT
                                ; ECHOING IT

    ;SET THE WINDOW BACK TO ASCII MODE
    MOV     AL,COL$8025
    MOV     AH,SET$MODE
    INT     10H

    ENDM
```

```

;
; MACRO NAME : GETDATA
; PARAMETERS : MFIC      -- RESOLVED VALUE FOR MFIC
;               KEYBOARD-- RESOLVED VALUE FOR KEYBOARD
;               HOST$ID  -- SESSION ID FOR THE HOST
;               PCTSKID  -- PC TASK ID
;               STRTPROG-- THE SCAN CODES  TO START THE HOST
;                           PROGRAM.
;               SENDACK  -- THE NAME OF THE BUFFER SENT TO THE
;                           HOST TO ACKNOWLEDGE THE FACT THAT
;                           WE RECEIVED THE DATA FROM THE HOST
;               BUFFER   -- MEMORY LOCATION OF RECEIVE BUFFER
; FUNCTION :
;   THIS WILL GET DATA FROM THE HOST.
;
;
GETDATA  MACRO  MFIC,KEYBOARD,HOST$ID,PCTSKID,STRTPROG,SENDACK,BUFFER

; CONNECT TO MFIC, CONNECT TO THE HOST KEYBOARD,
; DISABLE INPUT TO THE HOST AND CREATE A FIXED LENGTH QUEUE
CALL      CONNHOST

; PRINT A MESSAGE THAT SAYS WE ARE GETTING DATA FROM THE HOST
CALL      GETDATAM

; DEFINE A BUFFER TO RECEIVE INPUT FROM THE HOST
; THIS BUFFER IS DEFINED BEFORE THE HOST APPLICATION IS STARTED
; IN ORDER TO HAVE A BUFFER READY FOR THE HOST'S DATA.
DEF$RBUF WAITCMPQ,MFIC,HOST$ID,PCTSKID,BUFFER
GET$COMP 40H
CHEK4ERR 19,DBRETNCD

; START UP THE HOST APPLICATION, I.E. SEND THE KEYSTROKES TO THE
; HOST TO INVOKE A PROGRAM NAMED EXAMP
MOV       AL,IN1SCNCD      ; THE USERS SELECTION FROM THE MENU IS USED
MOV       RECNUM,AL        ; AS A RECORD NUMBER. THE HOST WILL SEND
                           ; A RECORD OF INFORMATION FROM A DATA FILE
                           ; ON THE HOST.
WRIT$KEY  KEYBOARD,HOST$ID,,STRTPROG
CMP       WKPARLST.WKRETNCD,AIDKEY
                           ; IF THE LAST KEY WAS SENT THEN AN AID
JNE       PRNTER           ; KEY WAS GENERATED (RETURN CODE = 12)
MOV       WKPARLST.WKRETNCD,0
PRNTER:   CHEK4ERR 20,WKPARLST.WKRETNCD
                           ; OTHERWISE CHECK FOR ANOTHER KEY

; GET THE COMMUNICATION STATUS FROM THE FIXED LENGTH QUEUE
GTSTATUS: DEQUEUE 02H,SF$Q$ID
CHEK4ERR 21

; IF THERE IS COMMUNICATION STATUS AND THERE IS NOT
; A MESSAGE AVAILABLE FROM THE HOST THEN CLEAN UP AND EXIT THIS
; ROUTINE BECAUSE THERE IS A PROBLEM WITH THE HOST
CMP       DQSTATUS,MSGAVAIL
JE        READMSG
JMP       PERRMSG

; THERE IS A MESSAGE FROM THE HOST SO READ IT INTO OUR BUFFER

```

Sample Program 4

```
READMSG:  READ$SF  40H,40H,MFIC,HOST$ID,PCTSKID
          GET$COMP 40H
          CHEK4ERR 22,RSRETNCD

          ; TELL THE HOST THAT WE RECEIVED THE INFORMATION
          WRIT$SF RPLYCMPS,WAITCMPS,MFIC,HOST$ID,PCTSKID,SENDACK
          CHEK4ERR 23,WSRETNCD
          JMP      CLNUP

          ; PRINT AN ERROR MESSAGE
PERRMSG:  MOV      DX,OFFSET HOSTPROB
          MOV      CX,DQSTATUS
          CHEK4ERR 21
          DOSFXN   DISPSTRG

          ; DELETE THE FIXED LENGTH QUEUES, ENABLE INPUT TO THE HOST AND
          ; DISCONNECT FROM THE HOST
CLNUP :   CALL DISCHOST

          ENDM

; 3270 P.C. API MACROS

;
;      MACRO : CHEK4ERR
;      FUNCTION :
;          SET UP THE REGISTERS FOR THE ERROR CHECKER PROCEDURE.
;
;
CHEK4ERR MACRO  CODE,RETNCODE

          MOV     AL,CODE

          IFNB    <RETNCODE>          ; IF THERE IS A PARAMETER LIST RETURN CODE
          MOV     BL,RETNCODE          ; SPECIFIED, PASS THE RETURN CODE AND THE
          MOV     BH,RETNCODE+1        ; FUNCTION ID TO THE ERROR CHECKER IN BX
          ELSE                                ; OTHERWISE, SEND A 0 IN BL
          MOV     BL,0
          ENDIF

          CALL    CHECKERR              ; CALL THE ERROR CHECKER

          ENDM

;
;      MACRO NAME : DEF$RBUF
;      PARAMETERS : WAITTYPE -- WAIT TYPE(40H=WAIT FOR COMPLETION
;                               00H=DO NOT WAIT
;                               MFIC      -- RESOLVED VALUE FOR MFIC
;                               HOSTID    -- HOST SESSION ID
;                               PCTSKID   -- PC TASK ID
;                               BUFFER    -- MEMORY LOCATION NAME OF THE MESSAGE
;                               BUFFER
;      FUNCTION :
;          USE THIS SERVICE TO DEFINE A BUFFER THAT WILL BE USED
;          TO RECEIVE A MESSAGE FROM THE SPECIFIED HOST SESSION.
;          THIS SERVICE IS VALID FOR DFT HOST SESSIONS ONLY.
;
;
```

```

DEF$RBUF  MACRO  WAITTYP,MFIC,HOSTID,PCTSKID,BUFFER

; INITIALIZE PARAMETER LIST FOR DEF$RBUF
MOV  DBRETNCD,00H      ; DBRETNCD MUST BE 0 BEFORE REQUEST
MOV  DBFXNID,00H      ; DBFXNID MUST BE 0 BEFORE REQUEST
MOV  AL,HOSTID        ; HOST ID IN
MOV  DBHOSTID,AL      ; THE LIST
MOV  AX,PCTSKID       ; PC TASK ID
MOV  DBTASKID,AX      ; IN THE LIST
MOV  AX,OFFSET BUFFER ; OFFSET OF MESSAGE BUFFER
MOV  DBOFFSET,AX      ; IN THE LIST
MOV  AX,SEG BUFFER    ; SEGMENT OF THE MESSAGE BUFFER
MOV  DBSEGMNT,AX      ; IN THE LIST

; INITIALIZE THE 8 BYTE HEADER OF THE MESSAGE BUFFER
MOV  BUFFER,0
MOV  WORD PTR BUFFER + 2,0
MOV  WORD PTR BUFFER + 4,800H ;LENGTH OF RECEIVE BUFFER
MOV  WORD PTR BUFFER + 6,0

; INITIALIZE REGISTERS FOR DEF$RBUF
MOV  AH,09H
MOV  AL,05H
MOV  BH,40H
MOV  BL,WAITTYP      ; WAIT TYPE IN BL
MOV  CX,0            ; PRIORITY IN CX
MOV  DX,MFIC         ; RESOLVED VALUE FOR MFIC
MOV  DI, SEG DBRETNCD ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI           ; IN ES
MOV  DI,OFFSET DBRETNCD ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR DEF$RBUF SERVICE
INT  7AH

ENDM

;
;
; MACRO : WRIT$KEY
;
; PARAMETERS : SERVTYPE -- SERVICE TYPE
;              SESSID  -- SESSION ID
;              SCANCD  -- SCAN CODE
;              SHIFST  -- SHIFT STATE
;              LISTNAME -- THE NAME OF THE LIST OF KEYSTROKES
;              TASKID  -- CONNECTOR'S TASK ID (OPTIONAL)
;
; FUNCTION :
;           SEND A KEYSTROKE OR A LIST OF KEYSTROKES TO THE
;           SPECIFIED SESSION.
;
;
;
WRIT$KEY  MACRO  SERVTYPE,SESSID,SCANCD,SHIFST,LISTNAME,TASKID
LOCAL  WKEND

MOV  WKPARLST.WKRETNCD,0H
; WKRETCD MUST BE 0 FOR THE CALL
MOV  WKPARLST.WKFXNID,0H ; WKFXNID MUST BE 0 FOR THE CALL
MOV  AL,SESSID          ; PUT THE SESSION ID IN PARM LIST
MOV  WKPARLST.WKSESSID,AL

```

Sample Program 4

```
IFNB <SCANCD>                ; CHECK IF SENDING ONE KEYSTROKE
                                ; OR A LIST OF KEYSTROKES

; SENDING ONE KEYSTROKE

MOV AL,SCANCD                 ; PUT THE SCAN CODE IN THE PARM LIST
MOV WKPARGST.WKSCNCOD,AL
MOV AL,SHIFST                 ; PUT SHIFT STATE IN THE PARM LIST
MOV WKPARGST.WKSHFST,AL

MOV AL,20H                    ; PUT THE OPTION BYTE FOR SENDING
MOV WKPARGST.WKOPTION,AL ; ONE CHARACTER IN THE PARM LIST

ELSE
; SENDING A LIST OF KEYSTROKES

MOV AX,OFFSET LISTNAME ; PUT THE OFFSET ADDRESS OF THE LIST
MOV WKPARGST.WKLSTOFF,AX
                                ; INTO THE PARAMETER LIST
MOV AX,SEG LISTNAME ; PUT THE SEGMENT ADDRESS OF THE LIST
MOV WKPARGST.WKLSTSEG,AX; INTO THE PARAMETER LIST

MOV AL,30H                    ; PUT THE OPTION BYTE FOR SENDING A
MOV WKPARGST.WKOPTION,AL ; LIST OF CHARS. IN THE PARM LIST

ENDIF

IFNB <TASKID>                ; IF A CONNECTOR'S TASK ID WAS
MOV AX,TASKID                ; SPECIFIED, PUT IT IN THE LIST
ELSE
MOV AX,0                      ; OTHERWISE PUT A 0 IN THE LIST
ENDIF
MOV WKPARGST.WKTASKID,AX

; INITIALIZE THE REGISTERS FOR WRIT$KEY
MOV AH,09H
MOV AL,04H
MOV BH,80H
MOV BL,20H
MOV CX,0000H
MOV DX,SERVTYPE ; SERVICE TYPE IN DX
MOV DI, SEG WKPARGST ; GET SEGMENT ADDRESS OF PARM LIST
MOV ES,DI ; AND PUT IT IN ES
MOV DI, OFFSET WKPARGST ; SET DI TO OFFSET OF PARM LIST

INT 7AH ; PASS THE REQUEST TO THE API

WKEND: NOP

ENDM

;
; MACRO NAME : DEQUEUE
; PARAMETERS : WAITTYPE -- WAIT TYPE
;              QUEUEID -- ID OF THE FIXED-LENGTH QUEUE
; FUNCTION :
;           OBTAIN AN ELEMENT FROM A FIXED-LENGTH QUEUE.
;
;
```

```

DEQUEUE    MACRO    WAITTYPE,QUEUEID

; INITIALIZE REGISTERS FOR DEQUEUE
MOV    AH,13H
MOV    BL,WAITTYPE        ; WAIT TYPE IN BL
MOV    CX,0004H
MOV    DX,QUEUEID        ; FIXED-LENGTH QUEUE ID IN DX
MOV    DI, SEG DQSESSID   ; SEGMENT ADDRESS OF DATA AREA IN ES
MOV    ES,DI
MOV    DI,OFFSET DQSESSID ; OFFSET ADDRESS OF DATA AREA IN DI

; SIGNAL WORKSTATION PROGRAM FOR DEQUEUE SERVICE
INT    7AH

ENDM

;
;
; MACRO NAME : READ$SF
;
; PARAMETERS : RPLYTYPE -- REPLY TYPE
;              WAITTYPE -- WAIT TYPE
;              MFIC      -- RESOLVED VALUE FOR MFIC
;              HOSTID    -- HOST SESSION ID
;              PCTSKID   -- PC TASK ID
;
; FUNCTION :
;           USE THIS SERVICE TO READ STRUCTURED FIELD DATA FROM THE
;           SPECIFIED HOST SESSION.  THIS SERVICE IS VALID FOR DFT
;           HOST SESSIONS ONLY.
;
;
;
READ$SF    MACRO    RPLYTYPE,WAITTYP,MFIC,HOSTID,PCTSKID
LOCAL    RSEND

; INITIALIZE PARAMETER LIST FOR READ$SF
MOV    RSRETNCD,00H        ; RSRETNCD MUST BE 0 BEFORE REQUEST
MOV    RSFXNID,00H        ; RSFXNID MUST BE 0 BEFORE REQUEST
MOV    AL,HOSTID          ; HOST ID IN
MOV    RSHOSTID,AL        ; THE LIST
MOV    AX,PCTSKID         ; PC TASK ID
MOV    RSTASKID,AX        ; IN LIST
MOV    RSZERO,0          ; THIS FIELD MUST BE ZEROED

; INITIALIZE REGISTERS FOR READ$SF
MOV    AH,09H
MOV    AL,03H
MOV    BH,RPLYTYPE        ; REPLY TYPE IN BH
MOV    BL,WAITTYP         ; WAIT TYPE IN BL
MOV    CX,0               ; PRIORITY IN CX
MOV    DX,MFIC            ; RESOLVED VALUE FOR MFIC
MOV    DI, SEG RSRETNCD   ; SEGMENT ADDRESS OF PARAMETER LIST
MOV    ES,DI              ; IN ES
MOV    DI,OFFSET RSRETNCD ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR READ$SF SERVICE
INT    7AH

ENDM

```

Sample Program 4

```
;
;
;      MACRO NAME : WRIT$SF
;      PARAMETERS : RPLYTYPE -- REPLY TYPE
;                   WAITTYPE -- WAIT TYPE
;                   MFIC      -- RESOLVED VALUE FOR MFIC
;                   HOSTID    -- HOST SESSION ID
;                   PCTSKID   -- PC TASK ID
;                   STR$DATA  -- MEMORY LOCATION NAME OF STRUCTURED
;                               FIELD DATA
;
;      FUNCTION :
;      USE THIS SERVICE TO WRITE STRUCTURED FIELD DATA FROM THE
;      SPECIFIED HOST SESSION.  THIS SERVICE IS VALID FOR DFT
;      HOST SESSIONS ONLY.
;
;
WRIT$SF MACRO RPLYTYPE,WAITTYP,MFIC,HOSTID,PCTSKID,STR$DATA

; INITIALIZE PARAMETER LIST FOR READ$SF
MOV  WSRETNCD,00H      ; WSRETNCD MUST BE 0 BEFORE REQUEST
MOV  WSFXNID,00H       ; WSFXNID MUST BE 0 BEFORE REQUEST
MOV  AL,HOSTID         ; HOST ID IN
MOV  WSHOSTID,AL       ; THE LIST
; OFFSET AND SEGMENT OF DATA IN LIST
MOV  WSOFFSD,OFFSET STR$DATA
MOV  WSSEGTD,SEG STR$DATA
MOV  AX,PCTSKID        ; PC TASK ID
MOV  WTASKID,AX        ; IN LIST
MOV  WSZERO,0         ; THIS FIELD MUST BE ZEROED

; INITIALIZE THE FIELDS IN THE STRUCTURED FIELD 8 BYTE HEADER
MOV  STR$DATA,0
MOV  WORD PTR STR$DATA + 4,0
MOV  WORD PTR STR$DATA + 6,0

; INITIALIZE REGISTERS FOR WRIT$SF
MOV  AH,09H
MOV  AL,04H
MOV  BH,RPLYTYPE       ; REPLY TYPE IN BH
MOV  BL,WAITTYP        ; WAIT TYPE IN BL
MOV  CX,0              ; PRIORITY IN CX
MOV  DX,MFIC           ; RESOLVED VALUE FOR MFIC
MOV  DI, SEG WSRETNCD  ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI             ; IN ES
MOV  DI,OFFSET WSRETNCD ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR WRIT$SF SERVICE
INT  7AH

ENDM

;
;
;      MACRO : GET$COMP
;      FUNCTION:
;      USE THIS SERVICE TO OBTAIN THE CONTENTS OF A SPECIFIED
;      REQUEST QUEUE ELEMENT.
;      ONE PARAMETER IS PASSED WHICH INDICATES WHETHER THE USER
;      WANTS TO WAIT UNTIL RESULTS ARE READY (40H) OR TO CHECK IF
;      RESULTS ARE AVAILABLE AND GET THEM IF THEY ARE READY (00H).
;
;
;
```

```

GET$COMP MACRO WAIT

    MOV BL, WAIT
    MOV AH, 83H ; SET UP THE REGS FOR A GET$COMP CALL
    MOV CX, 0000H
    INT 7AH

    ENDM

STACKSEG SEGMENT STACK 'STACK'
    DB STAKSIZE DUP(?)
STACKSEG ENDS

DATASEG SEGMENT PUBLIC

; THESE VARIABLES ARE LOCATED IN THE SECOND LOAD MODULE
EXTRN PCSESSID:BYTE, KEYBOARD:WORD, MENU$PS:BYTE, HOST$ID:BYTE, PCTASKID:WORD
EXTRN MFIC:WORD, IN1SCNCD:BYTE, SF$Q$ID:WORD, SECNDINP:BYTE, SESSMGR:WORD
EXTRN FIRSTINP:BYTE

; THESE VARIABLES ARE USED BY THE SECOND LOAD MODULE
PUBLIC BUFFAREA, CURR$Y

CURR$Y DW 0 ; CURRENT Y POINTER, USED TO DRAW A
          ; VERTICAL LINE
CURR$X DW 0 ; CURRENT X POINTER, USED TO DRAW A
          ; HORIZONTAL LINE
DASH$BEG DW 0 ; USED TO DRAW A DASH ON THE Y AXIS
DASHINTV DW 0 ; INTERVAL BETWEEN DASHED ON THE Y AXIS
MONTHS DB 'J F M A M J J A S O N D$'
YEARS DB ' 1980 1981 1982 1983 1984$'
          ; COMMENTS TO BE DISPLAYED UNDER THE X AXIS
          ; OF THE GRAPH
SAVECX DW 0 ; SAVE A LOOP COUNTER
FIVE DB 5 ; USED FOR DIVIDE INSTRUCTION

; TABLES FOR PRINTING OUT COMMENTS BY THE Y AXIS
; THE FIRST TABLE IS FOR PRINTING OUT DATA BY THE MONTH
; THE SECOND TABLE IS FOR PRINTING OUT DATA BY THE YEAR
M$INTVL DB ' 20$ 40$ 60$ 80$100$'
        DB ' 8$ 16$ 24$ 32$ 40$'
        DB ' 3$ 6$ 9$ 13$ 15$'
        DB ' 20$ 40$ 60$ 80$100$'
        DB ' 5$ 10$ 15$ 20$ 25$'
Y$INTVL DB '180$360$540$720$900$'
        DB ' 80$160$240$320$400$'
        DB ' 30$ 60$ 90$130$150$'
        DB ' 60$120$180$240$300$'
        DB ' 25$ 50$ 75$100$125$'

; MESSAGES
HOSTPROB DB 'PROBLEM WITH HOST, CANNOT GET DATA $'
CORPNAME DB 'STEMIR CORPORATION$'
WAITMSG DB 'GETTING DATA FROM THE HOST$'
HITKEYM DB 'HIT ANY KEY TO RETURN TO MENU$'

; MESSAGES PRINTED UNDER THE GRAPH DESCRIBING THE DATA
DATATTAB DB 'GROSS REVENUE (THOUSANDS)$'
        DB 'NET REVENUE (THOUSANDS)$'
        DB 'PRODUCTS SOLD (THOUSANDS)$'
        DB 'EMPLOYEE OVERTIME (HOURS)$'
        DB 'EMPLOYEE ILLNESS (DAYS) $'

```


Sample Program 4

```
; LENGTH OF EACH STRING IN THE ABOVE TABLE
LENCMNT    DB          33

; A LIST OF SCAN CODES AND SHIFTSTATES SENT TO THE HOST TO START THE
; PROGRAM THAT IS ON THE HOST
STRTPROG   DW          54          ; LENGTH OF THE LIST
E          DB          24H,UPCASE
X          DB          22H,UPCASE
A          DB          1CH,UPCASE
M          DB          3AH,UPCASE
P          DB          4DH,UPCASE
SPACE1     DB          29H,UPCASE
S          DB          1BH,UPCASE
T          DB          2CH,UPCASE
E$2        DB          24H,UPCASE
M$2        DB          3AH,UPCASE
I          DB          43H,UPCASE
R          DB          2DH,UPCASE
SPACE2     DB          29H,UPCASE
D          DB          23H,UPCASE
A$2        DB          1CH,UPCASE
T$2        DB          2CH,UPCASE
A$3        DB          1CH,UPCASE
SPACE3     DB          29H,UPCASE
LFTPARN1   DB          46H,UPCASE
R$2        DB          2DH,UPCASE
E$3        DB          24H,UPCASE
C          DB          21H,UPCASE
LFTPARN2   DB          46H,UPCASE
RECNUM     DB          ?          ; THIS IS THE RECORD THAT WE WANT FROM
                                   ; THE HOST

          DB          0
RHTPARN1   DB          45H,UPCASE
RHTPARN    DB          45H,UPCASE
ENTERKEY   DB          58H,0
; 8 BYTE HEADER FOLLOWED BY ACKNOWLEDGMENT THAT IS SENT TO THE HOST PROGRAM
; VIA WRIT$SF
; THE FORMAT OF THE SUCCESSFUL TRANSMISSION RESPONSE CAN BE FOUND IN
; APPENDIX B, "FORMAT OF THE HOST INSERT AND INSERT
; DATA REQUESTS."
SENDACK    DB          0,0,0BH,0,0,0,0,0
          DB          0,0BH,0DOH,47H,05H,63H,06H
DATBLK     DB          0,0,0,1

; DEFINE A RECEIVE BUFFER
; THE RECEIVE BUFFER IS AN EIGHT BYTE HEADER FOLLOWED BY THE DATA AREA
; THE FORMAT OF THE HOST DATA IS DOCUMENTED IN APPENDIX B,
; "FORMAT OF THE HOST INSERT AND INSERT DATA REQUESTS."
BUFFER     DW          0
OBLN       DW          0          ; OUTBOUND TRANSMISSION LEN
                                   ; EXCLUDING THE BUFFER HEADER
RB$SIZ     DW          0          ; SIZE OF RECEIVE BUFFER ON DEF$RBUF
          DB          12 DUP(0)
          DW          0          ; OUTBOUND TRANSMISSION LEN
                                   ; INCLUDING THE BUFFER HEADER
          DB          6 DUP(0)
LD         DW          0          ; LENGTH OF DATA + 5
BUFFAREA   DB          2000 DUP(0) ; 2000 BYTES FOR THE BUFFER
```

; PARAMETER LIST FOR DEF\$RBUF

```
DBRETNCD DB 0 ; RETURN CODE
DBFXNID DB 0 ; FUNCTION NUMBER
DBHOSTID DB 0 ; HOST SESSION ID
          DB 0 ; UNCHANGED
          DW 0 ; NOT USED
          DW 0001H
          DW 0 ; UNUSED
DBOFFSET DW 0 ; SEGMENT AND OFFSET OF THE MESSAGE BUFFER
DBSEGMNT DW 0
          DW 0 ; UNUSED
DBTASKID DW 0 ; PC TASK ID
          DW 0
          DW 9 DUP(0) ; SYSTEM WORK AREA
```

; PARAMETER LIST STRUCTURE FOR WRIT\$KEY

```
WRKYPARM STRUC
WKRETNCD DB 0
WKFXNID DB 0
WKSESSID DB 0
WKSPARE DB 0
WKTASKID DW 0
WKOPTION DB 0
WKNUMKEY DB 0
WKSCNCOD DB 0
WKSHFST DB 0
WKRESRV2 DW 0
WRKYPARM ENDS

WRKYPAR2 STRUC
          DB 8 DUP(00)
WKLSTOFF DW 0
WKLSTSEG DW 0
WRKYPAR2 ENDS
```

; ALLOCATE STORAGE FOR THE WRITE KEYSTROKE PARAMETER LIST

WKPARLST WRKYPARM <>

; DATA AREA FOR DEQUEUE

```
DQSESSID DB 0 ; SESSION ID
DQRESERV DB 0 ; RESERVED
DQSTATUS DW 0 ; STATUS CODE FOLLOWED BY
              ; STATUS TYPE
```

; PARAMETER LIST FOR READ\$SF

```
RSRETNCD DB 0 ; RETURN CODE
RSFXNID DB 0 ; FUNCTION NUMBER
RSHOSTID DB 0 ; HOST SESSION ID
RSZERO DB 0 ; UNCHANGED
          DW 0 ; NOT USED
          DW 0001H ; STRUCTURED FIELD TYPE, (DEST/ORIG)
          DW 0 ; UNUSED
RSOFFSD DW 0 ; OFFSET ADDRESS OF STRUCTURED FIELD DATA
RSSEGTD DW 0 ; SEGMENT ADDRESS OF STRUCTURED FIELD DATA
          DW 0 ; UNUSED
RSTASKID DW 0 ; PC TASK ID
          DW 0
          DW 9 DUP(0) ; SYSTEM WORK AREA
```

Sample Program 4

; PARAMETER LIST FOR WRIT\$SF

WSRETNCD	DB	0	; RETURN CODE
WSFXNID	DB	0	; FUNCTION NUMBER
WSHOSTID	DB	0	; HOST SESSION ID
WSZERO	DB	0	; UNCHANGED
	DW	0	; NOT USED
	DW	0001H	; STRUCTURED FIELD TYPE, (DEST/ORIG)
	DW	0	; UNUSED
WSOFFSD	DW	0	; OFFSET ADDRESS OF STRUCTURED FIELD DATA
WSSEGTD	DW	0	; SEGMENT ADDRESS OF STRUCTURED FIELD DATA
	DW	0	; UNUSED
WSTASKID	DW	0	; PC TASK ID
	DW	0	
	DW	9 DUP(0)	; SYSTEM WORK AREA

DATASEG ENDS

;
; *** MAIN BODY ***
;

CODESEG SEGMENT PUBLIC

; THESE ENTRY POINTS ARE LOCATED IN THE SECOND LOAD MODULE
EXTRN INIT:NEAR,CHECKERR:NEAR,GET\$RESP:NEAR,CONNHOST:NEAR,DISCHOST:NEAR,
GRAFDATA:NEAR

; THESE ENTRY POINTS ARE USED BY THE SECOND LOAD MODULE
PUBLIC THE\$END,DISPD

; ESTABLISH ADDRESSABILITY OF CODE
ASSUME CS:CODESEG

; ESTABLISH ADDRESSABILITY OF DATA BY DS AND ES REGISTER
START: MOV AX,DATASEG
MOV DS,AX
MOV ES,AX
ASSUME DS:DATASEG,ES:DATASEG

; RESOLVE NAMES FOR SERVICES AND FIND THE BASE WINDOW SESSION ID
CALL INIT

; DISPLAY THE DATA
MAINLOOP: DISPMENU PCSESSID,MENU\$PS

; GET THE USERS REQUEST FROM THE MENU
CALL GET\$RESP

; IS THE USER FINISHED?
CMP AL,ESC ; DID THE USER HIT THE ESCAPE KEY?
JNE CONT ; NO, PROCESS HIS REQUEST
JMP THE\$END ; YES, THE USER IS FINISHED, CLEAN UP

; GET THE DATA, (CORRESPONDING TO THE USERS REQUEST) FROM THE HOST

```
CONT:      GETDATA MFIC,KEYBOARD,HOST$ID,PCTASKID,STRTPROG,SENDACK,BUFFER
           ; DISPLAY THE DATA
           DISPDATA FIRSTINP,SECNDINP

           ; CONTINUE GETTING USER'S REQUEST
           JMP      MAINLOOP
```

[illegible]

```

;      PROCEDURE: CLEARSCR
;      FUNCTION :
;      THIS PROCEDURE WILL CLEAR THE SCREEN.
;
;
;

```

```

; SET THE SEGMENT TO THE BEGINNING OF THE HARDWARE BUFFER
MOV      AX,0B000H
MOV      ES,AX

; INITIALIZE THE LOOP COUNTER
MOV      CX,2000          ; THE BUFFER HAS 2000 CHARACTERS AND EACH
                           ; CHARACTER IS FOLLOWED BY AN ATTRIBUTE
MOV      DI,0

```

```

;
;      MACRO NAME : GETDATAM
;      FUNCTION :
;          THIS PROCEDURE WILL PRINT A MESSAGE ON THE MENU SAYING
;          THAT THE DATA IS BEING GOTTEN FROM THE HOST.
;
;
;

```

Sample Program 4

```
        ; PRINT THE STRING
        MOV     DX,OFFSET WAITMSG
        DOSFXN  DISPSTRG

        RET                                ; TO CALLER

GETDATAM ENDP

CODESEG ENDS                                ; END OF THE CODE SEGMENT
        END    START                       ; IDENTIFY THE START ADDRESS OF THE SOURCE
```

Chapter 29. Sample Program 5

```

;
;
;      THIS PROGRAM IS THE SECOND MODULE OF THE STEMIR PROGRAM.  THE
;      TWO MODULES ARE TO BE ASSEMBLED SEPARATELY AND THEN LINKED TOGETHER.
;
      PAGE 60,132

;
;      ****  DATA  ****
;

DATASEG  SEGMENT PUBLIC

; DECLARE VARIABLES THAT ARE NEEDED BY THE FIRST MODULE OF THE PROGRAM

PUBLIC   PCSESSID,KEYBOARD,MENU$PS,HOST$ID,PCTASKID,MFIC,IN1SCNCD,SF$Q$ID
PUBLIC   FIRSTINP,SECNDINP,SESSMGR

; DECLARE VARIABLES THAT ARE LOCATED IN THE FIRST MODULE OF THE PROGRAM

EXTRN    BUFFAREA:WORD,CURR$Y:WORD

; PARAMETER LIST FOR CONN$KEY

CKRETNCD DB 0                      ; RETURN CODE
CKFXNID  DB 0                      ; FUNCTION NUMBER
CKSESSID DB 0                      ; SESSION ID
CKRESRV1 DB 0                      ; RESERVED
CKEVENTQ DW 0                     ; EVENT QUEUE ID
CKKEYSTQ DW 0                     ; KEYSTROKE QUEUE ID
CKOPTION DB 40H                   ; OPTION BYTE (INTERCEPT ALL)
CKRESRV2 DB 0                      ; RESERVED

; PARAMETER LIST FOR CONN$SF

CFRETNCD DB 0                      ; RETURN CODE
CFFXNID  DB 0                      ; FUNCTION NUMBER
CFSESSID DB 0                      ; SESSION ID
CFRESRV1 DB 0                      ; MUST BE 0
CFFLQID  DW 0                      ; FIXED-LENGTH QUEUE ID
          DW 0001H                 ; MUST BE 00001H
CFEVNTS  DW 0600H                 ; EVENTS TO BE ENQUEUED - DFT
CFQRPLY  DD QUERYREP              ; OFFSET AND SEGMENT OF THE QUERY REPLY
CFRESRV2 DW 0                     ; MUST BE 0
CFTASKID DW 0                     ; PC TASK ID
CFRESRV3 DW 0                     ; MUST BE 0
CFWORK   DW 9 DUP(0)              ; SYSTEM WORK AREA

```

Sample Program 5

; QUERY REPLY FOR DESTINATION/ORIGIN

QUERYREP	DW	1900H	; LENGTH OF THE STRUCTURE
	DB	81H	; QUERY REPLY
	DB	9DH	; ANOMALY IMPLEMENTATION
	DB	0	; MUST BE 0
	DB	01H	; STRUCTURED FIELD EXCHANGE
	DW	0008H	; MAXIMUM NUMBER OF BYTES IN INBOUND TRANSMISSION
	DW	0008H	; MAXIMUM NUMBER OF BYTES IN OUTBOUND TRANSMISSION
	DB	0FH	; RESERVED
	DW	4801H	; MY OWN DESTINATION ORIGIN ID
QRAPLNAM	DB	12 DUP(0)	; PC APPLICATION NAME IN EBCDIC

; PARAMETER LIST FOR CRT\$Q

CQOFFSET	DW	0	; OFFSET OF THE QUEUE
CQSEGMENT	DW	0	; SEGMENT ADDRESS OF THE QUEUE
CQNAME	DB	8 DUP(0)	; QUEUE NAME

; PARAMETER LIST FOR DISC\$KEY

DKRETNCD	DB	0	; RETURN CODE
DKFXNID	DB	0	; FUNCTION NUMBER
DKSESSID	DB	0	; SESSION ID
DKRESRV1	DB	0	; RESERVED
DKTASKID	DW	0	; CONNECTOR'S TASK ID
DKRESRV2	DB	0	; RESERVED

; PARAMETER LIST FOR DISC\$SF

DFRETNCD	DB	0	; RETURN CODE
DFFXNID	DB	0	; FUNCTION NUMBER
DFSESSID	DB	0	; SESSION ID
DFRESERV1	DB	0	; RESERVED
	DW	0	; NOT USED
DFTYPE	DW	0001H	; DISCONNECT TYPE - DESTINATION/ORIGIN
	DW	6 DUP(0)	; NOT USED
DFWORK	DW	9 DUP(0)	; SYSTEM WORK AREA

; DEFINE PARAMETER LIST FOR DISA\$INP

DIRETNCD	DB	0	; RETURN CODE
DIFXNID	DB	0	; FUNCTION CODE
DISESID	DB	0	; SESSION ID
DIRESRVD	DB	0	; RESERVED
DICONNID	DW	0	; CONNECTORS TASK ID

; DEFINE PARAMETER LIST FOR ENAB\$INP

EIRETNCD	DB	0	; RETURN CODE
EIFXNID	DB	0	; FUNCTION ID
EISESID	DB	0	; SESSID
EIRESRVD	DB	0	; RESERVED
EICONNID	DW	0	; CONNECTOR'S TASK ID

; PARAMETER LIST FOR QUERY\$ID

QDRETNCD	DB	0	; RETURN CODE
QDFXNID	DB	0	; FUNCTION NUMBER
QDOPTION	DB	0	; OPTION BYTE
QDDATA	DB	0	; DATA BYTE
QDNAMOFF	DW	0	; OFFSET OF NAME TABLE
QDNAMSEG	DW	0	; SEGMENT OF NAME TABLE
QDLNGNAM	DB	8 DUP(?)	; SESSION LONG NAME

; PARAMETER LIST FOR Q\$BAS\$W

QSRETNCD	DB	0	; RETURN CODE
QSFXNID	DB	0	; FUNCTION NUMBER
QSENVID	DB	0	; ENVIRONMENT ID
QSSESSID	DB	0	; SESSION ID
QSWINDOW	DB	0	; WINDOW SHORT NAME
QSRESERV	DB	0	; RESERVED

; PARAMETER LIST FOR TRANSLAT

TLRETNCD	DB	0	; RETURN CODE
TLFXNID	DB	0	; FUNCTION NUMBER
TLSRCOFF	DW	0	; OFFSET ADDRESS OF SOURCE BUFFER
TLSRCSEG	DW	0	; SEGMENT ADDRESS OF SOURCE BUFFER
TLTRGOFF	DW	0	; OFFSET ADDRESS OF TARGET BUFFER
TLTRGSEG	DW	0	; SEGMENT ADDRESS OF TARGET BUFFER
TLTYPE	DB	0	; TRANSLATION TYPE
TLRESERV	DB	0	; RESERVED
TLENGTH	DW	0	; LENGTH

NAMARRAY	DB	14	; NAME ARRAY FOR QUERY\$ID FUNCTION
NUMSESS	DB	0	; NUMBER OF MATCHING SESSIONS
SHRTNAME	DB	0	; SESSION SHORT NAME
SESSTYPE	DB	0	; SESSION TYPE
SESSID	DB	0	; SESSION ID
SPARE	DB	0	
LONGNAME	DB	8 DUP(0)	; LONG NAME OF SESSION

KYBDNAME	DB	'KEYBOARD'	; PARM LIST FOR NAME\$RES ON KEYBOARD
SMGRNAME	DB	'SESSMGR '	; PARM LIST FOR NAME\$RES ON SESSMGR
XLATNAME	DB	'XLATE '	; PARM LIST FOR NAME\$RES ON XLATE
MFICNAME	DB	'MFIC '	; PARM LIST FOR NAME\$RES ON MFIC

KEYBOARD	DW	0	; KEYBOARD SERVICE TYPE
SESSMGR	DW	0	; SESSMGR SERVICE TYPE
XLATE	DW	0	; XLATE SERVICE TYPE
MFIC	DW	0	; MFIC SERVICE TYPE

PCSESSID	DB	0	; SESSION ID OF THE GRAPHICS WINDOW
----------	----	---	-------------------------------------

HOST\$ID	DB	0	; SESSION ID OF THE HOST WINDOW
HOST\$WND	DB	0	; SHORT NAME OF THE HOST WINDOW

PCTASKID	DW	0	; TASK ID FOR THIS PC PROGRAM
----------	----	---	-------------------------------

COUNT	DW	0	; NUMBER OF BAR FIELDS ON THE X AXIS
-------	----	---	--------------------------------------

BFWIDTH	DW	0	; WIDTH IN PELS OF THE BAR FIELD
BARWIDTH	DW	0	; WIDTH OF ONE BAR (3/4 OF BFWIDTH)
HEIGHT	DW	0	; HEIGHT IN PELS OF A BAR

Sample Program 5

```
STARTX    DW    0          ; X POSITION OF THE LOWER LEFT CORNER OF THE BAR
STARTY    DW    0          ; Y POSITION OF THE LOWER LEFT CORNER OF THE BAR

YAXISLEN  DW    0          ; LENGTH OF THE Y AXIS IN PELS

YMAX      DW    0          ; MAXIMUM VALUE ON THE Y AXIS

IN1SCNCD  DB    0          ; SCAN CODE FOR THE FIRST INPUT IN THE MENU

SF$Q$ID   DW    0          ; ID OF THE HOST INTERACTIVE FIXED$LENGTH QUEUE
SF$Q$LEN  DW    64         ; LENGTH OF THE HOST INTERACTIVE QUEUE
SF$Q      DB    64 DUP(0)   ; THE HOST INTERACTIVE QUEUE

HOSTQ$ID  DW    0          ; ID OF THE HOST KEYSTROKE FIXED$LENGTH QUEUE
HOSTQLEN  DW    64         ; LENGTH OF THE HOST KEYSTROKE QUEUE
HOSTQ     DB    64  DUP(0)   ; THE HOST KEYSTROKE QUEUE

FIRSTINP  DB    0          ; THE CHARACTER IN THE FIRST INPUT FIELD
SECNDINP  DB    0          ; THE CHARACTER IN THE SECOND INPUT FIELD

MENU$PS   DB    201,7      ; THE PRESENTATION SPACE FOR THE SELECTION MENU
          DB    78 DUP(205,7)
          DB    187,7

          DB    186,7
          DW    78 DUP(0700H)
          DB    186,7

          DB    186,7
          DW    30 DUP(0700H)
          DB    'S',4,'T',4,'E',4,'M',4,'I',4,'R',4,0,0
          DB    'C',4,'O',4,'R',4,'P',4,'O',4,'R',4,'A',4
          DB    'T',4,'I',4,'O',4,'N',4
          DW    30 DUP(0700H)
          DB    186,7

          DB    186,7
          DW    31 DUP(0700H)
          DB    'G',4,'R',4,'A',4,'P',4,'H',4,'I',4,'C',4,0,0
          DB    'S',4,'U',4,'M',4,'M',4,'A',4,'R',4,'Y',4
          DW    32 DUP(0700H)
          DB    186,7

          DB    186,7
          DW    78 DUP(0700H)
          DB    186,7

          DB    186,7
          DW    78 DUP(0700H)
          DB    186,7

          DB    186,7
          DW    10 DUP(0700H)
          DB    '1',1,0,0,'-',1,0,0,'G',1,'R',1,'O',1,'S',1,'S',1,0,0
          DB    'R',1,'E',1,'V',1,'E',1,'N',1,'U',1,'E',1
          DW    51 DUP(0700H)
          DB    186,7
```

```

DB      186,7
DW      10 DUP(0700H)
DB      '2',1,0,0,'-',1,0,0,'N',1,'E',1,'T',1,0,0
DB      'R',1,'E',1,'V',1,'E',1,'N',1,'U',1,'E',1
DW      53 DUP(0700H)
DB      186,7

DB      186,7
DW      10 DUP(0700H)
DB      '3',1,0,0,'-',1,0,0,'N',1,'U',1,'M',1,'B',1,'E',1,'R',1,0,0
DB      'O',1,'F',1,0,0,'P',1,'R',1,'O',1,'D',1,'U',1,'C',1,'T',1,'S',1
DB      0,0,'S',1,'O',1,'L',1,'D',1
DW      41 DUP(0700H)
DB      186,7

DB      186,7
DW      10 DUP(0700H)
DB      '4',1,0,0,'-',1,0,0,'E',1,'M',1,'P',1,'L',1,'O',1,'Y',1
DB      'E',1,'E',1,0,0,'O',1,'V',1,'E',1,'R',1,'T',1,'I',1,'M',1,'E',1
DB      0,0,'H',1,'O',1,'U',1,'R',1,'S',1
DW      41 DUP(0700H)
DB      186,7

DB      186,7
DW      10 DUP(0700H)
DB      '5',1,0,0,'-',1,0,0,'E',1,'M',1,'P',1,'L',1,'O',1,'Y',1
DB      'E',1,'E',1,0,0,'I',1,'L',1,'L',1,'N',1,'E',1,'S',1,'S',1,0,0
DB      'D',1,'A',1,'Y',1,'S',1
DW      43 DUP(0700H)
DB      186,7

DB      186,7
DW      78 DUP(0700H)
DB      186,7

DB      186,7
DW      10 DUP(0700H)
DB      'S',3,'E',3,'L',3,'E',3,'C',3,'T',3,0,0,'T',3,'H',3,'E',3,0,0
DB      'D',3,'A',3,'T',3,'A',3,0,0,'T',3,'O',3,0,0
DB      'D',3,'I',3,'S',3,'P',3,'L',3,'A',3,'Y',3,0,0,205,3,205,3,'>',3
DB      0,0
DB      0,7
DW      36 DUP(0700H)
DB      186,7

DB      186,7
DW      78 DUP(0700H)
DB      186,7

DB      186,7
DW      78 DUP(0700H)
DB      186,7

DB      186,7
DW      10 DUP(0700H)
DB      'M',1,0,0,'-',1,0,0,'D',1,'I',1,'S',1,'P',1,'L',1,'A',1,'Y',1
DB      0,0,'D',1,'A',1,'T',1,'A',1,0,0,'F',1,'O',1,'R',1,0,0
DB      'T',1,'H',1,'E',1,0,0,'L',1,'A',1,'S',1,'T',1,0,0,'1',1,'2',1
DB      0,0,'M',1,'O',1,'N',1,'T',1,'H',1,'S',1
DW      29 DUP(0700H)
DB      186,7

```

Sample Program 5

```
DB      186,7
DW      10 DUP(0700H)
DB      'Y',1,0,0,'-',1,0,0,'D',1,'I',1,'S',1,'P',1,'L',1,'A',1,'Y',1
DB      0,0,'D',1,'A',1,'T',1,'A',1,0,0,'F',1,'O',1,'R',1,0,0
DB      'T',1,'H',1,'E',1,0,0,'L',1,'A',1,'S',1,'T',1,0,0,'5',1
DB      0,0,'Y',1,'E',1,'A',1,'R',1,'S',1
DW      31 DUP(0700H)
DB      186,7

DB      186,7
DW      78 DUP(0700H)
DB      186,7

DB      186,7
DW      10 DUP(0700H)
DB      'S',3,'E',3,'L',3,'E',3,'C',3,'T',3,0,0,'T',3,'H',3,'E',3,0,0
DB      'T',3,'I',3,'M',3,'E',3,0,0,'S',3,'P',3,'A',3,'N',3,0,0
DB      205,3,205,3,'>',3,0,0
DB      0,7
DW      42 DUP(0700H)
DB      186,7

DB      186,7
DW      78 DUP(0700H)
DB      186,7

DB      186,7
DW      78 DUP(8400H)
DB      186,7

DB      186,7
DW      10 DUP(0700H)
DB      'P',7,'R',7,'E',7,'S',7,'S',7,0,0,'E',7,'N',7,'T',7,'E',7,'R',7
DB      0,0,'(',7,17,7,196,7,217,7,')',7,0,0,'T',7,'O',7,0,0
DB      'D',7,'I',7,'S',7,'P',7,'L',7,'A',7,'Y',7,0,0,'T',7,'H',7,'E',7
DB      0,0,'S',7,'E',7,'L',7,'E',7,'C',7,'T',7,'E',7,'D',7,0,0
DB      'D',7,'A',7,'T',7,'A',7
DW      22 DUP(0700H)
DB      186,7

DB      186,7
DW      10 DUP(0700H)
DB      'P',7,'R',7,'E',7,'S',7,'S',7,0,0,'E',7,'S',7,'C',7,0,0
DB      'T',7,'O',7,0,0,'E',7,'X',7,'I',7,'T',7
DW      51 DUP(0700H)
DB      186,7

DB      186,7
DW      78 DUP(0700H)
DB      186,7

DB      200,7
DB      78 DUP(205,7)
DB      188,7

DW      80 DUP(0700H)

YMAXTAB  DW      100,40,15000,100,25,900,400,150,300,125
PROGNAME  DB      'STEMIR .EXE'
HSTPRMPT  DB      'ENTER THE SHORT NAME OF THE HOST WINDOW',13,10
          DB      205,205,16,' $'
```

```

NOTDFT    DB      13,10,'INCORRECT HOST WINDOW SPECIFIED.  WINDOW MUST BE A DFT '
          DB      'HOST SESSION.',13,10
          DB      'PROGRAM TERMINATED',13,10,'$'

ERRMSG     DB      13,10,'ERROR IN API CALL.  PROGRAM TERMINATED.',13,10,'$'

ASCII2SCN  DB      45H,16H,1EH,26H,25H,2EH,36H,3DH,3EH,46H

DATABUFF   STRUC

MNTHTDATA  DW      12 DUP(?)      ; DATA FOR THE LAST 12 MONTHS
YEARDATA   DW      5  DUP(?)      ; DATA FOR THE LAST 5 YEARS

DATABUFF   ENDS

DATASEG    ENDS

;
;          ****  DEFINE MACROS  ****
;

;
;          MACRO : DOSFXN
;          FUNCTION :
;                  THIS WILL CALL A DOS FUNCTION SPECIFIED BY THE PARAMETER FXNNUM.
;

DOSFXN     MACRO FXNNUM

          MOV      AH,FXNNUM      ; THE FUNCTION NUMBER BELONGS IN AL
          INT      21H            ; CALL DOS

          ENDM

;
;          MACRO : DISPLAY
;          FUNCTION :
;                  DISPLAY THE CHARACTER STRING PASSED IN STRING.
;

; ESTABLISH CONSTANTS

DISPSTRN   EQU      9              ; DOS DISPLAY STRING FUNCTION NUMBER

DISPLAY    MACRO STRING

          LEA      DX,STRING      ; DX POINTS TO THE STRING TO BE DISPLAYED
          DOSFXN    DISPSTRN      ; DISPLAY THE CHARACTER

          ENDM

;
;          MACRO : DRAW$BOX
;          FUNCTION :
;                  DRAWS A FILLED IN BOX WITH ITS LOWER LEFT CORNER AT X0,Y0 WITH
;                  THE SPECIFIED HEIGHT AND LENGTH.  THIS IS DONE BY DRAWING LENGTH
;                  NUMBER OF VERTICAL LINES NEXT TO EACH OTHER EACH WITH LENGTH HEIGHT.
;

```

Sample Program 5

```
; ESTABLISH CONSTANTS

BARCOLOR EQU 2 ; THE COLOR OF THE BOX

DRAW$BOX MACRO X0,Y0,HEIGHT,LENGTH

    MOV CX,LENGTH
    MOV STARTX,X0 ; SAVE A COPY OF THE X STARTING OFFSET
    MOV AX,Y0 ; CALCULATE THE OFFSET FROM THE TOP OF THE SCREEN
    SUB AX,HEIGHT ; OF THE TOP OF THE BOX

    MOV STARTY,AX ; SAVE THE TOP OF BOX OFFSET

NEXTLINE: PUSH CX ; SAVE THE COUNT
    DRAWVERT STARTX,STARTY,HEIGHT,BARCOLOR

    POP CX ; RESTORE THE COUNT
    INC STARTX ; POINT STARTX TO THE NEXT COLUMN TO DRAW
    LOOP NEXTLINE ; DRAW THE NEXT LINE

ENDM

;
; MACRO NAME : DRAWVERT
; PARAMETERS : X -- VALUE ON THE X AXIS,(COLUMN)
; YBEGIN -- VALUE ON THE Y AXIS,(ROW)
; LEN -- LENGTH OF THE LINE
; COLOR -- COLOR OF THE LINE
; FUNCTION :
; THIS WILL DRAW A VERTICAL LINE. THE LINE WILL BE DRAWN
; FROM POINT (X,YBEGIN) TO (X,YBEGIN + LEN).
;

DRAWVERT MACRO X,YBEGIN,LEN,COLOR
    LOCAL LP,EXIT

    ; SAVE THE CURRENT Y VALUE
    MOV AX,YBEGIN
    MOV CURR$Y,AX

    ; HAVE WE REACHED THE LAST PEL YET?
    MOV BX,LEN ; THE LAST PEL = THE FIRST PEL PLUS
    ADD BX,CURR$Y ; THE LENGTH OF THE LINE
LP: CMP CURR$Y,BX

    ; YES, WE ARE DONE
    JE EXIT

    ; NO, TURN THE PEL ON
    ; INITIALIZE REGISTERS FOR THE BIOS WRITE DOT FUNCTION
    MOV CX,X ; COLUMN NUMBER FOR VERTICAL LINE
    MOV AL,COLOR ; COLOR FOR VERTICAL
    MOV AH,WRITEDOT ; BIOS FUNCTION NUMBER

    MOV DX,CURR$Y ; ROW NUMBER OF PEL
    INT 10H ; CALL THE BIOS VIDEO FUNCTION

    ; INCREMENT THE ROW NUMBER AND CHECK THE PEL
    INC CURR$Y
    JMP LP

EXIT: NOP

ENDM
```

```

;
;   MACRO : CHEK4ERR
;   FUNCTION :
;           SET UP THE REGISTERS FOR THE ERROR CHECKER PROCEDURE.
;
CHEK4ERR MACRO RETNCODE

    IFNB <RETNCODE> ; IF THERE IS A PARAMETER LIST RETURN CODE
    MOV  BL,RETNCODE ; SPECIFIED, PASS THE RETURN CODE IN BL
    ELSE
    MOV  BL,0 ; OTHERWISE, SEND A 0 IN BL
    ENDIF

    CALL CHECKERR ; CALL THE ERROR CHECKER

    ENDM

;
;   MACRO NAME : CONN$KEY
;   PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'KEYBOARD'
;               SESSID  -- SESSION ID
;   FUNCTION :
;           CONNECT THE KEYBOARD TO THE SPECIFIED SESSION.
;
CONN$KEY MACRO SERVTYPE,SESSID,KEYSTQ,EVNTQ

    ; INITIALIZE PARAMETER LIST FOR CONN$KEY
    MOV  CKRETNCD,00H ; RETURN CODE MUST = 0 BEFORE REQUEST
    MOV  CKFXNID,00H ; FUNCTION ID MUST = 0 BEFORE REQUEST
    MOV  AL,SESSID ; SESSION ID INTO THE LIST
    MOV  CKSESSID,AL

    IFNB <KEYSTQ>
    MOV  AX,KEYSTQ ; IF A KEYSTROKE QUEUE WAS SPECIFIED,
    ELSE ; PUT IT INTO THE LIST
    MOV  AX,0
    ENDIF
    MOV  CKKEYSTQ,AX

    IFNB <EVNTQ>
    MOV  AX,EVNTQ ; IF AN EVENT QUEUE WAS SPECIFIED,
    ELSE ; PUT IT INTO THE LIST
    MOV  AX,0
    ENDIF
    MOV  CKEVENTQ,AX

    ; INITIALIZE REGISTERS FOR CONN$KEY
    MOV  AH,09H
    MOV  AL,01H
    MOV  BH,80H
    MOV  BL,20H
    MOV  CX,0000H

```

Sample Program 5

```
MOV    DX,SERVTYPE      ; RESOLVED VALUE FOR 'KEYBOARD'
MOV    DI, SEG CKRETNCD  ; SEGMENT ADDRESS OF PARAMETER LIST
MOV    ES,DI            ; IN ES
MOV    DI,OFFSET CKRETNCD ; OFFSET OF PARAMETER LIST IN DI
```

```
; SIGNAL WORKSTATION PROGRAM FOR CONN$KEY SERVICE
INT    7AH
```

```
ENDM
```

```
;
;
; MACRO NAME : CONN$SF
; PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'MFIC '
;              SESSID  -- SESSION ID
;              QUEUEID  -- FIXED-LENGTH QUEUE ID
;              PCTAKSID -- PC TASK ID
;              PCAPLNAM -- PC APPLICATION NAME
;
; FUNCTION :
; CONNECT TO THE SPECIFIED HOST SESSION FOR HOST INTER-
; ACTIVE SERVICES. THIS MACRO CONNECTS FOR DESTINATION/ORIGIN
; STRUCTURED FIELDS. DESTINATION/ORIGIN IS VALID FOR DFT
; HOST SESSIONS ONLY.
;
```

```
CONN$SF MACRO SERVTYPE,SESSID,QUEUEID,PCTASKID,PCAPLNAM
```

```
; INITIALIZE PARAMETER LIST FOR CONN$SF
MOV    CFRETNCD,00H      ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV    CFFXNID,00H      ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV    AL,SESSID        ; SESSION ID INTO THE LIST
MOV    CFSSESSID,AL
MOV    AX,QUEUEID       ; FIXED-LENGTH QUEUE ID INTO THE LIST
MOV    CFFLQID,AX
MOV    AX,PCTASKID      ; PC TASK ID INTO THE LIST
MOV    CFTASKID,AX
```

```
TRANSLAT XLATE,PCAPLNAM,QRAPLNAM,01H,12
; TRANSLATE THE PC APPLICATION NAME INTO
; EBCDIC AND PUT IT IN THE QUERY REPLY
```

```
; INITIALIZE REGISTERS FOR CONN$SF
MOV    AH,09H
MOV    AL,01H
MOV    BH,80H
MOV    BL,20H
MOV    CX,0000H
MOV    DX,SERVTYPE      ; RESOLVED VALUE FOR 'MFIC '
MOV    DI, SEG CFRETNCD  ; SEGMENT ADDRESS OF PARAMETER LIST
MOV    ES,DI            ; IN ES
MOV    DI,OFFSET CFRETNCD ; OFFSET OF PARAMETER LIST IN DI
```

```
; SIGNAL WORKSTATION PROGRAM FOR CONN$SF SERVICE
INT    7AH
```

```
ENDM
```

```

;
; MACRO NAME : CRT$Q
; PARAMETERS : QUEUE      -- THE QUEUE TO CREATE
;              QSIZE      -- SIZE OF QUEUE
; FUNCTION :
;             CREATE A FIXED LENGTH QUEUE.
;

CRT$Q MACRO  QUEUE,QNAME,QSIZE,RETURNID

; INITIALIZE PARAMETER LIST FOR CRT$Q
MOV  AX,SEG QUEUE      ; SEGMENT ADDRESS INTO THE LIST
MOV  CQSEGMEN,AX
MOV  AX,OFFSET QUEUE   ; OFFSET INTO THE LIST
MOV  CQOFFSET,AX

; INITIALIZE REGISTERS FOR CRT$Q
MOV  AH,04H
MOV  BL,00H            ; NO NAME SPECIFIED
MOV  CX,QSIZE          ; SIZE OF THE QUEUE
MOV  DX,0000H          ; DX MUST = 0 FOR THE REQUEST
MOV  DI, SEG CQOFFSET   ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI             ; IN ES
MOV  DI,OFFSET CQOFFSET ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR CRT$Q SERVICE
INT  7AH

MOV  RETURNID,DX      ; PASS THE QUEUE ID BACK TO THE CALLER

ENDM

;
; MACRO NAME : DEL$ENT
; PARAMETERS : QUEID - FIXED LENGTH QUEUE ID
; FUNCTION:
;             DELETE THE ENTRY FROM THE SYSTEM
;

DEL$ENT MACRO  QUEID

; INITIALIZE REGISTERS FOR DEL$ENT REQUEST
MOV  AH,06H           ; AH = X'06'
MOV  CX,0000H          ; CX = X'0000'
MOV  DX,QUEID          ; DX = FIXED LENGTH QUEUE ID

; REQUEST THE DEL$ENT SERVICE
INT  7AH

ENDM

;
; MACRO NAME : DISC$KEY
; PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'KEYBOARD'
;              SESSID   -- SESSION ID
; FUNCTION :
;             DISCONNECT THE KEY BOARD FROM THE SPECIFIED SESSION.
;

```


Sample Program 5

```
DISC$KEY  MACRO  SERVTYPE,SESSID,TASKID

; INITIALIZE PARAMETER LIST FOR DISC$KEY
MOV  DKRETNCD,00H      ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV  DKFXNID,00H       ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV  AL,SESSID         ; SESSION ID INTO THE LIST
MOV  DKSESSID,AL

IFNB  <TASKID>
MOV  AX,TASKID         ; IF A TASK ID WAS SPECIFIED, PUT IT
ELSE                                ; IN THE LIST
MOV  AX,0000H
ENDIF
MOV  DKTASKID,AX

; INITIALIZE REGISTERS FOR DISC$KEY
MOV  AH,09H
MOV  AL,02H
MOV  BH,80H
MOV  BL,20H
MOV  CX,0000H
MOV  DX,SERVTYPE       ; RESOLVED VALUE FOR 'KEYBOARD'
MOV  DI, SEG DKRETNCD   ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI              ; IN ES
MOV  DI,OFFSET DKRETNCD ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR DISC$KEY SERVICE
INT  7AH

ENDM

;
;
; MACRO NAME : DISC$SF
;
; PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'MFIC'
;              SESSID  -- SESSION ID
;
; FUNCTION :
;           DISCONNECT FROM THE SPECIFIED HOST SESSION.  THIS MACRO
;           IS CODED TO DISCONNECT FROM A DESTINATION/ORIGIN CONNECTION.
;
;

DISC$SF  MACRO  SERVTYPE,SESSID

; INITIALIZE PARAMETER LIST FOR DISC$SF
MOV  DFRETNCD,00H      ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV  DFFXNID,00H       ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV  AL,SESSID         ; SESSION ID INTO THE LIST
MOV  DFSESSID,AL

; INITIALIZE REGISTERS FOR DISC$SF
MOV  AH,09H
MOV  AL,02H
MOV  BH,80H
MOV  BL,20H
MOV  CX,0000H
MOV  DX,SERVTYPE       ; RESOLVED VALUE FOR 'MFIC'
MOV  DI, SEG DFRETNCD   ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI              ; IN ES
MOV  DI,OFFSET DFRETNCD ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR DISC$SF SERVICE
INT  7AH

ENDM
```

```

;
;   MACRO NAME : DISA$INP
;   PARAMETERS : SERVTYPE - RESOLVED VALUE FOR 'KEYBOARD'
;                 SESSID  - SESSION ID
;                 CONN$ID  - (OPTIONAL) CONNECTORS TASK ID IS
;                           NEEDED ONLY IF THE TASK THAT REQ-
;                           UESTED THE CONNECT TO KEYBOARD FOR
;                           THIS SESSION IS DIFFERENT FROM THE
;                           TASK REQUESTING THE DISABLE INPUT.
;
;   FUNCTION :
;       USE THIS SERVICE TO DISABLE OPERATOR INPUT TO THE
;       SESSION.
;
DISA$INP MACRO    SERVTYPE,SESSID,CONN$ID

    MOV    AX,SEG DIRETNCD    ; ADDRESSABILITY OF
    MOV    ES,AX              ; PARAMETER LIST
    MOV    DI,OFFSET DIRETNCD ; USING ES:DI

    ; INITIALIZE PARAMETER LIST FOR DISABLE INPUT
    MOV    DIRETNCD,00H       ; RETURN CODE MUST=0 ON REQUEST
    MOV    DIFXNID,00H       ; FUNCTION ID MUST=0 ON REQUEST
    MOV    AL,SESSID          ; KEYBOARD INPUT DISABLED FOR
    MOV    DISESID,AL         ; THIS SESSION
    IFNB   <CONN$ID>          ; IF THERE IS A CONNECTORS ID
    MOV    AX,CONN$ID         ; THEN PUT IT IN THE
    MOV    DICONNID,AX        ; PARAMETER LIST
    ENDIF

    ; INITIALIZE REGISTERS FOR DISABLE INPUT
    MOV    AX,0905H           ; LOCK INPUT SERVICE
    MOV    BX,8020H
    MOV    CX,0000H
    MOV    DX,SERVTYPE        ; DX=NAME RESOLUTION FOR THE
                                ; KEYBOARD SERVICES
    ; REQUEST DISABLE INPUT SERVICE
    INT    7AH

    ENDM

;
;   MACRO NAME : ENAB$INP
;   PARAMETERS : SERVTYPE - RESOLVED VALUE FOR 'KEYBOARD'
;                 SESSID  - SESSION ID
;                 CONN$ID  - (OPTIONAL) THE CONNECTORS TASK ID IS
;                           NEEDED ONLY IF THE TASK THAT RE-
;                           QUESTED THE CONNECT TO KEYBOARD FOR
;                           THIS SESSION IS DIFFERENT FROM THE
;                           TASK REQUESTING THE ENABLE INPUT
;
;   FUNCTION :
;       USE THIS SERVICE TO ENABLE OPERATOR INPUT TO THE
;       SESSION.
;
ENAB$INP MACRO    SERVTYPE,SESSID,CONN$ID

    MOV    AX,SEG EIRETNCD    ; ADDRESSABILITY OF
    MOV    ES,AX              ; PARAMETER LIST
    MOV    DI,OFFSET EIRETNCD ; USING ES:DI

```

Sample Program 5

```
; INITIALIZE PARAMETER LIST FOR ENABLE INPUT
MOV  EIRETNCD,00H      ; RETURN CODE MUST=0 ON REQUEST
MOV  EIFXNID,00H      ; FUNCTION ID MUST=0 ON REQUEST
MOV  AL,SESSID        ; KEYBOARD INPUT DISABLED FOR
MOV  EISESID,AL        ; THIS SESSION
IFNB <CONN$ID>        ; IF THERE IS A CONNECTOR'S ID
MOV  AX,CONN$ID        ; THEN STORE THE ID
MOV  EICONNID,AX      ; IN THE PARAMETER LIST
ENDIF
```

```
; INITIALIZE REGISTERS FOR ENABLE INPUT
MOV  AX,0906H          ; ENABLE INPUT SERVICE
MOV  BX,8020H
MOV  CX,0000H
MOV  DX,SERVTYPE      ; DX=NAME RESOLUTION FOR THE
                      ; KEYBOARD SERVICES
; REQUEST ENABLE INPUT SERVICE
INT  7AH
```

ENDM

```
;
;      MACRO      - NAME$RES
;      PARAMETERS - NR$SERVN - LOCATION OF THE 8 BYTE
;                      SERVICE NAME. I.E.'SESSMGR '
;                      NR$SERVT - RETURN CODE FROM PARAMETER LIST
;
```

NAME\$RES MACRO NR\$SERVN,NR\$SERVT

```
; SET UP REGISTERS NAME$RES
MOV  AX,SEG NR$SERVN   ; SEGMENT ADDRESS OF PARAM. LIST
MOV  ES,AX             ; ES = SEGM ADDRESS OF PARAM.LIST
MOV  AH,81H           ; AH = X'81'
MOV  CX,0000H         ; CX = X'0000'
MOV  DI,OFFSET NR$SERVN ; DI = OFFSET ADDR. OF PARAM LIST
```

```
; REQUEST SERVICE TYPE FROM WORKSTATION PROGRAM
INT  7AH
```

```
; RETURN SERVICE TYPE ID TO CALLER
MOV  NR$SERVT,DX
```

ENDM

```
;
;      MACRO NAME : QUERY$ID
;      PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'SESSMGR '
;                      NAMEARRAY -- NAME ARRAY
;                      OPTION    -- OPTION BYTE
;                      DATA     -- DATA BYTE
;                      LONGNAME  -- SESSION LONG NAME
;
;      FUNCTION :
;          GET THE SESSION ID(S) OF THE SESSION(S) SPECIFIED BY
;          THE OPTION AND DATA BYTES AND RETURNS THEM IN THE NAME
;          ARRAY.
;      NOTE: THE NAME ARRAY IS SET UP BY THE USER AND MUST HAVE
;            THE LENGTH OF THE ARRAY CONTAINED IN THE 1ST BYTE.
;
```

```

QUERY$ID  MACRO  SERVTYPE,NAMEARRY,OPTION,DATA,LONGNAME

; INITIALIZE PARAMETER LIST FOR QUERY$ID
MOV  QDRETNCD,00H      ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV  QDFXNID,00H      ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV  AL,OPTION         ; OPTION BYTE INTO THE LIST
MOV  QDOPTION,AL
MOV  AL,DATA           ; DATA BYTE INTO THE LIST
MOV  QDDATA,AL
MOV  AX,OFFSET NAMEARRY ; NAME ARRAY OFFSET INTO THE LIST
MOV  QDNAMOFF,AX
MOV  AX,SEG NAMEARRY   ; NAME ARRAY SEGMENT INTO THE LIST
MOV  QDNAMSEG,AX

IFNB  <LONGNAME>      ; CHECK IF A LONG NAME WAS SPECIFIED

CLD                                ; COPY DIRECTION = FORWARD
MOV  AX,SEG QDLNGNAM
MOV  ES,AX                    ; ES:DI POINTS TO DESTINATION IN PARM
MOV  DI,OFFSET QDLNGNAM      ; LIST
MOV  SI,OFFSET LONGNAME     ; DS:SI POINTS TO SOURCE OF LONG NAME
MOV  CX,8                    ; MOVE 8 BYTES
REP  MOVSB                  ; COPY LONG NAME INTO THE PARM LIST

ENDIF

; INITIALIZE REGISTERS FOR QUERY$ID
MOV  AH,09H
MOV  AL,01H
MOV  BH,80H
MOV  BL,20H
MOV  CX,0000H
MOV  DX,SERVTYPE            ; RESOLVED VALUE FOR 'SESSMGR '
MOV  DI,SEG QDRETNCD        ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI                  ; IN ES
MOV  DI,OFFSET QDRETNCD     ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR QUERY$ID SERVICE
INT  7AH

ENDM

;
;
; MACRO NAME : Q$BAS$W
;
; PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'SESSMGR '
;
; FUNCTION :
;
;         FIND THE SESSION ID AND SHORT NAME FOR THE BASE WINDOW
;
;         OF AN ENVIRONMENT.
;

Q$BAS$W  MACRO  SERVTYPE

; INITIALIZE PARAMETER LIST FOR Q$BAS$W
MOV  QSRETNCD,00H          ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV  QSFXNID,00H          ; FUNCTION ID MUST = 0 BEFORE REQUEST

```

Sample Program 5

```
; INITIALIZE REGISTERS FOR Q$BAS$W
MOV  AH,09H
MOV  AL,0AH
MOV  BH,80H
MOV  BL,20H
MOV  CX,0FFH
MOV  DX,SERVTYPE      ; RESOLVED VALUE FOR 'SESSMGR '
MOV  DI, SEG QSRETNCD  ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI             ; IN ES
MOV  DI,OFFSET QSRETNCD ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR Q$BAS$W SERVICE
INT  7AH

ENDM
```

```
;
;
; MACRO NAME : Q$TASK
; PARAMETERS : NONE
; FUNCTION :
;           GET THE ID OF THE CURRENT ACTIVE TASK.
;
```

```
Q$TASK  MACRO

; INITIALIZE REGISTERS FOR Q$TASK
MOV  AH,9CH

; SIGNAL WORKSTATION PROGRAM FOR Q$TASK SERVICE
INT  7AH

ENDM
```

```
;
;
; MACRO NAME : TRANSLAT
; PARAMETERS : SERVTYPE -- RESOLVED VALUE FOR 'XLATE '
;              SOURCE   -- SOURCE BUFFER
;              TARGET   -- TARGET BUFFER
;              TYPE     -- TYPE OF TRANSLATE
;              LENGTH   -- NUMBER OF BYTES TO TRANSLATE
;
; FUNCTION :
;           TRANSLATE THE DATA IN A BUFFER FROM ASCII CODES TO
;           MFI CODES, OR FROM MFI CODES TO ASCII CODES.
;
TRANSLAT MACRO  SERVTYPE,SOURCE,TARGET,TYPE,LENGTH
```

```
; INITIALIZE PARAMETER LIST FOR TRANSLAT
MOV  TLRETNCD,00H      ; RETURN CODE MUST = 0 BEFORE REQUEST
MOV  TLFXNID,00H       ; FUNCTION ID MUST = 0 BEFORE REQUEST
MOV  AX,OFFSET SOURCE  ; SOURCE OFFSET INTO THE LIST
MOV  TLSRCOFF,AX
MOV  AX,SEG SOURCE     ; SOURCE SEGMENT INTO THE LIST
MOV  TLSRCSEG,AX
MOV  AX,OFFSET TARGET  ; TARGET OFFSET INTO THE LIST
MOV  TLTRGOFF,AX
MOV  AX,SEG TARGET     ; TARGET SEGMENT INTO THE LIST
MOV  TLTRGSEG,AX
MOV  AL,TYPE           ; TRANSLATION TYPE INTO THE LIST
MOV  TLTYPE,AL
MOV  AX,LENGTH         ; LENGTH INTO THE LIST
MOV  TLLENGTH,AX
```

```

; INITIALIZE REGISTERS FOR TRANSLAT
MOV  AH,09H
MOV  AL,01H
MOV  BH,80H
MOV  BL,20H
MOV  CX,OFFH
MOV  DX,SERVTYPE      ; RESOLVED VALUE FOR 'XLATE  '
MOV  DI, SEG TLRETNCD  ; SEGMENT ADDRESS OF PARAMETER LIST
MOV  ES,DI             ; IN ES
MOV  DI,OFFSET TLRETNCD ; OFFSET OF PARAMETER LIST IN DI

; SIGNAL WORKSTATION PROGRAM FOR TRANSLAT SERVICE
INT  7AH

ENDM

;
;      ****  CODE  ****
;

CODESEG  SEGMENT PUBLIC

; DECLARE ENTRY POINTS THAT ARE NEEDED BY THE FIRST MODULE OF THE PROGRAM
PUBLIC  INIT,CHECKERR,GET$RESP,CONNHOST,DISCHOST,GRAFDATA,CONNSF

; DECLARE ENTRY POINTS THAT ARE LOCATED IN THE FIRST MODULE OF THE PROGRAM
EXTRN  THE$END:NEAR

        ASSUME CS:CODESEG,DS:DATASEG,ES:DATASEG

X$VERTEX EQU  50          ; X,Y CO-ORDINATES OF THE VERTEX OF THE GRAPH
Y$VERTEX EQU  167

LENXAXIS EQU  260        ; LENGTH IN PELS OF THE X AXIS
LENYAXIS EQU  160        ; LENGTH IN PELS OF THE Y AXIS

WRITEDOT EQU  0CH        ; BIOS FUNCTION NUMBER FOR WRITING A DOT

;      PROCEDURE : INIT
;      CALLED BY : MAIN
;      FUNCTION :
;      THIS PROCEDURE DOES THE INITIAL WORK NEEDED FOR THIS PROGRAM.
;      FIRST, IT FINDS THE RESOLVED VALUES FOR THE KEYBOARD, SESSION INFOR-
;      MATION, HOST INTERACTIVE, AND TRANSLATE SERVICES. THEN IT PROMPTS
;      THE USER FOR THE SHORT NAME OF THE HOST WINDOW. NEXT IT FINDS THE
;      SESSION IDS FOR THIS PC SESSION AND FOR THE HOST SESSION. IT CHECKS
;      TO MAKE SURE THE HOST CONNECTION IS FOR DFT. LAST IT FINDS THE TASK
;      ID FOR THIS PC APPLICATION.
;
; ESTABLISH CONSTANTS

DFTTYPE EQU  02H          ; NUMBER TO INDICATE SESSION IS DFT HOST
INKEY   EQU  01H          ; DOS INPUT KEY WITH ECHO FUNCTION NUMBER

```

Sample Program 5

```
INIT      PROC    NEAR

;;;;;;;;;;;;;
;; FIND THE RESOLVED VALUES FOR KEYBOARD, SESSION MANAGER, TRANS-  ;;
;; LATE, AND HOST INTERACTIVE SERVICES.                               ;;
;;;;;;;;;;;;;

NAME$RES  KYBDNAME,KEYBOARD
          ; FIND THE RESOLVED VALUE FOR KEYBOARD SERVICES
CHEK4ERR

NAME$RES  SMGRNAME,SESSMGR
          ; FIND THE RESOLVED VALUE FOR SESSION INFORMATION
          ; SERVICES
CHEK4ERR

NAME$RES  XLATNAME,XLATE
          ; FIND THE RESOLVED VALUE FOR TRANSLATE SERVICES
CHEK4ERR

NAME$RES  MFICNAME,MFIC
          ; FIND THE RESOLVED VALUE FOR HOST INTERACTIVE
          ; SERVICES
CHEK4ERR

;;;;;;;;;;;;;
;; PROMPT THE USER FOR THE SHORT NAME OF THE HOST WINDOW.  GET THE  ;;
;; USER'S RESPONSE AND CONVERT IT TO UPPER CASE.                     ;;
;;;;;;;;;;;;;

DISPLAY   HSTPRMPT  ; PROMPT THE USER FOR THE HOST WINDOW SHORT NAME

DOSFXN    INKEY      ; GET THE USERS RESPONSE IN AL
CMP        AL,' '      ; CHECK IF THE CHARACTER IS UPPER CASE
JL         OKINPUT    ; IF SO, STORE IT
ADD        AL,'A'-' '  ; OTHERWISE, CONVERT IT TO UPPERCASE
OKINPUT:   MOV        HOST$WND,AL  ; SAVE THE HOST WINDOW SHORT NAME

;;;;;;;;;;;;;
;; FIND THE SESSION ID AND TYPE OF THE SPECIFIED HOST WINDOW.  IF    ;;
;; THE SESSION TYPE IS NOT DFT THEN DISPLAY AN ERROR MESSAGE AND    ;;
;; EXIT.  THE HOST CONNECTION MUST BE DFT SINCE THIS PROGRAM USES   ;;
;; DESTINATION/ORIGIN STRUCTURED FIELDS WHICH CAN ONLY BE USED WITH ;;
;; DFT HOST CONNECTIONS.                                           ;;
;;;;;;;;;;;;;

QUERY$ID  SESSMGR,NAMARRAY,01H,HOST$WND
          ; GET THE HOST SESSION ID AND TYPE
CHEK4ERR  QDRETNCD

CMP        SESSTYPE,DFTTYPE
          ; CHECK IF THIS HOST IS DFT
JE         OKHOST

DISPLAY   NOTDFT      ; IF NOT, DISPLAY AN ERROR MESSAGE AND EXIT
JMP       THE$END
```

```
OKHOST:    MOV     AL,SESSID      ; MOVE THE HOST SESSION ID TO HOST$ID
           MOV     HOST$ID,AL
```

```

; ; FIND THE SESSION ID FOR THIS PC SESSION. ; ;

```

```
Q$BAS$W      SESSMGR      ; GET THE SESSION ID FOR THIS PC SESSION
CHEK4ERR     OSRETNCD
```

```
MOV     AL,QSSESSID      ; MOVE THE PC SESSION ID TO PCSESSID
MOV     PCSESSID,AL
```

[illegible]

```
Q$TASK          ; GET THIS PC APPLICATION'S TASK ID
MOV     PCTASKID,DX ; SAVE THE TASK ID IN PCTASKID
```

RET

INIT ENDP

```

;
;      PROCEDURE : GET$RESP
;      CALLED BY : MAIN
;      FUNCTION :
;
;          THIS PROCEDURE GETS THE USERS RESPONSES TO THE MENU SELECTIONS.
;          IT STARTS BY SETTING THE CURSOR AT THE FIRST INPUT FIELD.  IT THEN
;          WAITS FOR THE USER TO PRESS A KEY.  IF THE KEY WAS THE ESCAPE KEY,
;          MEANING USER WISHES TO EXIT THE PROGRAM, THEN THE PROCEDURE RETURNS
;          TO THE CALLER.  IF THE KEY WAS THE PC ENTER KEY, MEANING THE USER
;          WANTS THE SELECTED DATA TO BE DISPLAYED, THEN A CHECK IS MADE TO
;          MAKE SURE BOTH INPUT FIELDS ARE NOT BLANK.  IF BOTH ARE NOT BLANK,
;          THEN THE PROCEDURE RETURNS TO THE CALLER.  IF AT LEAST ONE FIELD IS
;          BLANK, THEN A BEEP IS ISSUED TO SIGNAL AN ERROR AND THE PROCEDURE
;          LOOPS BACK TO READ IN ANOTHER CHARACTER.  THIS KEEPS THE USER FROM
;          TRYING TO DISPLAY DATA WITHOUT SPECIFYING WHICH DATA ARE TO BE DIS-
;          PLAYED.
;
;          WHEN ENTERING INPUT INTO THE FIRST FIELD, A CHECK IS MADE TO
;          MAKE SURE THE INPUT CHARACTER IS BETWEEN '1' AND '5'.  IF THE CHAR-
;          ACTER IS NOT IN THIS RANGE, A BEEP IS ISSUED AND THE PROCEDURE LOOPS
;          BACK TO READ IN ANOTHER CHARACTER.  IF THE CHARACTER IS WITHIN RANGE
;          THE CHARACTER IS DISPLAYED ON THE MENU AND THE CURSOR IS MOVED TO
;          THE SECOND INPUT FIELD.  THE CHARACTER ENTERED IN THE SECOND INPUT
;          FIELD IS CHECKED TO MAKE SURE IT IS EITHER A 'Y' OR AN 'M'.  AS IN
;          THE CASE FOR THE FIRST FIELD, A BEEP IS ISSUED ON AN ERROR, OTHER-
;          WISE THE CHARACTER IS DISPLAYED ON THE MENU AND THE CURSOR IS MOVED
;          TO THE FIRST INPUT POSITION.
;
;

```


Sample Program 5

; ESTABLISH CONSTANTS

```
BEEP      EQU    07H      ; ASCII FOR A BEEP CHARACTER
CR        EQU    0DH      ; ASCII FOR A CARRIAGE RETURN
ESC       EQU    1BH      ; ASCII FOR AN ESCAPE
DISPCHAR  EQU    02H      ; DOS DISPLAY CHARACTER FUNCTION NUMBER
INNOECHO  EQU    08H      ; DOS INPUT WITH NO ECHO FUNCTION NUMBER
SETCURS   EQU    02H      ; BIOS SET CURSOR POSITION FUNCTION NUMBER
IN1ROW    EQU    12       ; ROW CO-ORDINATE FOR 1ST INPUT AREA
IN1COL    EQU    42       ; COLUMN CO-ORDINATE FOR 1ST INPUT AREA
IN2ROW    EQU    18       ; ROW CO-ORDINATE FOR 2ND INPUT AREA
IN2COL    EQU    36       ; COLUMN CO-ORDINATE FOR 2ND INPUT AREA
```

GET\$RESP PROC NEAR

```
;;;;;;;;;;;;;
;; CLEAR THE INPUT VARIABLES.                                ;;
;; SET THE CURSOR AT THE FIRST INPUT FIELD.                  ;;
;;;;;;;;;;;;;
```

```
MOV  FIRSTINP,0      ; CLEAR THE FIRST INPUT VARIABLE
MOV  SECNDINP,0      ; CLEAR THE SECOND INPUT VARIABLE

MOV  DH,IN1ROW       ; DH, DL = ROW, COLUMN OF THE CURSOR
MOV  DL,IN1COL
MOV  BH,0            ; BH = DISPLAY PAGE NUMBER = 0
MOV  AH,SETCURS
INT  10H
```

```
;;;;;;;;;;;;;
;; GET THE USER'S RESPONSE.  IF THE ESCAPE KEY WAS HIT THEN RETURN ;;
;; TO THE MAIN PROGRAM.  IF THE ENTER KEY WAS HIT THEN CHECK THE   ;;
;; INPUT FIELDS TO MAKE SURE THE USER HAS ENTERED BOTH INPUTS.  IF ;;
;; EITHER INPUT IS BLANK THEN BEEP AT THE USER, OTHERWISE RETURN TO ;;
;; THE MAIN PROGRAM.  IF ANY OTHER KEY WAS HIT, THEN IF THE CURSOR ;;
;; IS ON THE FIRST INPUT, GO TO THE CODE TO HANDLE THE FIRST INPUT ;;
;; FIELD, OTHERWISE GO TO THE CODE TO HANDLE THE SECOND INPUT FIELD.;;
;;;;;;;;;;;;;
```

INPUTCHR: DOSFXN INNOECHO ; GET AN INPUT CHARACTER INTO AL

```
CMP  AL,ESC          ; CHECK IF IT IS THE ESC KEY
JE   QUIT            ; IF SO, EXIT

CMP  AL,CR           ; CHECK IF IT IS THE ENTER KEY
JNE  NOTDONE         ; IF NOT, CONTINUE PROCESSING THE INPUT KEY

CMP  FIRSTINP,0      ; CHECK IF THE FIRST INPUT FIELD IS EMPTY
JE   BADKEY          ; IF SO, BEEP AT THE USER AND TRY FOR ANOTHER KEY

CMP  SECNDINP,0      ; CHECK IF THE SECOND INPUT FIELD IS EMPTY
JE   BADKEY          ; IF SO, BEEP AT THE USER AND TRY FOR ANOTHER KEY
```

QUIT: RET ; RETURN TO THE CALLER

```

NOTDONE:  CMP    DH,IN1ROW      ; CHECK IF THE CURSOR IS ON THE FIRST INPUT FIELD
          JE     ONINPUT1      ; IF NOT, GO TO THE SECTION FOR HANDLING THE 2ND
          JMP    ONINPUT2      ; INPUT FIELD

          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
          ;; THE CURSOR IS ON THE FIRST INPUT FIELD. CHECK TO MAKE SURE THE ;;
          ;; USER HAS ENTERED A VALID SELECTION, I.E. THE CHARACTER IS BETWEEN ;;
          ;; '1' AND '5'. IF THE CHARACTER IS OUT OF RANGE THEN GO TO THE ;;
          ;; CODE THAT BEEPS AT THE USER. OTHERWISE, SAVE THE CHARACTER IN ;;
          ;; FIRSTINP, CONVERT THE CHARACTER TO A SCAN CODE AND SAVE IT IN ;;
          ;; IN1SCNCD, AND DISPLAY THE CHARACTER ON THE MENU. THEN MOVE THE ;;
          ;; CURSOR TO THE SECOND INPUT FIELD AND LOOP BACK TO GET ANOTHER ;;
          ;; CHARACTER. ;;
          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

ONINPUT1: CMP    AL,'1'         ; CHECK IF THE CHARACTER IS BETWEEN '1' AND '5'
          JL     BADKEY        ; IF NOT, BEEP AT THE USER AND TRY FOR ANOTHER
          CMP    AL,'5'         ; CHARACTER
          JG     BADKEY

          MOV     FIRSTINP,AL    ; SAVE THE CHARACTER IN FIRSTINP

          MOV     BL,AL          ; USE THE BINARY VALUE OF THE ASCII CHARACTER
          SUB     BL,'1'-1      ; AS AN INDEX INTO THE ASCII TO SCAN CODE TABLE
          MOV     BH,0
          MOV     DL,ASCII2SCN[BX]
          MOV     IN1SCNCD,DL    ; SAVE THE SCAN CODE IN IN1SCNCD

          MOV     DL,AL          ; DISPLAY THE CHARACTER ON THE SCREEN
          DOSFXN  DISPCHAR

          MOV     DH,IN2ROW      ; MOVE THE CURSOR TO THE SECOND INPUT FIELD
          MOV     DL,IN2COL
          MOV     AH,SETCURS
          INT     10H

          JMP     INPUTCHR

          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
          ;; THIS CODE BEEPS AT THE USER TO INDICATED THAT AN INCORRECT KEY ;;
          ;; WAS PRESSED. TO BEEP, A CTRL-G (07H) IS SENT TO THE DISPLAY. ;;
          ;; THEN LOOP BACK TO GET ANOTHER CHARACTER. ;;
          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

BADKEY:   MOV     DL,BEEP        ; BEEP AT THE USER TO SIGNAL AN ERROR
          DOSFXN  DISPCHAR
          JMP     INPUTCHR      ; GET ANOTHER INPUT CHARACTER

          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
          ;; THE CURSOR IS ON THE SECOND INPUT FIELD. CONVERT THE CHARACTER ;;
          ;; IN AL TO UPPER CASE. CHECK TO MAKE SURE THE USER HAS ENTERED A ;;
          ;; VALID SELECTION, I.E. THE CHARACTER IS EITHER AN 'M' OR A 'Y'. ;;
          ;; IF THE CHARACTER IS OUT OF RANGE THEN GO TO THE CODE THAT BEEPS ;;
          ;; AT THE USER. OTHERWISE, SAVE THE CHARACTER IN SECNDINP AND DIS- ;;
          ;; PLAY THE CHARACTER ON THE MENU. THEN MOVE THE CURSOR TO THE ;;
          ;; FIRST INPUT FIELD AND LOOP BACK TO GET ANOTHER CHARACTER. ;;
          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

ONINPUT2: CMP    AL,' '         ; CHECK IF THE CHARACTER IS LOWER CASE
          JL     ISUPPER        ; IF NOT, SKIP CONVERTING TO UPPER CASE
          SUB     AL,' '-'A'     ; CONVERT TO UPPER CASE

```

Sample Program 5

```

ISUPPER:  CMP     AL,'M'           ; CHECK IF THE CHARACTER IS EITHER AN 'M' OR A 'Y'
          JE      OKKEY           ; IF NOT, BEEP AT THE USER AND TRY FOR ANOTHER
          CMP     AL,'Y'           ; CHARACTER
          JNE     BADKEY          ; GET ANOTHER INPUT CHARACTER

OKKEY:    MOV     SECNDINP,AL      ; SAVE THE CHARACTER IN SECNDINP

          MOV     DL,AL           ; DISPLAY THE CHARACTER ON THE SCREEN
          DOSFXN   DISPCHAR

          MOV     DH,IN1ROW       ; MOVE THE CURSOR TO THE FIRST INPUT FIELD
          MOV     DL,IN1COL
          MOV     AH,SETCURS
          INT     10H

          JMP     INPUTCHR        ; GET ANOTHER INPUT CHARACTER

GET$RESP  ENDP

```

```

;
; PROCEDURE : GRAFDATA
;

```

```

; CALLED BY : DISPDATA
;

```

```

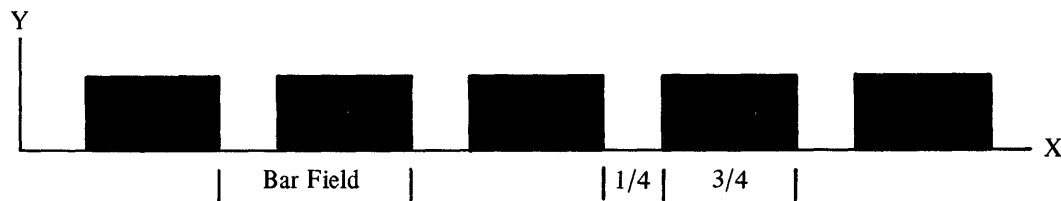
; FUNCTION :
;

```

```

; THIS PROCEDURE DISPLAYS THE BARCHART OF THE DATA OBTAINED FROM
; THE HOST. FIRST IT GETS THE SELECTED TIME SPAN. IF IT IS BY MONTH,
; THEN THERE ARE TWELVE BARS ON THE GRAPH. IF IT IS BY YEAR, THEN
; THERE ARE FIVE BARS ON THE CHART. THE X AXIS IS DIVIDED INTO THE
; EITHER FIVE OR TWELVE SECTIONS DEPENDING ON THE SELECTION. EACH OF
; THESE SECTIONS IS A BAR FIELD. THE BASE OF A BAR TAKES UP THE LATER
; 3/4 OF THE BAR FIELD. EXAMPLE:
;

```



```

; SI IS USED TO POINT TO THE STARTING POSITION OF EACH BAR. IT IS
; THE OFFSET IN PELS OF THE X POSITION OF THE LOWER LEFT CORNER OF THE
; BAR. SI STARTS BY POINTING 1/4 OF THE WAY INTO THE FIRST BAR FIELD.
; IT IS THEN INCREMENTED BY THE LENGTH OF THE BAR FIELD SO THAT IT
; POINTS 1/4 OF THE WAY INTO THE NEXT FIELD.
;

```

```

; DI IS USED TO POINT TO THE DATA OBTAINED FROM THE HOST. THE
; DATA FROM THE HOST IS 17 WORDS, 12 WORDS OF MONTH DATA FOLLOWED BY
; 5 WORDS OF YEAR DATA. DI IS SET TO POINT TO THE FIRST
; DATUM OF THE MONTH DATA IF THE USER SELECTED THE MONTH TIME
; SPAN OR TO THE FIRST
; DATUM OF THE YEAR DATA IF THE USER SELECTED THE YEAR TIME SPAN. DI
; IS THEN INCREMENTED BY 2 AS EACH DATUM IS GRAPHED.
;

```

```

; THE HEIGHT OF EACH BAR DEPENDS ON THE MAGNITUDE OF THE DATUM
; BEING GRAPHED RELATIVE TO THE MAXIMUM VALUE ON THE Y AXIS. TO CAL-
; CULATE THE NUMBER OF PELS NEEDED TO REPRESENT A DATUM THE FOLLOWING
; COMPUTATION IS USED:
;

```

$$\# \text{ PELS} = \frac{\text{DATUM VALUE}}{\text{MAXIMUM Y VALUE}} \times \text{NUMBER OF PELS IN THE Y AXIS}$$

$$\# \text{ PELS} = \left[\frac{(\text{DATUM VALUE} \times 65,536)}{\text{MAXIMUM Y VALUE}} \times \text{NUMBER OF PELS IN THE Y AXIS} \right] \times 65,536$$

THE MAXIMUM Y VALUE IS OBTAINED FROM A TABLE THAT CONTAINS THE MAXIMUM Y VALUE FOR EACH OF THE TEN GRAPHS. THE VALUES IN THE INPUT FIELDS ARE USED TO CALCULATE AN INDEX INTO THE TABLE. THE TABLE HAS THE MAXIMUM VALUES FOR THE MONTH OPTIONS 1, 2, 3, 4, AND 5 FOLLOWED BY THE MAXIMUM VALUES FOR THE YEAR OPTIONS 1, 2, 3, 4, AND 5.

GRAFDATA PROC NEAR

```
;; FIND WHICH TIME SPAN THE USER SELECTED. IF IT IS BY MONTH THEN  
;; SET THE NUMBER OF BARS (COUNT) TO 12 AND POINT DI TO THE START OF  
;; THE MONTH DATA. IF IT IS BY YEAR THEN SET THE NUMBER OF BARS TO  
;; 5 AND POINT DI TO THE START OF THE YEAR DATA.  
;
```

```

CMP      SECNDINP,'M'  ; CHECK IF 12 MONTH TIME SPAN WAS SELECTED
JE       BYMONTH

```

```

MOV     CX,5           ; IF 5 YEAR TIME SPAN, LOAD THE COUNT (CX)
                        ; WITH 5 TO DISPLAY 5 BARS
LEA     DI,BUFFAREA.YEARDATA
                        ; POINT DI TO THE BEGINNING OF THE YEAR DATA
JMP     FINDWIDT

```

```

BYMONTH:  MOV    CX,12          ; IF 12 MONTH TIME SPAN, LOAD THE COUNT (CX)
          ;          WITH 12 TO DISPLAY 12 BARS
          LEA     DI,BUFFAREA.MNTHDATA
          ;          POINT DI TO THE BEGINNING OF THE MONTH DATA

```

[illegible]

```

FINDWIDT:  MOV     COUNT,CX          ; STORE THE NUMBER OF BARS FOR DIVISION
           MOV     DX,0              ; CLEAR DX FOR DOUBLE WORD DIVISION
           MOV     AX,LENXAXIS       ; DIVIDE THE X AXIS INTO COUNT NUMBER OF
           DIV     COUNT              ; DIVISIONS.  EACH OF THESE DIVISIONS IS A
                                     ; BAR FIELD
           MOV     BFWIDTH,AX        ; STORE THE LENGTH OF THE BAR FIELD IN BFWIDTH
           MOV     BX,AX              ; BX = LENGTH OF THE BAR FIELD

```

Sample Program 5

```

SHR    AL,1          ; CALCULATE 1/4 OF THE BAR FIELD
SHR    AL,1          ; AX = 1/4 OF THE BAR FIELD

SUB     BX,AX         ; CALCULATE 3/4 OF BAR FIELD (BFWIDTH-1/4BFWIDTH)
MOV     BARWIDTH,BX   ; THIS IS THE WIDTH OF ONE BAR

MOV     SI,X$VERTEX   ; POINT SI TO THE START OF THE X AXIS
ADD     SI,AX         ; SI POINTS 1/4 OF THE WAY INTO FIRST BAR FIELD

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; GET THE MAXIMUM Y VALUE FOR THE SELECTED GRAPH. TO CONVERT THE ;;
;; INPUTS TO AN INDEX, FIRST FIND THE BINARY VALUE OF THE FIRST ;;
;; INPUT AND SUBTRACT 1. THIS IS DONE BY SUBTRACTING AN ASCII '1' ;;
;; FROM THE FIRST INPUT. THEN, IF THE SECOND INPUT IS A 'Y' ADD 5. ;;
;; USE THIS INDEX INTO THE TABLE TO OBTAIN THE MAXIMUM Y VALUE. ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

MOV     BH,0          ; GET THE USER'S CHOICES FROM THE MENU
MOV     BL,FIRSTINP    ; USE THESE AS AN INDEX INTO A TABLE THAT GIVES
SUB     BX,'1'         ; THE MAXIMUM Y VALUES
CMP     SECNDINP,'M'    ; 1M->1, 2M->2, ...,5M->5,1Y->6,2Y->7, ...,5Y->10
JE      DONTADD
DONTADD: ADD     BX,5
DONTADD: ADD     BX,BX   ; DOUBLE BX SINCE THE TABLE ENTRIES ARE WORDS
                        ; INDEX INTO THE TABLE TO GET THE OFFSET OF THE
                        ; PROPER TABLE AND...

MOV     AX,YMAXTAB[BX]
MOV     YMAX,AX        ; GET THE MAXIMUM Y VALUE FOR THIS GRAPH
                        ; SAVE IT IN YMAX

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; GRAPH EACH OF THE DATA. FIRST, STORE LENYAXIS IN MEMORY SO IT ;;
;; CAN BE USED FOR MULTIPLICATION. THEN, FOR EACH DATUM CALCULATE ;;
;; IT'S HEIGHT IN PELS AND DRAW A BOX WITH THAT HEIGHT AND A BASE OF ;;
;; BARWIDTH. SI IS INCREMENTED ALONG THE X AXIS TO POINT TO THE ;;
;; STARTING POINT OF EACH BOX. DI IS INCREMENTED THROUGH THE HOST ;;
;; DATA. CX COUNTS THE NUMBER OF BARS TO DISPLAY. ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

MOV     AX,LENYAXIS    ; STORE THE CONSTANT LENYAXIS IN MEMORY FOR
MOV     YAXISLEN,AX    ; THE UPCOMING MULTIPLIES

NEXTBAR: MOV     DX,[DI] ; PUT A HOST DATUM INTO DX (SHIFTED LEFT 1 WORD)
MOV     AX,0          ; CLEAR AX FOR DOUBLE WORD DIVISION
DIV     YMAX          ; CALCULATE THE NUMBER OF PELS NEEDED TO
MUL     YAXISLEN      ; REPRESENT THIS VALUE
MOV     HEIGHT,DX     ; SAVE DX (RESULT SHIFTED RIGHT 1 WORD) IN HEIGHT-
                        ; THE HEIGHT IN PELS OF THE BAR

PUSH    CX            ; SAVE THE BAR COUNT
DRAW$BOX SI,Y$VERTEX,HEIGHT,BARWIDTH
                        ; DRAW THE BOX - LOWER LEFT CORNER = (SI,Y$VERTEX)
                        ; HEIGHT = HEIGHT, WIDTH = BARWIDTH
POP     CX            ; RESTORE THE BAR COUNT

ADD     SI,BFWIDTH    ; POINT SI TO 1/4 OF THE WAY INTO NEXT BAR FIELD
ADD     DI,2          ; POINT DI TO THE NEXT WORD IN THE DATA
LOOP    NEXTBAR

RET

GRAFDATA  ENDP

```

```

;
;      PROCEDURE : CONNHOST
;      CALLED BY : GETDATA
;      FUNCTION :
;          THIS PROCEDURE DOES THE SET UP FOR GETTING THE DATA FROM THE
;          HOST.  FIRST IT CREATES A QUEUE FOR HOST INTERACTIVE COMMUNICATION
;          AND THEN CONNECTS TO THE HOST INTERACTIVE SERVICES.  NEXT IT CREATES
;          A QUEUE FOR THE HOST KEYSTROKES AND CONNECTS TO THE HOST KEYBOARD.
;          THEN IT DISABLES THE USER INPUT TO THE HOST KEYBOARD SO THAT THIS
;          PROGRAM'S INPUT TO THE HOST IS NOT DISRUPTED.
;
CONNHOST  PROC

          CRT$Q      SF$Q,,SF$Q$LEN,SF$Q$ID
                      ; CREATE A QUEUE FOR HOST INTERACTIVE
                      ;   COMMUNICATION
          CHEK4ERR

CONN$SF:  CONN$SF    MFIC,HOST$ID,SF$Q$ID,PCTASKID,PROGNAME
                      ; CONNECT FOR HOST INTERACTIVE SERVICES
          CHEK4ERR    CFRETNCD

          CRT$Q      HOSTQ,,HOSTQ$LEN,HOSTQ$ID
                      ; CREATE A QUEUE FOR HOST KEYSTROKES
          CHEK4ERR

          CONN$KEY    KEYBOARD,HOST$ID,HOSTQ$ID
                      ; CONNECT TO THE HOST KEYBOARD
          CHEK4ERR    CKRETNCD

          DISA$INP    KEYBOARD,HOST$ID
                      ; DISABLE USER INPUT TO THE HOST KEYBOARD
          CHEK4ERR    DIRETNCD

          RET

CONNHOST  ENDP

;
;      PROCEDURE : DISCHOST
;      CALLED BY : GETDATA
;      FUNCTION :
;          THIS PROCEDURE DOES THE CLEAN UP FROM GETTING THE DATA FROM THE
;          HOST.  FIRST IT REENABLES USER INPUT TO THE HOST KEYBOARD.  NEXT IT
;          DISCONNECTS FROM THE HOST KEYBOARD AND DELETES THE KEYSTROKE QUEUE.
;          THEN IT DISCONNECTS FROM HOST INTERACTIVE SERVICES AND DELETES THE
;          HOST INTERACTIVE COMMUNICATION QUEUE.
;
DISCHOST  PROC

          ENAB$INP    KEYBOARD,HOST$ID
                      ; ENABLE USER INPUT TO THE HOST KEYBOARD
          CHEK4ERR    EIRETNCD

          DISC$KEY    KEYBOARD,HOST$ID
                      ; DISCONNECT FROM THE HOST KEYBOARD
          CHEK4ERR    DKRETNCD

          DEL$ENT      HOSTQ$ID ; DELETE THE HOST KEYSTROKE QUEUE
          CHEK4ERR

```

Sample Program 5

```
DISC$SF MFIC,HOST$ID
; DISCONNECT FROM HOST INTERACTIVE SERVICES
CHEK4ERR DFRETNCD

DEL$ENT SF$Q$ID ; DELETE THE HOST INTERACTIVE COMMUNICATION Q
CHEK4ERR

RET

DISCHOST ENDP

;
;
; PROCEDURE : CHECKERR
; FUNCTION :
; THIS PROCEDURE IS PASSED TWO RETURN CODES IN CL AN BL. BL CON-
; TAINS A RETURN CODE FROM THE FIRST BYTE IN A PARAMETER LIST. BOTH
; CL AND BL ARE CHECKED FOR 0'S. IF EITHER CONTAINS A NON-ZERO RETURN
; CODE, AN ERROR MESSAGE IS DISPLAYED AND THE PROGRAM IS TERMINATED.
;
; NOTE: THIS IS A VERY SIMPLE ERROR HANDLER USED TO PRESERVE PROGRAM
; FLOW AND IS NOT LISTED AS AN EXAMPLE OF AN APPROPRIATE ERROR
; HANDLER. THIS ERROR HANDLER SIMPLY TERMINATES THE PROGRAM
; WHEN AN ERROR IS ENCOUNTERED LEAVING ANY RESOURCES, SUCH AS
; FIXED LENGTH QUEUES, PRESENTATION SPACES, AND A CONNECTION
; TO THE WINDOW SERVICES, STILL ALLOCATED. A MORE APPROPRIATE
; ERROR HANDLER WOULD DELETE ALL RESOURCES BEFORE TERMINATING.
;

CHECKERR PROC NEAR

CMP CL,0 ; CHECK THE RETURN CODE IN CL
JNE ERROR

CMP BL,0 ; CHECK THE PARAMETER LIST RETURN CODE
JNE ERROR

RET

ERROR: DISPLAY ERRMSG ; DISPLAY THE ERROR MESSAGE

INT 20H ; EXIT TO DOS

CHECKERR ENDP

CODESEG ENDS
END
```

Part 5. Appendixes

Part 5 contains additional information that deals with the Application Program Interface (API).

- Appendix A, “Scan-Code/Shift-State and ASCII/ASCII-Mnemonic Values,” describes the scan code/shift state values and the ASCII/ASCII mnemonics that can be sent to a session or intercepted from the keyboard for a particular session.
- Appendix B, “Destination/Origin Structured Fields,” describes the destination/origin structured field formats and protocol to use with the host interactive services.
- Appendix C, “Using Command Procedures for Save and Restore and for File Transfer,” describes ways to create programmed command procedures for using the Save, Restore, Send, and Receive commands.
- Appendix D, “Technical Notes,” contains technical information on the 3270 capabilities of the IBM 3270 Personal Computer and describes the 3270 data stream functions.
- Appendix E, “Problem Determination Procedures and Debugging Information,” describes problem determination procedures to use if a system error occurs during API activity in your application program, and describes some of the control blocks and data areas used by the workstation program that may assist you in debugging your application program or system extension.
- Appendix F, “Presentation Space Considerations,” describes presentation space considerations.

-
- Appendix G, “Calling Save, Restore, Send, and Receive from Your Application Program,” describes how to use DOS function calls to call Save, Restore, Send, and Receive from your application program.
 - Appendix H, “Return Codes,” describes the return codes that can be received while the workstation program or application programs that use the API services are running.
 - Appendix I, “Outbound Data Stream Preprocessor (ODSP) Option,” describes how to preprocess outbound data streams using ODSP.
-

Appendix A. Scan-Code/Shift-State and ASCII/ASCII-Mnemonic Values

Introduction	A-2
Scan-Code/Shift-State Values	A-2
Scan Code	A-2
Special Scan Codes	A-3
Shift State	A-4
ASCII/ASCII Mnemonics	A-4
Default Scan Codes for the IBM 3270 PC Keyboard (PC Mode)	A-5
Default Scan Codes for the IBM 3270 PC Keyboard (MFI Mode)	A-9
Default Scan Codes for the IBM Enhanced PC Keyboard (PC Mode)	A-13
Default Scan Codes for the IBM Enhanced PC Keyboard (MFI Mode)	A-16
Default Scan Codes for the PC XT Keyboard (PC Mode)	A-19
Default Scan Codes for the IBM PC XT Keyboard (MFI Mode)	A-22
Default Scan Codes for the IBM Personal Computer AT Keyboard (PC Mode)	A-25
Default Scan Codes for the IBM Personal Computer AT Keyboard (MFI Mode)	A-28
ASCII Characters Common to All Countries	A-31
ASCII Mnemonics Common to All Countries	A-34
Additional ASCII Characters Used by U.S. English	A-37

Introduction

Keystrokes sent to a session or read from the keyboard for a particular session via the Read and Write API are in the form of:

- Scan-code/shift-state values or
- ASCII/ASCII-mnemonic values.

If you are using the Keyboard Services API, then applications using the API must be well-behaved. If you are using the API to another PC session, then that PC session must also be well-behaved.

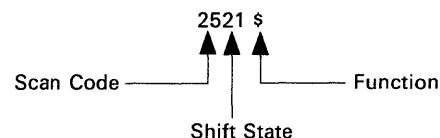
Scan-Code/Shift-State Values

Keystrokes sent in the form of scan code/shift state are represented by a 4-byte value. Byte 0 is the scan code, and byte 1 is the shift state. Bytes 2 and 3 are normally '01' and '00', respectively; these two bytes are not needed for the Write Keystroke service requests.

Scan Code

The scan code is a unique 1-byte hexadecimal value that is assigned to each key position on the IBM 3270 PC Keyboard, the Enhanced PC Keyboard, the PC XT Keyboard, and the Personal Computer AT Keyboard. Foldouts at the back of this book show the key position number and scan code for each key position on the keyboards.

Figures A-1 and A-2 list the default scan code values for each key position for the PC and MFI modes, respectively, of the IBM 3270 PC Keyboard. Figures A-3 and A-4 list the default value for each key position for the PC and MFI modes, respectively, of the IBM Enhanced PC Keyboard. Figures A-5 and A-6 list the default value for each key position for the PC and MFI modes, respectively, of the PC XT Keyboard. Figures A-7 and A-8 list the default value for each key position for the PC and MFI modes, respectively, of the Personal Computer AT Keyboard. The first byte of the 2-byte keystroke value represents the default scan code; the second byte, the shift state; and the function (if there is one) is listed last. In Figure A-4 for example, the uppercase keystroke value for key 5 is:



Note: The tables indicate the default values you will receive when reading the keyboard as well as the values you can send to a session. If you have modified the keyboard with the Keyboard Definition Utility, then the values you receive when reading the keyboard will differ.

Special Scan Codes

The workstation program uses some special scan codes not listed in Figure A-1 on page A-5. They are:

- X'7F', which is used by the workstation program to tell keystroke applications that the shift state may have changed since the last key was sent. Your application should adjust the shift state to match the one accompanying the X'7F' scan code if it is concerned with the shift state.
- X'F0', which indicates that a key is breaking (being released), and the next keystroke to be sent is the key being released.
- X'00', which indicates that the application or keyboard did not handle a series of rapid keystrokes, and some keystrokes were lost.

In all sessions except for personal computer sessions, all keys are “make only,” with the exception of the following keys, which are “make/break”:

- The Upshift keys (left and right)
- The Alt key
- The Ctrl key
- The Caps Lock key.

When in the personal computer session, the keyboard is reprogrammed to make all keys that are sent to the personal computer session “make/break” as well as typematic. Therefore, an application program that uses the Read Input service to obtain keystrokes destined for a personal computer session should expect to receive three separate scan codes for each key that is pressed. The scan codes generated each time a key is pressed and released are as follows:

1. When a key is pressed, the scan code for that key is generated. This scan code is continuously generated until the key breaks (is released).
2. When the key breaks (is released), the X'F0' scan code is generated.
3. Finally, the scan code of the key that was pressed is generated again to indicate that this key is breaking.

In all other sessions, only one scan code is received for most keys. The “make/break” sequence of three scan codes only appears for the keys listed as “make/break” keys above.

The Shift State

Shift State

The shift state is a 1-byte value that indicates which of the functions or characters printed on the keytop of a given position is being sent. The following shows the format of the shift state byte:

0, 1	2	3	4	5	6	7
Reserved	Right shift	Left shift	Control key	Alt keys	Caps Lock	Upshift keys

Bits 0 and 1 are reserved.

Bit 2 represents the right upshift key.

Bit 3 represents the left upshift key.

Bit 4 represents the Control shift state.

Bit 5 represents the Alt shift state.

Bit 6 represents the Caps Lock state.

Bit 7 represents the upshift state. Bit 7 indicates that one of the two upshift keys was pressed. If your application program must distinguish between the right upshift key and the left upshift key, use bits 2 and 3.

Lower shift is represented by a value of X'00'.

Note: When sending keystrokes to another session, only bits 4 through 7 control the shift state. Bits 2 and 3 are used by the PC sessions and PC application programs to determine which of the two shift keys caused the upshift state condition represented by bit 7.

ASCII/ASCII Mnemonics

The ASCII/ASCII mnemonic is the 1- to 6-byte value representing the functions on the keyboard. Figures A-9 and A-10 list the ASCII/ASCII mnemonics that are common for all countries. Figure A-11 lists the additional ASCII/ASCII mnemonics that can be used by U.S. English.

The tables give the ASCII code (both decimal and hexadecimal) and the ASCII mnemonic and specify the functions they perform in PC, Host/Notepad, and Work Station Control sessions.

Default Scan Codes for the IBM 3270 PC Keyboard (PC Mode)

Key	Lowercase	Uppercase	Alternate Case
1	0E00 `	0E21 ~	0E04
2	1600 1	1621 !	1604
3	1E00 2	1E21 @	1E04
4	2600 3	2621 #	2604
5	2500 4	2521 \$	2504
6	2E00 5	2E21 %	2E04
7	3600 6	3621 ^	3604
8	3D00 7	3D21 &	3D04
9	3E00 8	3E21 *	3E04
10	4600 9	4621 (4604
11	4500 0	4521)	4504
12	4E00 -	4E21 _	4E04
13	5500 =	5521 +	5504
15	6600 Backspace	6621 Backspace	6604
16	0D00 Right Tab	0D21 Left Tab	0D04
17	1500 q	1521 Q	1504
18	1D00 w	1D21 W	1D04
19	2400 e	2421 E	2404
20	2D00 r	2D21 R	2D04
21	2C00 t	2C21 T	2C04
22	3500 y	3521 Y	3504
23	3C00 u	3C21 U	3C04
24	4300 i	4321 I	4304
25	4400 o	4421 O	4404
26	4D00 p	4D21 P	4D04
27	5400 [5421 {	5404
28	5B00 \	5B21	5B04
30	1400 Caps Lock	1421 Caps Lock	1404 Caps Lock
31	1C00 a	1C21 A	1C04
32	1B00 s	1B21 S	1B04
33	2300 d	2321 D	2304
34	2B00 f	2B21 F	2B04
35	3400 g	3421 G	3404

Figure A-1 (Part 1 of 4). Default Scan Codes for IBM 3270 PC Keyboard (PC Mode)

Scan Code Tables

Key	Lowercase	Uppercase	Alternate Case
36	3300 h	3321 H	3304
37	3B00 j	3B21 J	3B04
38	4200 k	4221 K	4204
39	4B00 l	4B21 L	4B04
40	4C00 ;	4C21 :	4C04
41	5200 '	5221 "	5204
42	5300]	5321 }	5304
43	5A00 Enter	5A21 Enter	5A04
44	1200 Left Shift	1221 Left Shift	1204 Left Shift
45	1300 <	1321 >	1304
46	1A00 z	1A21 Z	1A04
47	2200 x	2221 X	2204
48	2100 c	2121 C	2104
49	2A00 v	2A21 V	2A04
50	3200 b	3221 B	3204
51	3100 n	3121 N	3104
52	3A00 m	3A21 M	3A04
53	4100 ,	4121 <	4104
54	4900 .	4921 >	4904
55	4A00 /	4A21 ?	4A04
57	5900 Right Shift	5921 Right Shift	5904 Right Shift
58	1100	1121	1104 Quit
60	1900 Left Alt	1921 Left Alt	1904 Left Alt
61	2900 Spacebar	2921 Spacebar	2904 Spacebar
62	3900 Right Alt	3921 Right Alt	3904 Right Alt
64	5800 Enter	5821 Enter	5804
65	0600	0621	0604 Test
66	0C00 Finish	0C21 Finish	0C04
67	0B00	0B21	0B04
68	0A00	0A21	0A04 Pause
69	0900 Ctrl	0921 Ctrl	0904 Ctrl
70	0500	0521	0504 SysRq
71	0400 WSCtrl	0421 WSCtrl	0404
72	0300 Jump	0321 ChgScr	0304
73	8300 *	8321 Print	8304
74	0100 Enlarge	0121 Enlarge	0104
75	6700	6721	6704

Figure A-1 (Part 2 of 4). Default Scan Codes for IBM 3270 PC Keyboard (PC Mode)

Key	Lowercase	Uppercase	Alternate Case
76	6400	6421	6404
78	6100 Left Cursor	6121 Left Cursor	6104
80	6E00	6E21	6E04
81	6500 Insert	6521 Insert	6504
82	6300 Up Cursor	6321 Up Cursor	6304
83	6200	6221	6204
84	6000 Dn Cursor	6021 Dn Cursor	6004
85	6F00	6F21	6F04
86	6D00 Delete	6D21 Delete	6D04
88	6A00 Right Cursor	6A21 Right Cursor	6A04
90	7600 Esc	7721 Esc	7704
91	6C00 Home	6C21 7	6C04
92	6B00 Left Cursor	6B21 4	6B04
93	6900 End	6921 1	6904
95	7700 NumLk	7721 NumLk	7704
96	7500 Up Cursor	7521 8	7504
97	7300	7321 5	7304
98	7200 Down Cursor	7221 2	7204
99	7000 Insert	7021 0	7004
100	7E00 ScrLk	7E21 ScrLk	7E04
101	7D00 PgUp	7D21 9	7D04
102	7400 Right Cursor	7421 6	7404
103	7A00 PgDn	7A21 3	7A04
104	7100 Delete	7121 .	7104
105	8400 Spacebar	8421 Spacebar	8404 Spacebar
106	7C00 Right Tab	7C21	7C04
107	7B00 -	7B21 -	7B04
108	7900 +	7921 +	7904
110	0700 F1	0721	0704
111	0F00 F2	0F21	0F04
112	1700 F3	1721	1704
113	1F00 F4	1F21	1F04
114	2700 F5	2721	2704
115	2F00 F6	2F21	2F04
116	3700 F7	3721	3704
117	3F00 F8	3F21	3F04
118	4700 F9	4721	4704

Figure A-1 (Part 3 of 4). Default Scan Codes for IBM 3270 PC Keyboard (PC Mode)

Scan Code Tables

Key	Lowercase	Uppercase	Alternate Case
119	4F00 F10	4F21	4F04
120	5600	5621	5604
121	5E00	5E21	5E04
122	0800	0821	0804
123	1000	1021	1004
124	1800	1821	1804
125	2000	2021	2004
126	2800	2821	2804
127	3000	3021	3004
128	3800	3821	3804
129	4000	4021	4004
130	4800	4821	4804
131	5000	5021	5004
132	5700	5721	5704 Alt Cr
133	5F00	5F21	5F04
*This key position is not used.			

Figure A-1 (Part 4 of 4). Default Scan Codes for IBM 3270 PC Keyboard (PC Mode)

Default Scan Codes for the IBM 3270 PC Keyboard (MFI Mode)

Note: MFI = host/notepad.

Key	Lowercase	Uppercase	Alternate Case
1	0E00 `	0E21 ~	0E04
2	1600 1	1621	1604
3	1E00 2	1E21 @	1E04
4	2600 3	2621 #	2604
5	2500 4	2521 \$	2504
6	2E00 5	2E21 %	2E04
7	3600 6	3621 ▯	3604
8	3D00 7	3D21 &	3D04
9	3E00 8	3E21 *	3E04
10	4600 9	4621 (4604
11	4500 0	4521)	4504
12	4E00 -	4E21 _	4E04
13	5500 =	5521 +	5504
15	6600 Backspace	6621 Backspace	6604
16	0D00 Right Tab	0D21 Right Tab	0D04
17	1500 q	1521 Q	1504
18	1D00 w	1D21 W	1D04
19	2400 e	2421 E	2404
20	2D00 r	2D21 R	2D04
21	2C00 t	2C21 T	2C04
22	3500 y	3521 Y	3504
23	3C00 u	3C21 U	3C04
24	4300 i	4321 I	4304
25	4400 o	4421 O	4404
26	4D00 p	4D21 P	4D04
27	5400 ¢	5421 !	5404
28	5B00 \	5B21 !	5B04
30	1400 Shift Lock	1421 Shift Lock	1404 Shift Lock
31	1C00 a	1C21 A	1C04
32	1B00 s	1B21 S	1B04
33	2300 d	2321 D	2304
34	2B00 f	2B21 F	2B04
35	3400 g	3421 G	3404

Figure A-2 (Part 1 of 4). Default Scan Codes for IBM 3270 PC Keyboard (MFI Mode)

Scan Code Tables

Key	Lowercase	Uppercase	Alternate Case
36	3300 h	3321 H	3304
37	3B00 j	3B21 J	3B04
38	4200 k	4221 K	4204
39	4B00 l	4B21 L	4B04
40	4C00 ;	4C21 :	4C04
41	5200 '	5221 "	5204
42	5300 {	5321 }	5304
43	5A00 Newline	5A21 Newline	5A04
44	1200 Left Shift	1221 Left Shift	1204 Left Shift
45	1300 <	1321 >	1304
46	1A00 z	1A21 Z	1A04
47	2200 x	2221 X	2204
48	2100 c	2121 C	2104
49	2A00 v	2A21 V	2A04
50	3200 b	3221 B	3204
51	3100 n	3121 N	3104
52	3A00 m	3A21 M	3A04
53	4100 ,	4121 ,	4104
54	4900 .	4921 .	4904
55	4A00 /	4A21 ?	4A04
57	5900 Right Shift	5921 Right Shift	5904 Right Shift
58	1100 Reset	1121 Reset	1104 Quit
60	1900 Left Alt	1921 Left Alt	1904 Left Alt
61	2900 Spacebar	2921 Spacebar	2904 Spacebar
62	3900 Right Alt	3921 Right Alt	3904 Right Alt
64	5800 Enter	5821 Enter	5804
65	0600 Clear	0621 Clear	0604 Test
66	0C00 Finish	0C21 Finish	0C04 Attn
67	0B00 Erase EOF	0B21 Erase EOF	0B04 ErInp
68	0A00 AUTO	0A21 COPY	0A04 Pause
69	0900 Ctrl	0921 Ctrl	0904 Ctrl
70	0500 Help	0521 Help	0504 Sys Req
71	0400 WSCtrl	0421 WSCtrl	0404 ExSel
72	0300 Jump	0321 ChgScr	0304 CrSel
73	8300 Print	8321 Print	8304 Ident
74	0100 Enlarge	0121 Enlarge	0104 Window Delete

Figure A-2 (Part 2 of 4). Default Scan Codes for IBM 3270 PC Keyboard (MFI Mode)

Key	Lowercase	Uppercase	Alternate Case
75	6700 PA1	6721 Dup	6704
76	6400 Left Tab	6421 Left Tab	6404
78	6100 Left Cursor	6121 Left Cursor	6104 Fast Left Cursor
80	6E00 PA2	6E21 Field Mark	6E04
81	6500 Insert	6521 Insert	6504
82	6300 Up Cursor	6321 Up Cursor	6304
83	6200	6221	6204 Home
84	6000 Down Cursor	6021 Down Cursor	6004
85	6F00 PA3	6F21	6F04
86	6D00 Delete	6D21 Delete	6D04
88	6A00 Right Cursor	6A21 Right Cursor	6A04 Fast Right Cursor
90	7600	7721	7704
91	6C00 7	6C21 7	6C04
92	6B00 4	6B21 4	6B04
93	6900 1	6921 1	6904
95	7700 NumLk	7721 NumLk	7704 NumLk
96	7500 8	7521 8	7504
97	7300 5	7321 5	7304
98	7200 2	7221 2	7204
99	7000 0	7021 0	7004
100	7E00 ,	7E21 ,	7E04
101	7D00 9	7D21 9	7D04
102	7400 6	7421 6	7404
103	7A00 3	7A21 3	7A04
104	7100 .	7121 .	7104
105	8400 Spacebar	8421 Spacebar	8404 Space
106	7C00 Right Tab	7C21 Right Tab	7C04
107	7B00 -	7B21 -	7B04
108	7900 Enter	7921 Enter	7904
110	0700 PF1	0721 PF1	0704 PSA
111	0F00 PF2	0F21 PF2	0F04 PSB
112	1700 PF3	1721 PF3	1704 PSC
113	1F00 PF4	1F21 PF4	1F04 PSD
114	2700 PF5	2721 PF5	2704 PSE
115	2F00 PF6	2F21 PF6	2F04 PSF
116	3700 PF7	3721 PF7	3704 PS Reset

Figure A-2 (Part 3 of 4). Default Scan Codes for IBM 3270 PC Keyboard (MFI Mode)

Scan Code Tables

Key	Lowercase	Uppercase	Alternate Case
117	3F00 PF8	3F21 PF8	3F04
118	4700 PF9	4721 PF9	4704 Reverse Video
119	4F00 PF10	4F21 PF10	4F04 Blink
120	5600 PF11	5621 PF11	5604 Underscore
121	5E00 PF12	5E21 PF12	5E04 Attribute Reset
122	0800 PF13	0821 PF13	0804 Red
123	1000 PF14	1021 PF14	1004 Pink
124	1800 PF15	1821 PF15	1804 Green
125	2000 PF16	2021 PF16	2004 Yellow
126	2800 PF17	2821 PF17	2804 Blue
127	3000 PF18	3021 PF18	3004 Turquoise
128	3800 PF19	3821 PF19	3804 White
129	4000 PF20	4021 PF20	4004 Black
130	4800 PF21	4821 PF21	4804 Color Reset
131	5000 PF22	5021 PF22	5004
132	5700 PF23	5721 PF23	5704 Alt Cr
133	5F00 PF24	5F21 PF24	5F04
*This key position is not used.			

Figure A-2 (Part 4 of 4). Default Scan Codes for IBM 3270 PC Keyboard (MFI Mode)

Default Scan Codes for the IBM Enhanced PC Keyboard (PC Mode)

Key	Lowercase	Uppercase	Alternate Case
1	0E00 `	0E21 ~	0E04
2	1600 1	1621 !	1604
3	1E00 2	1E21 @	1E04
4	2600 3	2621 #	2604
5	2500 4	2521 \$	2504
6	2E00 5	2E21 %	2E04
7	3600 6	3621 ^	3604
8	3D00 7	3D21 &	3D04
9	3E00 8	3E21 *	3E04
10	4600 9	4621 (4604
11	4500 0	4521)	4504
12	4E00 -	4E21 _	4E04
13	5500 =	5521 +	5504
15	6600 Backspace	6621 Backspace	6604
16	0D00 Right Tab	0D21 Left Tab	0D04
17	1500 q	1521 Q	1504
18	1D00 w	1D21 W	1D04
19	2400 e	2421 E	2404
20	2D00 r	2D21 R	2D04
21	2C00 t	2C21 T	2C04
22	3500 y	3521 Y	3504
23	3C00 u	3C21 U	3C04
24	4300 i	4321 I	4304
25	4400 o	4421 O	4404
26	4D00 p	4D21 P	4D04
27	5400 [5421 {	5404
28	5300]	5321 }	5304
29	5B00 \	5B21	5B04
30	1400 Caps Lock	1421 Caps Lock	1404 Caps Lock
31	1C00 a	1C21 A	1C04
32	1B00 s	1B21 S	1B04
33	2300 d	2321 D	2304
34	2B00 f	2B21 F	2B04

Figure A-3 (Part 1 of 3). Default Scan Codes for IBM Enhanced PC Keyboard (PC Mode)

Scan Code Tables

Key	Lowercase	Uppercase	Alternate Case
35	3400 g	3421 G	3404
36	3300 h	3321 H	3304
37	3B00 j	3B21 J	3B04
38	4200 k	4221 K	4204
39	4B00 l	4B21 L	4B04
40	4C00 ;	4C21 :	4C04
41	5200 '	5221 "	5204
43	5A00 Enter	5A21 Enter	5A04
44	1200 Left Shift	1221 Left Shift	1204 Left Shift
46	1A00 z	1A21 Z	1A04
47	2200 x	2221 X	2204
48	2100 c	2121 C	2104
49	2A00 v	2A21 V	2A04
50	3200 b	3221 B	3204
51	3100 n	3121 N	3104
52	3A00 m	3A21 M	3A04
53	4100 ,	4121 <	4104
54	4900 .	4921 >	4904
55	4A00 /	4A21 ?	4A04
57	5900 Right Shift	5921 Right Shift	5904 Right Shift
58	0900 Left Ctrl	0921 Left Ctrl	0904 Left Ctrl
60	1900 Left Alt	1921 Left Alt	1904
61	2900 Spacebar	2921 Spacebar	2904 Spacebar
62	3900 Right Alt	3921 Right Alt	1104 Quit
64	0200 Right Ctrl	0221 Right Ctrl	0204 Right Ctrl
75	6500 Insert	6521 Insert	0100 Enlarge
76	6D00 Delete	6D21 Delete	6D04
79	6100 Left Cursor	6121 Left Cursor	6104
80	A500 Home	A521 Home	0321 ChgSc
81	A600 End	0C00 Finish	A604
83	6300 Up Cursor	6321 Up Cursor	6304
84	6000 Dn Cursor	6021 Dn Cursor	6004
85	A700 Page Up	A721 Page Up	0300 Jump
86	A800 Page Down	A821 Page Down	A804
89	6A00 Right Cursor	6A21 Right Cursor	6A04
90	7700 NumLock	7721 NumLock	7704 NumLock
91	6C00 Home	6C21 7	6C04

Figure A-3 (Part 2 of 3). Default Scan Codes for IBM Enhanced PC Keyboard (PC Mode)

Key	Lowercase	Uppercase	Alternate Case
92	6B00 Left Cursor	6B21 4	6B04
93	6900 End	6921 1	6904
95	5D00 /	5D21 /	5D04
96	7500 Up Cursor	7521 8	7504
97	7300	7321 5	7304
98	7200 Dn Cursor	7221 2	7204
99	7000 Ins	7021 0	7004
100	6800 *	6821 *	6804
101	7D00 PgUp	7D21 9	7D04
102	7400 Right Cursor	7421 6	7404
103	7A00 PgDn	7A21 3	7A04
104	7100 Del	7121 .	7104
105	7B00 -	7B21 -	7B04
106	7900 +	7921 +	7904
108	7800 Enter	7821 Enter	7804
110	7600 Esc	0400 WSCtrl	7604
112	0700 F1	0721	0704
113	0F00 F2	0F21	0F04
114	1700 F3	1721	1704
115	1F00 F4	1F21	1F04
116	2700 F5	2721	2704
117	2F00 F6	2F21	2F04
118	3700 F7	3721	3704
119	3F00 F8	3F21	3F04
120	4700 F9	4721	4704
121	4F00 F10	4F21	4704
122	5600 F11	5621	5604
123	5E00 F12	5E21	5E04
124	5100 Print Screen	5121 Print Screen	5104 SysRq
125	7E00 Scroll Lock	7E21 Scroll Lock	0604 Test
126	8000 Pause (PC)	0A04 Pause (Autokey)	8004
<i>Note: Key 124 in the control state toggles Screen Echo. Key 126 in the control state is Break.*</i>			

Figure A-3 (Part 3 of 3). Default Scan Codes for IBM Enhanced PC Keyboard (PC Mode)

Default Scan Codes for the IBM Enhanced PC Keyboard (MFI Mode)

Key	Lowercase	Uppercase	Alternate Case
1	0E00 `	0E21 ~	0E04
2	1600 1	5421 !	1604
3	1E00 2	1E21 @	1E04
4	2600 3	2621 #	2604
5	2500 4	2521 \$	2504
6	2E00 5	2E21 %	2E04
7	3600 6	5400 ¢	3604
8	3D00 7	3D21 &	3D04
9	3E00 8	3E21 *	3E04
10	4600 9	4621 (4604
11	4500 0	4521)	4504
12	4E00 -	4E21 _	4E04
13	5500 =	5521 +	5504
15	6600 Backspace	6621 Backspace	6604 Backspace
16	0D00 Right Tab	6400 Left Tab	0D04
17	1500 q	1521 Q	1504
18	1D00 w	1D21 W	1D04
19	2400 e	2421 E	2404
20	2D00 r	2D21 R	2D04
21	2C00 t	2C21 T	2C04
22	3500 y	3521 Y	3504
23	3C00 u	3C21 U	3C04
24	4300 i	4321 I	4304
25	4400 o	4421 O	4404
26	4D00 p	4D21 P	4D04
27	3621 ⌏	5300 {	5304
28	1621	5321 }	5304
29	5B00 \	5B21 !	5B04
30	1400 Shift Lock	1421 Shift Lock	1404 Shift Lock
31	1C00 a	1C21 A	1C04
32	1B00 s	1B21 S	1B04
33	2300 d	2321 D	2304
34	2B00 f	2B21 F	2B04

Figure A-4 (Part 1 of 3). Default Scan Codes for IBM Enhanced PC Keyboard (MFI Mode)

Key	Lowercase	Uppercase	Alternate Case
35	3400 g	3421 G	3404
36	3300 h	3321 H	3304
37	3B00 j	3B21 J	3B04
38	4200 k	4221 K	4204
39	4B00 l	4B21 L	4B04
40	4C00 ;	4C21 :	4C04
41	5200 '	5221 "	5204
43	5A00 New Line	5A21 New Line	5A04
44	1200 Left Shift	1221 Left Shift	1204 Left Shift
46	1A00 z	1A21 Z	1A04
47	2200 x	2221 X	2204
48	2100 c	2121 C	2104
49	2A00 v	2A21 V	2A04
50	3200 b	3221 B	3204
51	3100 n	3121 N	3104
52	3A00 m	3A21 M	3A04
53	4100 ,	1300 <	4104
54	4900 .	1321 >	4904
55	4A00 /	4A21 ?	4A04
57	5900 Right Shift	5921 Right Shift	5904 Right Shift
58	0900 Ctrl	0921 Ctrl	0904 Ctrl
60	1900 Alt	1921 Alt	1904
61	2900 Spacebar	2921 Spacebar	2904 Spacebar
62	1100 Reset	1121 Reset	1104 Quit
64	5800 Enter	5821 Enter	5804
75	6500 Insert	6721 Dup	0100 Enlarge
76	6D00 Delete	6D21 Delete	6D04 Word Del
79	6100 Cursor Left	6121 Cursor Left	6104 Fast Cursor Left
80	6204 Home	6E21 FldMk	0321 ChgSc
81	0B00 ErEOF	0C00 Finish	0B04 ErInp
83	6300 Cursor Up	6321 Cursor Up	6304
84	6000 Cursor Dn	6021 Cursor Dn	6004
85	6700 PA1	6F00 PA3	0300 Jump
86	6E00 PA2	DE00 NF	DE00 NF
89	6A00 Cursor Right	6A21 Cursor Right	6A04 Fast Cursor Right
90	0700 PF1	7721 NumLk	0704 PSA

Figure A-4 (Part 2 of 3). Default Scan Codes for IBM Enhanced PC Keyboard (MFI Mode)

Scan Code Tables

Key	Lowercase	Uppercase	Alternate Case
91	1F00 PF4	3D00 7	0F04 PSB
92	3700 PF7	2500 4	1704 PSC
93	4F00 PF10	1600 1	1F04 PSD
95	0F00 PF2	DE00 NF	2704 PSE
96	2700 PF5	3E00 8	2F04 PSF
97	3F00 PF8	2E00 5	3704 ➤
98	5600 PF11	1E00 2	DE00 NF
99	6500 Ins	4500 0	DE00 NF
100	1700 PF3	4121 ,	4704 ☐
101	2F00 PF6	4600 9	4F04 :a:
102	4700 PF9	3600 6	5604 a
103	5E00 PF12	2600 3	5E04 ➤
104	6D00 Del	4921 .	7104 Del
105	7B00 -	7B21 -	7B04
106	7C00 Right Tab	7C21 Right Tab	7C04
108	7900 Enter	7921 Enter	7904
110	0C04 Attn	0400 WSCtrl	0404 ExSel
112	0700 PF1	0800 PF13	0804
113	0F00 PF2	1000 PF14	1004
114	1700 PF3	1800 PF15	1804
115	1F00 PF4	2000 PF16	DE00 NF
116	2700 PF5	2800 PF17	0A00
117	2F00 PF6	3000 PF18	0A21
118	3700 PF7	3800 PF19	0104
119	3F00 PF8	4000 PF20	DE00 NF
120	4700 PF9	4800 PF21	0304
121	4F00 PF10	5000 PF22	5004
122	5600 PF11	5700 PF23	5704
123	5E00 PF12	5F00 PF24	DE00 NF
124	8300 Print	8304 Indent	0504
125	0500 Help	0521 Help	0604 Test
126	0600 Clear	0A04 Pause	DE00 NF
<i>Note: NF (no function) means no function has been assigned to the specified key location.</i>			

Figure A-4 (Part 3 of 3). Default Scan Codes for IBM Enhanced PC Keyboard (MFI Mode)

Default Scan Codes for the PC XT Keyboard (PC Mode)

Key	Lowercase	Uppercase	Alternate Case
1	7600 Escape	0400 WS Ctrl	0300 Jump
2	1600 1	1621 !	1604
3	1E00 2	1E21 @	1E04
4	2600 3	2621 #	2604
5	2500 4	2521 \$	2504
6	2E00 5	2E21 %	2E04
7	3600 6	3621 ^	3604
8	3D00 7	3D21 &	3D04
9	3E00 8	3E21 *	3E04
10	4600 9	4621 (4604
11	4500 0	4521)	4504
12	4E00 -	4E21 _	4E04
13	5500 =	5521 +	5504
14	6600 Backspace	6621 Backspace	6604
15	0D00 Right tab	0D21 Left tab	0D04
16	1500 q	1521 Q	1504
17	1D00 w	1D21 W	1D04
18	2400 e	2421 E	2404
19	2D00 r	2D21 R	2D04
20	2C00 t	2C21 T	2C04
21	3500 y	3521 Y	3504
22	3C00 u	3C21 U	3C04
23	4300 i	4321 I	4304
24	4400 o	4421 O	4404
25	4D00 p	4D21 P	4D04
26	5400 [5421 {	5404
27	5300]	5321 }	5304
28	5A00 Enter	5A21 Enter	5A04
29	0900 Ctrl	0921 Ctrl	0904 Ctrl
30	1C00 a	1C21 A	1C04
31	1B00 s	1B21 S	1B04
32	2300 d	2321 D	2304
33	2B00 f	2B21 F	2B04
34	3400 g	3421 G	3404
35	3300 h	3321 H	3304

Figure A-5 (Part 1 of 3). Default Scan Codes for IBM PC XT Keyboard (PC Mode)

Scan Code Tables

Key	Lowercase	Uppercase	Alternate Case
36	3B00 j	3B21 J	3B04
37	4200 k	4221 K	4204
38	4B00 l	4B21 L	4B04
39	4C00 ;	4C21 :	4C04
40	5200 '	5221 "	5204
41	0E00 `	0E21 ~	0E04
42	1200 Left Shift	1221 Left Shift	1204 Left Shift
43	5B00 \	5B21 !	5B04
44	1A00 z	1A21 Z	1A04
45	2200 x	2221 X	2204
46	2100 c	2121 C	2104
47	2A00 v	2A21 V	2A04
48	3200 b	3221 B	3204
49	3100 n	3121 N	3104
50	3A00 m	3A21 M	3A04
51	4100 ,	4121 <	4104
52	4900 .	4921 >	4904
53	4A00 /	4A21 ?	4A04
54	5900 Right Shift	5921 Right Shift	5904 Right Shift
55	8300 *	8321 Prnt Scr	8304
56	1900 Alt	1921 Alt	1904 Alt
57	2900 Spacebar	2921 Spacebar	2904 Spacebar
58	1400 Caps lock	1421 Caps lock	1404 Caps lock
59	0700 F1	0721	0704
60	0F00 F2	0F21	0F04
61	1700 F3	1721	1704
62	1F00 F4	1F21	1F04
63	2700 F5	2721	2704
64	2F00 F6	2F21	2F04
65	3700 F7	3721	3704
66	3F00 F8	3F21	3F04
67	4700 F9	4721	4704
68	4F00 F10	4F21	4F04
69	7700 Num Lock	7721 Num Lock	0100 Enlarge
70	7E00 Scr Lk	1104 Quit	0321 Chg Scr
71	6C00 Home	6C21 Keypad 7	6C04
72	7500 Up Cursor	7521 Keypad 8	7504

Figure A-5 (Part 2 of 3). Default Scan Codes for IBM PC XT Keyboard (PC Mode)

Key	Lowercase	Uppercase	Alternate Case
73	7D00 Page Up	7D21 Keypad 9	7D04
74	7B00 -	7B21 -	7B04
75	6B00 Left Cursor	6B21 Keypad 4	6B04
76	7300	7321 Keypad 5	7304
77	7400 Right Cursor	7421 Keypad 6	7404
78	7900 +	7921 +	0604 Test
79	6900 End	6921 Keypad 1	6904
80	7200 Down Cursor	7221 Keypad 2	7204
81	7A00 PgDn	7A21 Keypad 3	7A04
82	7000 Insert	7021 Keypad 0	7004
83	7100 Delete	7121 .	7104

Figure A-5 (Part 3 of 3). Default Scan Codes for IBM PC XT Keyboard (PC Mode)

Default Scan Codes for the IBM PC XT Keyboard (MFI Mode)

Note: MFI = host/notepad.

Key	Lowercase	Uppercase	Alternate Case
1	DE00 NF	0400 WS Ctrl	0300 Jump
2	1600 1	5421 !	1604
3	1E00 2	1E21 @	1E04
4	2600 3	2621 #	2604
5	2500 4	2521 \$	2504
6	2E00 5	2E21 %	2E04
7	3600 6	3621 ^	3604
8	3D00 7	3D21 &	3D04
9	3E00 8	3E21 *	3E04
10	4600 9	4621 (4604
11	4500 0	4521)	4504
12	4E00 -	4E21 _	4E04
13	5500 =	5521 +	5504
14	6600 Backspace	6621	6604
15	0D00 Right tab	0D21 Left tab	0D04
16	1500 q	1521 Q	1504
17	1D00 w	1D21 W	1D04
18	2400 e	2421 E	2404
19	2D00 r	2D21 R	2D04
20	2C00 t	2C21 T	2C04
21	3500 y	3521 Y	3504
22	3C00 u	3C21 U	3C04
23	4300 i	4321 I	4304
24	4400 o	4421 O	4404
25	4D00 p	4D21 P	4D04
26	5400 ¢	5300 {	5304
27	1621	5321 }	5304
28	5A00 Newline	5A21 Newline	5A04
29	0900 Ctrl	0921 Ctrl	0904 Ctrl
30	1C00 a	1C21 A	1C04
31	1B00 s	1B21 S	1B04
32	2300 d	2321 D	2304

Figure A-6 (Part 1 of 3). Default Scan Codes for IBM PC XT Keyboard (MFI Mode)

Key	Lowercase	Uppercase	Alternate Case
33	2B00 f	2B21 F	2B04
34	3400 g	3421 G	3404
35	3300 h	3321 H	3304
36	3B00 j	3B21 J	3B04
37	4200 k	4221 K	4204
38	4B00 l	4B21 L	4B04
39	4C00 ;	4C21 :	4C04
40	5200 '	5221 "	5204
41	0E00 `	0E21 ~	0E04
42	1200 Left Shift	1221 Left Shift	1204 Left Shift
43	5B00 \	5B21	5B04
44	1A00 z	1A21 Z	1A04
45	2200 x	2221 X	2204
46	2100 c	2121 C	2104
47	2A00 v	2A21 V	2A04
48	3200 b	3221 B	3204
49	3100 n	3121 N	3104
50	3A00 m	3A21 M	3A04
51	4100 ,	1300 <	4104
52	4900 .	1321 >	4904
53	4A00 /	4A21 ?	4A04
54	5900 Right Shift	5921 Right Shift	5904 Right Shift
55	5800 Enter	8300 Prnt Scr	DE00 NF
56	1900 Alt	1921 Alt	1904 Alt
57	2900 Spacebar	2921 Spacebar	2904 Spacebar
58	1400 Shift Lock	1421 Shift Lock	1404 Shift Lock
59	0700 PF1	5600 PF11	0C04 Attn
60	0F00 PF2	5E00 PF12	0600 Clear
61	1700 PF3	0800 PF13	0304 CrSel
62	1F00 PF4	1000 PF14	0404 ExSel
63	2700 PF5	1800 PF15	1804 ChFmt
64	2F00 PF6	2000 PF16	0B00 ErEOF
65	3700 PF7	2800 PF17	8300 Print
66	3F00 PF8	3000 PF18	6721 Dup
67	4700 PF9	3800 PF19	DE00 NF
68	4F00 PF10	4000 PF20	1100 Reset
69	7700 Num Lock	0504 SysRq	0100 Enlarge

Figure A-6 (Part 2 of 3). Default Scan Codes for IBM PC XT Keyboard (MFI Mode)

Scan Code Tables

Key	Lowercase	Uppercase	Alternate Case
70	6E21 FldMk	1104 DvCnl	0321 Chg Scr
71	6204 Home	6C21 Keypad 7	6C04
72	6300 Up Cursor	7521 Keypad 8	5004 CrPos
73	6700 PA1	7D21 Keypad 9	0B04 ErInp
74	6E00 PA2	7B00 -	4704 ReVid
75	6100 Left Cursor	6B21 Keypad 4	6104 Fast Left Cursor
76	8304 Ident	7321 Keypad 5	1004 Wrap
77	6A00 Right Cursor	7421 Keypad 6	6A04 Fast Right Cursor
78	7900 Enter	7921 Enter	0604 Test
79	DE00 NF	6921 Keypad 1	4F04 Blink
80	6000 Down Cursor	7221 Keypad 2	5604 UndSc
81	6F00 PA3	7A21 Keypad 3	5E04 Ex Highlight
82	6500 Insert	7021 Keypad 0	0804 DocMd
83	6D00 Delete	7121 .	6D04 WdDel
<i>Note: NF (no function) means no function has been assigned to the specified key location.</i>			

Figure A-6 (Part 3 of 3). Default Scan Codes for IBM PC XT Keyboard (MFI Mode)

Default Scan Codes for the IBM Personal Computer AT Keyboard (PC Mode)

Key	Lowercase	Uppercase	Alternate Case
1	0E00 `	0E21 ~	0E04
2	1600 1	1621 !	1604
3	1E00 2	1E21 @	1E04
4	2600 3	2621 #	2604
5	2500 4	2521 \$	2504
6	2E00 5	2E21 %	2E04
7	3600 6	3621 ^	3604
8	3D00 7	3D21 &	3D04
9	3E00 8	3E21 *	3E04
10	4600 9	4621 (4604
11	4500 0	4521)	4504
12	4E00 -	4E21 _	4E04
13	5500 =	5521 +	5504
14	5B00 \	5B21	5B04
15	6600 Backspace	6621 Backspace	6604
16	0D00 Right Tab	0D21 Left Tab	0D04
17	1500 q	1521 Q	1504
18	1D00 w	1D21 W	1D04
19	2400 e	2421 E	2404
20	2D00 r	2D21 R	2D04
21	2C00 t	2C21 T	2C04
22	3500 y	3521 Y	3504
23	3C00 u	3C21 U	3C04
24	4300 i	4321 I	4304
25	4400 o	4421 O	4404
26	4D00 p	4D21 P	4D04
27	5400 [5421 {	5404
28	5300]	5321 }	5304
30	0900 Ctrl	0921 Ctrl	0904 Ctrl
31	1C00 a	1C21 A	1C04
32	1B00 s	1B21 S	1B04
33	2300 d	2321 D	2304
34	2B00 f	2B21 F	2B04

Figure A-7 (Part 1 of 3). Default Scan Codes for IBM Personal Computer AT Keyboard (PC Mode)

Scan Code Tables

Key	Lowercase	Uppercase	Alternate Case
35	3400 g	3421 G	3404
36	3300 h	3321 H	3304
37	3B00 j	3B21 J	3B04
38	4200 k	4221 K	4204
39	4B00 l	4B21 L	4B04
40	4C00 ;	4C21 :	4C04
41	5200 '	5221 "	5204
43	5A00 Enter	5A21 Enter	5A04
44	1200 Left Shift	1221 Left Shift	1204 Left Shift
46	1A00 z	1A21 Z	1A04
47	2200 x	2221 X	2204
48	2100 c	2121 C	2104
49	2A00 v	2A21 V	2A04
50	3200 b	3221 B	3204
51	3100 n	3121 N	3104
52	3A00 m	3A21 M	3A04
53	4100 ,	4121 <	4104
54	4900 .	4921 >	4904
55	4A00 /	4A21 ?	4A04
57	5900 Right Shift	5921 Right Shift	5904 Right Shift
58	1900 Alt	1921 Alt	1904 Alt
61	2900 Spacebar	2921 Spacebar	2904 Spacebar
64	1400 Caps Lock	1421 Caps Lock	1404 Caps Lock
65	0F00 F2	0F21	0F04
66	1F00 F4	1F21	1F04
67	2F00 F6	2F21	2F04
68	3F00 F8	3F21	3F04
69	4F00 F10	4F21	4F04
70	0700 F1	0721	0704
71	1700 F3	1721	1704
72	2700 F5	2721	2704
73	3700 F7	3721	3704
74	4700 F9	4721	4704
90	7600 Esc	0400 WS Ctrl	0300 Jump
91	6C00 Home	6C21 Keypad 7	6C04
92	6B00 Left Cursor	6B21 Keypad 4	6B04
93	6900 End	6921 Keypad 1	6904

Figure A-7 (Part 2 of 3). Default Scan Codes for IBM Personal Computer AT Keyboard (PC Mode)

Key	Lowercase	Uppercase	Alternate Case
95	7700 NumLk	7721 NumLk	0100 Enlarge
96	7500 Up Cursor	7521 Keypad 8	7504
97	7300	7321 Keypad 5	7304
98	7200 Down Cursor	7221 Keypad 2	7204
99	7000 Insert	7021 Keypad 0	7004
100	7E00 ScrLk	1104 Quit	0321 ChScr
101	7D00 PgUp	7D21 Keypad 9	7D04
102	7400 Right Cursor	7421 Keypad 6	7404
103	7A00 PgDn	7A21 Keypad 3	7A04
104	7100 Delete	7121 .	7104
105	0504 SysReq	DE00 NF	DE00 NF
106	8300 *	8321 PrtScr	8304
107	7B00 -	7B21 -	7B04
108	7900 +	7921 +	0604 Test
<i>Note: NF (no function) means no function has been assigned to the specified key location.</i>			

Figure A-7 (Part 3 of 3). Default Scan Codes for IBM Personal Computer AT Keyboard (PC Mode)

Default Scan Codes for the IBM Personal Computer AT Keyboard (MFI Mode)

Key	Lowercase	Uppercase	Alternate Case
1	0E00 `	0E21 ~	0E04
2	1600 1	5421 !	1604
3	1E00 2	1E21 @	1E04
4	2600 3	2621 #	2604
5	2500 4	2521 \$	2504
6	2E00 5	2E21 %	2E04
7	3600 6	3621 ^	3604
8	3D00 7	3D21 &	3D04
9	3E00 8	3E21 *	3E04
10	4600 9	4621 (4604
11	4500 0	4521)	4504
12	4E00 -	4E21 _	4E04
13	5500 =	5521 +	5504
14	5B00 \	5B21	5B04
15	6600 Backspace	6621 Backspace	6604
16	0D00 Right Tab	6400 Left Tab	0D04
17	1500 q	1521 Q	1504
18	1D00 w	1D21 W	1D04
19	2400 e	2421 E	2404
20	2D00 r	2D21 R	2D04
21	2C00 t	2C21 T	2C04
22	3500 y	3521 Y	3504
23	3C00 u	3C21 U	3C04
24	4300 i	4321 I	4304
25	4400 o	4421 O	4404
26	4D00 p	4D21 P	4D04
27	5400 {	5300 {	5404
28	1621	5321 }	5304
30	0900 Ctrl	0921 Ctrl	0904 Ctrl
31	1C00 a	1C21 A	1C04
32	1B00 s	1B21 S	1B04
33	2300 d	2321 D	2304
34	2B00 f	2B21 F	2B04

Figure A-8 (Part 1 of 3). Default Scan Codes for IBM Personal Computer Keyboard (MFI Mode)

Key	Lowercase	Uppercase	Alternate Case
35	3400 g	3421 G	3404
36	3300 h	3321 H	3304
37	3B00 j	3B21 J	3B04
38	4200 k	4221 K	4204
39	4B00 l	4B21 L	4B04
40	4C00 ;	4C21 :	4C04
41	5200 '	5221 "	5204
43	5A00 Enter	5A21 Enter	5A04
44	1200 Left Shift	1221 Left Shift	1204 Left Shift
46	1A00 z	1A21 Z	1A04
47	2200 x	2221 X	2204
48	2100 c	2121 C	2104
49	2A00 v	2A21 V	2A04
50	3200 b	3221 B	3204
51	3100 n	3121 N	3104
52	3A00 m	3A21 M	3A04
53	4100 ,	1300 <	4104
54	4900 .	1321 >	4904
55	4A00 /	4A21 ?	4A04
57	5900 Right Shift	5921 Right Shift	5904 Right Shift
58	1900 Alt	1921 Alt	1904 Alt
61	2900 Spacebar	2921 Spacebar	2904 Spacebar
64	1400 Shift lock	1421 Shift lock	1404 Shift lock
65	0F00 PF2	5E00 PF12	0600 Clear
66	1F00 PF4	1000 PF14	0404 ExSel
67	2F00 PF6	2000 PF16	0B00 ErEOF
68	3F00 PF8	3000 PF18	6721 Dup
69	4F00 PF10	4000 PF20	1100 Reset
70	0700 PF1	5600 PF11	0C04 Attn
71	1700 PF3	0800 PF13	0304 CrSel
72	2700 PF5	1800 PF15	1804 ChFmt
73	3700 PF7	2800 PF17	8300 Print
74	4700 PF9	3800 PF19	DE00 NF
90	DE00 NF	0400 WS Ctrl	0300 Jump
91	6204 Home	6C21 Keypad 7	DE00 NF
92	6100 Left Cursor	6B21 Keypad 4	6B04 Fast Cursor Left

Figure A-8 (Part 2 of 3). Default Scan Codes for IBM Personal Computer Keyboard (MFI Mode)

Scan Code Tables

Key	Lowercase	Uppercase	Alternate Case
93	DE00 NF	6921 Keypad 1	4F04 Blink
95	7700 NumLk	7721 NumLk	0100 Enlarge
96	6300 Up Cursor	7521 Keypad 8	5004 CrPos
97	8304 Ident	7321 Keypad 5	1004 Wrap
98	6000 Down Cursor	7221 Keypad 2	5604 UndSc
99	6500 Insert	7021 Keypad 0	0804 DocMd
100	6E21 FldMk	1104 Quit	0321 ChScr
101	6700 PA1	7D21 Keypad 9	0B04 ErInp
102	6A00 Right Cursor	7421 Keypad 6	6A04 Fast Right
103	6F00 PA3	7A21 Keypad 3	5E04 FIHiCr
104	6D00 Delete	7121 .	6D04 WdDel
105	0504 SysReq	DE00 NF	DE00 NF
106	3E21 *	8300 PrtScr	DE00 NF
107	6E00 PA2	7B21 -	4704 ReVid
108	7900 Enter	7921 Enter	0604 Test
<i>Note: NF (no function) means no function has been assigned to the specified key location.</i>			

Figure A-8 (Part 3 of 3). Default Scan Codes for IBM Personal Computer Keyboard (MFI Mode)

ASCII Characters Common to All Countries

ASCII Code (hex) (dec)		PC and Host/Notepad Character	WsCtrl Function
20	32	Blank space	Space
22	34	"	Invalid
25	37	%	Invalid
26	38	&	Invalid
27	39	'	Invalid
28	40	(Invalid
29	41)	Invalid
2A	42	*	Invalid
2B	43	+	Invalid
2C	44	,	Invalid
2D	45	-	Invalid
2E	46	.	Invalid
2F	47	/	Invalid
30	48	0	Select screen
31	49	1	Select screen
32	50	2	Select screen
33	51	3	Select screen
34	52	4	Select screen
35	53	5	Select screen
36	54	6	Select screen
37	55	7	Select screen
38	56	8	Select screen
39	57	9	Select screen
3A	58	:	Invalid
3B	59	;	Invalid
3C	60	<	<
3D	61	=	Invalid
3E	62	>	>
3F	63	?	Invalid
41	65	A	Select window
42	66	B	Select window
43	67	C	Select window
44	68	D	Select window
45	69	E	Select window

Figure A-9 (Part 1 of 3). Valid ASCII Characters Common to All Countries

ASCII Code (hex) (dec)		PC and Host/Notepad Character	WsCtrl Function
46	70	F	Select window
47	71	G	Select window
48	72	H	Select window
49	73	I	Select window
4A	74	J	Select window
4B	75	K	Select window
4C	76	L	Select window
4D	77	M	Select window
4E	78	N	Select window
4F	79	O	Select window
50	80	P	Select window
51	81	Q	Select window
52	82	R	Select window
53	83	S	Select window
54	84	T	Select window
55	85	U	Select window
56	86	V	Select window
57	87	W	Select window
58	88	X	Select window
59	89	Y	Select window
5A	90	Z	Select window
5F	95	–	Invalid
61	97	a	Select window
62	98	b	Select window
63	99	c	Select window
64	100	d	Select window
65	101	e	Select window
66	102	f	Select window
67	103	g	Select window
68	104	h	Select window
69	105	i	Select window
6A	106	j	Select window
6B	107	k	Select window
6C	108	l	Select window
6D	109	m	Select window
6E	110	n	Select window

Figure A-9 (Part 2 of 3). Valid ASCII Characters Common to All Countries

ASCII Code (hex) (dec)	PC and Host/Notepad Character	WsCtrl Function
6F 111	o	Select window
70 112	p	Select window
71 113	q	Select window
72 114	r	Select window
73 115	s	Select window
74 116	t	Select window
75 117	u	Select window
76 118	v	Select window
77 119	w	Select window
78 120	x	Select window
79 121	y	Select window
7A 122	z	Select window

Figure A-9 (Part 3 of 3). Valid ASCII Characters Common to All Countries

ASCII Mnemonics Common to All Countries

ASCII Mnemonic	PC Function	Host/Notepad Function	WsCtrl Function
@ <	Backspace	Backspace	Backspace
@x	Invalid	PA1	Invalid
@S@x	Invalid	Dup	Invalid
@y	Invalid	PA2	Invalid
@S@y	Invalid	Field mark	Invalid
@z	Invalid	PA3	Invalid
@w	Esc	Invalid	Invalid
@t	Numlk	Numlk (DFT)	Invalid
@s	ScrIk	Invalid	Invalid
@r@s	Break	Invalid	Invalid
@T	Right tab	Right tab	Right tab
@B	Left tab	Left tab	Left tab
@I	Insert	Insert	Invalid
@A@D	Invalid	Delete word (CUT)	Invalid
@D	Delete character	Delete character	Invalid
@N	Newline	Newline	Newline
@E	Enter	Enter	Enter
@U	Cursor up	Cursor up	Cursor up
@V	Cursor down	Cursor down	Cursor down
@Z	Cursor right	Cursor right	Cursor right
@L	Cursor left	Cursor left	Cursor left
@A@Z	Invalid	Cursor right fast	Invalid
@A@L	Invalid	Cursor left fast	Invalid
@0	Home	Home	Home
@u	Pg Up	Invalid	Invalid
@v	Pg Down	Invalid	Invalid
@q	End	Invalid	Invalid
@H	Invalid	Invalid	Help
@C	Invalid	Clear	Invalid
@A@H	Invalid	Sys Req	Invalid
@A@C	Invalid	Test	Invalid
@W	Invalid	Invalid	WsCtrl
@Q	Invalid	Invalid	Finish
@A@W	Invalid	ExSel	Invalid

Figure A-10 (Part 1 of 3). Valid ASCII Mnemonics Common to All Countries

ASCII Mnemonic	PC Function	Host/Notepad Function	WsCtrl Function
@A@Q	Invalid	Attn	Invalid
@J	Invalid	Invalid	Jump
@S@J	Invalid	Invalid	ChgScr
@A@J	Invalid	CrSel	CrSel
@F	Invalid	Erase EOF	Erase EOF
@A@F	Invalid	Erinp	Erase
@P	Print	Print	Print
@A@P	Invalid	Ident	Invalid
@K	Invalid	Invalid	Copy
@S@K	Invalid	Invalid	Auto
@r@t	Pause	Invalid	Invalid
@M	Invalid	Invalid	Enlarge
@A@M	Invalid	Invalid	Delete Window
@Y	Caps Lock	Shift Lock	Shift Lock
@R	Invalid	Reset	Invalid
@A@R	Invalid	Dev Cancel	Quit
@1	F1	PF1	List
@2	F2	PF2	Setup
@3	F3	PF3	Browse
@4	F4	PF4	Play
@5	F5	PF5	Invalid
@6	F6	PF6	Invalid
@7	F7	PF7	Invalid
@8	F8	PF8	Color
@9	F9	PF9	Hide
@a	F10	PF10	Corner
@b	F11	PF11	Move
@c	F12	PF12	Size
@d	Invalid	PF13	Source
@e	Invalid	PF14	Target
@f	Invalid	PF15	Invalid
@q	Invalid	PF16	Record
@h	Invalid	PF17	Invalid
@i	Invalid	PF18	Invalid
@j	Invalid	PF19	Invalid
@k	Invalid	PF20	Invalid

Figure A-10 (Part 2 of 3). Valid ASCII Mnemonics Common to All Countries

ASCII Tables

ASCII Mnemonic	PC Function	Host/Notepad Function	WsCtrl Function
@l	Invalid	PF21	Base Color
@m	Invalid	PF22	Scr Bk Color
@n	Invalid	PF23	Foregr Color
@o	Invalid	PF24	Background Color
@A@1	Invalid	PSA (DFT only)	Invalid
@A@2	Invalid	PSB (DFT only)	Invalid
@A@3	Invalid	PSC (DFT only)	Invalid
@A@4	Invalid	PSD (DFT only)	Invalid
@A@5	Invalid	PSE (DFT only)	Invalid
@A@6	Invalid	PSF (DFT only)	Invalid
@A@7	Invalid	Reset program symbols (DFT only)	Invalid
@A@9	Invalid	Reverse video (DFT and notepad)	Invalid
@A@a	Invalid	Hilite (DFT and notepad)	Invalid
@A@b	Invalid	Underscore (DFT and notepad)	Invalid
@A@c	Invalid	Reset reverse video, etc. (DFT and notepad)	Invalid
@A@d	Invalid	Red (DFT and notepad) Doc mode (CUT)	Red
@A@e	Invalid	Pink (DFT and notepad) Wrap (CUT)	Pink
@A@f	Invalid	Green (DFT and notepad) Chg format (CUT)	Green
@A@g	Invalid	Yellow (DFT and notepad)	Yellow
@A@h	Invalid	Blue (DFT and notepad)	Blue
@A@i	Invalid	Turq (DFT and notepad)	Turq
@A@j	Invalid	White (DFT and notepad)	White
@A@k	Invalid	Black (DFT and notepad)	Black
@A@l	Invalid	Reset host colors (DFT and notepad)	Invalid
@A@m	Invalid	Cr Position (CUT)	Invalid
@/	Queue full, keystrokes lost	Queue full, keystrokes lost	Queue full, keystrokes lost
@@	@	@	Invalid

Figure A-10 (Part 3 of 3). Valid ASCII Mnemonics Common to All Countries

Additional ASCII Characters Used by U.S. English

ASCII Code (hex) (dec)		Character Description	PC	CUT	DFT/Notepad
21	33	!	x	x	x
23	35	#	x	x	x
24	36	\$	x	x	x
5B	91	[x		
5C	92	\	x	x	x
5D	93]	x		
5E	94	^	x		
60	96	`	x	x	x
7B	123	{	x	x	x
7C	124			x	x
7D	125	}	x	x	x
7E	126	~	x	x	x
9B	155	¢		x	x
AA	170	¬		x	x
DD	221	¡	x	x	x

Figure A-11. Additional ASCII Characters Used by U.S. English

Appendix B. Destination/Origin Structured Fields

Introduction	B-3
The 3270 Outbound Data Stream	B-3
The 3270 Inbound Data Stream	B-3
Verifying That the IBM 3270 Personal Computer Interface Is	
Operational	B-4
The Read Partition Query Structured Field	B-4
The Query Reply Structured Field	B-4
Query Reply	B-6
Input Control	B-6
PC Application Program and Display Interaction	B-7
Exception Handling	B-8
Structured Fields	B-9
Destination/Origin	B-9
Exception Condition	B-10
X'D0' Structured Fields for Sending Data from the Host to the 3270	
Personal Computer	B-11
The Open X'D0' Structured Field	B-12
Format of the Host Open Request	B-12
Format of the Successful Transmission Response	B-14
Format of the Unsuccessful Transmission Response	B-14
The Insert and Insert Data X'D0' Structured Fields	B-15
Format of the Host Insert and Insert Data Requests	B-15
Format of the Successful Transmission Response	B-16
Format of the Unsuccessful Transmission Response	B-17
The Close X'D0' Structured Field	B-18
Format of the Host Close Request	B-18
Format of the Successful Transmission Response	B-18
Format of the Unsuccessful Transmission Response	B-19
X'D0' Structured Fields for Sending Data from Personal Computer to	
Host	B-20
The Open X'D0' Structured Field	B-21
Format of the Host Open Request	B-21
Format of the Successful Transmission Response	B-22
Format of the Unsuccessful Transmission Response	B-23
The Set Cursor and Get X'D0' Structured Field	B-23
Format of the Host Set Cursor and Get Requests	B-24
Format of the Successful Transmission Response	B-25
Format of the Unsuccessful Transmission Response for Set	
Cursor	B-26
Format of the Unsuccessful Transmission Response for Get	B-27
The Close X'D0' Structured Field	B-28
Format of the Host Close Request	B-28
Format of the Successful Transmission Response	B-28
Format of the Unsuccessful Transmission Response	B-29

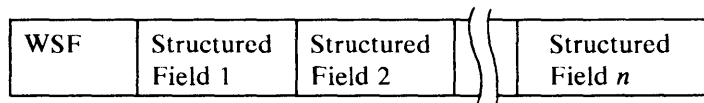
Introduction

This appendix describes the destination/origin structured field formats and protocol used by your IBM 3270 PC application program to move data between a host session and the personal computer session (using the host interactive services). Structure types accepted by the 3270 Workstation Program are designated Open (X'D000'), Close (X'D041'), Set Cursor (X'D045'), Get (X'D046'), and Insert and Insert Data (X'D047'). No other types are allowed when using the 3270 PC application program interface.

The 3270 data stream was defined for use between a host application program and a single display; it allows support of a 3270 data stream work station. A 3270 data stream work station consists of a 3270 data stream display and one or more personal computer (PC) application programs. A PC application program does not accept the usual 3270 data stream (for example, 3270 commands, orders, and so forth). However, the 3270 data stream is used to carry the data streams associated with the PC application programs. The data to and from PC application programs must be in the form of structured fields.

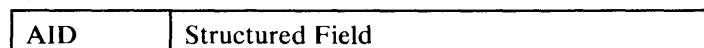
The 3270 Outbound Data Stream

The 3270 outbound data stream is a data stream sent from the host to the 3270 Personal Computer. The 3270 outbound data stream containing structured fields begins with a Write Structured Field (WSF) command X'F3' or X'11'. Multiple structured fields can be sent with one WSF command.



The 3270 Inbound Data Stream

The 3270 inbound data stream is a data stream sent from the 3270 Personal Computer to the host. The 3270 inbound data stream containing structured fields begins with an attention identifier (AID):



The structures used by the 3270 Personal Computer follow the 3270 data stream format. The maximum number of bytes that can be sent in one transmission in either direction is 3.5K bytes (K equals 1024). (For more information on structured fields, refer to the *IBM 3274 Control Unit Description and Programmer's Guide*.)

Verifying That the IBM 3270 Personal Computer Interface Is Operational

Prior to a request from the host application to the 3270 Personal Computer, the host application must verify (with the control unit) that the 3270 Personal Computer interface is operational. This is done with a Read Partition Query structured field. The workstation program then returns a specific Query Reply back to the host.

The Read Partition Query Structured Field

The read partition query and query reply structures verify that a path exists between the host application and the 3270 PC application. The host application inquires about the 3270 PC application.

Figure B-1 shows the format of the read partition query structured field.

Byte	Contents	Meaning
0, 1	X'0005'	Length of structured field in bytes
2	X'01'	Read-partition ID code
3	X'FF'	Partition identifier physical terminal: query operation
4	Type X'02'	Query (used in both implicit partition and explicit partitioned states)

Figure B-1. Read Partition Query Structured Field Format

The Query Reply Structured Field

Figure B-2 shows the format of the query reply structured field. The field is returned only when the PC application is loaded and active.

The maximum bytes per transmission allowed on inbound and outbound transmissions is the 3.5K-byte length restriction enforced by the control unit on structures to and from the 3270 Personal Computer. (Refer to the *IBM 3274 Control Unit Description and Programmer's Guide* for more information.)

Offset	Length	Contents	Meaning
0	1 byte	Must be zero	Not used
1	1 byte	X'19'	Length of structure
2	1 byte	X'81'	Query Reply
3	1 byte	X'9D'	Query Reply type
4	1 byte	Must be zero	Reserved flags
5	1 byte	X'01'	Structured Field Exchange
6, 7	2 bytes	Maximum of X'0E00'	Maximum number of bytes allowed in an inbound transmission
8, 9	2 bytes	Maximum of X'0E00'	Maximum number of bytes allowed in an outbound transmission
10	1 byte	Must be X'0F'	Identifies the next two bytes as being the destination/origin ID.
11	1 word	Must be zero	Destination/origin ID supplied by the 3270 Workstation Program.
13 - 24	12 bytes	APLNME	Application name (in EBCDIC)

Figure B-2. Query Reply Structured Field Format

When the 3270 Personal Computer powers off, the interface to the control unit is disabled. If a host application attempts to exchange data with a PC application, the control unit returns a Data Stream Error-OP CHECK. (Refer to the component description card in your *Guide to Operations* for more information about OP CHECK.)

The presentation space associated with a PC application program is independent of the display presentation space. Data directed to a PC application program does not alter the display presentation space, and data directed to the display presentation space does not alter the presentation space associated with a PC application program.

A different type of Query Reply is defined for each different IBM data stream used by PC application programs. The Query Reply identifies the IBM data stream supported.

The display is the default destination or origin if the data destination or origin is not explicitly identified by a destination/origin structured field. Data of a type not supported that is directed to the display or a PC application program will be rejected.

At the start of each outbound transmission the destination is the display, and at the start of each inbound transmission the origin is the display. The destination/origin remains the display unless changed by a destination/origin structured field. Once a destination/origin structured field has established the destination/origin of the data, that destination/origin applies for all structured fields that follow until the end of the transmission or until changed by a subsequent destination/origin structured field.

Input Control

Query Reply

The PC application program Query Reply is sent in reply to either a Query or Query List.

Return of the AUXDA Query Reply indicates a 3270 Data Stream Work Station implementation (that is, support of the destination/origin structured field and one or more PC application programs). The AUXDA Query Reply is returned in reply to either a Query List (= AUXDA or All) or a Query.

The workstation program inserts the Destination/Origin Identification (DOID) value into the Query Reply for the Destination/Origin structured field in individual PC application programs.

A Query or Query List directed to a PC application program instead of to the display will be rejected.

A *separate* Query Reply must be returned for each PC application program supported. For example, if two identical PC application programs were supported, a Query Reply would be returned for each. The DOID reported would be different for each.

Input Control

The host application controls when the PC application is permitted to send in data. The control is achieved with the INCTRL (input control) flag in the destination/origin structured field. The INCTRL flag has meaning only on outbound transmissions (to the 3270 PC) and is ignored on inbound transmissions. When the destination/origin structured field is directed to the display (ID = X'0000'), the INCTRL flag provides a global control.

The default (for example, Power-On-Reset from the control unit) is input-disabled. Once input is enabled, it remains enabled until disabled by one of the following:

- A destination/origin structured field with the INCTRL flag set to B'10' (input disable) is sent outbound to the PC application.
- A destination/origin structured field with the INCTRL flag set to B'10' (global input disable) is sent outbound to the display.
- An Erase Write or Erase Write Alternate command with the Write control character set to reset is sent outbound.
- A clear local function (for example, the Clear key is pressed).
- A Power-On-Reset.
- The 3270 PC receives a Bind (SNA only).

Receiving a destination/origin structured field from the host application with INCTRL set to B'01' will not cause a change in the input enable/disable state of the PC application. Also, if the INCTRL flag value is the same as the existing input enable/disable state, the state is unchanged. For example, if the input enable/disable state is input-enabled, receiving a destination/origin structured field with INCTRL set to B'00' (input enable) will be accepted and the input enable/disable state will remain enabled.

Note: There is one exception where input may be sent without being enabled. An exception condition structured field, reporting unavailability of the PC application, may be sent in reply to a destination/origin structured field sequence attempting to use it.

PC Application Program and Display Interaction

The PC application programs must conform to the read operations described in the *IBM 3270 Information Display System Data Stream Programmer's Reference*, except where otherwise noted here.

When data is read in from a PC application, the rules or states for Read Retry and Read Acknowledgment apply. For example, once a transmission is sent from a PC application, additional data from that application cannot be sent inbound until a Read Acknowledgment is received. If the data from a PC application is transmitted in multiple transmissions, each transmission requires an acknowledgment. An inbound transmission may contain data from the display and/or data from one or more PC applications. When display data is sent in the same transmission as PC application data, the Inbound 3270DS structured field must be used for the display data. An inbound transmission containing data from PC applications must start with an AID of X'88', which indicates structured fields follow. The same conditions that acknowledge a Query Reply will acknowledge an inbound transmission from a PC application.

An outbound transmission to a PC application constitutes a read acknowledgment per the description for outbound display transmissions. The fact that the transmission is to a PC application adds no additional acknowledgment function. For example, a transmission to a PC application would acknowledge an outstanding Query Reply transmission because the transmission contained a WSF. As another example, in the SNA environment a transmission to a PC application would constitute an acknowledgment to an outstanding enter transmission only if the transmission put the work station in a send or contention state.

Only one display-type read may occur in an outbound transmission, and when in structured field form, it must be the last structured field in the transmission. A display-type read is defined as any of the following:

- A query or query list structured field
- A read partition structured field
- A Read Buffer, Read Modified, or Read Modified All command.

Exception Handling

In an outbound transmission, data to a PC application can initiate inbound data from that application. Inbound data can be initiated from multiple PC applications by a single outbound transmission containing multiple destination/origin structured fields. Inbound data from one or more PC applications can be initiated in an outbound transmission that also contains a display-type read. When this occurs, the display-type read is executed first.

A display-type read always takes priority over pending inbound data from a PC application. A display operator enter action is considered a display-type read. If inbound data is pending from one or more PC applications, an operator enter action will take priority and use the next available inbound transmission.

When the data from a PC application must be sent in multiple transmissions (for example, a transmission size limit imposed for certain data), each inbound transmission is treated like an Enter, to the extent that sending of the data is initiated by the application. A host Read Acknowledgment is required prior to sending the next part of the data. Therefore, data from a PC application that is sent in multiple transmissions could have some interspersed display transmissions. Also, the display operator is not "locked out" as a result of a PC application condition (for example, power off, diskette removed, and so forth).

Exception Handling

An exception condition in a PC application does not cause the session between the host and the 3270DS work station to be terminated. That is, a PC application program exception condition must not cause a negative response. Exception conditions must be reported at the application level.

In general, the exception handling is defined by the data stream used by the PC application.

Some exception conditions are handled within the 3270DS. For example, if the PC application is not available (such as when power is off or processing code is not resident), the unavailability is reported by returning a destination/origin structured field followed by an exception condition structured field with the code field set to X'0801' (resource not available). Another example is where the host exceeds the transmission size specified in a PC application Query Reply. In this case, the code field is set to X'084C' (permanent insufficient resource).

Structured Fields

Destination/Origin

The destination/origin structured field is used to designate the destination or origin of the structured fields that follow in the data stream.

Format

Byte	Bit	Contents	Meaning
0, 1		X'0008'	Length of this structure
2, 3		X'0F02'	Destination/origin
4	0, 1	Flags: INCTRL B'00' B'01' B'10' B'11'	Input Control Enable input No change Disable input Reserved
	2 – 7	RES	Reserved – must be zeros
5		Flags	Must be zeros
6, 7		ID	Identifies the destination or origin of the structured fields that follow in the data stream

Flags: INCTRL applies only on outbound transmissions (to the PC application). The INCTRL flag is ignored on inbound retransmissions.

1. B'00' – The PC application is allowed to send data. If the PC application is already enabled, it will remain enabled.
2. B'01' – A change does not occur in the enabled/disabled status.
3. B'10' – The PC application is not permitted to send data until subsequently enabled by a destination/origin structured field with INCTRL flag = B'01'. If the PC application is already disabled, the INCTRL flag = B'10' will cause no change.

If a destination/origin structured field is directed to the base display (ID = X'0000'), the INCTRL flag applies on a global basis. That is, all the supported PC applications are enabled, disabled, or unchanged as a group.

Note: There is one case where a PC application may send input without being enabled. An exception condition structured field, reporting unavailability of the PC application, may be sent in reply to a destination/origin structured field sequence attempting to use the PC application.

Structured Fields

ID: The valid values for the ID are:

- X'0000' (permanently assigned to the primary display)
- All ID values returned in the Query Reply(s) for PC applications.

All other values are invalid and are rejected.

Operation: The function of the destination/origin structured field is to identify the destination or origin of the structured fields in a single-session multidevice (work station) implementation.

Outbound (from the host) the ID identifies the destination of the structured fields that follow. Inbound (to the host) the ID identifies the origin of the structured fields that follow.

At the beginning of the transmission, the destination/origin is the default (primary display).

Once a destination/origin structured field establishes the destination/origin, it applies until either another destination/origin structured field establishes a new destination/origin or a new transmission starts.

Exception Condition

The exception condition structured field allows the reporting of exception information at the application level.

Format

Byte	Bit	Contents	Meaning
0, 1		L	Length of structured field
2, 3		X'0F22'	Exception condition
4		PID	Partition identifier
5, 6		Flags	Reserved — must be zeros

Self-Defining Parameter — Application Program Exception Condition

Byte	Bit	Contents	Meaning
0		X'06'	Length of parameter
1		X'01'	Application program exception condition
2, 3		RES	Reserved — must be zero
4, 5		EXCODE	Exception code

PID: The PID should be set to X'FF'.

EXCODE: This defines the specific direct-accessed PC application exception condition.

Code	Meaning
------	---------

0801	Resource not available. The required processing code is not resident.
------	---

084B	Temporary insufficient resource. The application did not provide a buffer.
------	--

084C	Permanent insufficient resource. The host sent more than X'0E00' bytes of data.
------	---

1003	Invalid function code
------	-----------------------

Operation: The exception condition structured field is allowed to carry only one exception condition.

When used for reporting an exception condition for a direct-accessed PC application, the exception condition structured field must be preceded by a destination/origin structured field.

X'D0' Structured Fields for Sending Data from the Host to the 3270 Personal Computer

The following structures describe the requests sent by the host and the responses sent by the 3270 PC application through the 3270 PC Application Program Interface.

The data part of the Insert and Insert Data X'D0' structured fields is defined by the host application file formats for the 3270 Personal Computer. All numbers given in the request and response formats are hexadecimal values.

Transferring data from the host to the 3270 Personal Computer is accomplished by the following sequence of X'D0' structured fields:

1. The Open X'D0' Structured Field (X'D000')

The host application sends an Open X'D0' structured field request to the 3270 PC application program. The Open request contains the ASCII format name of the data to be sent and the ASCII file specification of the file to be created on the 3270 Personal Computer. The application program must check the host request for validity, and send a X'D0' structured field response to the host indicating whether the Open request was successful or unsuccessful.

X'D0' Structured Fields, Host to 3270 PC

2. The Insert and Insert Data X'D0' Structured Fields (X'D047')

The host application sends the Insert and Insert Data X'D0' structured field requests to the application program. These two requests are always sent in the same transmission. The Insert request indicates to the application program that an insert operation is to be done on the opened file. The Insert Data request contains both the length of the data to be inserted into the opened file and the data itself. The application program must check the host requests for validity, and send a X'D0' structured field response indicating whether the Insert and Insert Data requests were successful or unsuccessful. The host program continues to send the Insert and Insert Data requests to the 3270 PC application program until all the data is sent.

3. The Close X'D0' Structured Field (X'D041')

The host application sends a Close X'D0' structured field request to the 3270 PC application program when all the data has been sent. The application program must check the host request for validity, and send a X'D0' structured field response to the host indicating whether the Close request was successful.

The Open X'D0' Structured Field

The Open X'D0' structured field request forms a logical connection between an application on the host system and a file on the 3270 Personal Computer system. Once the connection has been made, requests and data may flow from the host to the 3270 Personal Computer.

Format of the Host Open Request

The buffer sent by the host to the 3270 PC application program for the Open X'D0' structured field request must be formatted as follows:

Byte (Decimal)	Contents (Hex)
0, 1	LL, a 2-byte length field (the length of the transmission, including LL)
2 through 10	D0 00 12 01 06 01 01 04 03
11 through 19	0A 0A 00 00 00 00 11 01 01
20 through 26	00 50 05 52 03 F0 08
27	L0, a 1-byte length field (8 + length of format NAME)
28	28
29	L1, a 1-byte length field (2 + length of format NAME)
30 through n	NAME, a variable-length field containing the format name (in ASCII)

X'D0' Structured Fields, Host to 3270 PC

Byte (Decimal)	Contents (Hex)
n + 1	03
n + 2	L2 , a 1-byte length field (2 + length of the 3270 Personal Computer file specification)
n + 3 through m	FILSPEC , a variable-length field containing the 3270 Personal Computer file specification in ASCII

Note: The host Open request must be coded as shown above.

When the Workstation Program receives the request from the host, it puts it into the buffer defined by the DEF-BUF service. The 3270 PC application uses the READ-SF and GET-COMP services to receive the data in the following format:

Byte (Decimal)	Contents (Hex)
0, 1	00 00
2, 3	xx xx , a 2-byte length field containing the length of the outbound transmission received from the host, excluding the buffer header, which is 8 bytes long. This is in the PC format with low-order byte first.
4 through 7	00 00 C0 00
8, 9	LL , a 2-byte length field (the length of the transmission, including LL)
10 through 18	D0 00 12 01 06 01 01 04 03
19 through 27	0A 0A 00 00 00 00 11 01 01
28 through 34	00 50 05 52 03 F0 08
35	L0 , a 1-byte length field (8 + length of format NAME)
36	28
37	L1 , a 1-byte length field (2 + length of format NAME)
38 through n	NAME , a variable-length field containing the format name (in ASCII)
n + 1	03
n + 2	L2 , a 1-byte length field (2 + length of the 3270 Personal Computer file specification)
n + 3 through m	FILSPEC , a variable-length field containing the 3270 Personal Computer file specification in ASCII

Bytes 0 through 7 are the buffer header.

X'D0' Structured Fields, Host to 3270 PC

Format of the Successful Transmission Response

To indicate a successful transmission of the Open X'D0' structured field request, the 3270 PC application uses the WRITE-SF service to send a message to the host in the following format:

Byte (Decimal)	Contents (Hex)
0 through 7	00 00 05 00 00 00 00 00
8 through 12	00 05 D0 00 09

Bytes 0 through 7 are the buffer header. Bytes 2 and 3 are the length of the structured field message that begins in byte 8. The successful transmission response **must** be coded as shown above.

When the Workstation Program receives the response from the 3270 PC application program, it sends it to the host in the following format:

Byte (Decimal)	Contents (Hex)
0 through 4	00 05 D0 00 09

Format of the Unsuccessful Transmission Response

To indicate an unsuccessful transmission of the Open X'D0' structured field request, the 3270 PC application program uses the WRITE-SF service to send a message to the host in the following format:

Byte (Decimal)	Contents (Hex)
0 through 7	00 00 09 00 00 00 00 00
8 through 16	00 09 D0 00 08 69 04 xx 00

Bytes 0 through 7 are the buffer header. Bytes 2 and 3 are the length of the structured field message that begins in byte 8. The unsuccessful transmission response **must** be coded as shown above.

xx is one of the following:

- 01 Open failed exception
- 02 Arrival sequence not allowed
- 1A File name invalid
- 1B File not found
- 1C File size invalid
- 20 Function/open error
- 2A Path not found
- 5D Unsupported type

- 60 Command sequence error
- 62 Parameter is missing
- 63 Parameter not supported
- 65 Parameter value not supported
- 71 Invalid format

When the Workstation Program receives the response from the 3270 PC application program, it sends it to the host in the following format:

Byte (Decimal)	Contents (Hex)
0 through 8	00 09 D0 00 08 69 04 xx 00

xx is one of the values listed above.

The Insert and Insert Data X'D0' Structured Fields

The Insert and Insert Data X'D0' structured field requests are always sent in the same transmission. The Insert X'D0' structured field request indicates that an insert operation is to be performed on the opened file. The Insert Data X'D0' structured field request contains both the length of the data to be inserted into the opened file and the data itself.

Format of the Host Insert and Insert Data Requests

The buffer sent by the host to the 3270 PC application program for the Insert and Insert Data X'D0' structured field requests must be formatted as follows:

Byte (Decimal)	Contents (Hex)
0 through 9	00 0A D0 47 11 01 05 00 80 00
10, 11	LL , a 2-byte length field (the length of the transmission, from LL to the end of the data)
12 through 17	D0 47 04 C0 80 61
18, 19	LD , a 2-byte length field (5 + length of the data)
20 through n	DATA , the data being sent

*Note: The host Insert and Insert Data requests **must** be coded as shown above.*

X'D0' Structured Fields, Host to 3270 PC

When the Workstation Program receives the request from the host, it puts it into the buffer defined by the DEF-BUF service. The 3270 PC application program uses the READ-SF and GET-COMP services to receive the data in the following format:

Byte (Decimal)	Contents (Hex)
0, 1	00 00
2, 3	xx xx , a 2-byte length field containing the length of the outbound transmission received from the host excluding the buffer header, which is 8 bytes long. This is in PC format with low-order byte first.
4 through 7	00 00 C0 00
8 through 17	00 0A D0 47 11 01 05 00 80 00
18, 19	LL , a 2-byte length field (the length of the transmission, including LL)
20 through 25	D0 47 04 C0 80 61
26, 27	LD , a 2-byte length field (5 + length of the data)
28 through n	DATA , the data being sent

Bytes 0 through 7 are the buffer header.

Format of the Successful Transmission Response

To indicate a successful transmission of the Insert and Insert Data X'D0' structured field requests, the 3270 PC application program uses the WRITE-SF service to send a message to the host in the following format:

Byte (Decimal)	Contents (Hex)
0 through 7	00 00 0B 00 00 00 00 00
8 through 14	00 0B D0 47 05 63 06
15 through 18	DATBLK , a 4-byte field containing the data block number received. For the first data block, DATBLK = 00 00 00 01.

Bytes 0 through 7 are the buffer header. Bytes 2 and 3 are the length of the structured field message that begins in byte 8. The successful transmission response **must** be coded as shown above.

X'D0' Structured Fields, Host to 3270 PC

When the Workstation Program receives the response from the 3270 PC application program, it sends it to the host in the following format:

Byte (Decimal)	Contents (Hex)
0 through 6	00 0B D0 47 05 63 06
7 through 10	DATBLK , a 4-byte field containing the data block number received. For the first data block, DATBLK = 00 00 00 01.

Format of the Unsuccessful Transmission Response

To indicate an unsuccessful transmission of the Insert and Insert Data X'D0' structured field requests, the 3270 PC application program uses the WRITE-SF service to send a message to the host in the following format:

Byte (Decimal)	Contents (Hex)
0 through 7	00 00 09 00 00 00 00 00
8 through 16	00 09 D0 47 08 69 04 xx 00

Bytes 0 through 7 are the buffer header. Bytes 2 and 3 are the length of the structured field message that begins in byte 8. The unsuccessful transmission response **must** be coded as shown above.

xx is one of the following:

- 02 Arrival sequence not allowed
- 3E Operation not authorized
- 47 Record not added, storage limit
- 5D Unsupported type
- 60 Command syntax error
- 62 Parameter is missing
- 63 Parameter not supported
- 65 Parameter value not supported
- 6E Data element missing
- 70 Record length = 0
- 71 Invalid flag value

When the Workstation Program receives the response from the 3270 PC application program, it sends it to the host in the following format:

Byte (Decimal)	Contents (Hex)
0 through 8	00 09 D0 47 08 69 04 xx 00

xx is one of the values listed above.

X'D0' Structured Fields, Host to 3270 PC

The Close X'D0' Structured Field

The Close X'D0' structured field request performs the logical termination of a connection between a file on the host system and a previously opened file on the personal computer system.

Format of the Host Close Request

The buffer sent by the host to the 3270 PC application program for the Close X'D0' structured field request must be formatted as follows:

Byte (Decimal)	Contents (Hex)
0 through 4	00 05 D0 41 12

*Note: The host close request **must** be coded as shown above.*

When the Workstation Program receives the request from the host, it puts it into the buffer defined by the DEF-BUF service. The 3270 PC application program uses the READ-SF and GET-COMP services to receive the data in the following format:

Byte (Decimal)	Contents (Hex)
0 through 7	00 00 05 00 00 00 C0 00
8 through 12	00 05 D0 41 12

Bytes 0 through 7 are the buffer header.

Format of the Successful Transmission Response

To indicate a successful transmission of the Close X'D0' structured field request, the 3270 PC application program uses the WRITE-SF service to send a message to the host in the following format:

Byte (Decimal)	Contents (Hex)
0 through 7	00 00 05 00 00 00 00 00
8 through 12	00 05 D0 41 09

Bytes 0 through 7 are the buffer header. Bytes 2 and 3 are the length of the structured field message that begins in byte 8. The successful transmission response **must** be coded as shown above.

X'D0' Structured Fields, Host to 3270 PC

When the Workstation Program receives the response from the 3270 PC application program, it sends it to the host in the following format:

Byte (Decimal)	Contents (Hex)
0 through 4	00 05 D0 41 09

Format of the Unsuccessful Transmission Response

To indicate an unsuccessful transmission of the Close X'D0' structured field request, the 3270 PC application program uses the WRITE-SF service to send a message to the host in the following format:

Byte (Decimal)	Contents (Hex)
0 through 7	00 00 09 00 00 00 00 00
8 through 16	00 09 D0 41 08 69 04 xx 00

Bytes 0 through 7 are the buffer header. Bytes 2 and 3 are the length of the structured field message that begins in byte 8. The unsuccessful transmission response **must** be coded as shown above.

xx is one of the following:

03 Close of an unopened file
5D Unsupported type
60 Command syntax error
71 Invalid format

When the Workstation Program receives the response from the 3270 PC application program, it sends it to the host in the following format:

Byte (Decimal)	Contents (Hex)
0 through 8	00 09 D0 41 08 69 04 xx 00

xx is one of the values listed above.

X'D0' Structured Fields for Sending Data from Personal Computer to Host

The following structures detail the requests sent by the host and the responses sent by the 3270 PC application program through the 3270 Personal Computer Application Program Interface.

The data part of the Set Cursor and Get X'D0' structured fields is defined by the host application file formats for the 3270 Personal Computer. All numbers given in the request and response formats are hexadecimal values.

Data is transferred from the 3270 Personal Computer to the host by the following sequence of X'D0' structured fields:

1. The Open X'D0' Structured Field (X'D000')

The host application sends an Open X'D0' structured field request to the 3270 PC application program. The Open request contains the ASCII format name of the data to be sent and the ASCII file specification of the file to be sent to the host from the 3270 Personal Computer. The application program must check the host request for validity, and send a X'D0' structured field response to the host indicating whether the Open request was successful or unsuccessful.

2. The Set Cursor and Get X'D0' Structured Fields (X'D045' and X'D046')

The host application sends the Set Cursor and Get X'D0' structured field requests to the application program. These two requests are always sent in the same transmission. The Set Cursor request sets the logical block pointer (cursor) of the opened file to the next data block to be sent. The Get request requests a block of data from the 3270 Personal Computer. The PC application sends the specified data block of the opened file to the host. The application program must check the host requests for validity, and send a X'D0' structured field response indicating whether the Set Cursor and Get requests were successful or unsuccessful. The host program continues to send the Set Cursor and Get requests to the 3270 PC application program until all the data is sent.

3. The Close X'D0' Structured Field (X'D041')

The host application sends a Close X'D0' structured field request to the 3270 PC application program when all the data has been sent. The application program must check the host request for validity, and send a X'D0' structured field response to the host indicating whether the Close request was successful.

The Open X'D0' Structured Field

The Open X'D0' structured field request forms a logical connection between an application on the host system and a file on the 3270 Personal Computer system. Once the connection has been made, requests may flow from the host to the 3270 Personal Computer, and data may flow back from the 3270 Personal Computer to the host.

Format of the Host Open Request

The buffer sent by the host to the 3270 PC application program for the Open X'D0' structured field request must be formatted as follows:

Byte (Decimal)	Contents (Hex)
0, 1	LL , a 2-byte length field (the length of the transmission, including LL)
2 through 10	D0 00 12 01 06 01 01 04 03
11 through 19	0A 0A 00 01 00 00 00 00 01
20 through 26	00 50 05 52 03 F0 08
27	L0 , a 1-byte length field (4 + length of format NAME)
28	28
29	L1 , a 1-byte length field (2 + length of format NAME)
30 through n	NAME , a variable-length field containing the format name (in ASCII)
n + 1	03
n + 2	L2 , a 1-byte length field (2 + length of the 3270 Personal Computer file specification)
n + 3 through m	FILSPEC , a variable-length field containing the 3270 Personal Computer file specification (in ASCII)

*Note: The host Open request **must** be coded as shown above.*

X'D0' Structured Fields, PC to Host

When the Workstation Program receives the request from the host, it puts it into the buffer defined by the DEF-BUF service. The 3270 PC application program uses the READ-SF and GET-COMP services to receive the data in the following format:

Byte (Decimal)	Contents (Hex)
0, 1	00 00
2, 3	xx xx , a 2-byte length field containing the length of the outbound transmission received from the host excluding the buffer header, which is 8 bytes long. This is in PC format with low-order byte first.
4 through 7	00 00 C0 00
8, 9	LL , a 2-byte length field (the length of the transmission, including LL)
10 through 18	D0 00 12 01 06 01 01 04 03
19 through 27	0A 0A 00 01 00 00 00 00 01
28 through 34	00 50 05 52 03 F0 08
35	L0 , a 1-byte length field (4 + length of format NAME)
36	28
37	L1 , a 1-byte length field (2 + length of format NAME)
38 through n	NAME , a variable-length field containing the format name (in ASCII)
n + 1	03
n + 2	L2 , a 1-byte length field (2 + length of the 3270 Personal Computer file specification)
n + 3 through m	FILSPEC , a variable-length field containing the 3270 Personal Computer file specification (in ASCII)

Bytes 0 through 7 are the buffer header.

Format of the Successful Transmission Response

To indicate a successful transmission of the Open X'D0' structured field request, the 3270 PC application program uses the WRITE-SF service to send a message to the host in the following format:

Byte (Decimal)	Contents (Hex)
0 through 7	00 00 05 00 00 00 00 00
8 through 12	00 05 D0 00 09

Bytes 0 through 7 are the buffer header. Bytes 2 and 3 are the length of the structured field message that begins in byte 8. The successful transmission response **must** be coded as shown above.

When the Workstation Program receives the response from the 3270 PC application program, it sends it to the host in the following format:

X'D0' Structured Fields, PC to Host

Byte (Decimal)	Contents (Hex)
0 through 4	00 05 D0 00 09

Format of the Unsuccessful Transmission Response

To indicate an unsuccessful transmission of the Open X'D0' structured field request, the 3270 PC application program uses the WRITE-SF service to send a message to the host in the following format:

Byte (Decimal)	Contents (Hex)
0 through 7	00 00 09 00 00 00 00 00
8 through 16	00 0B D0 00 08 69 04 xx 00

Bytes 0 through 7 are the buffer header. Bytes 2 and 3 are the length of the structured field message that begins in byte 8. The unsuccessful transmission response **must** be coded as shown above.

xx is one of the following:

- 01 Open failed exception
- 02 Arrival sequence not allowed
- 1A File name invalid
- 1B File not found
- 1C File size invalid
- 20 Function/open error
- 2A Path not found
- 5D Unsupported type
- 60 Command syntax error
- 62 Parameter is missing
- 63 Parameter not supported
- 65 Parameter value not supported
- 71 Invalid format

When the Workstation Program receives the response from the 3270 PC application program, it sends it to the host in the following format:

Byte (Decimal)	Contents (Hex)
0 through 8	00 09 D0 00 08 69 04 xx 00

xx is one of the values listed above.

The Set Cursor and Get X'D0' Structured Field

The Set Cursor and Get X'D0' structured field requests are always sent in the same transmission. The Set Cursor X'D0' structured field request sets the logical block pointer (cursor) in the opened file to the next data block to be sent. The Get request requests a block of data from the 3270 Personal Computer. The PC application program sends the specified data block of the opened file to the host in its reply.

X'D0' Structured Fields, PC to Host

Format of the Host Set Cursor and Get Requests

The buffer sent by the host to the 3270 PC application program for the Set Cursor and Get X'D0' structured field requests must be formatted as follows:

Byte (Decimal)	Contents (Hex)
0 through 6	00 0F D0 45 11 01 05
7 through 14	00 06 00 09 05 01 03 00
15 through 23	00 09 D0 46 11 01 04 00 80

*Note: The host Set Cursor and Get requests **must** be coded as shown above.*

When the Workstation Program receives the request from the host, it puts it into the buffer defined by the DEF-BUF service. The 3270 PC application program uses the READ-SF and GET-COMP services to receive the data in the following format:

Byte (Decimal)	Contents (Hex)
0, 1	00 00
2, 3	xx xx , a 2-byte length field containing the length of the outbound transmission received from the host excluding the buffer header, which is 8 bytes long. This is the PC format with low-order byte first.
4 through 7	00 00 C0 00
8 through 17	00 0F D0 45 11 01 05 00 06 00
18 through 22	09 05 01 03 00
23 through 31	00 09 D0 46 11 01 04 00 80

Bytes 0 through 7 are the buffer header.

X'D0' Structured Fields, PC to Host

Format of the Successful Transmission Response

To indicate a successful transmission of the Set Cursor and Get X'D0' structured field requests, the 3270 PC application program uses the WRITE-SF service to send a message to the host in the following format:

Byte (Decimal)	Contents (Hex)
0, 1	00 00
2, 3	xx xx , a 2-byte length field containing the length of the data that begins in byte 8. This is the PC format with low-order byte first.
4 through 7	00 00 00 00
8, 9	LL , a 2-byte length field (the length of the structure including LL)
10 through 14	D0 46 05 63 06
15 through 18	DATBLK , a 4-byte field, the data block number being sent
19, 20	C0 80
21, 22	LD , a 2-byte length field (5 + length of the data)
23 through n	DATA , the data being sent

Bytes 0 through 7 are the buffer header. Bytes 2 and 3 are the length of the structured field message that begins in byte 8. The successful transmission response **must** be coded as shown above.

When the Workstation Program receives the response from the 3270 PC application program, it sends it to the host in the following format:

Byte (Decimal)	Contents (Hex)
0, 1	LL , a 2-byte length field (the length of the structure including LL)
2 through 6	D0 46 05 63 06
7 through 10	DATBLK , a 4-byte field, the data block number being sent
11, 12	C0 80
13, 14	LD , a 2-byte length field (5 + length of the data)
15 through n	DATA the data being sent

X'D0' Structured Fields, PC to Host

Format of the Unsuccessful Transmission Response for Set Cursor

To indicate an unsuccessful transmission of the Set Cursor X'D0' structured field request, the 3270 PC application program uses the WRITE-SF service to send a message to the host in the following format:

Byte (Decimal)	Contents (Hex)
0 through 7	00 00 09 00 00 00 00 00
8 through 16	00 09 D0 45 08 69 04 xx 00

Bytes 0 through 7 are the buffer header. Bytes 2 and 3 are the length of the structured field message that begins in byte 8. The unsuccessful transmission response **must** be coded as shown above.

xx is one of the following:

- 02 Arrival sequence not allowed
- 5D Unsupported type
- 60 Command syntax error
- 62 Parameter is missing
- 63 Parameter not supported
- 65 Parameter value not supported
- 6E Data element missing
- 70 Record length = 0
- 71 Invalid flag value

When the Workstation Program receives the response from the 3270 PC application program, it sends it to the host in the following format:

Byte (Decimal)	Contents (Hex)
0 through 8	00 09 D0 45 08 69 04 xx 00

xx is one of the values listed above.

Format of the Unsuccessful Transmission Response for Get

To indicate an unsuccessful transmission of the Get X'D0' structured field request, the 3270 PC application program uses the WRITE-SF service to send a message to the host in the following format:

Byte (Decimal)	Contents (Hex)
0 through 7	00 00 09 00 00 00 00 00
8 through 16	00 09 D0 46 08 69 04 xx 00

Bytes 0 through 7 are the buffer header. Bytes 2 and 3 are the length of the structured field message that begins in byte 8. The unsuccessful transmission response **must** be coded as shown above.

xx is one of the following:

- 02 Arrival sequence not allowed
- 22 Get past end of file
- 3E Operation not authorized
- 5D Unsupported type
- 60 Command syntax error
- 62 Parameter is missing
- 63 Parameter not supported
- 65 Parameter value not supported
- 71 Invalid flag value

When the Workstation Program receives the response from the 3270 PC application program, it sends it to the host in the following format:

Byte (Decimal)	Contents (Hex)
0 through 8	00 09 D0 46 08 69 04 xx 00

xx is one of the values listed above.

X'D0' Structured Fields, PC to Host

The Close X'D0' Structured Field

The Close X'D0' structured field request performs the logical termination of a connection between a file on the host system and a previously opened file on the personal computer system.

Format of the Host Close Request

The buffer sent by the host to the 3270 PC application program for the Close X'D0' structured field request must be formatted as follows:

Byte (Decimal)	Contents (Hex)
0 through 4	00 05 D0 41 12

*Note: The host Close request **must** be coded as shown above.*

When the Workstation Program receives the request from the host, it puts it into the buffer defined by the DEF-BUF service. The 3270 PC application program uses the READ-SF and GET-COMP services to receive the data in the following format:

Byte (Decimal)	Contents (Hex)
0 through 7	00 00 05 00 00 00 C0 00
8 through 12	00 05 D0 41 12

Bytes 0 through 7 are the buffer header.

Format of the Successful Transmission Response

To indicate a successful transmission of the Close X'D0' structured field request, the 3270 PC application program uses the WRITE-SF service to send a message to the host in the following format:

Byte (Decimal)	Contents (Hex)
0 through 7	00 00 05 00 00 00 00 00
8 through 12	00 05 D0 41 09

Bytes 0 through 7 are the buffer header. Bytes 2 and 3 are the length of the structured field message that begins in byte 8. The successful transmission response **must** be coded as shown above.

X'D0' Structured Fields, PC to Host

When the Workstation Program receives the response from the 3270 PC application program, it sends it to the host in the following format:

Byte (Decimal)	Contents (Hex)
0 through 4	00 05 D0 41 09

Format of the Unsuccessful Transmission Response

To indicate an unsuccessful transmission of the Close X'D0' structured field request, the 3270 PC application program uses the WRITE-SF service to send a message to the host in the following format:

Byte (Decimal)	Contents (Hex)
0 through 7	00 00 09 00 00 00 00 00
8 through 16	00 09 D0 41 08 69 04 xx 00

Bytes 0 through 7 are the buffer header. Bytes 2 and 3 are the length of the structured field message that begins in byte 8. The unsuccessful transmission response **must** be coded as shown above.

xx is one of the following:

- 03 Close of an unopened file
- 5D Unsupported type
- 60 Command syntax error
- 71 Invalid format

When the Workstation Program receives the response from the 3270 PC application program, it sends it to the host in the following format:

Byte (Decimal)	Contents (Hex)
0 through 8	00 09 D0 41 08 69 04 xx 00

xx is one of the values listed above.

Appendix C. Using Command Procedures for Save and Restore and for File Transfer

Introduction	C-2
Command Procedures for Save and Restore	C-2
Creating an AUTOEXEC.Bat File	C-3
Programmed Command Procedures	C-4
File Transfer Command Procedures	C-6

Introduction

Any of the Save and Restore commands, Send and Receive commands, IBM PC DOS commands, or user-written commands (see the IBM Personal Computer *Disk Operating System* manual) can be stored as records in special DOS files called **batch files**, whose file extension is **.BAT**. Each record in the file is a command, and the sequence of such commands is called a **command procedure**. Records stored in such a **filename.BAT** file may include any of the commands mentioned above.

The commands stored in a **filename.BAT** file can be invoked by typing

filename

in the personal computer window.

This appendix shows you how to create various kinds of command procedures for the Save, Restore, Send, and Receive commands.

Command Procedures for Save and Restore

Suppose you want to set up your IBM 3270 Personal Computer periodically for a particular application and need to:

- Restore a set of screen profiles from APPL1.SCR
- Restore a set of autokey recordings from APPL1.REC
- Restore a set of notepads from APPL1.NOT.

Instead of entering three individual Restore commands and having to remember three user file names, you can set up a command procedure, such as CONFIG1.BAT, that contains the three commands. Then you need only enter a single word to cause the three commands to be executed. This is illustrated below:

A >

CONFIG1 —————> CONFIG1.BAT FILE

INDRSTR SCREEN APPL1.SCR
INDRSTR RECORD APPL1.REC
INDRSTR NOTEPAD APPL1.NOT

Creating an AUTOEXEC.Bat File

Besides using command procedures to prepare the 3270 Personal Computer quickly for a particular application, most users will have a special command procedure named **AUTOEXEC.BAT**. This file initially contains a statement that the system supplies at the conclusion of the customization task. Do not **alter** or **remove** this name. You may include other desired commands as long as they **follow** the statement that the system supplies.

If you build such a file, it is invoked **automatically** any time DOS is initialized, causing the commands in AUTOEXEC.BAT to be executed. By putting the appropriate Restore commands into the AUTOEXEC.BAT file with other desired commands, you can set up the 3270 Personal Computer with standard screen profiles, autokey recordings, and notepads whenever DOS is initialized.

DOS initialization occurs automatically whenever the customized system diskette is loaded and initialized. Also, when the personal computer window is active and in application mode, you can reinitialize DOS at any time by pressing the following key sequence:

Ctrl + Alt + Del (press these keys at the same time)

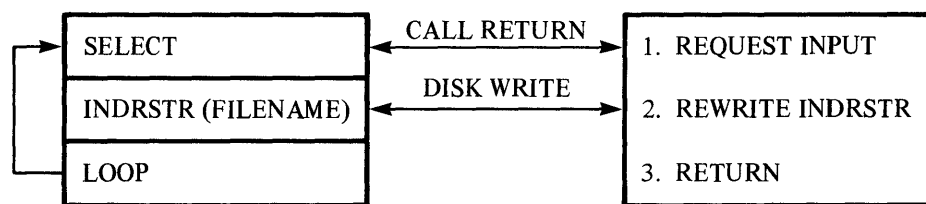
AUTOEXEC.BAT or other command procedure files can be created by using the EDLIN editor supplied with DOS (see the IBM Personal Computer *Disk Operating System* manual) or any of the other file editors available for the IBM Personal Computer.

Programmed Command Procedures

A special programmed command procedure can be created to quickly recall a series of notepads previously saved. This technique requires that the user write a DOS command program (in BASIC or another language) to interface with the operator. The program can prompt the operator to press certain keys to page forward or backward through notepad screens, or ask for the "name" of a particular notepad screen to be restored. The operator program, after receiving input about which notepad to restore next, converts this input into an INDRSTR (Restore) command record with the correct notepad file name. The operator program then writes this new INDRSTR record to the disk to overlay an existing INDRSTR command record contained in the command procedure file. This file also contains a command that invokes the operator program itself. If a Loop command is put into this file as well, a programmed command procedure, controlled by operator input, is created. This is illustrated below:

A >

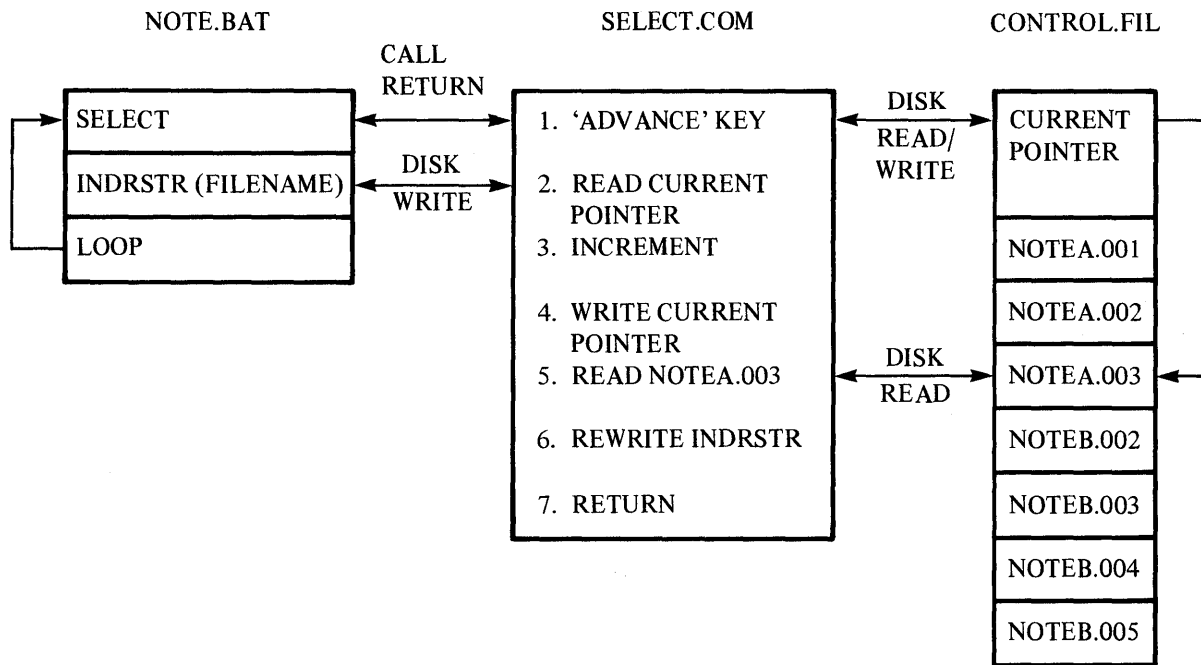
NOTE → NOTE.BAT



When NOTE.BAT is invoked, it first calls the SELECT.COM operator program. The operator program requests input that indicates which notepad record is to be restored next. This input is converted into an INDRSTR command record with the correct notepad file name. This record is written into the NOTE.BAT file to overlay the existing INDRSTR record. The SELECT.COM program then returns, causing the next command, INDRSTR, which was just created, to be executed. When this command has finished restoring the notepad, the next command, LOOP, is executed, causing the SELECT command to be invoked again. This starts the cycle over again, with new operator input into the SELECT.COM operator program.

Programmed Command Procedures

A related series of notepads can be created and their file names stored in a control file along with a "current notepad" pointer. This control file can be accessed by the operator program to step through the related notepads, using the pointer. Single keys can then be used to cause the operator program to step forward or backward through the related notepad file names to quickly restore them. The notepads can be used for operator viewing or, for example, as a source for copying saved data into 3270 edit screens for document creation. The following illustrates the control file concept:



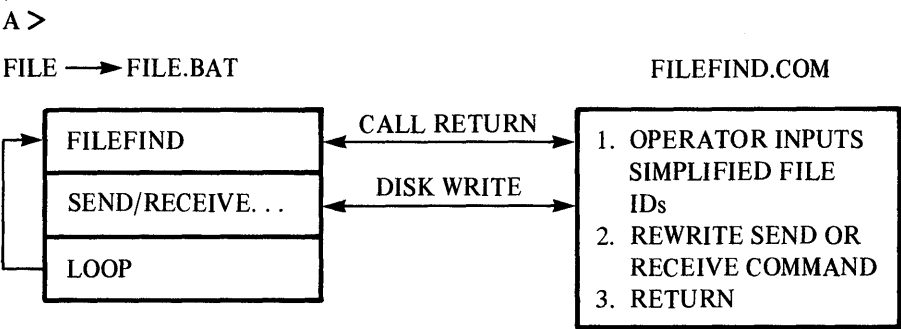
The SELECT.COM program reads an operator keystroke indicating advance to the next related notepad. The current pointer in the control file is read and updated. The file name of the notepad being pointed to, NOTEA.003, is read in from the control file and used to build a new INDRSTR command record. This record is written into the NOTE.BAT file. SELECT.COM returns, and NOTEA.003 is restored by the INDRSTR command for operator viewing or other uses.

Many variations of the illustrated programmed command procedure can be developed to allow program control, not only of notepad Save and Restore, but also of screen profiles or autokey Save and Restore. In addition, similar techniques can be used to control file transfer commands (see the next section), IBM Personal Computer DOS commands, or other user-written commands.

File Transfer Command Procedures

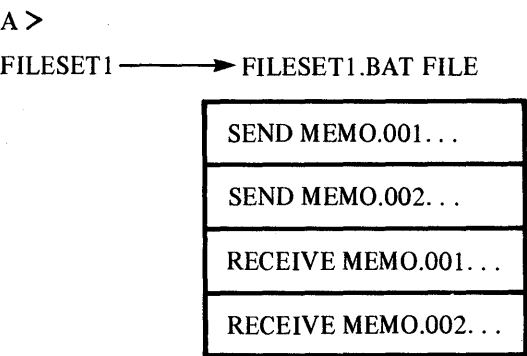
The file transfer commands can be entered directly from the keyboard or invoked indirectly by use of standard or programmed command procedures. Such command procedures can relieve the operator of remembering the complex parameters required when directly entering Send or Receive commands.

For example, you could write a general programmed command procedure. An operator interface program could prompt for simplified filenames and convert them into the required parameters for multiple Send or Receive operations. This is illustrated below:



Refer to “Programmed Command Procedures” in this appendix for a more detailed discussion of this technique.

Another option is to set up fixed command procedures that transfer multiple files to and from the host by typing in a single command, as shown below:



Note: To use the file transfer functions, the IBM host-supported file transfer program, Program No. 5664-281 for VM/CMS or 5665-311 for TSO, must be installed at your host site. You can verify that you have the host file transfer program by checking the host for the presence of the file named IND\$FILE MODULE.

Appendix D. Technical Notes

Introduction	D-2
3270 Limitations	D-2
3270 Data Stream Functions	D-3
Interface Codes	D-3
Attributes	D-6
Field Attributes	D-6
Extended Field Attributes and Character Attributes	D-7
Commands	D-8
3270 Data Stream Commands	D-8
Non-SNA Channel Commands	D-8
Write Control Character	D-9
3270 Data Stream Orders	D-11
Outbound 3270 Data Stream Structured Fields	D-12
Set Reply Mode Structured Field Format	D-13
Erase/Reset Structured Field Format	D-14
Outbound 3270DS Structured Field Format	D-14
Read Partition Structured Field Format	D-15
Inbound 3270 Data Stream	D-16
Inbound 3270 Data Stream for Partition 0	D-17
Inbound Structured Fields	D-21
Query Reply Structured Field	D-21
Transmission of Buffer Addresses	D-31
Buffer Addresses	D-31
Transmission of a Buffer Address	D-31
Transmission of a Buffer Address in 16-Bit Address Mode	D-32
Transmission of a Buffer Address in 12/14-Bit Address Mode	D-32
Changes or Limitations to the Personal Computer Session	D-34
Non-3270 PC Hardware Restrictions	D-34
Personal Computer Physical Cursor	D-35
Personal Computer Print Spooling	D-35
Control Unit Communication Session Termination	D-36
IBM 3270 Personal Computer Failure	D-36
Color Limitations	D-36
Notes on All-Points-Addressable Graphics	D-37
Using the Full-Screen APA Mode	D-37
Changing the Cursor Size or Position	D-38
Personal Computer Session Screen Size	D-38

Introduction

This appendix is for anyone who needs technical information about

- 3270 limitations
- 3270 data stream functions
- Personal computer application mode operation.

3270 Limitations

The following 3270 capabilities are limited or are not available with the IBM 3270 Personal Computer:

- The magnetic slot reader, magnetic hand scanner, and selector light pen are not available as attachments.
- The functions of port 0 of the 3270 control unit attachment that are available in control unit terminal (CUT) mode are Load Print Matrix and Test key. Port 0 of the control unit attachment is not available in distributed function terminal (DFT) mode.
- The binary synchronous host copy command is not available.
- The security keylock is not available.
- Explicit partition is not supported on the DFT session.
- The control unit Entry Assist feature is supported in CUT mode only.
- X.25 and X.21 keystroke support is available in CUT mode only.
- The APL character set is not available.
- Encryption/decryption is available for control unit terminals (CUT mode) only.
- The 3270 graphic escape is not supported.
- The diagnostic reset dump to the control unit is supported as a reset only; no dump is generated.
- The response time monitor is supported on the DFT session.

3270 Data Stream Functions

This section describes the 3270 data stream functions supported by the IBM 3270 Personal Computer. For general information on *all* the 3270 data stream functions, refer to the *IBM 3270 Information Display System: Data Stream Programmer's Reference*.

In this section, those functions of the 3270 data stream that are supported by the 3270 Personal Computer are listed, and any information unique to 3270 Personal Computer is provided. For those familiar with the 3270 data stream, this information should be sufficient for most purposes.

Interface Codes

Data commands and orders transmitted between the 3270 Personal Computer and the host system are in the form of extended binary-coded decimal interchange code (EBCDIC) interface codes. Figure D-1 shows the interface codes. Figure D-2 lists the EBCDIC control character I/O codes. Refer to *IBM 3270 Information Display System: Character Set Reference*, for further details.

		00				01				10				11				Bits 0,1
		00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11	2,3
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Hex 0
0000	0	NUL				SP	&	-						()	\	0	
0001	1		SBA				/			a	j	~		A	J		1	
0010	2		EUA							b	k	s		B	K	S	2	
0011	3		IC							c	l	t		C	L	T	3	
0100	4									d	m	u		D	M	U	4	
0101	5	PT	NL							e	n	v		E	N	V	5	
0110	6									f	o	w		F	O	W	6	
0111	7									g	p	x		G	P	X	7	
1000	8	GE		SA						h	q	y		H	Q	Y	8	
1001	9		EM	SFE						~	i	r	z		I	R	Z	9
1010	A					¢	!	!	:									
1011	B					.	\$.	#									
1100	C	FF	DUP	MF	RA	<	*	%	@									
1101	D	CR	SF			()	-	'									
1110	E		FM			+	;	>	=									
1111	F				SUB		⌋	?	"									EO

Figure D-1 (Part 1 of 2). United States EBCDIC I/O Interface Code

3270 Data Stream Functions - Interface Codes

Notes:

1. Codes X'00' through X'1F' and X'FF' are control codes, and codes X'40' through X'FE' are character codes. All blank squares in this chart represent undefined codes. All defined character codes are enclosed by heavy lines. Undefined control codes are rejected with a negative response (SNA) or an OP CHECK (non-SNA). If the extended attribute buffer (EAB) is enabled, all undefined character codes are accepted, stored, and returned without change on a subsequent read operation. If the EAB is disabled, character codes X'CE', X'CF', X'DD', X'DE', X'EE', X'EF', and X'FE' are stored, displayed, and returned as the character (X'60'), and all other undefined character codes are accepted, stored, and returned without change. IBM reserves the right to change at any time the character displayed or printed and the I/O interface code returned for an undefined character code.
2. CR, NL, EM, and FF control characters are displayed and printed as spaces. If extended attributes are not enabled, the DUP and FM control characters are respectively displayed as * and ; in dual-case mode. If extended attributes are enabled, DUP and FM are always displayed as * and ; respectively. DUP and FM are always printed as * and ; respectively.
3. Bits 0 and 1 are assigned for the following characters: AID, attribute, write control (WCC), device address, buffer address, sense, and status. Bits 0 and 1 are assigned so that each character can be represented by a graphic character within the solid outlined areas of the chart. See Figure D-2.

Figure D-1 (Part 2 of 2). United States EBCDIC I/O Interface Code

3270 Data Stream Functions - Interface Codes

Bits 2 – 7	Graphic	EBCDIC (Hex)	Bits 2 – 7	Graphic	EBCDIC (Hex)
00 0000	SP	40	10 0000	.	60
00 0001	A	C1	10 0001	/	61
00 0010	B	C2	10 0010	S	E2
00 0011	C	C3	10 0011	T	E3
00 0100	D	C4	10 0100	U	E4
00 0101	E	C5	10 0101	V	E5
00 0110	F	C6	10 0110	W	E6
00 0111	G	C7	10 0111	X	E7
00 1000	H	C8	10 1000	Y	E8
00 1001	I	C9	10 1001	Z	E9
00 1010	┐	4A	10 1010		6A
00 1011		4B	10 1011	,	6B
00 1100	<	4C	10 1100	┐	6C
00 1101	(4D	10 1101		6D
00 1110	+	4E	10 1110	>	6E
00 1111		4F	10 1111	?	6F
01 0000	&	50	11 0000	0	F0
01 0001	J	D1	11 0001	1	F1
01 0010	K	D2	11 0010	2	F2
01 0011	L	D3	11 0011	3	F3
01 0100	M	D4	11 0100	4	F4
01 0101	N	D5	11 0101	5	F5
01 0110	O	D6	11 0110	6	F6
01 0111	P	D7	11 0111	7	F7
01 1000	Q	D8	11 1000	8	F8
01 1001	R	D9	11 1001	9	F9
01 1010		5A	11 1010	:	7A
01 1011	┐	5B	11 1011	#	7B
01 1100	*	5C	11 1100	┐	7C
01 1101)	5D	11 1101	'	7D
01 1110	;	5E	11 1110	=	7E
01 1111	┐	5F	11 1111	"	7F

Notes:

1. The characters in Figure D-2 are used as attribute, write control character (WCC), attention identifier (AID), CU and device address, and buffer address.
2. To use this table to determine the hexadecimal code transmitted for an address or control character, first determine the values of bits 2–7. Select this bit configuration from the Bits 2–7 column. The hexadecimal code that will be transmitted is to the right of the bit configuration.
3. Use this table also to determine equivalent EBCDIC hex codes and their associated graphic characters. Graphic characters for the United States I/O interface codes are shown. Graphic characters might differ for particular World Trade I/O interface codes. For possible graphic differences when these codes are used, refer to IBM 3270 Information Display System: Character Set Reference.

Figure D-2. EBCDIC Control Character I/O Codes

3270 Data Stream Functions - Attributes

Attributes

The IBM 3270 Personal Computer display stations support field attributes, extended field attributes, and character attributes.

Field Attributes

Figure D-3 shows the bit positions in the field attribute byte. Figure D-4 shows the bit assignments for the field attributes supported by the IBM 3270 Personal Computer.

X	1	U/P	A/N	D/SPD	0	MDT
0	1	2	3	4 5	6	7

Figure D-3. Field Attribute Byte Bit Positions

EBCDIC Bit	Field Characteristics
0	Value determined by contents of bits 2–7. See Figure D-2 for hexadecimal values.
1	Always 1
2	0 = Unprotected 1 = Protected (see Note)
3	0 = Alphanumeric 1 = Numeric (if numeric lock capability is activated, causes automatic numeric shift of keyboard) (see Note)
4, 5	00 = Display not detectable by CrSel key 01 = Display detectable by CrSel key 10 = Intensified display detectable by CrSel key 11 = Nondisplay, nonprint, nondetectable
6	Reserved – Always 0
7	Modified data tag (MDT); identifies modified fields during Read Modified command operation: 0 = Field has not been modified 1 = Field has been modified by the operator. Can also be set by program in data stream.

Note: Bits 2 and 3 equal to 11 causes an automatic skip.

Figure D-4. Field Attribute Character Bit Assignments

Extended Field Attributes and Character Attributes

The IBM 3270 Personal Computer supports the attribute types and attribute values in Figures D-6 through D-8 as extended field attributes (EFAs) and as character attributes (CAs). All other attribute types and attribute values are rejected with a negative response (SNA) or an OP CHECK (non-SNA).

The attribute structure used for extended field attributes defines all characteristics with attribute type-value pairs, as shown in Figure D-5. Each attribute type has associated with it a set of attribute values.

Attribute type	Attribute value
----------------	-----------------

Figure D-5. The Structure of an Attribute Pair

Attribute Type (Hex)	Attribute Value (Hex)	EFA	CA
41	00 F1 F2 F4	Default Blink Reverse Video Underscore	Default (to EFA) Blink Reverse Video Underscore

Figure D-6. Attribute Type X'41' - Extended Highlighting

Attribute Type (Hex)	Attribute Value (Hex)	EFA	CA
42	00 F1 F2 F3 F4 F5 F6 F7	Default Blue Red Pink Green Turquoise Yellow White	Default (to EFA) Blue Red Pink Green Turquoise Yellow White

Figure D-7. Attribute Type X'42' - Color

Attribute Type (Hex)	Attribute Value (Hex)	EFA	CA	LCID Default
43	00 F1 F2 F3 F4 F5 F6 F7	Default Reserved PSA PSB PSC PSD PSE PSF	Default (to EFA) PSA PSB PSC PSD PSE PSF	 FF FF FF FF FF FF

Figure D-8. Attribute Type X'43' - Character Set Selection

3270 Data Stream Functions - Commands

Commands

For a description of the functions of the following commands, see the *IBM 3270 Information Display System: 3274 Control Unit Description and Programmer's Guide*, GA23-0061.

3270 Data Stream Commands

The control unit internally translates the non-SNA channel command codes into the SNA/BSC form before transmitting this information to the 3270 Personal Computer. Figure D-9 shows these commands.

Command	Mnemonic	SNA/BSC	Non-SNA Channel	Graphic
Write	WRITE	F1	01	1
Erase/Write	EW	F5	05	5
Erase/Write Alternate	EWA	7E	0D	=
Write Structured Field	WSF	F3	11	NA
Erase All Unprotected	EAU	6F	0F	?
Read Buffer	RB	F2	02	2
Read Modified	RM	F6	06	6
Read Modified All	RMA	6E	NA	>

Figure D-9. 3270 Data Stream Commands

Non-SNA Channel Commands

Although not part of the 3270 data stream, the channel commands in Figure D-10 are valid for non-SNA channel-attached 3270 Personal Computer display stations.

The 3270 Control Unit internally decodes the following commands and parses the appropriate Write, Read Modified, or Read Buffer SNA/BSC command code values to the IBM 3270 Personal Computer.

Command	Mnemonic	Non-SNA Channel Command Code EBCDIC (Hex)
No Operation		03
Sense		04
Select Read Modified	Select RM	0B
Select Read Buffer	Select RB	1B
Select Read Modified from Position	Select RMP	2B
Select Read Buffer from Position	Select RBP	3B
Select Write	Select WRT	4B

Figure D-10. Non-SNA Channel Commands

Write Control Character

The write control character (WCC) bits have the following significance for the 3270 Personal Computer:

WCC

Bit	Explanation
-----	-------------

0	No function.
1	No function for the WRT command. Reset function, if set to 1, for EW and EWA commands. See Figure D-11 for the effects of the reset function when the 3270 Personal Computer is in implicit partition state.
2, 3	Reserved.
4	Start printer (SNA only). When set to 1, initiates a local-copy operation at the completion of the write operation. If no printer is available, a negative response X'0801', X'082E', or X'082F' is returned.
5	Sound alarm. When set to 1, causes the audible alarm to sound.
6	Keyboard restore. When set to 1, causes keyboard operation to be restored (by resetting the system lock or WAIT indicator) and resets the AID byte to X'60'.
7	Reset MDT bits in field attributes (WRT command only). When set to 1, causes MDT bits to be reset to 0 in all field attribute bytes in the specified partition, before any orders or data characters are processed.

Write Control Character

Reset Condition	Implicit Partition State
1. WCC following Erase/Write or an Erase/Write Alternate command. a. WCC = Reset. b. WCC = No Reset.	Execute the command; reset the inbound reply mode to field. Execute the command.
2. WCC following a Write command. a. WCC = Reset or no reset.	Execute the command.
3. WCC in outbound 3270DS header, and the function is Erase/Write or Erase/Write Alternate a. WCC = Reset. b. WCC = No reset.	If the PID equals 0, execute the function (except for screen-size changes) and reset the inbound reply mode to field. If the PID does not equal 0, reject with a negative response If the PID equals 0, execute the function. If the PID does not equal 0, reject with a negative response.
4. WCC in outbound 3270DS header, and the function is Write. a. WCC = Reset or no reset.	Execute the function if the PID equals 0; otherwise, reject with a negative response.

Figure D-11. Write Control Character Reset Actions

3270 Data Stream Orders

The IBM 3270 Personal Computer supports the 3270 data stream orders shown in Figure D-12.

Order	Mnemonic	Order Code EBCDIC (Hex)
Start Field	SF	1D
Start Field Extended ¹	SFE	29
Set Buffer Address	SBA	11
Set Attribute ¹	SA	28
Modify Field ¹	MF	2C
Insert Cursor	IC	13
Program Tab	PT	05
Repeat to Address	RA	3C
Erase Unprotected to Address	EUA	12

¹ The SFE, SA, and MF orders are valid only when the extended attribute buffer (EAB) is enabled for the logical terminal to which the order is directed. When the EAB is not enabled, these orders are rejected with a negative response (SNA) or an OP CHECK (non-SNA).

Figure D-12. 3270 Data Stream Orders

Outbound 3270 Data Stream Structured Fields

Outbound 3270 Data Stream Structured Fields

The Write Structured Field (WSF) command (X'F3') is used by the host program to transmit the following outbound structured fields to the addressed logical terminal.

Outbound Structured Field	ID Code	Type
Set Reply Mode	X'09'	
Erase/Reset	X'03'	
Outbound 3270DS	X'40'	
Write		X'F1'
Erase/Write		X'F5'
Erase/Write Alternate		X'7E'
Erase All Unprotected		X'6F'
Rcad Partition	X'01'	
Read Buffer		X'F2'
Read Modified		X'F6'
Read Modified All (SNA only)		X'6E'
Query		X'02'
Query List		X'03'
Load Programmed Symbols	X'06'	
Destination/Origin	X'0F'	X'02'

All other structured-field ID codes are rejected with a negative response (SNA) or an OP CHECK (non-SNA).

Set Reply Mode Structured Field Format

The set reply mode structured field specifies the reply mode required in all subsequent inbound data streams. The specified reply mode remains in effect until it is changed by one of the following:

- Another set reply mode structured field, specifying a different reply mode
- An erase/reset structured field
- An EW or EWA command or a structured field with the WCC bit 1 set to 1, causing a reset function.

Figure D-13 shows the contents and meanings for bytes 0 through 5 and 7.

Byte	Contents	Meaning
0, 1	X'0005' through X'0008'	Length of structured field in bytes
2	X'09'	ID code of set reply mode
3	PID	Partition identifier (must be X'00' for implicit partition state) X'00' – X'0F'
4	X'00' X'01' X'02'	Reply mode requested Field mode (default) Extended field mode Character mode
5	X'41'	Highlighting selection (Notes 1 and 3)
7	X'42'	Color selection (Notes 1, 2, and 3)

Notes:

1. *More than five bytes can be present only when character mode (X'02') is requested. Only when the length code is greater than X'0005' can the operator select from the keyboard the attribute value to be associated with the keyed data as shown in byte 5 above.*
2. *If color selection (X'42') is specified, the structured field is accepted, but the operator is not allowed to select color attributes from the keyboard.*
3. *X'41' and X'42' can appear in any order, not necessarily as shown in bytes 5 and 7 above.*

Figure D-13. Set Reply Mode Structured Field Format

Outbound 3270 Data Stream Structured Fields

Erase/Reset Structured Field Format

The erase/reset structured field creates an implicit partition 0 of default or alternate size, as specified in the structured field. Inbound-reply mode is reset. Figure D-14 shows the contents and meanings for bytes 0 through 3.

Byte	Contents	Meaning
0, 1	X'0004'	Length of structured field in bytes
2	X'03'	Erase/Reset ID code
3	Size X'00' X'80'	Size of usable area: Default size Alternate size

Figure D-14. Erase/Reset Structured Field Format

If byte 3 contains any value other than X'00' or X'80', the structured field is rejected with a negative response (SNA) or an OP CHECK (non-SNA).

Outbound 3270DS Structured Field Format

The outbound 3270DS structured field is used to direct the write commands to a specified partition. The write commands are:

- Write
- Erase/Write
- Erase/Write Alternate
- Erase All Unprotected.

Figure D-15 lists the contents and meanings for this structured field.

Byte	Contents	Meaning
0, 1	Length	Length of structured field in bytes
2	X'40'	Outbound 3270DS ID code
3	PID	Partition identifier
4	Wrt Cmd X'F1' X'F5' X'7E' X'6F'	Write-type command code Write Erase/Write Erase/Write Alternate Erase All Unprotected

Figure D-15. Outbound 3270DS Structured Field Format

The remaining bytes of the structured field are the same as for the specified Write command.

WCC bit 1 set to 1 does not reset the IBM 3270 Personal Computer to implicit partition state, as it does for the nonstructured-field command, but resets the reply mode of the inbound partition to field mode. WCC bit 4 set to 1 specifies Start Print and must be in the last structured field of the string of structured fields.

Read Partition Structured Field Format

Figure D-16 describes the contents and meanings for the read partition structured field.

Byte	Content	Meaning
0, 1	X'0000' X'0005'	Length of structured field in bytes
2	X'01'	Read-partition ID code
3	PID	Partition identifier X'00' through X'0F': read operations X'FF' (physical terminal): query operations
4	Type X'F6' X'6E' X'F2' X'02' X'03'	The type of operation to be performed: Read Modified (RM) Read Modified All (RMA) Read Buffer (RB) Query Query List
5	REQTYP	Request Type – present only for Type = X'03' (Query List):
Bits: 0 – 1	B'00' B'01' B'10' B'11'	Only list (bytes 6 through n) Query Equivalent + list All Query Replies Reserved
2 – 7		Reserved

Figure D-16. Read Partition Structured Field Format

For Query (Type = X'02'), the structured field ends after byte 4.

For Query List (Type = X'03'), byte 5 is a flag byte called REQTYP, described below. Bytes 6 through n contain the type codes of the Query Reply (or Replies) being requested.

- REQTYP – Request Type (present only if TYPE = X'03'):
 - B'00' indicates the only Query Replies being requested are those specified in bytes 6 through n. If the value is B'00' but no list is present (count field is valid), a Null Query Reply is returned.
 - B'01' indicates all the Query Replies that would be sent in reply to a Query are sent, in addition to those (if any) that are specified in the list (bytes 6 through n). No duplicate Query Replies are sent. For example, if the list requests a Query Reply that would be sent anyway, because of the B'01' flag, the Query Reply is sent only once.

Inbound 3270 Data Stream

- B'10' indicates that all the query replies that are supported are sent. If a list is present (bytes 6 through n), B'10' overrides the list (that is, the list is ignored).

The same type code may appear more than once in the list (bytes 6 through n). However, only one Query Reply will be returned for a particular type code value, regardless of how many times it appears in the list.

All type code values are valid in the list. Those type codes not supported are ignored. However, if none of the type codes in the list are supported, a null Query Reply is returned.

Restrictions:

1. The Read Partition structured field must be the last structured field in the current outbound transmission.
2. The IBM 3270 Personal Computer must be in normal-read state, and the PID must specify a valid partition.

Any of the following conditions causes an error and the return of an appropriate sense code in SNA sessions or OP-CHECK in non-SNA sessions:

- The IBM 3270 Personal Computer display is in normal-read state, and the identified partition does not exist: sense code X'1005'.
- The command code is invalid: sense code X'1003'.
- The PID value is invalid: sense code X'1005'.
- The read-partition structured field is not the last structured field to be sent in the current outbound transmission: sense code X'1005'.
- The IBM 3270 Personal Computer is in retry state: sense code X'0871'.
- The transmission is sent with end bracket: sense code X'0829'.
- The transmission does not specify "Change Direction": sense code X'0829'.

Inbound 3270 Data Stream

All inbound 3270 data streams are preceded by an attention identifier (AID) byte that identifies the cause of the inbound transmissions. Figure D-17 lists the possible causes of an inbound 3270 data stream, the associated AID values, and the resulting operations when the IBM 3270 Personal Computer is in normal-read state or in retry state.

Cause	AID	Operation (Normal-Read State)	Operation (Retry State)
Clear key	X'6D'	Short read (implicit partition 0)	
PA1 key PA2 key PA3 key	X'6C' X'6E' X'6B'	Short read Short read Short read	
PF1 – PF9 keys PF10 – PF12 keys PF13 – PF21 keys PF22 – PF24 keys Cursor-select field with & designator or Enter key	X'F1'-X'F9' X'7A'-X'7C' X'C1'-X'C9' X'4A'-X'4C' X'7D'	Read modified (addresses and data of all modified fields – nulls suppressed)	
Cursor-select field with null or space designator	X'7E'	Read modified (addresses of modified fields)	
Read Modified command	X'60'	Read modified	Retry last AID action from inbound partition
Read Modified All command	X'60'	Read modified all	Retry read modified all from inbound partition
Read Buffer command	X'60'	Read buffer	Retry read buffer from inbound partition
Read Partition- Modified SF	X'61'	Read modified	Reject
Read Partition-Read Modified ALL SF	X'61'	Read modified all	Negative response (SNA) or OP CHECK (non-SNA)
Read Partition-Read Buffer SF	X'61'	Read buffer	
Read Partition-Query SF	X'88'	Query replies	

Figure D-17. Inbound 3270 Data Stream

Inbound 3270 Data Stream for Partition 0

Each inbound 3270 data stream for partition 0 is preceded by an AID code that identifies the cause of this inbound data stream. Figure D-17 lists each cause, the AID associated with that cause, and the resulting type of inbound operation. The types of inbound operations are:

- Short read
- Read modified
- Read modified all
- Read buffer.

Short Read Format: Figure D-18 shows that the short-read operation results in an inbound data stream consisting of only the AID byte.

Byte	Content	Meaning
0	AID X'6D' X'6C' X'6E' X'6B'	Attention identifier byte: Clear key PA1 key PA2 key PA3 key

Figure D-18. Short Read Format

Read Modified and Read Modified All Format: The read-modified operation and the read-modified-all (SNA only) operation can take place in a field, an extended field, or a character reply mode. Figure D-19 shows a general format for all three reply modes.

Byte	Contents	Meaning
0	AID	Attention identifier byte
1, 2	CCP	Hexadecimal address of the cursor (current cursor position)
3	X'11'	SBA order code
4, 5	X'—'	Hexadecimal address of the first character in the modified field (attribute address + 1)
6 through n	Data	Data from the addressed field but with null characters suppressed. See Notes 1 and 2.

Notes:

1. *In character reply mode, the string of data characters can also include Set Attribute orders to indicate changes in the character attribute value.*
2. *For the read-modified operation, with an AID of X'7E' (indicating cursor selection of a space or null designator character), only the addresses of modified fields are transmitted.*
3. *If none of the fields has been modified, the inbound data stream consists of bytes 0 through 2 only.*
4. *If the partition buffer is unformatted, the data stream consists of bytes 0 through 2 and bytes 6 through n (all data in the partition buffer, with nulls suppressed).*

Figure D-19. Read Modified and Read Modified All Format

The pattern of bytes 3 through n is repeated for all other modified (MDT bit set to 1) fields.

Read Buffer Format: The read-buffer operation can take place in field, extended field, or character reply mode. The read-buffer operation transmits the contents of all buffer locations in the presentation space, starting at the current cursor position.

Read Buffer Format in Field Reply Mode: Figure D-20 shows the read buffer format in field reply mode.

Byte	Contents	Meaning
0	AID	Attention identifier byte
1, 2	CCP	Hexadecimal address of the cursor (current cursor position)
3 through n	Data	Data characters (including null characters, but excluding Set Attribute orders), from the current cursor position to the start of the next field (see Note)
$n + 1$	X'1D'	Start Field order code
$n + 2$	X'—'	Field-attribute character
m	Data	All data characters to the start of the next field

Note: Set Attribute orders are defined in the IBM 3270 Information Display System 3274 Control Unit Description and Programmer's Guide.

Figure D-20. Read Buffer Format in Field Reply Mode

The pattern of bytes $n + 1$ through m is repeated for all other fields being transmitted.

Read Buffer Format in Extended Field and Character Mode: The read buffer format in extended field and character mode is shown in Figure D-21.

Inbound 3270 Data Stream

Byte	Contents	Meaning
0	AID	Attention identifier byte
1, 2	CCP	Hexadecimal address of the cursor (current cursor position)
3 through n	Data	Data characters (including null characters), from the current cursor position to the start of the next field.
n + 1	X'29'	Start Field Extended order code.
n + 2	ATTC X'C0' X'—' X'—' X'—'	Attribute count (X'00' through X'04'). The number of attribute specifications (byte pairs) that follow. Byte pairs are transmitted only for those extended field attributes with specified (nondefault) values. Field-attribute type Field-attribute type Extended-field-attribute type Extended-field-attribute value: includes highlighting, color, and PS. (See "Attributes" in Appendix F, "Presentation Space Considerations.")
m	Data	All data characters to the start of the next field.

Notes:

1. *Set Attribute orders are defined in the IBM 3270 Information Display System: 3274 Control Unit Description and Programmer's Guide, GA23-0061.*
2. *When in character mode, each string of data characters can include Set Attribute orders to indicate changes in the character attribute value. The format for these Set Attribute orders is the same as for outbound Set Attribute orders. In extended-field mode, no Set Attribute orders are included.*

Figure D-21. Read Buffer Format in Extended Field and Character Mode

The pattern of bytes **n** + 1 through **m** is repeated for all other fields being transmitted.

Inbound Structured Fields

The inbound structured field AID code (X'88') precedes and identifies the following inbound structured fields for transmission to the host program:

Query Reply Structured Field	ID Code	Type
Destination/Origin	X'0F'	X'02'
Inbound 3270DS	X'80'	
Query reply	X'81'	
Usable area		X'81'
Character sets		X'85'
Color		X'86'
Highlight		X'87'
Reply modes		X'88'
Implicit partition		X'A6'
DDM		X'95'
Auxiliary device		X'99'
Document Interchange Architecture		X'97'

Query Reply Structured Field

The logical terminal to which the read partition query function was addressed responds with the transmission of a series of structured fields indicating the field and character attributes, the screen or page size characteristics, the symbol sets, and the reply modes available on the logical terminal. Since each structured field contains its own unique identification, the order in which the fields are transmitted is not important. The query reply structured fields and their associated reply codes are as follows:

Query Reply Structured Field	ID Code	Type
Query reply	X'81'	
Usable area		X'81'
Character sets		X'85'
Color		X'86'
Highlight		X'87'
Reply modes		X'88'
Implicit partition		X'A6'
DDM		X'95'
Auxiliary device		X'99'
Document Interchange Architecture		X'97'

Inbound Structured Fields

Usable Area Query Structured Field Format: The usable area query reply structured field indicates to the host program the dimensions of the logical screen for the logical terminal. Figure D-22 describes the contents and meaning of the usable area query reply structured field.

Byte	Contents	Meaning
0, 1	Length X'0017'	Length of structured field in bytes, if partitions are not supported (0 partitions defined)
2	X'81'	ID code of query reply
3	X'81'	ID code of usable area reply
4	X'01'	12/14-bit addressing allowed
5	X'00'	Variable-character cells not supported (omit bytes 23 through 26) per logical-terminal definition table
6, 7	LSW	Width of usable screen for logical screen
8, 9	LSH	Height of usable screen for logical screen
10	X'01'	Units of measure (millimeters)
11 through 14	Xr	Horizontal pitch: the distance between points in the x-direction as a fraction measured in units 2-byte numerator 2-byte denominator
15 through 18	Yr	Vertical pitch: the distance between points in the y-direction as a fraction measured in units 2-byte numerator 2-byte denominator
19	HS 09	Horizontal character cell size
20	VS 0E	Vertical character cell size
21, 22	BUFFSZ	Character buffer size

Figure D-22. Usable Area Query Reply Structured Field Format

Character Sets Query Reply Structured Field Format: The character sets query reply structured field is transmitted to inform the host program what character sets are defined for the logical terminal. Figure D-23 describes the contents and meaning of the character sets query reply structured field.

Byte	Bit	Contents	Meaning
0, 1		Length	Length of structured field in bytes (13 + lengths of descriptors)
2		X'81'	ID code of query reply
3		X'85'	ID code of character sets reply
4		Flags	Based on logical-terminal definition table (EAB/PS) definition:
	0	0	Graphic escape not supported
	1	0	Reserved
	2	0 or 1	Load PS (EAB and PS) supported (0 = no; 1 = yes)
	3	0 or 1	Load PS extension (EAB and PS) supported (0 = no; 1 = yes)
	4	0	One size of character cell supported
	5	000	Reserved
	—		
	7		
5		X'00'	Reserved
6		X'09'	Default-character cell width
7		X'0E'	Default-character cell depth
8 – 11		X'6000 0000'	Supported Load PS format types; bit-encoded (if bit i = 1, then type is supported).
12		DL	Length of each descriptor

Figure D-23. Character Sets Query Reply Structured Field Format

Inbound Structured Fields

Character Set Descriptors: Figure D-24 describes the length of each descriptor.

Byte	Bit	Contents	Meaning
1		Set ID	Device-specific character set ID (RWS or ROS number)
2		Flags	
	0	Load B'0' B'1'	Loadable: Nonloadable character set Loadable character set
	1	Triple B'0' B'1'	Triple plane: Single-plane character set Triple-plane character set
	2	Reserved	Reserved – must be zero
	3	CB B'0' B'1'	Compare bit: LCID compare No LCID compare
	4 – 7	Reserved	Reserved
3		LCID	Local character set ID

Figure D-24. Character Set Descriptors

Reply Modes Query Reply Structured Field Format: The reply modes query reply structured field is transmitted, if EAB is defined, to inform the host program which reply modes are supported by the logical terminal. Figure D-25 lists the contents and meanings for bytes 0 through 6 of the structured field.

Byte	Contents	Meaning
0, 1	X'0007'	Length of structured field in bytes
2	X'81'	ID code of query reply
3	X'88'	ID code of reply modes reply
4	X'00'	Field reply mode supported
5	X'01'	Extended-field reply mode supported
6	X'02'	Character reply mode supported

Figure D-25. Reply Modes Query Reply Structured Field Format

DDM Query Reply Structured Field Format: Figure D-26 lists the contents and meanings for bytes 0 through 11 of the structured field.

Byte	Contents	Meaning
0, 1	Length	Length of structured field in bytes
2	X'81'	ID code of query reply
3	X'95'	ID code of DDM reply
4	X'00'	File not available
5	X'00'	Reserved
6, 7	X'0800'	Maximum DDM bytes per transmission allowed inbound
8, 9	X'0800'	Maximum DDM bytes per transmission allowed outbound
10	X'01'	Number of subsets supported
11	X'01'	DDM subset identifier

Figure D-26. DDM Query Reply Structured Field Format

Auxiliary Device Query Reply Structured Field Format: Figure D-27 lists the contents and meanings of the bytes in the auxiliary device query reply structured field, which indicates direct access support of one or more auxiliary devices.

Byte	Contents	Meaning
0, 1	X'0006'	Length of this structured field in bytes
2	X'81'	ID code of query reply
3	X'99'	ID code of auxiliary device reply
4, 5	Reserved flags	Must be X'0000'

Figure D-27. Auxiliary Device Query Reply Structured Field Format

When one or more auxiliary devices is supported (3270 data stream work stations), this query reply is transmitted inbound to a Query List or to a Query. This query reply indicates support of:

- Destination/origin structured field
- Query list structured field
- One or more auxiliary devices.

Document Interchange Architecture Query Reply Structured Field

Format: The contents and meanings of the bytes in the document interchange architecture query reply structured field are shown in Figure D-28.

Byte	Contents	Meaning
0, 1	Length	Length of structured field in bytes
2	X'81'	ID code of query reply
3	X'97'	ID code of Document Interchange Architecture query reply
4, 5	Reserved flags	Must be X'0000'
6, 7	X'0800'	Maximum DIA bytes per transmission allowed inbound
8, 9	X'0800'	Maximum DIA bytes per transmission allowed outbound
10	X'01'	Number of 3-byte function set identifiers that follow
11 – 13	X'01000B'	DIAL function set identifier

Figure D-28. Document Interchange Architecture Query Reply Structured Field Format

Direct Access ID Self-Defining Parameter: Figure D-29 lists the contents and meanings of the bytes in the direct access ID self-defining parameter.

Byte	Content	Meaning
0	X'04'	Parameter length in bytes
1	X'01'	Direct access ID
2, 3	ID	Destination/origin identification

Figure D-29. Direct Access Self-Defining Parameter

Color Query Reply Structured Field Format: The color query reply structured field indicates to the host the color that will be displayed for each color attribute value. Figure D-30 describes the contents and meanings of the bytes in this structured field.

Byte	Contents	Meaning
0, 1	X'0016'	Length of this structured field in bytes
2	X'81'	ID code of query reply
3	X'86'	ID code of color reply
4	X'00'	Reserved
5	X'08'	Length of attribute list (number of color attribute pairs)
6	X'00'	Default color attribute
7	X'F4'	Green is the default device color.
8 through 21	X'F1F1F2F2 F3F3F4F4 F5F5F6F6 F7F7'	For color display, each color maps to itself.
8 through 21	X'F1F4F2F4 F3F4F4F4 F5F4F6F4 F7F4'	For monochrome display, each color maps to green.

Notes:

1. Bytes 6 and 7 are always used to indicate the IBM 3270 Personal Computer default color.
2. If "extended attributes" is not enabled by the logical-terminal definition table, the color query reply is not returned.

Figure D-30. Color Query Reply Structured Field Format

Highlight Query Reply Structured Field Format: The highlight query reply is transmitted inbound as a structured field with the format shown in Figure D-31.

Byte	Contents	Meaning
0, 1	X'000D'	Length of this structured field in bytes
2	X'81'	ID code of query reply
3	X'87'	ID code of highlight reply
4	X'04'	Number of byte pairs that follow
5	X'00'	Acceptable attribute value
6	X'F0'	Default-highlight action code
7	X'F1'	Accepted attribute value
8	X'F1'	Blink action code
9	X'F2'	Accepted attribute value
10	X'F2'	Reverse action code
11	X'F4'	Accepted attribute value
12	X'F4'	Underscore action code

Notes:

1. Bytes 5, 7, 9, and 11 of a highlight query reply indicate to a host the attribute values that the IBM 3270 Personal Computer:
 - a. Can accept in an outbound extended-highlight attribute byte
 - b. Will preserve and will return to the host in a subsequent inbound operation (unless changed by the operator)
2. Bytes 6, 8, 10, and 12 indicate to a host what action the IBM 3270 Personal Computer performs for each of those attribute values.
3. If "extended attributes" is not enabled by the logical-terminal definition table, the highlight query reply is not returned.

Figure D-31. Highlight Query Reply Structured Field Format

Implicit Partition Query Reply Structured Field Format: The implicit partition query reply structured field is always returned by the IBM 3270 Personal Computer logical terminal. It informs the host of the default and alternate sizes for the target logical terminal's implicit partition. It may also return the cell size in effect for the default and alternate screen sizes. Figure D-32 shows this structured field format.

Byte	Contents	Meaning
0, 1	Length	Length of structured field in bytes
2	X'81'	ID code of query reply
3	X'A6'	ID code of implicit partition reply
4, 5	X'0000'	Reserved

Figure D-32. Implicit Partition Query Reply Structured Field Format

Implicit Partition Default and Alternate Screen Size: For an SNA session, the default and alternate screen sizes returned in this reply are those established by the BIND for this logical terminal.

For a non-SNA session, the default and alternate screen sizes returned in this reply are those in effect for this logical terminal at the time the reply is generated.

Character Cell Dimensions: The dimensions of the character cells in effect for the default and alternate screen sizes are returned only when the character cell size for either the default or the alternate implicit partition is different from the cell size specified in the usable area query reply.

Inbound Structured Fields

Self-Defining Parameters: The implicit partition size parameter shown in Figure D-33 is always returned as part of the implicit partition query reply.

Byte	Contents	Meaning
0	Length	Length of parameter in bytes
1	X'01'	Parameter of screen size dimension
2	X'00'	Reserved
3, 4	WD	Width of default screen size in cells
5, 6	HD	Height of default screen size in cells
7, 8	WA	Width of alternate screen size in cells
9, 10	HA	Height of alternate screen size in cells

Notes:

1. *Bytes 3 and 4: For an SNA session, the width of the default implicit partition is the value established at logical-terminal bind time. For a non-SNA session, the width of the default implicit partition is 80.*
2. *Bytes 5 and 6: For an SNA session, the height of the default implicit partition is the value established at logical-terminal bind time. For a non-SNA session, the height of the default implicit partition is 24.*
3. *Bytes 7 through 10: For an SNA session, the width and the height of the alternate implicit partition are the values established at logical-terminal bind time. For a non-SNA session, the width and the height of the alternate implicit partition are the values defined in the logical-terminal definition table.*

Figure D-33. Self-Defining Parameters

Transmission of Buffer Addresses

The following text describes:

1. The relationship between a given row/column position in a logical terminal's presentation space and its address in the logical terminal's buffer storage.
2. How a buffer storage address is transmitted to and from the IBM 3270 Personal Computer in each of the two address modes.

Buffer Addresses

The relationship between any given row/column position in a logical terminal and its address in the logical terminal's buffer storage is expressed thus:

$$\text{Buffer address} = W \times (R-1) + (C-1)$$

Where:

W is the logical terminal width (number of columns).

R is the row number (counting from 1) of the position being addressed.

C is the column number (counting from 1) of the position being addressed.

Note: For column 1, row 1, the buffer address is 0.

Example of Buffer Address Calculation: Given a logical terminal width (*W*) of 50 columns, the buffer address of the character location in column 46 and row 21 is:

$$\text{Buffer Address} = (50 \times 20) + 45 = 1045$$

Transmission of a Buffer Address

A buffer address is always transmitted in a 2-byte field.

The way in which a buffer address is transmitted depends on the address mode for the logical terminal. For example, it may be transmitted in the two bytes after the X'11' byte in a Set Buffer Address order, or in the two bytes that specify the current cursor position in an inbound read operation.

A logical terminal has one of two address modes:

- 16-bit address mode
- 12/14-bit address mode.

Transmission of Buffer Addresses

Transmission of a Buffer Address in 16-Bit Address Mode

In 16-bit address mode, buffer addresses are transmitted outbound and inbound as 16-bit binary numbers.

Transmission of a Buffer Address in 12/14-Bit Address Mode

In 12/14-bit address mode, bits 0 and 1 of the first address byte have the following bit significance:

- 00 – 14-bit binary address follows
- 01 – 12-bit coded address follows
- 11 – 12-bit coded address follows

(In an outbound transmission, if bits 0 and 1 of the first byte are set to B'10', a sense code of X'1005' is returned.)

For outbound transmissions, the IBM 3270 Personal Computer uses these first two bits to determine whether the address in the transmission has 12 bits or 14 bits. For inbound transmissions, the size of the logical terminal determines whether the address in the transmission has 12 bits or 14 bits. If the logical terminal size is greater than 4096 bytes (excluding character attributes if an EAB device), 14-bit addresses are used. Otherwise, 12-bit addresses are used.

14-Bit Addresses: 14-bit buffer addresses are transmitted outbound and inbound with bits 0 and 1 of the first byte equal to B'00', followed by the buffer address expressed as a 14-bit binary number.

12-Bit Addresses: 12-bit buffer addresses are transmitted outbound and inbound with the following format:

First byte	Second byte
B B a a a a a a	B B a a a a a a

Where:

a represents the binary value of the address (12 bits), and *BB* is B'01' or B'11' (see text below).

The 12-bit address EBCDIC values for decimal addresses 0 through 10879 are derived as follows:

1. The 12-bit binary representation of the decimal address is split into two 6-bit groups.
2. The more significant 6-bit group occupies bits 2 through 7 of the first byte.
3. The less significant 6-bit group occupies bits 2 through 7 of the second byte.

4. According to the value of bits 2 through 7 in each byte, bits 0 and 1 of each byte are set to obtain the hexadecimal representations, as defined in Figure D-34. The resulting EBCDIC value represents a graphic character in the range X'40' through X'F9'.

Bits 2-7	EB*	Bits 2-7	EB*	Bits 2-7	EB*	Bits 2-7	EB*
00 0000	40	01 0000	50	10 0000	60	11 0000	F0
00 0001	C1	01 0001	D1	10 0001	61	11 0001	F1
00 0010	C2	01 0010	D2	10 0010	E2	11 0010	F2
00 0011	C3	01 0011	D3	10 0011	E3	11 0011	F3
00 0100	C4	01 0100	D4	10 0100	E4	11 0100	F4
00 0101	C5	01 0101	D5	10 0101	E5	11 0101	F5
00 0110	C6	01 0110	D6	10 0110	E6	11 0110	F6
00 0111	C7	01 0111	D7	10 0111	E7	11 0111	F7
00 1000	C8	01 1000	D8	10 1000	E8	11 1000	F8
00 1001	C9	01 1001	D9	10 1001	E9	11 1001	F9
00 1010	4A	01 1010	5A	10 1010	6A	11 1010	7A
00 1011	4B	01 1011	5B	10 1011	6B	11 1011	7B
00 1100	4C	01 1100	5C	10 1100	6C	11 1100	7C
00 1101	4D	01 1101	5D	10 1101	6D	11 1101	7D
00 1110	4E	01 1110	5E	10 1110	6E	11 1110	7E
00 1111	4F	01 1111	5F	10 1111	6F	11 1111	7F

* EB = EBCDIC

Note: The 12-bit address EBCDIC values for decimal addresses 0 through 3563 are listed in the IBM 3270 Buffer Address Codes.

Figure D-34. Hexadecimal Representations

Changes or Limitations to the Personal Computer Session

Following are the changes or limitations to the PC session in application mode when operating under the workstation program. Note that if you are running a PC application program through the IBM 3270 Workstation Program, there may be unpredictable results in some cases.

Non-3270 PC Hardware Restrictions

The following restrictions apply to non-3270 PC hardware (XT and AT). Failure to follow these guidelines on the use of non-3270 PC hardware could result in system failure.

- For Uni-DOS on non-3270 PC hardware, an application is assumed to be ill-behaved. The application will be suspended when:
 - It is not the top or active window
 - The WS Ctrl key is pressed.
- Ill-behaved applications will run only in an active session. If an ill-behaved application exits and stays resident in the active session, then any other application you run in that session will be seen as ill-behaved and will never run in the background.
- On non-3270 PC hardware, ill-behaved applications will be displayed full-screen when they are made active even if they are sized.
- Even if you sized your windows using the API, they may be forced to full-screen and appear enlarged when active under the following conditions:
 - Your application uses graphics mode
 - Your application uses 40-column mode
 - Your application writes directly to the screen.

Note: The Browse function will not work on applications forced to full-screen size.

- When using workstation control API, the WS Ctrl OIA will not be displayed on either a Uni-DOS or Multi-DOS system under the following conditions:
 - Your application uses graphics mode
 - Your application uses 40-column mode
 - Your application writes directly to the screen.

Changes or Limitations to the Personal Computer Session

- If you are running an ill-behaved application in a PC session, the shift state of that session may not remain as you originally set it after jumping to other windows and back again. For instance, if you have set Caps Lock on in this PC session, your session may be in lowercase mode when you jump back to it from another window.
- Input Control API is not supported for sessions running ill-behaved applications that read port 60 directly.
- Do not run PC applications that reprogram the timer. This could cause host communication failure.
- Batch files will only run in the active window. To IPL your system completely, you must jump to the session that contains your AUTOEXEC.BAT file.
- Applications that send or receive keystrokes from the host session will not run under Uni-DOS in a non-3270 PC XT system.
- Applications that write directly to the display adapter registers may not be restored properly after jumping to another window and back again.

Personal Computer Physical Cursor

Auto-windowing in the PC session occurs only in response to the movement of the PC session physical cursor. That is, the IBM 3270 PC window always follows and keeps the physical cursor within its viewable area. Some IBM Personal Computer application programs do not use the physical cursor, but use other types of cursors. These other types of cursors will not support the auto-windowing function. If you want auto-windowing to occur, you must convert such applications to use the physical cursor, or keep the physical cursor updated and in synchronization with the logical cursor.

If an application has inhibited the default cursor but you chose an alternate cursor, the alternate cursor will be the cursor you see.

On non-3270 PC hardware, your cursor may have changed its position in a PC session after you have jumped to another session. Press the spacebar to return the cursor to its original position.

Personal Computer Print Spooling

If you are running Uni-DOS, the DOS print spooler will not operate in a PC session and *should never be activated*. If you use the print spooler, it may, and in most instances will, cause the control unit communication session to terminate. See "Control Unit Communication Session Termination" below for other causes of session termination.

If you are using DOS Version 3.3 in a Multi-DOS PC session, you can use print spooling without any control unit communication problems.

Changes or Limitations to the Personal Computer Session

Control Unit Communication Session Termination

The PC application actions that may, and in most instances will, cause the control unit communication session to terminate are:

- Use of the disable/enable functions for an extended period of time
- Failure to issue an end-of-interrupt on a hardware interrupt level for an extended period of time
- Masking of selected interrupt levels for an extended period of time
- Failure to issue an IRET for an extended period of time.

IBM 3270 Personal Computer Failure

The 3270 Personal Computer will, in most cases, fail in an unpredictable manner if you:

- Use interrupt vectors X'50' through X'57' or X'7A'.
- Program the Intel 8259 Interrupt Controller timer chip.
- Program channel 0 or 1 of the Intel 8253 timer chip.
- Program the mode bits of any channel of the Intel 8253 timer chip.
- Program group A of the Intel 8255 programmable peripheral interface (PPI) chip.
- Program the Mode register of the Intel 8255 PPI chip.
- Take over hardware interrupt 2 (3270 PC adapter).
- Disable interrupts for an extended period of time.

Color Limitations

Note: This section does not apply to non-3270 PC hardware.

The 3270 Personal Computer supports eight colors for personal computer applications. If you use the Personal Computer color printer to print a Personal Computer color graphics screen in medium resolution graphics, the 3270 Workstation Program uses the blue color to determine which color palette is displayed. In palette 0 there is no blue, and the colors printed are black, blue, and pink. If you change the background color to green, blue, turquoise, or white while using palette 0, the blue color gun is turned on and the colors for palette 1 are used. The color palette select register is Write only and cannot be checked.

Notes on All-Points-Addressable Graphics

Note: This section does not apply to non-3270 PC hardware.

If you use the IBM Personal Computer all-points-addressable (APA) graphics applications on the IBM 3270 Personal Computer, your graphics images may appear differently. For example, you may have ovals in place of circles. For images to appear normally on the IBM 3270 Personal Computer, you must set the aspect ratio parameters in the application program. For example, in BASIC, add the aspect ratio parameter to the CIRCLE command to get circles instead of ovals on the display screen. Following is a table that shows the aspect ratios for the IBM 5153 Color Graphics Display, the IBM 5154 Enhanced Color Graphics Display, and the IBM 3270 Personal Computer Color Display.

Full-Screen Mode	5153 Personal Computer	5154 Personal Computer	3270 Personal Computer Color Display
Medium resolution	5:6	35:24	35:27 (approx 5:4)
High resolution	5:12	35:48	35:54 (approx 5:8)

Note: The current aspect ratio parameters are also available dynamically. A video interrupt (INT X'10') must be made to BIOS. Register AH must contain X'30'. This function call causes an address to be returned in CX (segment) and DX (offset), which points to the aspect values in two hexadecimal bytes: X'23' and X'36' for high resolution, or X'23' and X'1B' for medium resolution. The BIOS character generator cannot be used while in full-screen mode.

Using the Full-Screen APA Mode

Note: This section does not apply to non-3270 PC hardware.

When emulating the IBM Personal Computer Color Graphics Adapter, note that approximately one-half of the display screen is used (640 × 200 PELs). In addition, the APA adapter is capable of displaying 720 × 350 PELs. To set this full-screen mode, a video interrupt (INT X'10') must be made to BIOS. For high-resolution full-screen graphics mode (720 × 350 PELs), register AH must contain X'00' and register AL must contain X'30' or X'32'. For medium-resolution full-screen graphics mode (360 × 350 PELs), register AH must contain X'00' and register AL must contain X'31'. Colors associated with these modes are shown in the table below.

Changes or Limitations to the Personal Computer Session

AH	AL	Colors	Display Screen (PELs)
0	30	2	720 × 350
0	31	4	360 × 350
0	32	8	720 × 350

For either full-screen graphics mode, the storage addressing scheme changes from the standard IBM Personal Computer odd/even scans to a simpler, contiguous map of 32K bytes starting at 'B8000'. For example, the first byte of the second scan line is addressed as 90 (X'5A'). The IBM Personal Computer screens use the standard IBM Personal Computer map.

Changing the Cursor Size or Position

Note: This section does not apply to non-3270 PC hardware.

PC applications control the size and position of the cursor by writing to a pair of I/O registers, as described in the *IBM Personal Computer Technical Reference*.

The 6845 index registers X'0A' and X'0B' are used for the cursor size. Index registers X'0E' and X'0F' are used for the cursor position. Applications that do not use BIOS or DOS services for this must write these register pairs in the following order. For size, specify:

X'0A' followed by X'0B'

For position, specify:

X'0E' followed by X'0F'

Personal Computer Session Screen Size

Note: This section does not apply to non-3270 PC hardware.

PC applications can control the screen size of the PC session. Only the screen size of 25 × 80 is supported by the 3270 Workstation Program.

Appendix E. Problem Determination Procedures and Debugging Information

Introduction	E-2
System Error Problem Determination Procedures	E-2
Dump Data Utilities	E-3
Preparing Formatted Dump Diskettes	E-3
Taking the Dump	E-4
Using the Display Utility	E-5
Using the Trace Command	E-8
Workstation Program Loading Procedure	E-9
Debugging a PC Application Program	E-10
Debugging a System Extension	E-10
Control Blocks You Can Use during Debugging	E-10
The Supervisor's Data Area	E-11
The Dispatcher's Data Area	E-11
The Task Control Block	E-12
The Supervisor Name Table	E-14

Introduction

This appendix describes the problem determination procedures to follow if a system error occurs during API activity in your application program.

System Error Problem Determination Procedures

If a system error occurs during API activity, you should follow your local procedures for problem determination and:

1. If the return code indicates that you are out of system resources such as RQEs or TCBs, or if the name table is full, you must do the following:
 - a. If the error occurred in a system extension, you must increase the resource requirements in the SIF for that system extension. See Chapter 24, "Coding System Extensions," for information on SIFs.
 - b. If the error occurred in an application program, you must increase the requirements in the INDIBM2 SIF file. See the *IBM 3270 Workstation Program User's Guide and Reference* for information on updating the INDIBM2 SIF file.
2. Record the return code that indicates that a system error occurred, and also record whether the return code was in the CL register, in the parameter list, or in the communication status code.
3. Dump the system, using the procedures outlined under "Dump Data Utilities" in this appendix.
4. Turn on trace events 45, 46, 47, and 48. Instructions on using the trace facility are given under "Using the Trace Command" in this appendix.
5. Rerun the application that caused the system error.
6. If the system error occurs again:
 - a. Save the results of the first system dump, and dump the system again.
 - b. Follow your local procedures, and report the problem to your service coordinator.
7. If the system error does not occur again, follow your local procedures and report the system error occurrence to your service coordinator.

Dump Data Utilities

The dump data utilities prepare formatted dump diskettes, take a dump, and then display:

- Dumps of memory
- Dumps of the trace buffer
- Dumps of error counters.

The three basic steps you follow to use these utilities are:

1. Preparing formatted dump diskettes
2. Taking the dump
3. Displaying the dump.

Preparing Formatted Dump Diskettes

To use the dump facility, you must start by preparing a formatted dump diskette(s). To do this, perform the following steps:

1. IPL DOS and have available the diskette that contains the INDPREP utility.
2. Format and externally label the number of diskettes according to the chart below.

Remember:

- a. The diskettes cannot contain any bad sectors.
- b. Do **NOT** use /s as a FORMAT command parameter.

Type of Diskette	Number of Diskettes Needed	Externally Label the Diskette(s):
5-1/2-inch 360 Kb	3	DUMPDAT.A.001 DUMPDAT.A.002 DUMPDAT.A.003
3-1/2-inch 720 Kb	2	DUMPDAT.A.001 DUMPDAT.A.002
High-Density 1.2 Mb or greater	1	DUMPDAT.A.001

With the DOS diskette in the active drive, type:

format a:

or:

format b:

Dump Data Utilities

Note: Refer to the DOS manual to ensure that you are using the correct parameters for the FORMAT command (for example, /4 to format a 360K double-sided diskette in a high-capacity drive).

3. After formatting the recommended number of diskettes, insert the diskette containing the INDPREP utility into the active diskette drive and type:

INDPREP a:

or:

INDPREP b:

4. Press Enter.
5. You will be prompted to insert the diskette(s) just formatted.

At the conclusion of the INDPREP utility, the following message appears:

INDDP003 Dump diskette(s) ready for use

You have generated the dump diskette(s) to use only when you need to take a dump.

Taking the Dump

Notes:

1. *If you have an XMA (expanded memory adapter) installed, it may take up to 6 minutes to complete the dumping process.*
2. *If you used the IBM 3270 Workstation Program Keyboard Definition Utility to redefine the keyboard layout, when you are prompted to "Press D to take a dump..." you must press the "original" D key to take the dump.*

If you need to use the dump facility for problem determination procedures, you may take a dump in one of four ways:

1. By pressing the **D** key in response to an INDSY001 or INDSY002 message.
2. By pressing the Alt + Ctrl + Test keys (Alt + Ctrl + Scroll Lock on an enhanced PC keyboard, Alt + Ctrl + {+} on an XT or AT keyboard). The following prompt message will appear:

INDSY001 Unrecoverable System Error - 72060000 Press D to take a dump or any other key to re-IPL

3. By turning TRACE off with the dump option. To do this, type TRACE OFF/D. The following prompt message will appear:

4. By pressing the Non-Maskable Interrupt (NMI) pushbutton, if present, on the back of the system unit. Use the NMI pushbutton if the keyboard does not respond. The following prompt message will appear:

INDSY001 Unrecoverable System Error - 72050000 Press D to take a dump or any other key to re-IPL

After pressing the **D** key in response to one of the above system messages:

1. You will be prompted to insert diskette DUMPDATA.001.
2. The message INDSY011 Dumping ... will appear.
3. You will then be prompted to insert DUMPDATA.002 and DUMPDATA.003 diskettes if they were created. Insert them as requested.
4. When the dump is completed, messages INDSY011 and INDSY012 appear. Insert the system diskette and press any key to re-IPL.

This completes the dump process.

Using the Display Utility

The display utility, INDDISP, resides on your customized utilities system diskette, and is used to display dumps of memory, trace buffers, and error counters. Note that to use the display utility, you must have previously taken a dump or issued the command INDSAVE, discussed below.

To save dumps of trace buffers or error counters:

1. You must be in an active personal computer session.
2. At the DOS prompt, type:

INDSAVE TRACE

or:

INDSAVE COUNTER

This action creates TRACE.DMP or COUNTER.DMP files on the diskette in the active drive, depending on the parameter you chose. This is a quick way to provide trace or error counter information without dumping the entire system.

Dump Data Utilities

Displaying the Dump

To display dumps, dumps of trace buffers, or dumps of error counters:

1. You must be in an active personal computer session.
2. At the DOS prompt, type:

INDDISP a:DUMP

or

INDDISP b:DUMP

3. Press Enter.
4. The following message appears:

INDDDD004 Insert DUMPDATA.001. Press Enter, or End to quit
5. Insert DUMPDATA.001 in the currently active drive you specified above and press Enter.
6. Notice the display of the dump. It always begins at address 0000:0000.

You can use the following keys:

Home Takes you to another panel with additional options

PgUp Takes you to the next higher 256 bytes of memory

PgDn Takes you to the next lower 256 bytes of memory

End Ends the use of this display utility

7. If you press the Home key, a panel appears with prompts that allow you to display the following options:
 - a. Trace buffer
 - b. Registers at the time of the dump. If the registers are all zeros, they were not meaningful at the time of the dump.
 - c. Error counters
 - d. PC Presentation Space Work Areas
 - e. PC Presentation Space Buffers.

- f. Segment address in hexadecimal. Type the first four digits of the 8-character address and press Enter. The following prompt appears:

Enter offset to display

Type the last four digits that follow the colon. Then that section of memory is displayed.

- g. The frequently used buffers listed in Figure E-1 may be found at the segment addresses indicated.

Buffer You Want Displayed	Type of PC You Have	Segment Address
DFT/CUT communication buffers	3270 PC unique hardware or non-3270 PC hardware	CE00
PC display buffer	3270 PC unique hardware or non-3270 PC Mono Display hardware	B000
	Non-3270 PC Color Display hardware	B800* or A000**
PC graphics buffer	3270 PC unique hardware	B800
	Non-3270 PC hardware	B800* or A000**
3270 display buffer	3270 PC unique hardware	A000
	Non-3270 PC hardware	Not applicable

* BIOS Modes 0 through 6

** BIOS Modes D, E, 10, 11, and 13

As described in the "IBM Enhanced Graphics Adapter" section of the *Technical Reference Options and Adapter* manual.

Figure E-1. Frequently Used Trace Buffers

Displaying the Error Counters

To display the counter:

1. You must be in an active personal computer session.
2. At the DOS prompt, type:
INDDISP COUNTER
3. Press Enter.

Trace Command

4. The following message appears:

INDDDD006 Insert diskette with COUNTER.DMP. Press Enter,
or End to quit

5. Insert diskette with file COUNTER.DMP in the currently active drive,
and press Enter.
6. You will see a display of the counters where:

RC = Return code.

TH = Threshold or the point at which the error-handling program
takes action.

CT = Count or the number of times this error has been reported to the
error-handling program.

SV = Severity level of error: 1, 2, 3....

Displaying the Trace Buffer

To display the trace buffer:

1. You must be in an active personal computer session.
2. At the DOS prompt, type:

INDDISP TRACE

3. Press Enter. The following message appears:

INDDDD005 Insert diskette with TRACE.DMP. Press Enter, or
End to quit

4. Insert the diskette with file TRACE.DMP in the currently active drive,
and press Enter.
5. You will see a display of the trace buffers.

Using the Trace Command

Trace is used in gathering data required to isolate the causes of software failures in the workstation program and is to be used during problem determination. If you encounter problems running Trace, refer to the *IBM 3270 Workstation Program Problem Determination Guide*.

The Trace command runs as a PC application and is used to turn Trace on and off. While running in IBM Personal Computer DOS application mode with the PC window active, the operator issues a Trace On command for the trace recordings he wants to start or a Trace Off command to stop all trace recording. The traces in effect are those specified on the most recent Trace On command.

The discussion of solving IBM 3270 Personal Computer problems in the *IBM 3270 Workstation Program Problem Determination Guide* identifies which events you must turn on to obtain the correct documentation for a given problem.

The syntax of the command is:

$$\text{TRACE} \left\{ \begin{array}{l} \text{ON} \left[\begin{array}{c} ,M \\ M \end{array} \right] \left[\begin{array}{c} ,n \\ n \\ -n \end{array} \right] \dots \left[\begin{array}{c} ,o \\ o \end{array} \right] \left[\begin{array}{c} ,p \\ p \\ -p \end{array} \right] \\ \text{OFF} \left[/d \right] \end{array} \right\}$$

- The Trace commands may be typed and entered in any combination of uppercase and lowercase characters.
- *TRACE OFF /d* turns off all trace requests and causes the error handler to take a dump.
- One of each of the parameters shown in uppercase in a stack must be selected.
- Lowercase parameters are optional.
- *M*, *n*, *o*, and *p* represent decimal numbers corresponding to unique trace identification recording requests.
- Single or multiple blanks and/or commas delineate individual requests.
- The */d* parameter indicates that a dump is desired.

When a dump is requested, your system will request you to insert a diskette for the dump. This diskette will be formatted; therefore, **all existing files on the diskette will be erased**. After the dump gets control, the workstation program is no longer running. The only way to restart the system is by turning the system off and then on.

Workstation Program Loading Procedure

When the workstation program is IPLed, the INDCIPL.EXE loads the first system extension belonging to the workstation program, the supervisor, called INDSNUM.COM. Next, the INDCIPL.EXE loads the DOS subsystem extension INSDOM.COM, followed by the remaining system extensions from low storage to high storage. When all the system extensions have been loaded, the DOS subsystem divides the rest of the remaining storage into the PC environments. The last PC environment gets all of the storage left over, which fits within the 640K address space.

Debugging a PC Application Program

A PC application program running under the workstation program can be debugged by loading the application with the DEBUG utility, or any other software debugger that runs under the workstation program. You can use the debugger to help you find the errors in your PC application program.

Debugging a System Extension

Because system extensions do not have a logical refresh buffer, debugging your system extension can be done in the following ways:

- You can use a hardware debugger. In this case you could code a stop condition in the beginning of the system extension to find out where it is loaded and debug the system extension from there.
- You can run the system extension as a PC application and debug it using a software debugger. When the system extension is running correctly, you can remove any PC session dependencies such as DOS or BIOS calls.

Control Blocks You Can Use during Debugging

Following are some guidelines on how to find some of the control blocks used by the workstation program. These control blocks will assist you in debugging your system extension or PC application program.

Warning: These offsets will change across releases. No coding dependencies should be implemented based on these offsets.

Addresses are given in the format `xx:yy`, where `xx` is the segment portion of the address and `yy` is the offset portion of the address.

Address `0:1E8` (interrupt vector `X'7A'`) contains the address of `INDKSY1`, a module that belongs to the supervisor. The segment portion of that address is the beginning of the 3270 Workstation Program code. The first load module is named `INDSNUM.COM`.

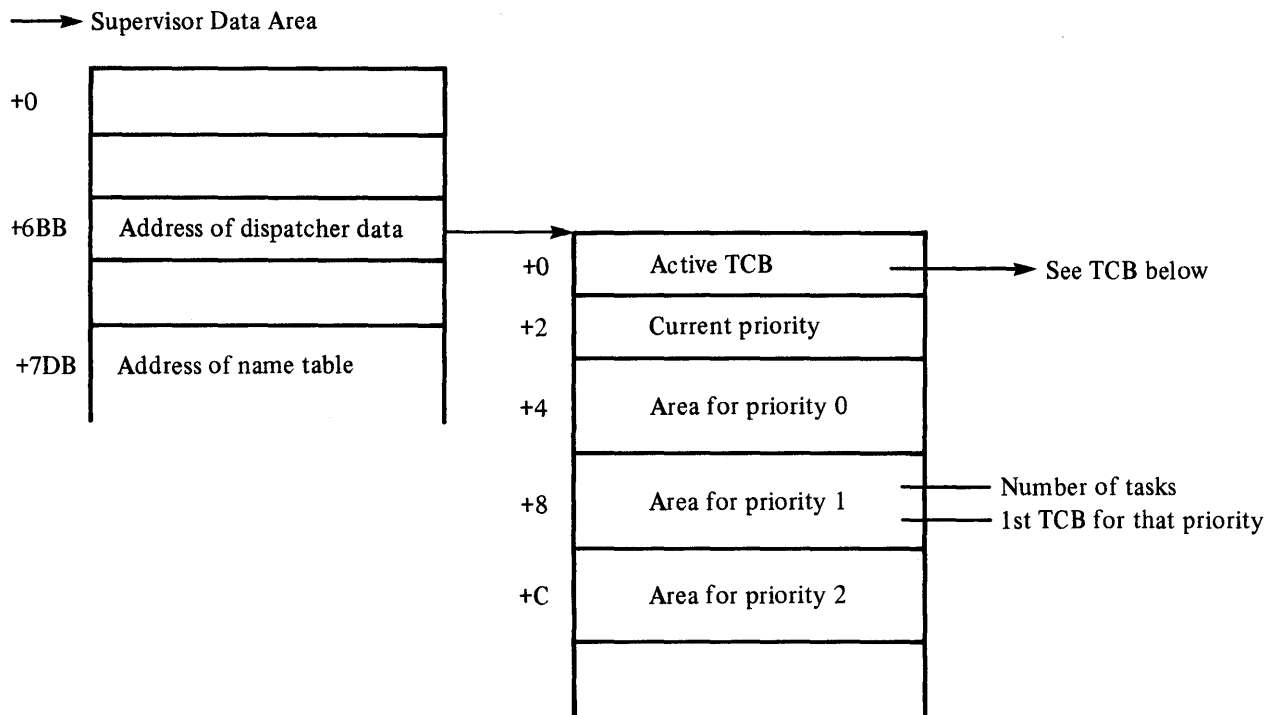
That address minus 5 points to the 1-word segment address of the supervisor's data area.

The Supervisor's Data Area

The supervisor's data area is on a segment boundary. All addresses within the supervisor's data area are 16-bit addresses. They are offset addresses and are to be used with the segment address of the supervisor data area to refer to any address within the supervisor's data area.

Warning: These offsets will change across releases. No coding dependencies should be implemented based on these offsets.

The supervisor's data area is shown below.



The Dispatcher's Data Area

To locate the dispatcher's data area, use an offset of X'671' into the supervisor's data segment. The first word in the dispatcher's data area is an offset address of the dispatched active task's task control block (TCB). The next word in the dispatcher's data area contains the priority of the highest priority dispatchable task. Next are 65 four-byte areas, one for each of the 65 task priority levels, 0 through 64. In each 4-byte area, the first 2 bytes contain the number of dispatchable tasks at that priority. The next 2 bytes contain the offset address of the TCB for the first task in the round robin at that priority level.

Priority zero is reserved for the supervisor. The PC task that runs an application has a priority of 60. Tasks created by a PC application program are restricted to priority levels 36 through 64.

Warning: These offsets will change across releases. No coding dependencies should be implemented based on these offsets.

See Figure E-2 for offsets in previous releases.

Release 2.1	Release 3.0	Release 4.0
+ 447	+ 671	+ 6BB

Figure E-2. Dispatcher Data Address Offsets

The Task Control Block

Warning: These offsets will change across releases. No coding dependencies should be implemented based on these offsets.

The offset address of a TCB, used together with the supervisor's data segment, can be used to locate that TCB. The fields of the TCB shown below may be helpful when you are debugging a program:

+0	TCB status byte
+1	Wait byte
+2	Priority + 1
+3	Preemption type
+4	Stack address (ss/sp)
+8	Request queue head pointer
+10	Request queue tail pointer
+12	Completion queue head pointer
+14	Completion queue tail pointer
+16	Pointer to next TCB in round robin
+18	SVC ID of task
+24	Waiting for data from this FLQ ID
+28	ID of semaphore being waited on

The TCB Status Byte

The TCB status byte is formatted as follows:

Bit 0	Bit 1	Bit 2	Bit 3	Bits 4, 5	Bit 6	Bit 7
Wait	Unready	Reserved	Nonpreemptable	Reserved	Susp	Reserved

- Bit 0 – Task is waiting
- Bit 1 – Task is “unready”
- Bit 2 – Reserved
- Bit 3 – Task is nonpreemptable within the round robin or environment
- Bit 4 – Reserved
- Bit 5 – Reserved
- Bit 6 – Task is suspended
- Bit 7 – Reserved

The TCB Wait Byte

The TCB wait byte is formatted as follows:

Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Request queue	Comp queue	Comp signal	Sema-phore	Timer	Signal	Data	Reserved

- Bit 0 – Waiting for an RQE in the request queue
- Bit 1 – Waiting for an RQE in the completion queue
- Bit 2 – Waiting for a ‘completion’ signal
- Bit 3 – Waiting to for a ‘semaphore claimed’ signal
- Bit 4 – Waiting for a ‘timer tick’ signal
- Bit 5 – Waiting for a ‘generic’ signal
- Bit 6 – Waiting for a ‘data available’ signal
- Bit 7 – Reserved

The TCB Preemption Type

The TCB preemption type is specified as follows:

- X’00’ – Task is preemptable
- X’01’ – Task is nonpreemptable within round robin
- X’02’ – Task is nonpreemptable within environment

The Request Queue Head Pointer

The request queue head pointer points to the first request queue element (RQE) on the task’s request queue. The request queue tail pointer points to the last RQE on the task’s request queue. These RQEs were placed on the task’s request queue after the Make a Request supervisor service was requested to make a request of that task. These RQEs may not have been processed by the task yet, or the task may be currently working on processing one of the RQEs. When the task has finished processing an RQE, it will use the Reply to a Request service so that the supervisor will

either move the RQE from its request queue to the completion queue of the requesting task or discard the RQE if the requesting task did not want a reply.

The Completion Queue Head Pointer

The completion queue head pointer points to the first RQE on the task's completion queue. The completion queue tail pointer points to the last RQE on the task's completion queue. These RQEs were placed on the task's completion queue as the result of a Make a Request service that had a reply type of "completion queue" specified. When the task requests the Get Request Completion service, the supervisor copies the contents of the RQE to the parameter list used for the service request, and returns the RQE to the system RQE pool.

The Request Queue Element

The following fields in an RQE may be helpful to you when you are debugging:

RQE	
+0	
+2	ID of requestor
+4	Next RQE on chain

The ID should be the ID of a task or component that was created as part of your system extension or PC application program. This ID was returned to you when the Create Task Entry or Create Component Entry service request was completed.

The Supervisor Name Table

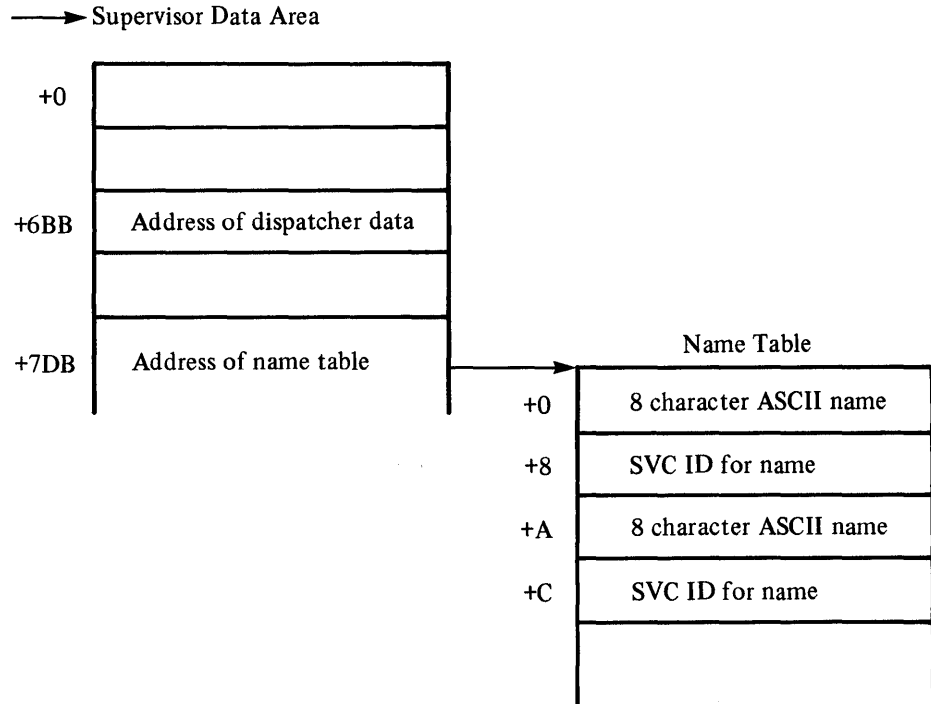
Warning: These offsets will change across releases. No coding dependencies should be implemented based on these offsets.

See Figure E-3 for offsets in previous releases.

Release 2.1	Release 3.0	Release 4.0
+55B	+78B	+7DB

Figure E-3. Name Table Address Offsets

At offset X'78B' into the supervisor's data area is the pointer to the name table. The name table contains all the names in the system that were added to the table through the use of the supervisory objects services. Each name is followed by the index of the object that was created with that name. The name table has the following format:



Following is a list of names used by the supervisor. Do not assign these same names to system objects that you create.

- Names that begin with the letters "IND" or 3270KS
- SYSKILL
- MEMORY
- DOSINT21
- DOSIOR
- DOSBADP
- XLATE
- SESSMGR
- KEYBOARD
- WSCTRL
- OIAM
- CPYUET
- MFIC
- 3270EML
- PCPSM
- COPY
- BSMUET

Appendix F. Presentation Space Considerations

Introduction	F-2
Attributes	F-2
Field Attributes	F-3
Extended Field Attributes and Character Attributes	F-4
Presentation Space Character Tables	F-6
Host and Notepad Session Character Codes	F-6
Personal Computer Session Character Codes	F-7
Presentation Space Sizes	F-7
Distributed Function Terminal (DFT) Host Presentation Space Sizes	F-7
Control Unit Terminal (CUT) Host Presentation Space Size	F-8
Notepad Presentation Space Size	F-8
Personal Computer Presentation Space Size	F-8

Introduction

Presentation space represents the area that contains the data that is shown in an image on your display screen. The data is stored in a format that includes both the character to be displayed and information about how that character is to be displayed, called the *attribute* of the character. This appendix discusses attribute types, character codes, and presentation space sizes for DFT host, CUT host, notepad, and personal computer sessions on the IBM 3270 Personal Computer.

Warning: Altering the contents of the presentation space can cause unpredictable results in the host application program that accesses the presentation space. Do not alter the contents of a host session presentation space unless the host application program is designed to interpret the changes that you make.

Note: For information concerning presentation spaces that are defined to accept 3270 keystroke emulation, refer to Chapter 9, "Coding 3270 Keystroke Emulation Service Requests."

Attributes

Display images may be unformatted or formatted. An unformatted display is one that has no defined fields. A formatted display is one that has separate fields defined by the host application program. The first character position in each field contains a control character, or attribute, that defines the characteristics of that field.

For detailed information on attributes and their use in the IBM 3270 data stream, see these manuals:

- *IBM 3270 Information Display System: 3274 Control Unit Description and Programmer's Guide*
- *IBM 3270 Information Display System: Data Stream Programmer's Reference*

The IBM 3270 Personal Computer display station supports field attributes, extended field attributes, and character attributes. Field attributes are supported in CUT and DFT host sessions, and are used to define the start of a field and control the characteristics of that field. Character attributes are supported in CUT host sessions, DFT host sessions, and notepad sessions, and are used to control the attributes of a character.

Note: If your DFT host session is customized to have an extended attributes buffer (EAB=Yes), the session will support field attributes, extended field attributes, and character attributes. Otherwise, the session supports only field attributes.

Field Attributes

The field attribute is used by the host application program to define the start of a field and to assign characteristics to that field. A field consists of the field attribute and all the data following it up to (but not including) the next field attribute. A field can wrap (continue) from the end of one row to the beginning of the next row within the presentation space. A field can also wrap from the last location in the presentation space to the first location. In any case, the field is terminated by the next field attribute. There is no limit to the number of fields that can be defined, other than that imposed by the screen size.

The characteristics that can be assigned to a field are:

- Protected or unprotected. A protected field is protected from modification by the operator. An unprotected field is available for the operator to enter or modify data. The unprotected definition classifies a field as an input field.
- Alphanumeric or numeric. Subject to its being protected, an alphanumeric field is one into which an operator enters data normally, using the shift keys as required.

When the numeric lock is active, fields defined as numeric will only accept characters 0 through 9, . , Dup, and - . Numeric lock can be overridden by pressing and holding the upshift key while typing. Overriding Numlock by this method will allow only upper shift characters to be entered.

- Autoskip. A field defined as protected **and** numeric causes the cursor to skip to the next unprotected field.
- Nondisplay or display/intensified display. The selected characteristics apply to the entire field. Nondisplay means that any characters entered from the keyboard are entered into the buffer for subsequent transmission to the application program but they are not displayed. Intensified display means the intensified characters appear on the screen brighter than the nonintensified characters. Some devices may not be able to intensify characters on the screen and will therefore highlight in a different manner.
- Detectable or nondetectable. A field defined as detectable can be detected by the cursor select key (CrSel), subject to the use of a designator character.

Figure F-1 shows the bit positions in the field attribute byte. Bit 0 is the leftmost bit in the byte, and bit 7 is the rightmost bit in the byte. Figure F-2 shows the bit assignments for the field attributes supported by the 3270 Personal Computer.

Upon entry of a character into the last character location of an unprotected data field, the cursor is repositioned based on the attributes of the next field.

Attributes

If the field attribute defines the next field as (1) alphanumeric and either unprotected or protected, or (2) numeric and unprotected, the cursor skips the attribute character and is positioned to the first character location in that field.

If the field attribute defines the field as numeric and protected, the cursor automatically skips that field and is positioned to the first character location of the next unprotected field.

0, 1	2	3	4, 5	6	7
Attribute	U/P	A/N	D/SPD	Reserved	MDT

Figure F-1. Field Attribute Bit Positions

EBCDIC Bit	Field Characteristics
0, 1	11 = This byte is an attribute
2	0 = Unprotected 1 = Protected (see Note)
3	0 = Alphanumeric 1 = Numeric (if numeric lock capability is activated, causes automatic numeric shift of keyboard) (see Note)
4, 5	00 = Display not detectable by Cursor Select key 01 = Display detectable by Cursor Select key 10 = Intensified display detectable by Cursor Select key 11 = Nondisplay, nonprint, nondetectable
6	Reserved – Always 0
7	Modified data tag (MDT); identifies modified fields during Read Modified command operation 0 = Field has not been modified 1 = Field has been modified by the operator. Can also be set by a program in the data stream.

Note: Bits 2 and 3 equal to binary 11 causes an automatic skip.

Figure F-2. Field Attribute Bit Assignment

Extended Field Attributes and Character Attributes

Extended field attributes and character attributes are used to give fields and single characters the attributes of highlighting, color, and character set. The extended field attribute is always associated with a field, and is positioned in the byte following the field attribute in the presentation space. The character attribute is associated with a single character, and is positioned in the byte following the character in the presentation space. The extended field attributes of any single character are always overridden

by the character attributes associated with it. However, characters in nondisplay fields are never displayed. The attribute structure used for character attributes is the same as for extended field attributes.

Figure F-3 shows the bit positions in the extended field/character attribute byte. Bit 0 is the leftmost bit in the byte, and bit 7 is the rightmost bit in the byte. Figure F-4 shows the bit assignments for the extended field attributes and character attributes supported by the IBM 3270 Personal Computer.

Highlighting	Color	Character set
0, 1	2 through 4	5 through 7

Figure F-3. Extended Field Attribute and Character Attribute Bit Positions

EBCDIC Bit	Character Characteristics
0, 1	Character highlighting: 00 = Normal ¹ 01 = Blink 10 = Reverse video 11 = Underscore ²
2 through 4	Character color: 000 = Default ¹ 001 = Blue 010 = Red 011 = Pink 100 = Green 101 = Turquoise 110 = Yellow 111 = White
5 through 7	Character set: 000 = Base character set ¹ 001 = Reserved 010 = Programmable Symbol Set A ³ 011 = Programmable Symbol Set B ³ 100 = Programmable Symbol Set C ³ 101 = Programmable Symbol Set D ³ 110 = Programmable Symbol Set E ³ 111 = Programmable Symbol Set F ³

¹ If this is a character attribute, a zero value in this field indicates that the value in the extended field attribute for this field is to be used. If this is an extended field attribute, a zero value in this field indicates that the default value for the display is to be used.

² On the Enhanced Graphics Adapter (EGA), the character highlighting underscore will be displayed as a normal attribute.

³ The Programmable Symbol Sets are not supported in notepad sessions on non-3270 PC hardware.

Figure F-4. Extended Field Attribute and Character Attribute Bit Assignment

For additional information on PC character attributes, see the *IBM Personal Computer Technical Reference*.

Presentation Space Character Tables

Characters in the presentation space are represented by hexadecimal codes.

Host and Notepad Session Character Codes

Figure F-5 shows the hexadecimal codes found in the DFT host, CUT host, and notepad presentation spaces, and the characters they represent.

	0x	1x	2x	3x	4x	5x	6x	7x	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx
x0	NUL	SP	0	&	à	ä	À	Ä	a	q	A	Q	↖	^	P	☒
x1	EM	=	1	–	è	ë	È	Ë	b	r	B	R	–		S	☒
x2	FF	'	2	.	ì	ï	Ì	Ï	c	s	C	S	z	☒	→	↵
x3	NL	”	3	,	ò	ö	Ò	Ö	d	t	D	T	_	°	↑	↵
x4	STP	/	4	:	ù	ü	Ù	Ü	e	u	E	U	☒	°	↑	☒
x5	CR	\	5	+	ã	â	Ã	Â	f	v	F	V	☒	–	↓	–
x6			6	¬	õ	ê	Õ	Ê	g	w	G	W	✕	☒	☒	–
x7			7	—	ÿ	î	Y	Î	h	x	H	X	☒	☒	☒	☒
x8	>	?	8	°	à	ô	À	Ô	i	y	I	Y	←	☒	☒	☒
x9	<	!	9		è	û	E	Û	j	z	J	Z	☒	☒	☒	☒
xA	[\$	β	^	é	á	E	Á	k	ae	K	Æ	☒	☒	☒	☒
xB]	¢	§	~	ì	é	I	É	l	ø	L	Ø	☒	☒	☒	☒
xC)	£	#	..	ò	í	O	Í	m	à	M	À	☒	☒	☒	☒
xD	(¥	@	`	ù	ó	U	Ó	n	ç	N	Ç	☒	☒	☒	☒
xE	}	pts	%	'	ü	ú	Y	Ú	o	;	O	;	☒	☒	☒	☒
xF	{	✱	—	5	ç	ñ	C	Ñ	p	*	P	*	☒	☒	☒	Not Supported

Notes:

1. Values X'C0' through X'FF' are used as attributes in CUT and DFT host sessions, and as characters in notepad sessions.
2. Characters X'68' through X'6F' are replaced in the refresh buffer by X'E8', X'69', X'6A', X'F8', X'FE', X'D4', X'CE', and X'D3', respectively. These characters are not used in the U.S..

Figure F-5. Host and Notepad Presentation Space Character Table

Personal Computer Session Character Codes

Figure F-6 shows the hexadecimal codes found in the personal computer presentation space, and the characters they represent.

HEXAD. DECIMAL VALUE	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	BLANK (NULL)	▶	BLANK (SPACE)	0	@	P	'	p	Ç	É	á	▤	▥	▦	∞	≡
1	☺	◀	!	1	A	Q	a	q	ü	æ	í	▧	▨	▩	β	±
2	☹	↑	"	2	B	R	b	r	é	Æ	ó	▪	▫	▬	Γ	≥
3	♥	!!	#	3	C	S	c	s	â	ô	ú	▭	▮	▯	π	≤
4	♦	¶	\$	4	D	T	d	t	ä	ö	ñ	▰	▱	▲	Σ	∫
5	♣	§	%	5	E	U	e	u	à	ò	Ñ	▴	▵	▶	σ	∫
6	♠	■	&	6	F	V	f	v	å	û	ä	▷	▸	▹	μ	÷
7	•	↓	'	7	G	W	g	w	ç	ù	o	▹	►	▻	τ	≈
8	•	↑	(8	H	X	h	x	ê	ÿ	ï	▻	▼	▽	ϕ	°
9	○	↓)	9	I	Y	i	y	ë	Ö	Γ	▻	▾	▿	θ	•
A	○	→	*	:	J	Z	j	z	è	Ü	Γ	▻	▿	▻	Ω	•
B	♂	←	+	;	K	I	k	{	ï	ç	½	▻	▻	▻	δ	√
C	♀	└	,	<	L	\	l		î	£	¼	▻	▻	▻	∞	n
D	♪	↔	-	=	M		m		ì	¥	ì	▻	▻	▻	φ	²
E	♪	▲	.	>	N	^	n	~	Ä	R	«	▻	▻	▻	€	■
F	☼	▼	/	?	O	—	o	Δ	Å	f	»	▻	▻	▻	∩	BLANK

Figure F-6. Personal Computer Presentation Space Character Table

For further information regarding the personal computer character set and attributes, refer to the *IBM Personal Computer Technical Reference*.

Presentation Space Sizes

Distributed Function Terminal (DFT) Host Presentation Space Sizes

Figure F-7 lists the presentation space size for DFT host sessions, based on the screen size and on whether the session is customized to support an extended attributes buffer (EAB). For additional information on color attributes, see the *IBM Personal Computer Technical Reference*.

Presentation Space Sizes

Screen Size	Attributes	Presentation Space Size
1920 characters (24 rows of 80 characters)	No EAB (4-color)	1920 bytes
1920 characters (24 rows of 80 characters)	EAB (7-color)	3840 bytes
2560 characters (32 rows of 80 characters)	No EAB (4-color)	2560 bytes
2560 characters (32 rows of 80 characters)	EAB (7-color)	5120 bytes
3440 characters (43 rows of 80 characters)	No EAB (4-color)	3440 bytes
3440 characters (43 rows of 80 characters)	EAB (7-color)	6880 bytes
3564 characters (27 rows of 132 characters)	No EAB (4-color)	3564 bytes
3564 characters (27 rows of 132 characters)	EAB (7-color)	7128 bytes

Figure F-7. DFT Host Presentation Space Sizes

Control Unit Terminal (CUT) Host Presentation Space Size

The presentation space size for a CUT host session is 1920 bytes, one byte in the presentation space for each character position on the screen.

Notepad Presentation Space Size

The presentation space size for the notepad session depends on the configured size of the notepad session. Each character in the notepad session is represented by two bytes in the presentation space. The first byte representing a character is the character itself, and the second byte is the character attribute associated with the character. For example, a notepad defined to have 24 rows of 80 characters will have a presentation space of 3840 bytes, since there can be 1920 characters in the notepad.

Personal Computer Presentation Space Size

The presentation space size for the personal computer session is 4000 bytes. Each character in the personal computer session is represented by two bytes. The first byte representing a character is the character itself, and the second byte is the attribute associated with the character.

Appendix G. Calling Save, Restore, Send, and Receive from Your Application Program

Introduction	G-2
The DOS SETBLOCK Function Call	G-2
The DOS EXEC Function Call	G-3
The Environment String	G-3
The Command Line	G-4
The File Control Blocks	G-4

The DOS SETBLOCK Function Call

Introduction

Your application program can invoke the Save, Restore, Send, and Receive commands by using the DOS function calls SETBLOCK and EXEC. This appendix describes how to set up these DOS function calls in your application program. The DOS function calls are described in the DOS manuals that were shipped with the version of DOS you are using.

The DOS SETBLOCK Function Call

The DOS SETBLOCK function call modifies the number of allocated storage blocks. You use the SETBLOCK function to free all available storage other than the storage used by your application program, so that there is room for the Send, Receive, Save, or Restore .COM file. To call the SETBLOCK function, set up the registers as follows:

AH = X'4A'

BX = the amount of storage used by your application program (in paragraphs)

ES = the segment address of the block of storage used by your application program (that is, the segment address of the application program's code segment).

To determine the amount of storage used by your application program (in paragraphs), use the following formula:

$$\text{paragraphs} = [(\text{program size} + 15) / 16]$$

Program size is the number of bytes shown for your .COM file in a directory listing obtained by the DOS DIR command, plus 16 paragraphs for the program segment prefix (PSP). If your application program allocates additional storage during execution, *program size* should include the number of bytes in the additional allocated storage as well.

Note: Be sure that the SS:SP registers point to a stack area within the same block of storage as your program.

Use the INT 21H instruction to invoke the DOS SETBLOCK function.

For more information on the DOS SETBLOCK function and other DOS function calls, see the DOS manuals that were shipped with the version of DOS you are using.

The DOS EXEC Function Call

The DOS EXEC function call loads and executes a program. You use the EXEC function to identify the program that you want to execute (Send, Receive, Save, or Restore), and to pass a parameter string to the program that contains the command line to be executed. To call the EXEC function, set up the registers as follows:

AH = X'4B'

AL = X'00'

BX = offset address of the parameter block

ES = segment address of the parameter block

DX = offset address of an ASCIIZ string containing the drive, path, and file name of the program to be loaded

DS = segment address of an ASCIIZ string containing the drive, path, and filename of the program to be loaded.

The block pointed to by ES:BX has the following format:

Data Type	Contents
Word	Segment address of the environment string to be passed
Dword	Pointer to the command line to be placed at PSP + X'80'
Dword	Pointer to the default FCB to be passed at PSP = X'5C'
Dword	Pointer to the default FCB to be passed at PSP = X'6C'

The Environment String

The environment string is a series of ASCII strings (totaling less than 32K bytes) in the form:

NAME = parameter

Each string is terminated by a byte of zeros, and the entire set of strings is terminated by another byte of zeros. The environment built by the command processor (and passed to all programs it invokes) contains a COMSPEC = string at a minimum. The parameter on the COMSPEC string is the path used by DOS to locate the command processor on the disk. The last PATH and PROMPT command issued will also be in the environment, along with any environment strings entered through the DOS SET command.

If you do not wish to change the environment string for the program being executed, set the segment address of the environment string to be passed to zero. Otherwise, build the new environment string and store the segment

The DOS EXEC Function Call

address of the string in this word. Additional information on the environment string can be found in the DOS manuals that were shipped with the version of DOS you are using.

The Command Line

The command line contains any parameters that you wish to send to the program being executed. The format of the command line is as follows:

Byte 0 is the number of bytes in the command line (a hexadecimal value).

Byte 1 is the ASCII code for a space (X'20').

The remaining bytes in the command line are the ASCII codes for the rest of the characters in the command line.

The File Control Blocks

The two default file control blocks (FCBs) are used to contain file names that may be needed by the program being executed. The format of the default FCBs is as follows:

Byte 0 is a decimal number representing the drive, where

- 00 represents the default drive
- 01 represents drive A
- 02 represents drive B
- 03 represents drive C

Bytes 1 through 8 contain the ASCII code for the file name, padded to the right with blanks if necessary. If a reserved drive name is placed here (such as LPT1), do not include the optional colon.

Bytes 9 through 11 contain the ASCII code for the file name extension, padded to the right with blanks if necessary. The file name extension can be all blanks. Additional information on the format of the FCB can be found in the DOS manuals that were shipped with the version of DOS you are using.

Use the INT 21H instruction to invoke the DOS EXEC function.

Note: When control is returned to your application program, all registers are changed, including the stack. You must restore SS, SP, and any other required registers before proceeding.

For more information on the DOS EXEC function and other DOS function calls, see the DOS manuals that were shipped with the version of DOS you are using.

Appendix H. Return Codes

Introduction	H-2
Function ID X'12': System Services Return Codes	H-3
Function ID X'13': Environment Manager Services Return Codes ..	H-11
Function ID X'22' or X'23': DOS Subsystem Services Return Codes ..	H-16
Function IDs X'24' or X'25': System Loader Return Codes	H-22
Function ID X'30': DFT Operations Return Codes	H-26
Function ID X'32': Host Interactive Services Return Codes	H-32
Function ID X'46': CUT Return Codes	H-33
Function ID X'51': Notepad Operations Return Codes	H-34
Function ID X'62': Keyboard Services Return Codes	H-35
Function ID X'63': Window Management Services Return Codes ...	H-38
Function ID X'64': Copy Services Return Codes	H-41
Function ID X'67': Draw Service Return Codes	H-43
Function ID X'69': Presentation Space Services Return Codes	H-44
Function ID X'6B': Session Information Services Return Codes	H-47
Function ID X'6C': Translate Services Return Codes	H-49
Function ID X'6D': OIA Services Return Codes	H-50
Function ID X'6E': 3270 Keystroke Emulation Services Return Codes	H-51
Function ID X'6F': Keystroke Definition Return Codes	H-52
Function ID X'72': Error Handler Return Codes	H-53
Function ID X'7F': Dump Task Return Codes	H-54
Function ID X'81': Enhanced Connectivity Router Return Codes ...	H-55
Function IDs X'Dx through Fx': User System Extension Return Codes	H-56
Return Code Error Steps	H-57

Introduction

This appendix contains explanations of the return codes issued by the workstation program. These return codes can appear in messages on the screen, or can be returned to your application program when it requests an API service.

Return codes are two bytes long. The first byte of the return code is the function ID, and the second byte is the error number. The function ID indicates the portion of the workstation program that is issuing the return code. The error number indicates the specific condition being reported. The possible function IDs are:

Function ID	Code Reported By
X'12'	Supervisor services
X'13'	Environment manager services
X'22' or X'23'	Multiple DOS support services
X'24' or X'25'	System Loader
X'30'	DFT system extension
X'32'	Host interactive services
X'46'	CUT system extension
X'52'	Notepad system extension
X'62'	Keyboard services
X'63'	Window management services
X'64'	Copy services
X'69'	Presentation space services
X'6B'	Session information services
X'6C'	Translate services
X'6D'	Operator Information Area services
X'6E'	3270 keystroke emulation services
X'72'	Error handler
X'7F'	Dump task
X'81'	Enhanced Connectivity Router

Notes:

- 1. Return codes with a function ID of X'Dx' through X'Fx' are generated by user-supplied system extensions. Consult local documentation for the meaning of these return codes and the action to take when they are encountered. If you get any return codes that are not listed, use the procedures at your location for diagnosing the problem.*
- 2. "Local procedures," to which you are frequently referred during this chapter, are the procedures followed in your location for isolating problems or making repairs.*

Function ID X'12': System Services Return Codes

Return codes beginning with function code X'12' indicate that an error occurred during supervisor operations, except return code X'1200', which indicates that the requested supervisor service was completed successfully.

Code	Explanation	Action to Take
1200	The requested supervisor service was completed successfully.	None.
1201	The object being created does not have a unique name.	Ensure that the name is unique. If it is, rerun the application that caused the error. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.
1202	The supervisor cannot create any more objects, because the SVC table is full; service failed.	You must increase the SVC table resource. See "Return Code Error Step" 8 on page H-57.
1203	The supervisor cannot create any more named objects, because the system name table is full; service failed.	You must increase the system name table resource. See "Return Code Error Step" 8 on page H-57.
1204	The supervisor cannot create any more tasks, since it ran out of task control blocks; service failed.	You must increase the number of task control blocks. See "Return Code Error Step" 8 on page H-57.
1205	The SVC index specified in the DX register or the parameter list is not valid for the service requested.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.
1206	The specified priority was out of range; requested service failed.	Check the input priority index parameter to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.

System Services Return Codes - 12xx

Code	Explanation	Action to Take
1207	The requested reply is not valid; service failed.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.
1208	The requested wait is not valid; service failed.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.
1209	The queue is empty.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the problem persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5 and 6 on page H-57.
120A	The nonpreemption type specified on create task service is invalid, defaulted to preemptable; service successful.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.
120B	The system request queue element pool is depleted; the system cannot continue.	You must increase the number of system RQEs. See "Return Code Error Step" 8 on page H-57.
120D	A Release Semaphore request was issued for a semaphore that was already free.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5 and 6 on page H-57.
120E	An invalid interrupt vector or level was specified; service failed.	Check the input parameters to the supervisor. Check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.
120F	An invalid environment access was attempted; service failed.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.

System Services Return Codes - 12xx

Code	Explanation	Action to Take
1210	The timer is not owned by the requester; service failed.	Check the input parameter to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.
1211	No more timers are available.	You must increase the number of timer resources. See "Return Code Error Step" 8 on page H-57.
1212	A request was made to a terminating task; service failed.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.
1213	The dequeue request failed; the request is too big.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.
1215	The Install User Exit Table Entry service was requested with an entry index that is out of range.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.
1216	Invalid "count" parameter, which specifies the number of entries to be placed in a user exit table.	Check the count parameter in the input parameter list to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.
121A	The system is running with an XMA card. On Install User Exit Table Entry Service, the user exit table is in an address space that is not available to the requester. For more information on system extensions and the XMA card, see the <i>Workstation Program Programming Guide</i> .	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.

System Services Return Codes - 12xx

Code	Explanation	Action to Take
121D	There is sufficient memory for the supervisor to allocate requested resources.	Check the SIF file resource requests. Correct any errors that exist, or if there are no errors, reduce the number of system extensions.
121E	A first-level interrupt handler has run out of stacks.	<p>Increase the number of stacks used by the first-level interrupt handler in the INDIBM2.SIF file.</p> <p>Warning: Increments of only 1 are advisable since each increment represents another 384 bytes.</p>
121F	A version of PC Land is lower than 1.2.	Use a version of the PC Land program higher than 1.2, or run Version 1.2 in redirector mode only.
1220	No more interrupt handlers can be installed.	You must increase the number of interrupt handler resources. See return code error step 8.
1221	The environment ID specified in the DL register or the parameter list is not valid for the service requested.	Check the environment ID input parameter to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.
1223	No free environment control blocks are available.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.
1224	The resource manager index specified in the request is invalid.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.
1225	The maximum number of resource managers was already defined.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.
1226	The maximum number of software interrupt vectors (32) were already taken. The mix of program applications is using too many software vectors for the supervisor to handle.	Stop any unnecessary applications to reduce the number of software interrupt vectors that are used.

System Services Return Codes - 12xx

Code	Explanation	Action to Take
1228	The buffer provided on a Query Environment request was too small to contain the output; service failed.	Check the input parameter to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5 and 6 on page H-57.
122B	The environment already was suspended.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5 and 6 on page H-57.
122C	The semaphore was not claimed, even though "wait for semaphore" was specified. (Some other specified wait condition was satisfied first.)	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5 and 6 on page H-57.
122D	The stoppable environment was not allowed to create environments.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5 and 6 on page H-57.
122E	The name does not exist.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5 and 6 on page H-57.
122F	The supervisor service does not exist.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5 and 6 on page H-57.
1230	A task, fixed-length queue, or semaphore cannot be deleted if requests are pending.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5 and 6 on page H-57.

System Services Return Codes - 12xx

Code	Explanation	Action to Take
1231	The task cannot be deleted, because it owns a timer.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5 and 6 on page H-57.
1232	The supervisor cannot stop or delete a nonstopable environment.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5 and 6 on page H-57.
1233	The supervisor cannot find the specified resource.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5 and 6 on page H-57.
1234	The object to be installed in a gate is not a task or component, the gate length is invalid, or an invalid index (service number) was specified in the AL register on request.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5 and 6 on page H-57.
1235	A user exit table cannot be created with a length of zero.	Correct the length and retry.
1236	No request queue elements are on the request queue.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5 and 6 on page H-57.
1237	The dequeue with no wait failed because it is not the requester's turn.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5 and 6 on page H-57.

System Services Return Codes - 12xx

Code	Explanation	Action to Take
1238	There is an error in opening a file.	Check the first message on the screen for the name of the file. Verify that the file exists on your system diskette. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 6, and 7 on page H-57.
1239	There is not enough room in the fixed-length queue to enqueue the specified data.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5 and 6 on page H-57.
123A	There is an error reading in a file.	Check the first message on the screen for the name of the file. Verify that the file has not been damaged on your system diskette. (Follow the procedures in your DOS manual to run a Check Disk.) If there is damage, customize again on a new formatted diskette. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 6, and 7 on page H-57.
123B	The supervisor cannot create any more gates, because the system gate table is full.	Increase the gate table size. See "Return Code Error Step" 8 on page H-57.
123C	The type specified is not a valid semaphore type.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5 and 6 on page H-57.
123D	This code was returned on a claim semaphore with a no-wait; it means that the semaphore is already claimed.	Check the program logic. Correct the wait status if needed, and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.
123F	The gates cannot be deleted.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5 and 6 on page H-57.

System Services Return Codes - 12xx

Code	Explanation	Action to Take
1240	The delete environment is already pending.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5 and 6 on page H-57.
1241	The fixed length queue size is in error.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5 and 6 on page H-57.

Function ID X'13': Environment Manager Services Return Codes

Return codes beginning with function code X'13' indicate that an error occurred during environment manager operations, except return code X'1300', which indicates that the requested environment manager service was completed successfully.

Code	Explanation	Action to Take
1300	The requested environment manager service was completed successfully.	None.
1305	The SVC index specified in the DX register or the parameter list is not valid for the service requested.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5 and 6 on page H-57.
1306	The priority specified is not in the range of valid priorities.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5 and 6 on page H-57.
130C	A request to stop, reset, suspend, or resume an environment failed because the return code field in the parameter list of the work request was not set to zero.	Set the return code field of the input parameter list to zero. Then rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.
130F	The requester is not allowed to complete the type of request that was made, because of invalid environment access.	Check to be sure the requester is allowed to complete the specified request. Then check the input parameters to the supervisor, check program logic, and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.

Environment Manager Services Return Codes - 13xx

Code	Explanation	Action to Take
1314	A request was made to a task in an environment that was stopped, reset, or involved in an INDSPLIT or INDMERGE operation before the request could be acted upon.	Once the stop, reset, INDSPLIT, or INDMERGE is completed, the parameter list can be set up and the request made again. If the request was not made by an application and no error can be found in your procedures, check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3a, 4, 5, and 6 on page H-57.
1317	The requester is in a stoppable environment and is trying to stop, suspend, or resume another environment.	Check that the requester is allowed to complete the specified request. Check the input parameters to the supervisor. Check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3a, 4, 5, and 6 on page H-57.
1321	The environment ID specified in the DL register or the parameter list is not valid for the service requested.	Check the environment ID input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.
1322	An attempt to stop a personal computer environment that was made either through the API by using ALT-CTRL-DEL or by a request to delete an environment (using INDSPLIT or INDMERGE) failed because some resources were not successfully released. Some internal error occurred, and the stop is not recoverable.	The environment on which the stop was not completed cannot be used. It may be that a system error has occurred or that some resource manager or its device has hung. In this case, you may want to turn power off and on again. If no error can be found in your procedures, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3a, 4, 5 and 6 on page H-57.
1323	No free environment control blocks are available.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.

Environment Manager Services Return Codes - 13xx

Code	Explanation	Action to Take
1324	The resource manager index specified in the request is invalid.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.
1325	A Resource Manager Open request failed because no resource manager indexes are available.	Increase the number of resource managers. See return code error step 8.
1327	An attempt to stop a personal computer environment that was made either through the API by using ALT-CTRL-DEL or by a request to delete an environment (using INDSPLIT or INDMERGE) failed because the process took too long to complete. The request may be completed at a later time.	<p>The environment in which the stop request was not completed cannot be used. If possible, wait until the condition clears. If it does not clear, another environment may be taking too much processor time, so that this stop request cannot be completed. You may reduce the system load by stopping another environment. If the condition does not clear, a system error may have occurred or some resource manager or its device has hung. In this case, you may want to turn power off/on again. If power off/on is not attempted, the system will continue to try to clean up. If the cleanup is completed, the environment manager will post a different return code, and the environment may be reused. If the cleanup is not completed, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3a, 4, 5, and 6 on page H-57.</p> <p>Some common reasons that an environment cleanup will not be completed:</p> <ul style="list-style-type: none">• The application is still holding a code serialization semaphore.• The cleanup component did not delete all its resources.• A task in the environment being cleaned up is waiting for a reply to a request from a system extension.
1328	The size of the output buffer specified in a Query Environment Characteristics request is too small to hold the data requested.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.

Environment Manager Services Return Codes - 13xx

Code	Explanation	Action to Take
1329	A request was made to resume an environment that was not in a suspended state.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3b, 4, 5, and 6 on page H-57.
132D	A request was made to create an environment, but the requester is in a stoppable environment.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.
1332	A request was made to stop a nonstoppable system environment.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3a, 4, 5, and 6 on page H-57.
1333	A request to move a resource to the top of a resource chain or to delete a resource from a resource chain failed because the resource specified could not be found.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.
1340	A previous request to delete the specified environment using INDSPLIT or INDMERGE is already in progress.	Check the input parameters to the supervisor. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3a, 4, 5, and 6 on page H-57.

Environment Manager Services Return Codes - 13xx

Code	Explanation	Action to Take
1342	A previous request to stop an environment made through the API by using ALT-CTRL-DEL or by a previous request to delete an environment (using INDSPLIT or INDMERGE) failed with a return code of X'1327', indicating a time-out has occurred. That request is now completed, and the environment is now available for reuse.	Make the window active that previously returned the error, and begin another application.
1343	A request to stop, reset, suspend, or resume an environment failed because the request type field in the parameter list was not a valid request type.	Check the input parameters to the supervisor, including the values in the parameter list. Then check program logic and rerun the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 3c, 4, 5 and 6 on page H-57.

Function ID X'22' or X'23': DOS Subsystem Services Return Codes

Return codes beginning with function code X'22' or X'23' indicate that an error occurred during DOS subsystem operations, except return code X'2200', which indicates that the requested DOS subsystem service was completed successfully. In some cases, the return code indicates that the error was generated by DOS when the DOS subsystem issued a DOS function call.

Code	Explanation	Action to Take
2200	The requested DOS subsystem service was completed successfully.	None.
2201 or 2301	The DOS subsystem issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 1. INVALID FUNCTION NUMBER.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5, and 6 on page H-57.
2202 or 2302	The DOS subsystem issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 2. FILE NOT FOUND.	Place the file that could not be found on the disk being used and retry the operation. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5, and 6 on page H-57.
2203 or 2303	The DOS subsystem issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 3. PATH NOT FOUND.	Place a disk in the drive with the correct path or create the path on the disk and then retry the operation. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5, and 6 on page H-57.
2204 or 2304	The DOS subsystem issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 4. TOO MANY OPEN FILES (NO HANDLES LEFT).	Create a CONFIG.SYS on the IPL disk or edit the existing one and increase the number of file handles. The command in the file is FILES = xx, where xx is the number of file handles. See the DOS manual for details on setting up your CONFIG.SYS file. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5, and 6 on page H-57.
2205 or 2305	The DOS subsystem issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 5. ACCESS DENIED.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5, and 6 on page H-57. his chapter.

DOS Subsystem Services Return Codes - 22xx or 23xx

Code	Explanation	Action to Take
2206 or 2306	The DOS subsystem issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 6. INVALID HANDLE.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2 4, 5, and 6 on page H-57.
2207 or 2307	The DOS subsystem issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 7. MEMORY CONTROL BLOCKS DESTROYED.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5 and 6 on page H-57.
2208 or 2308	The DOS subsystem issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 8. INSUFFICIENT MEMORY.	Make more storage available and retry the request. If it appears that there should have been enough storage, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5, and 6 on page H-57.
2209 or 2309	The DOS subsystem issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 9. INVALID MEMORY BLOCK ADDRESS.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5 and 6 on page H-57.
220A or 230A	The DOS subsystem issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 10. INVALID ENVIRONMENT.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5 and 6 on page H-57.
220B or 230B	The DOS subsystem issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 11. INVALID FORMAT.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5 and 6 on page H-57.
220C or 230C	The DOS subsystem issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 12. INVALID ACCESS CODE.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5 and 6 on page H-57.

DOS Subsystem Services Return Codes - 22xx or 23xx

Code	Explanation	Action to Take
220D or 230D	The DOS subsystem issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 13. INVALID DATA.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5 and 6 on page H-57. chapter.
220F or 230F	The DOS subsystem issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 15. INVALID DRIVE WAS SPECIFIED.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5 and 6 on page H-57.
2210 or 2310	The DOS subsystem issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 16. ATTEMPT TO REMOVE THE CURRENT DIRECTORY.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5 and 6 on page H-57.
2211 or 2311	The DOS subsystem issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 17. NOT SAME DEVICE.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5 and 6 on page H-57.
2212 or 2312	The DOS subsystem issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 18. NO MORE FILES.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5 and 6 on page H-57.
2213 thru 2253 or 2313 thru 2353	The DOS subsystem issued a DOS interrupt X'21' function call, and DOS failed the request with a DOS ERROR CODE nn, where nn is in hexadecimal.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5 and 6 on page H-57.

DOS Subsystem Services Return Codes - 22xx or 23xx

Code	Explanation	Action to Take
22E2	An error occurred in the DOS subsystem. A Split or Merge command was rejected, because it was issued for an environment that does not exist.	Issue a Display Environment (INDDENV *) command to see what environments do exist. If it appears that the Split command should have been completed, take a system dump by pressing Alt + Ctrl + Test (Alt + Ctrl + Scroll Lock on an enhanced PC keyboard, Alt + Ctrl + {+} on an XT or AT keyboard), then follow your local procedures and have available the dump and the data from "Return Code Error Steps" 2 and 6 on page H-57.
22E3	An error occurred in the DOS subsystem. A Split or Merge command was rejected, because it was issued for an environment that is being terminated.	Wait until the original Split command is completed. If it hangs, take a system dump by pressing Alt + Ctrl + Test (Alt + Ctrl + Scroll Lock on an enhanced PC keyboard, Alt + Ctrl + {+} on an XT or AT keyboard), then follow your local procedures and have available the dump and the data from "Return Code Error Steps" 2 and 6 on page H-57.
22E4	An error occurred in the DOS subsystem when the DOS environment task received an invalid request. The only valid requests are "Create" and "Clean Up."	<p>If the problem can be re-created, follow your local procedures and have available the dump and the data from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.</p> <p>If the problem cannot be re-created, take a system dump by pressing Alt + Ctrl + Test (Alt + Ctrl + Scroll Lock on an enhanced PC keyboard, Alt + Ctrl + {+} on an XT or AT keyboard), then follow your local procedures and have available the dump and the data from "Return Code Error Steps" 2 and 6 on page H-57.</p>

DOS Subsystem Services Return Codes - 22xx or 23xx

Code	Explanation	Action to Take
22E5	An error occurred in the DOS subsystem when the DOS environment task received an invalid parameter list. The return code field of the input parameter list was nonzero.	<p>Ensure that the parameter list passed to the DOS environment task has a zero return code field. If the problem persists, follow your local procedures and have available the dump and the data from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.</p> <p>If the problem cannot be recreated, take a system dump by pressing Alt + Ctrl + Test (Alt + Ctrl + Scroll Lock on an enhanced PC keyboard, Alt + Ctrl + {+} on an XT or AT keyboard), then follow your local procedures and have available the dump and the data from "Return Code Error Steps" 2 and 6 on page H-57.</p>
22E6	An error occurred in the DOS subsystem when the DOS Query Environment request received an invalid parameter list. The return code field of the input parameter list was nonzero.	<p>Ensure the parameter list passed to DOS Query Environment has a zero return code field. If the problem persists, follow your local procedures and have available the dump and the data from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.</p> <p>If the problem cannot be recreated, take a system dump by pressing Alt + Ctrl + Test (Alt + Ctrl + Scroll Lock on an enhanced PC keyboard, Alt + Ctrl + {+} on an XT or AT keyboard), then follow your local procedures and have available the dump and the data from "Return Code Error Steps" 2 and 6 on page H-57.</p>
22E7	A request was made with an invalid environment ID. A DOS Query Environment request was issued for an environment that does not exist, or a memory request was issued for an invalid environment.	<p>Issue a Display Environment (INDDENV) command to see what environments do exist. If it seems that the Environment request should have been completed successfully, follow local procedures and have available the dump and the data from "Return Code Error Steps" 1, 2, 3c, 4, 5, and 6 on page H-57.</p>
22E8	An error occurred in the DOS subsystem. A Create Environment request was issued that was not completed.	<p>This return code is always accompanied by a second return code (XXXX) that explains why the Create Environment request failed. Look up the second return code and take the action recommended for that return code.</p>

DOS Subsystem Services Return Codes - 22xx or 23xx

Code	Explanation	Action to Take
23FD	A request was made using the Asynchronous DOS Function Request service without a prior request to connect for asynchronous DOS function requests.	Request the Asynchronous DOS Function Request service with a request type of X'00' to connect for asynchronous DOS function requests.
23FE	The request to the DOS subsystem to add a device to the DOS subsystem redirection function failed.	Run fewer programs that are adding entries into the redirection tables. Re-IPL to reset the DOS subsystem and retry the request. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5, and 6 on page H-57.
23FF	The DOS subsystem encountered an error while processing a request for a personal computer session for which there is no way to report the error to the application. The environment in which the application was running stopped.	This return code is always accompanied by a second return code (XXXX) that explains what the initial failure was. Look up the second return code and take the action recommended for that return code. Correct the problem in the application or system and retry the application. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5, and 6 on page H-57.

Function IDs X'24' or X'25': System Loader Return Codes

Return codes beginning with function code X'24' or X'25' indicate that an error occurred during system loader operations.

Code	Explanation	Action to Take
2404	A request was made to the loader for storage to be allocated from the XMA card and assigned to a bank. This return code indicates there were no available banks.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, and 7 on page H-57.
2405	A request was made to the loader for storage to be allocated from the XMA card and assigned to a bank. This return code indicates that the requested storage size was invalid.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, and 7 on page H-57.
2406	A request was made to the loader for storage to be allocated from the XMA card and assigned to a bank. This return code indicates that there was not enough storage available on the XMA card.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, and 7 on page H-57.
	A 22E8 preceding the 2406 return code indicates that the workstation program ran out of XMA storage while trying to create a PC session. For example, you may receive 22E82406 if you customized the system for a 2-megabyte card and ran the system with a 1-megabyte card; or if you customized for multiple PC sessions, there may not be enough storage for the last session if you have device drivers and user system extensions.	Recustomize the system, referring to the <i>User's Guide (Setting Up and Learning the Workstation Program)</i> to calculate the session sizes.

System Loader Return Codes - 24xx or 25xx

Code	Explanation	Action to Take
241B	A request was made to the loader for storage on the XMA card, and there was not enough storage available for the request.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, and 7 on page H-57.
2501	The loader issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 1. INVALID FUNCTION NUMBER.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5, and 6 on page H-57.
2502	The loader issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 2. FILE NOT FOUND.	Place the file that could not be found on the disk being used and retry the operation. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5, and 6 on page H-57.
2503	The loader issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 3. PATH NOT FOUND.	Place a disk in the drive with the correct path or create the path on the disk and then retry the operation. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5, and 6 on page H-57.
2504	The loader issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 4. TOO MANY OPEN FILES (NO HANDLES LEFT).	Create a CONFIG.SYS on the IPL disk or edit the existing one and increase the number of file handles. The command in the file is FILES=xx, where xx is the number of file handles. See the DOS manual for details on setting up your CONFIG.SYS file. If the error persists, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5, and 6 on page H-57.
2505	The loader issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 5. ACCESS DENIED.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5, and 6 on page H-57.
2506	The loader issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 6. INVALID HANDLE.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5, and 6 on page H-57.

System Loader Return Codes - 24xx or 25xx

Code	Explanation	Action to Take
2507	The loader issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 7. MEMORY CONTROL BLOCKS DESTROYED.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5 and 6 on page H-57.
2508	The loader issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 8. INSUFFICIENT MEMORY.	Make more storage available and retry the request. If it appears that there should have been enough storage, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5, and 6 on page H-57.
2509	The loader issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 9. INVALID MEMORY BLOCK ADDRESS.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5 and 6 on page H-57.
250A	The loader issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 10. INVALID ENVIRONMENT.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5 and 6 on page H-57.
250B	The loader issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 11. INVALID FORMAT.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5 and 6 on page H-57.
250C	The loader issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 12. INVALID ACCESS CODE.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5 and 6 on page H-57.
250D	The loader issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 13. INVALID DATA.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5 and 6 on page H-57.

System Loader Return Codes - 24xx or 25xx

Code	Explanation	Action to Take
250F	The loader issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 15. INVALID DRIVE WAS SPECIFIED.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5 and 6 on page H-57.
2510	The loader issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 16. ATTEMPT TO REMOVE THE CURRENT DIRECTORY.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5 and 6 on page H-57.
2511	The loader issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 17. NOT SAME DEVICE.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5 and 6 on page H-57.
2512	The loader issued a DOS interrupt X'21' function call, and DOS failed the request with DOS ERROR CODE 18. NO MORE FILES.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5 and 6 on page H-57.
2513 thru 2553	The loader issued a DOS interrupt X'21' function call, and DOS failed the request with a DOS ERROR CODE nn, where nn is in hexadecimal.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5 and 6 on page H-57.

Function ID X'30': DFT Operations Return Codes

Return codes beginning with function code X'30' indicate that an error occurred during DFT operations, except return code X'3000', which indicates that the requested DFT service was completed successfully.

If these return codes were issued because of some API interaction, they will be followed by another return code that better describes the problem and the best action to take; otherwise, follow the "Action to Take" information provided with the return code.

Code	Explanation	Action to Take
3000	The requested DFT service was completed successfully.	None.
3001	An error occurred during DFT operations during an attempt to sound a bell alarm.	None.
30C7	An error occurred during DFT initialization operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30C8	An error occurred during DFT operations because a nonresettable machine check was received.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30C9	An error occurred during DFT error-handling operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30CA	An error occurred during DFT error-handling operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30CB	An error occurred during DFT error-handling operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30CC	An error occurred during DFT operations while the screen size was being changed.	Take a system dump, follow local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.

DFT Operations Return Codes - 30xx

Code	Explanation	Action to Take
30CE	An error occurred during DFT operations while the active logical terminal session was being found.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30CF	An error occurred during DFT operations while the active logical terminal session was being found.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30D0	An error occurred during DFT initialization operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30D1	An error occurred during DFT operations while the DFT environment was being reset.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30D2	An error occurred during DFT operations while a keystroking task was being reinitialized for any of the configured logical terminals.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30D3	An error occurred during DFT operations while a DFT inbound data task was being reinitialized for any of the configured logical terminals.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30D4	An error occurred during DFT operations while a DFT outbound data task was being reinitialized for any of the configured logical terminals.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30D5	An error occurred during DFT operations while a DFT link task was being reinitialized.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30D6	An error occurred during DFT operations while a window was being defined for each customized logical terminal.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.

DFT Operations Return Codes - 30xx

Code	Explanation	Action to Take
30D7	An error occurred during DFT initialization operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30D8	An error occurred during DFT initialization operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30D9	An error occurred during DFT operations while a task was being linked to.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30DA	An error occurred during DFT operations while the keyboard was being connected for a logical terminal.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30DB	An error occurred during DFT operations while the keyboard connection was being made.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30DC	An error occurred during DFT operations while a keystroke was received from a logical terminal keyboard.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57. Record the number of logical terminals you are operating and the number of the logical terminal into which you were keystroking.
30DD	An error occurred during DFT operations while the keystroke was being received.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57. Then record the number of logical terminals with which you are operating and the number of the logical terminal into which you were keystroking.
30DE	An error occurred during DFT keystroke operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30DF	An error occurred during DFT keystroke operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.

DFT Operations Return Codes - 30xx

Code	Explanation	Action to Take
30E0	An error occurred during DFT keystroke operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57. Record the number of logical terminals with which you are operating and the number of the logical terminal into which you were keystroking.
30E1	An error occurred during DFT keystroke operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30E2	An error occurred during DFT inbound operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30E3	An error occurred during DFT inbound operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30E4	An error occurred during DFT inbound operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30E5	An error occurred during DFT outbound operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30E6	An error occurred during DFT outbound operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30E7	An error occurred during DFT outbound operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30E8	An error occurred during DFT operations while a task was being linked to.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30E9	An error occurred during DFT keystroke operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30EA	An error occurred during DFT inbound operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.

DFT Operations Return Codes - 30xx

Code	Explanation	Action to Take
30EB	An error occurred during DFT outbound operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30EC	An error occurred during DFT keystroke operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30ED	An error occurred during DFT inbound operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30EE	An error occurred during DFT outbound operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30EF	An error occurred during DFT keystroke operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30F0	An error occurred during DFT keystroke operations.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30F1	An error occurred during DFT operations; a logical terminal number cannot be found.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30F2	An error occurred during DFT operations while 7-color mode was being requested.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30F3	An error occurred during DFT operations while the cursor was being drawn.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30F4	An error occurred during DFT operations while 4-color mode was being requested.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30F5	An error occurred during DFT operations while the cursor was being drawn.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.

DFT Operations Return Codes - 30xx

Code	Explanation	Action to Take
30F6	An error occurred during DFT operations while the OIA was being drawn.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30F7	An error occurred during DFT operations while the OIA was being drawn.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30F8	An error occurred during DFT operations while the cursor was being drawn.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30F9	An error occurred during DFT operations while a character was being drawn.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30FA	An error occurred during DFT operations while the cursor was being drawn.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30FB	An error occurred during DFT operations while the cursor was being drawn.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30FC	An error occurred during DFT operations while the screen was being drawn.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.
30FD	An error occurred during DFT operations while the screen was being drawn.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL the system.

Function ID X'32': Host Interactive Services Return Codes

Return codes beginning with function code X'32' indicate that an error occurred during host interactive services operations, except return code X'3200', which indicates that the requested host interactive service was completed successfully.

Code	Explanation	Action to Take
3200	The request was completed successfully.	None.
3201	The host session is not active.	The port is not geared for the host session that the application attempted to connect to. Do not attempt to connect to a host session that you have no attachment for.
3202	There was an invalid service request parameter.	Check the host session ID, fixed-length queue ID, and task ID. The task ID must be the same one specified on the connect request.
3204	The session is not connected.	Connect to the host interactive services and retry.
3208	A system error occurred.	Follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5, and 6 on page H-57.
320C	Byte 0 of the parameter list was nonzero on request.	Set byte 0 of the parameter list to zero and retry.
3210	Three requesters (the limit) are already connected.	No more than three applications may connect to a host session at one time.
	The message you sent was rejected.	The device is not in a state to receive inbound transmissions. If the host keyboard is inhibited, pressing Clear or Reset may allow the inbound to work on retry.
		For destination/origin, the host application may not have indicated that it wants a reply from the personal computer program. If this does not seem to be the case, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, 4, 5, and 6 at the end of this chapter.

Function ID X'46': CUT Return Codes

Return codes beginning with function code X'46' indicate that an error occurred during CUT operations.

These return codes will be followed by another return code that better describes the problem and the best action to take. Otherwise, follow the "Action to Take" information provided with the return code.

Code	Explanation	Action to Take
4601 thru 4603	An error occurred during the CUT second-level interrupt handler task.	Re-IPL the system or take a system dump. Then follow local procedures and have the dump available as well as the data from "Return Code Error Steps" 1, 2, and 6 on page H-57.
4604 thru 4607	An error occurred during the CUT hardware initialization task.	Re-IPL the system or take a system dump. Then follow local procedures and have available the dump and the data from "Return Code Error Steps" 1, 2, and 6 on page H-57.
4608 thru 4611	An error occurred during the CUT keystroke handling task.	Re-IPL the system or take a system dump. Then follow local procedures and have available the dump and the data from "Return Code Error Steps" 1, 2, and 6 on page H-57.
4612	An error occurred during the CUT keystroke transmit task.	Re-IPL the system or take a system dump. Then follow local procedures and have available the dump and the data from "Return Code Error Steps" 1, 2, and 6 on page H-57.
4613 thru 4616	An error occurred during the CUT outbound control task.	Re-IPL the system or take a system dump. Then follow local procedures and have available the dump and the data from "Return Code Error Steps" 1, 2, and 6 on page H-57.

Function ID X'51': Notepad Operations Return Codes

Return codes beginning with function code X'51' indicate that an error occurred during notepad operations, except return code X'5100', which indicates that the requested notepad service was completed successfully.

Code	Explanation	Action to Take
5100	The requested notepad service was completed successfully.	None.
5101	The notepad cannot connect to the keyboard.	Refer to keyboard problem determination in the <i>Problem Determination Guide and Reference</i> .
5102	The notepad received an error indication on a READ KEYBOARD operation.	Refer to keyboard problem determination in the <i>Problem Determination Guide and Reference</i> .
5103	The notepad received an error indication while attempting a DRAW operation.	This error is generally caused by the incorrect insertion of a patch or code fix. To correct the problem, remove the last series of changes to the Workstation Program. If this does not correct the problem, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, and 6 on page H-57.
5104	The notepad received an error indication during autokey operations.	This error generally occurs by an incorrect insertion of a patch or code fix. To correct the problem, remove the last series of changes to the Workstation Program. If this does not correct the problem, follow local procedures and have the data available from "Return Code Error Steps" 1, 2, and 6 on page H-57.

Function ID X'62': Keyboard Services Return Codes

Return codes beginning with function code X'62' indicate that an error occurred during keyboard operations, except return code X'6200', which indicates that the requested keyboard service was completed successfully.

Code	Explanation	Action to Take
6200	The requested service was completed successfully.	None.
6201	An invalid intercept option was specified on the Connect to Keyboard service request. Connectors that wish to do READs must set one of the intercept option bits on and specify an input queue ID. The first connector to a session input must specify "intercept all" and must provide an input queue ID. Others must specify "both" or "neither."	<p>If the user is doing a READ, disconnect and reconnect using the correct intercept option bit and an input queue. If the user is connecting, ensure that either both or neither of the above inputs is in the parameter list.</p> <p>This error can also occur if the user is trying to connect to a host session that has not yet received a power-on reset from the control unit. (Missing a power-on reset can result from an inoperative control unit, a line configuration mismatch between the control unit and the workstation, or a disconnected coaxial cable.) Byte 9 of the Connect to Keyboard parameter list containing a X'FF' is a further indication of this and should be treated as an abnormal condition. Under these conditions, the session has not yet completed preparations to accept keystrokes.</p>
6202	An error occurred during input operations. An invalid session ID was found in bytes 2 and 3 of the parameter list, or the length specified in the list of keystrokes is greater than 255.	Use the session information services to determine the correct session ID and reissue the request, or correct the length of the list of keystrokes to be less than or equal to 255.

Keyboard Services Return Codes - 62xx

Code	Explanation	Action to Take
6204	<p>On a Connect request, two connections are already made to the requested session ID. No more connections are allowed until one of them disconnects first.</p> <p>On all other keyboard API requests, you are not connected to the session whose ID is in bytes 2 and 3 of the parameter list.</p>	<p>If your program previously did a Connect, you are still connected. You must issue a Disconnect before reconnecting. Otherwise, if your program is not one of those connected, wait and try again later or notify the terminal operator to determine which other program is connected so it may be terminated, allowing yours to run.</p> <p>Determine the correct session ID to use and ensure that Connect is issued before any other API function is used.</p>
6209	On a Read Keystroke request with a No Wait option, there is no keystroke available on the queue.	Poll again for a keystroke, or continue with other processing.
620C	A nonzero return code was passed in byte 0 of the parameter list when the service was requested.	Set byte 0 of the parameter list to zero and retry the request.
6210	An error occurred during input operations. On a Write Keystroke request, the last key sent was rejected either because it was an invalid scan code for the session to which it was sent, or an inhibit condition was present in that session.	Determine whether the last key sent is valid for the target session (for example, the ESC key is invalid for a DFT session or a PA1 key has no meaning to a personal computer session). Other responses depend on the indications present in the target session (for example, if a key was sent and entered into a protected field of a DFT session, a reset key must be sent to clear the inhibit condition before any more keystrokes will be accepted by that session).

Keyboard Services Return Codes - 62xx

Code	Explanation	Action to Take
6212	On a Connect operation, the Connect has been rejected because an autokey record is in progress.	Notify the terminal operator that the autokey operation must be terminated before this or any keyboard API function can be processed.
	On a Write Key operation, the last key sent was detected as an AID key. If the user is sending a list of keys, the processing of that list ended with that key.	If the user is processing a list of keys, determine how many of them were sent by looking at byte 7 of the returned parameter list. When the session is able to receive more keys, send the remainder.
	<i>Note: This does not mean that an error occurred.</i>	
6214	On a Connect operation, the connect was rejected because an autokey playback is in progress.	Notify the terminal operator that the autokey operation must be terminated before this or any other keyboard API function can be processed.

Function ID X'63': Window Management Services Return Codes

Return codes beginning with function code X'63' indicate that an error occurred during work station control operations, except return code X'6300', which indicates that the requested window management service was completed successfully.

Code	Explanation	Action to Take
6300	The requested service was completed successfully.	None.
6301	There is no space for additional windows.	To add a window, a window must first be deleted from this or another screen.
6302	An invalid session ID was specified. The ID did not match the one specified on the Connect to Work Station Control request. If the function that failed was the connect, then the session ID specified is not within the valid range of session IDs.	Use the session ID that was specified on the Connect to Work Station Control service request to perform the function. If this return code occurs during the connect process, then the proper session ID for the session is needed.
6303	There is not enough storage to relocate initialization code.	Additional storage must be obtained in order to load the module.
6304	The caller is not connected to the work station control session. The work station control session is already in use by one of the following: <ul style="list-style-type: none">• Another application program• The user (by pressing the WSCRTL key)• The workstation program.	Connect to the workstation control session before attempting to perform a function. Try to connect to the workstation control session when it is available.
6305	The specified window ID already exists on the specified screen ID.	Either delete the desired window from the screen (so it may be put back later) or specify another window ID to be added.

Window Management Services Return Codes - 63xx

Code	Explanation	Action to Take
6306	An invalid screen ID was specified. The desired screen either does not exist or cannot be used for the requested function.	Specify a valid ASCII screen ID to the function.
6307	The specified window ID was not found on the specified screen ID.	Specify a valid ASCII ID of a window on a screen to perform the function.
6309	The specified window ID was not found on Screen 0.	Specify a window ID of a window that exists on Screen 0.
630A	The user attempted to hide a window when it is the only window on the screen.	Add at least a second window before attempting to hide a window.
630B	All windows on the screen are hidden; the next window on the chain will be unhidden.	None.
630C	A nonzero return code was passed in byte 0 of the parameter list when the request was issued.	Set byte 0 of the parameter list to zero and retry the request.
630D	The specified screen ID is not that of the active screen.	Specify the ID of the active screen or make the desired screen active to perform the function.
630E	No windows exist on the specified screen ID.	The function cannot be performed when no windows exist on the requested screen.
630F	Colors cannot be set on for a PC session.	Provide a non-PC window ID to set colors.
6310	Either the row or column values sent caused the window not to fit fully on the screen or presentation space, or one or both of the values were equal to zero.	This is an informational return code. The window has been placed on the screen but has been modified to allow it to fit on the screen with correct values. The changes will be sent back via the parameter list.

Window Management Services Return Codes - 63xx

Code	Explanation	Action to Take
6311	Some or all of the values sent in the parameter list were either not correct or caused the window not to fit fully on the screen or presentation space.	This is an informational return code. The window has been placed on the screen but has been modified to allow it to fit on the screen with correct values. The changes will be sent back to the calling program via the parameter list.
6312	The foreground and background colors are the same.	This is an informational return code. You should change colors as desired.

Function ID X'64': Copy Services Return Codes

Return codes beginning with function code X'64' indicate that an error occurred during copy operations, except return code X'6400', which indicates that the requested copy service was completed successfully.

Code	Explanation	Action to Take
6400	The requested copy service was completed successfully.	None.
6401	The selected source is not allowed. It is a personal computer window in graphics mode.	Select a different source.
6402	An invalid session ID was passed on request to the Connect or Disconnect for Copy to Personal Computer Session services. The specified session is not a personal computer session.	Correct the session ID and retry.
6403	Input is inhibited in the target. A copy operation was attempted while the keyboard was in an input-inhibited state for the selected target window.	<ol style="list-style-type: none">1. Wait for the keyboard to "unlock."2. Try the copy again.3. Verify that the host is operating. If the keyboard remains locked, refer to keyboard problem determination in the <i>Problem Determination Guide and Reference</i>.
6404	There is not enough storage to relocate the initialization code.	Additional storage must be obtained in order to load this module.
6405	Warning: There is an overlapping source and target area. The copy was successful.	Verify the target area.
6406	The source definition in the parameter list is missing a parameter or has invalid information.	Correct the source definition and retry the copy.
6407	The target definition in the parameter list is missing a parameter or has invalid information.	Correct the target definition and retry the copy.

Copy Services Return Codes - 64xx

Code	Explanation	Action to Take
6409	Warning: The source and target are not the same size. If the source is larger than the target, truncation occurs. If the source is smaller than the target, the target area is padded with blanks and copy.	Verify the target area.
640C	The return code passed in the parameter list on request was not zero.	Set the return code field in the parameter list to zero and retry.
640D	The selected target is not allowed. Either the selected target is a PC window that did not do a copy connect first, or the PC target is in graphics mode.	Select a different target.
640E	The target window is protected.	Redefine the target area.
640F	The copying of field attributes is not allowed unless the target window is a PC window that is attached to 3270 keystroking or the target is a PC buffer form.	Attach the window to 3270 keystroking or remove the bit in the parameter list to copy field attributes or make the target a PC buffer.

Function ID X'67': Draw Service Return Codes

Return codes beginning with function code X'67' indicate that an error occurred during draw operations, except return code X'6700', which indicates that the requested draw service was completed successfully.

Code	Explanation	Action to Take
6700	The draw request was completed successfully.	None.
6703	There is not enough storage to relocate initialization code.	Obtain additional storage in order to load the module.
6708	The parameter list definition has a missing parameter on the request.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL.
670C	The return code passed in the parameter list on request was not zero.	Follow your local procedures, and have the data available from "Return Code Error Steps" 1, 2, 5, 6, and 7 on page H-57, or re-IPL.

Function ID X'69': Presentation Space Services Return Codes

Return codes beginning with function code X'69' indicate that an error occurred during presentation space operations, except return code X'6900', which indicates that the requested presentation space management service was completed successfully.

Code	Explanation	Action to Take
6900	The requested presentation space management service was completed successfully.	None.
6902	The specified session ID is unknown.	The Define Presentation Space function will return the session ID to be used with all subsequent requests concerning this new presentation space. Ensure that the specified session ID is one returned from the Define Presentation Space request.
6903	The specified offset for the display is not within the address of the presentation space.	Correct the offset supplied in the parameter list and retry.
6906	An invalid cursor type was specified in the parameter list for the Display Cursor service request.	Correct the cursor type and retry.
6907	An invalid cursor address was specified in the parameter list for the Display Cursor service request.	Correct the cursor address and retry.
6909	The specified length is invalid in the parameter list for the Display Presentation Space service request.	Correct the length and retry.
690A	An invalid number of commands are in the presentation space data stream.	Correct the number of commands in the header of the presentation space data stream and retry.
690B	An invalid number of rows/columns are in the presentation space data stream for the Define Presentation Space service request.	Correct the row/column information in command type 1 of the presentation space data stream and retry.

Presentation Space Services Return Codes - 69xx

Code	Explanation	Action to Take
690C	Byte 0 of the parameter list is nonzero on request.	Set byte 0 of the parameter list to zero and retry.
690D	There is invalid data in the Set Presentation Space Type data stream command of the Define Presentation Space service request.	Correct the presentation space type and retry.
690F	A command that had no data was found in the presentation space data stream of the Define Presentation Space service request.	The address supplied on command 03 or command 04 was zero. Correct and retry.
6910	A Delete Presentation Space request was issued for a session ID that is an initial resource (that is, a configured personal computer session).	A Delete Presentation Space request can only be issued for a presentation space that was previously defined by the Define Presentation Space request.
6911	One or more of the input parameters is not valid.	Review input parameters and ensure they are valid.
6913	The address of the work area on request to the Define Presentation Space service was zero.	Correct the work area segment address and retry the request.
6914	The Define Presentation Space service was requested, and the maximum number of PC presentation spaces has already been created.	Another Define Presentation Space request cannot be completed until an existing presentation space is deleted.
6915	The Set Presentation Space Buffer command was missing from the presentation space data stream of the Define Presentation Space service request.	Correct the presentation space data stream to include command 03 and retry.
6918	The Set Presentation Space Size command was missing from the presentation space data stream of the Define Presentation Space service request.	Correct the presentation space data stream to include command 01 and retry.

Presentation Space Services Return Codes - 69xx

Code	Explanation	Action to Take
6919	The Set Presentation Space Type command was missing from the presentation space data stream of the Define Presentation Space service request.	Correct the presentation space data stream to include command 02 and retry.

Function ID X'6B': Session Information Services Return Codes

Return codes beginning with function code X'6B' indicate that an error occurred during session management operations, except return code X'6B00', which indicates that the requested session information service was completed successfully.

Code	Explanation	Action to Take
6B00	The requested service was completed successfully.	None.
6B01	All short window names are currently in use.	Delete a window to free a short window name.
6B02	The session ID in the parameter list is outside the legal range.	Correct the session ID in the parameter list and retry.
6B03	The long window name was not found in the session manager table.	Check that the long name is in ASCII. Then check the long name spelling. Correct and retry.
6B05	Too many attachments were made. The maximum attachments allowed are 255.	You must request the Detach Session ID service for the given session ID before further attachments will be allowed.
6B06	The session ID in the parameter list was not found in the session manager table, indicating that the session is no longer defined.	Correct the session ID in the parameter list and either retry or ignore the error.
6B07	The short window name is already in use.	Choose an unused short window name and retry.
6B09	There is an invalid type field in the parameter list.	Correct the parameter list and retry.
6B0A	The environment ID in the parameter list was not found in the session manager table. This indicates that either the specified environment ID is invalid or the specified environment ID was valid at one time but is not currently active.	Correct the environment ID in the parameter list and either retry or ignore the error.

Session Information Services Return Codes - 6Bxx

Code	Explanation	Action to Take
6B0B	The window short name was not found in the session manager table.	Correct the window short name in the parameter list and either retry or ignore the error.
6B0C	The return code in the parameter list is not zero on call.	Set the return code field in the parameter list to zero and retry.
6B0D	There is an invalid option code in the parameter list.	Correct the option code and retry.
6B0E	The base window was not found. This indicates that either the specified environment ID is invalid or the specified environment ID was valid at one time but is not currently active.	Correct the environment ID in the parameter list and either retry or ignore the error.
6B0F	There are no available entries in the session manager table. No additional session can be established.	Detach a session ID or wait until a session ID becomes free.
6B11	The session type was not found in the session manager table.	Correct the type field in the parameter list or ignore the error.
6B12	The length of the name array is incorrect.	Correct the name array length and retry.
6B13	The window short name is not in uppercase ASCII alphanumeric characters.	The short window name must be uppercase ASCII alphanumeric characters. Correct and retry.
6B14	Cannot detach from this session now.	Check to make sure you have not issued more detaches than attaches.

Function ID X'6C': Translate Services Return Codes

Return codes beginning with function code X'6C' indicate that an error occurred during translate operations, except return code X'6C00', which indicates that the requested translate service was completed successfully.

Code	Explanation	Action to Take
6C00	The requested service was completed successfully.	None.
6C01	There is an invalid translate type in the parameter list.	Change the translate type in the parameter list and retry.
6C0C	Byte 0 of the parameter list was nonzero on request.	Set byte 0 of the parameter list to zero and retry.

Function ID X'6D': OIA Services Return Codes

Return codes beginning with function code X'6D' indicate that an error occurred during operator information area operations, except return code X'6D00', which indicates that the requested operator information area service was completed successfully.

Code	Explanation	Action to Take
6D00	The requested service was completed successfully.	None.
6D02	A invalid session ID was specified in the parameter list.	Correct the session ID and retry.
6D0C	Byte 0 of the parameter list was nonzero on request.	Set byte 0 of the parameter list to zero and retry.

Function ID X'6E': 3270 Keystroke Emulation Services Return Codes

Return codes beginning with function code X'6E' indicate that an error occurred during 3270 keystroke emulation operations, except return code X'6E00', which indicates that the requested 3270 keystroke emulation service was completed successfully.

Code	Explanation	Action to Take
6E00	The requested service was completed successfully.	None.
6E02	On a Connect request, the specified presentation space has not been defined to accept 3270 keystroking emulation.	Use the Define Presentation Space service to create a presentation space that is defined to accept 3270 keystroking emulation. Specify this presentation space on the Connect for 3270 Keystroke Emulation service request.
	On a Read AID Key request, the specified presentation space has not been connected for 3270 keystroking emulation.	Correct the specified session ID and request the READ AID Key service again.
6E08	A system error occurred during 3270 keystroke emulation operations.	Follow local procedures and have the data available from return code error step 9 at the end of this chapter.
6E0C	Byte 0 of the parameter list was nonzero on request.	Set byte 0 of the parameter list to zero and retry.

Function ID X'6F': Keystroke Definition Return Codes

Return codes beginning with function code X'6F' indicate that an error occurred during 3270 keystroke definition initialization, except return code X'6F00', which indicates that the requested 3270 keystroke definition service was completed successfully.

Code	Explanation	Action to Take
6F00	The keystroke definition initialization was completed successfully.	None
6F01	An ID request was issued to the keyboard with no response.	Refer to the <i>Guide to Operations</i> and run the keyboard diagnostics.
6F02	An unsupported or invalid ID was returned from the keyboard.	Check that the switch settings on the Model 1A keyboard are off. If they are off, the keyboard is defective. In other cases, the keyboard is defective or incompatible.

Function ID X'72': Error Handler Return Codes

Return codes beginning with function code X'72' indicate that an error occurred during error handler operations.

Code	Explanation	Action to Take
7201	A component is trying to report an undefined return code to the error handler.	This return code is followed by a second return code. Follow the directions given in the "Action to Take" column for that return code.
7202	A component is trying to add a return code to the error handler error table, but the table is full.	This return code is followed by a second return code. Follow the directions given in the "Action to Take" column for that return code.
7203	A component is trying to add a return code to the error handler error table with an invalid severity.	This return code is followed by a second return code. Follow the directions given in the "Action to Take" column for that return code.
7204	A dump was requested using "TRACE OFF /d."	None.
7205	A dump was requested by pressing the NMI pushbutton.	None.
7206	A dump was requested by pressing the ALT + CTRL + TEST keys.	None.

Function ID X'7F': Dump Task Return Codes

Return codes beginning with function code X'7F' indicate that an error occurred during dump task operations.

Code	Explanation	Action to Take
7F01	An error occurred during system startup before the error handler could be loaded into storage and successfully initialized.	Re-IPL the system. If the error recurs, follow local procedures and submit a problem report.
7FFF	An error occurred in the multiple DOS portion of the workstation program before the error handler could be loaded into storage and successfully initialized.	Re-IPL the system. If the error recurs, follow local procedures and submit a problem report.
7F02	An error occurred while a graphics application was being run without a graphics adapter.	Press any key to re-IPL your system. If the error recurs, follow local procedures and submit a problem report.
7F03	A graphics application is already in progress in another PC window.	Press any key to re-IPL your system. If the error recurs, follow local procedures and submit a problem report.

Function ID X'81': Enhanced Connectivity Router Return Codes

Return codes beginning with function code X'81' indicate that an error occurred during enhanced connectivity router operations.

Code	Explanation	Action to Take
8101	No DFT sessions exist.	Rerun customization to add a DFT session.
8104	Insufficient space to relocate initialization code.	Obtain additional storage in order to load the module.

Function IDs X'Dx through Fx': User System Extension Return Codes

Return codes with a function ID of X'Dx' through X'Fx' are generated by user-supplied system extensions. Consult local documentation for the meaning of these return codes and the action to take when they are encountered.

Return Code Error Steps

Use these steps only when you are directed to do so by 'action-to-take' instructions in this chapter.

1. Record the return code.
2. Record the sequence of events that caused the failure, including the keys pressed and in what order.
3. Turn on the Trace events:
 - a. 95 96, 97, 98, 99, 101, and 102
 - b. 93, 94, 101, and 102
 - c. 101 and 102
4. Rerun the application that caused the error until the error recurs.
5. If the problem persists, issue the command `TRACE OFF /D` to take a system dump.
6. Record the system level. To do this, look at your APAR list as described in the *Problem Determination Guide and Reference*.
7. Record the system configuration, which is a list of the hardware, including installed options. This may be found in the *Guide to Operations* and the contents of the summary panels.
 - a. Insert the customized system diskette in the active drive.
 - b. If you have a printer, type:
`TYPE INDCFIG.FIL>PRN` and press Enter.
 - c. If you do not have a printer, type:
`MORE < INDCFIG.FIL` and press Enter.
Write down the contents of the summary panels.
8. Increase the resource requirements in the SIF for the system extension in which the error occurred. Refer to the *User's Guide* for information on SIFs.
9. The return code received was accompanied by a message to take a dump. Record the return code and take a dump if the error persists.

Return Code Error Steps

Appendix I. Outbound Data Stream Preprocessor (ODSP) Option

Introduction	I-2
Customizing for ODSP	I-2
Initializing ODSP	I-2
Using ODSP	I-3
Entry Parameters	I-4
Return Parameters	I-4
ODSP Restrictions and Recommendations	I-5
Sample Program for Outbound Data Stream Preprocessing	I-5

Introduction

The Outbound Data Stream Preprocessor (ODSP) Option allows you to preprocess a 3270 outbound data stream with a user system extension program. To preprocess an outbound data stream, you must customize your workstation program for ODSP.

Customizing for ODSP

Before you can customize your workstation program for ODSP, you must create a user system extension program. Once you have done this, update the customization panels as explained below:

1. Update the Home Panel of customization where it says System Extensions under WORKSTATION PROGRAM OPTIONS. If this is your only user system extension program, type a 1 under System Extensions on the Home Panel. If you already have a number of system extension programs indicated, increase this number by 1.
2. Complete customization Panel 1.1, filling in all the pertinent information about your user system extension program.
3. Type "yes" under ODSP on Panel 2 of customization. As a result of indicating "yes" to ODSP, the workstation program creates a user exit table called INDODSP. INDODSP contains four 4-byte routine address entries. Each entry corresponds to one of your four possible host sessions and will be used when you initialize ODSP.

See the *IBM 3270 Workstation Program User's Guide and Reference* for more information on how to customize for ODSP.

Initializing ODSP

Now that you have customized for ODSP, each time you IPL your system the workstation program loads and gives control to the user system extension program you just specified in customization. That system extension program must do the following:

- Issue the Supervisory Object Service X'81':Name Resolution, to locate the user exit table named INDODSP.
- Issue the Supervisory Object Service X'0E':Install User Exit Table Entries, to initialize the INDODSP table, which contains the routine addresses that will process the outbound data stream for each host session. If you wish to preprocess data streams from a subset of host sessions, then fill in the entries pertaining to those host sessions only.

- After initialization, return to DOS using the Exit and Remain Resident function.

See Chapter 15 for more information on name resolutions and installing user exit table entries.

Using ODSP

When an outbound data stream is received and a routine address exists in the user extension table, a parameter list is presented to the user system extension routine containing pointers and count information pertaining to the data stream currently in process.

ES = Segment address of the parameter list

DI = Offset address of the parameter list

The parameter list has the following format on entry to and return from your user system extension routine:

Offset	Length	Contents on Entry	Contents on Return
0	1 word	Offset address of current buffer	Offset address of current buffer
2	1 word	Segment address of current buffer	Segment address of current buffer
4	1 word	Count of bytes in current buffer	Count of bytes in current buffer
6	1 byte	Chain indicator	Reserved
7	1 byte	Host session number	Host session number
8	1 word	Reserved	Offset address of supplementary buffer
10	1 word	Reserved	Segment address of supplementary buffer
12	1 word	Reserved	Count of bytes in supplementary buffer

Entry Parameters

- The current buffer address contains offset and segment of outbound data stream.
- The current buffer count contains the count of bytes present in the current buffer.
- Chain indicator
 - '100xxxxx'X First
 - '010xxxxx'X Last
 - '000xxxxx'X Middle
 - '110xxxxx'X Only
 - 'xx1xxxxx'X Local channel command
- The host session number contains the number (0 to 3) of the host session for which the data stream was received. This will prevent the user extension code from having to declare separate entry points to determine the host session number.

Note: For locally attached 3274/3270PCs, the command is sent first and is followed by the remaining 3270 data stream, if present. The user system extension will first be passed the command and, subsequently, will be called with the data as it is received.

Return Parameters

- The current buffer address contains offset and segment of the data stream buffer to process first.
- The current buffer count contains the count of bytes present in the data stream buffer to process first.
- The supplementary buffer address contains offset and segment of the data stream buffer to process second.
- The supplementary buffer count contains the count of bytes present in the data stream buffer to process second.

Note: No change in the Supplementary Buffer fields is necessary if no supplementary buffer is provided. If you wish the user extension to process a stored data stream first, move the current buffer address and count in the parameter list to the respective fields for the supplementary buffer. Then store the buffer address and count provided by the user extension in the respective fields of the current buffer.

You may change data in the current buffer and use the address and count fields of the parameter list to shorten it, either at the front (increase the address, decrease the count) or at the end (leave address the same, decrease the count). Do not append data to either end of the current buffer. This may cause unpredictable results and eventual disconnection from the control unit. 3270 buffer addresses (12/14-16 bit) should be consistent with the current session.

ODSP Restrictions and Recommendations

The User System Extension routines that preprocess outbound data streams for each logical terminal operate as an internal 3270 Workstation Program subroutine. Therefore, the following design restrictions must be observed to avoid time-out problems with the control unit:

- Avoid system waits, including implied waits for I/O, and other workstation program API functions.
- Do not disable interrupts.

Note: Routines requiring lengthy processing time degrade performance.

In a multiple-host-session configuration, when processing a data stream for one host session, all other host sessions will be locked out from processing data streams. Also, this routine gets control from the workstation program data stream processor so errors could cause damage to the workstation program or system control blocks and modules.

For systems with XMA, the user system extension resides in common memory and, therefore, should be as compact as possible, since it will reduce the size of your PC session. See Chapter 24 for more information about user system extensions.

Sample Program for Outbound Data Stream Preprocessing

Use the following as a sample user system extension program for the ODSP option of the 3270 Workstation Program.

```

;
; ODSPTTEST.ASM <===== Test code for Outbound Data Stream
;                      Preprocessing Option
;
; It is intended for use in testing the Data Stream Preprocessor
; option of 3270 Workstation Program. It operates under all three
; transmission environments (i.e. Local Chnl, SNA, and Bisynch).
; Three variations of data stream modification are tested:
;   - Prefixing a stored data stream to the beginning of the
;     current host outbound data stream.
;   - Post fixing a stored data stream to the end of the
;     current host outbound data stream.
;   - Modification of the data stream without pre or post fixing.
; The host outbound data stream contains an escape character to
; signal which of the operations are required (if any).
; The character follows the WCC character in the data stream and
; is followed in turn by an eight character format name. The
; escape characters are:
;   - The FM character (1EH) is used to signal pre fixing of a
;     stored format.
;   - The DUP character (1CH) is used to signal post fixing of a
;     stored format.
; If modification only is required then the stored format name is
; ended with an 'R'.
;
; =====> Program Operation <=====
;
; The parameter list is first moved into local storage.
;
; If the data stream segment is first in chain Then
;   If the character following the WCC is Field Mark (1EH) Then
;
;     Pick up the next eight characters as a stored data stream name
;     and place the stored format as first in the processing.
;
;   Else If the character is DUP (1CH) Then
;
;     Flag for later processing and pick up the next eight characters
;     as the stored format name to be appended to the data stream.
;
; If the data stream segment is last in chain Then
;
;   If an end of chain escape character (i.e. DUP) Then
;     Append stored data stream to current data stream
;
; Move local copy of parameters to 3270 PC's copy
;
; Return
;
; =====
;
; Initialization attempts to read an ODSP control file 'ODSPSF.CTL'
; from drive C:\ODSPTEST directory, if not found, drive A: is used.
; All test data streams are defined in the control file and are
; read into a control area of ODSPTTEST storage.
; Each directory named ODSPTTEST on which ever drive contains
;   ODSPSF.CTL
;
;
;
;
;
;

```

```

;-----
; EQU Section
;-----
ESCP_INC      EQU      10          ;Escape character increment
NAM_SIZE      EQU      8          ;Maximum size of a stored name
SFCFSIZE     EQU      NAM_SIZE + 11 ;Size of record in SFCF
MAX_NTRS      EQU      16         ;Maximum entries in SFI
MAX_SIZE      EQU      1920       ;Maximum size of a screen
;
BLANK         EQU      BYTE PTR ' ' ;Blank space
A             EQU      BYTE PTR 'A' ;ASCII character A
DOSOPEN       EQU      3DH        ;DOS interrupt to open a file
DISPLAY       EQU      9H         ;DOS interrupt to display char.
DOSCALL       EQU      21H        ;Call to DOS commands
FIRSTINC      EQU      80H        ;First in chain flag
LASTINC       EQU      40H        ;Last in chain flag
LCLCMD        EQU      20H        ;Local channel command flag
POSTFLAG      EQU      01H        ;Post fix flag
DATA_NOP      EQU      13H        ;Data stream NOP
;
;-----
PROGRAM        SEGMENT            ;Define program segment
                ASSUME CS:PROGRAM, DS:PROGRAM, ES:PROGRAM ;
                ORG      0100H     ;Make into a COM file
;
;-----
; DEFINE THE PROGRAM SEGMENT AND SET UP STACK TO MAIN PROGRAM
;-----
;
START:
                JMP      INIT      ;Start Initialization
;
                PAGE
;-----
; Define the structures to be used
;-----
;
; * Input Parameters *
;
INPUT_CB       STRUC
    OFFSTCB    DW      0          ;OFFSET of current buffer
    SEGADCB    DW      0          ;SEGMENT of current buffer
    LENCURB    DW      0          ;Length of current buffer
    CHAINID    DB      0          ;Chain Indicator
    HOSTNUM    DB      0          ;Host Session
    OFFSTSB    DW      0          ;OFFSET of Suppl. buffer
    SEGADSB    DW      0          ;SEGMENT of Suppl. buffer
    LENSUPB    DW      0          ;Length of Suppl. buffer
INPUT_CB       ENDS
;
; * Stored Format Name Table *
;
DEFSFI         STRUC              ;Definition of SFI
    SCREENME   DB      NAM_SIZE DUP (' ') ; EBCDIC Name
    SCREENLN   DW      0          ; Length of Screen Data
    SCREENAD   DW      0          ; Offset of Screen Data
DEFSFI         ENDS
;
                PAGE

```

```

;-----
; Define the working storage to be used
;-----
;
LOCAL_CB      INPUT
_CB <>        ;Local copy of Control Block
;
; * Stored Format Information *
;
COUNT        DW      0          ;Count length of screen
;
SF_LOC         DW      0          ;Stored format offset
SF_COUNT       DW      0          ;Stored format count
ESCAPE_LOC     DW      0          ;Location of escape code
LCLFLAGS       DB      0          ;Local processing flags
;
;
; Error message
; Erase the screen and highlite top line
NO_FIND        DB      0F1H,0C7H  ; Write command with unl. kbd.
               DB      011H,040H,040H ; Set buffer address at R1/C1
               DB      03CH,05CH,0F0H,000H ; RA to R24/C1 (00H)
               DB      012H,040H,040H ; EUA R1/C1
               DB      011H,040H,040H ; Set buffer address at R1/C1
               DB      01DH,0E8H      ; Start field (prot. + hilite)
;
               DB      0C5H,0D9H,0D9H,0D6H,0D9H,040H,05CH,05CH,05CH,040H
;
               DB      0E4H,095H,081H,082H,093H,085H,040H,0A3H,096H,040H
;
               DB      093H,096H,083H,081H,0A3H,085H,040H,0A2H,0A3H,096H
;
               DB      099H,085H,084H,040H,086H,096H,099H,094H,081H,0A3H
;
               DB      040H,095H,081H,094H,085H,084H,040H,07EH,07EH,06EH,040H
;
REQFNAME       DB      NAM_SIZE DUP(0) ;Requested format name (EBCDIC)
;
NO_FINDL       EQU      OFFSET SFI - OFFSET NO
_FIND          ;Length of message
;
SFI             DEFSFI  MAX_NTRS + 1 DUP (<>) ;Screen Format Information
;
               PAGE
;-----
;=====> Main Procedure <=====
;-----
;
MAIN           PROC      FAR
;-----
; Move parameter information to local storage
;-----
;
               PUSH     ES          ;Save pointers to Control Block
               PUSH     DI          ;
               PUSH     CS          ;Set up to put code segment into
               POP      DS          ; DS (i.e COM file format)
               CLD                ;Clear direction flag - auto inc.
               MOV      CX,TYPE LOCAL_CB ;number of bytes to transfer
               LEA      SI,LOCAL_CB.OFFSTCB ;OFFSET of local CB
;
               XCHG     DI,SI       ;Put contents of parameter
               PUSH     ES          ; list into my data segment

```

```

        POP     DS                ; to free up ES and DI
        PUSH    CS                ; registers
        POP     ES
    REPE MOVSB                    ;
;
        PUSH    CS                ;Restore DS
        POP     DS                ;
        MOV     DI,LOCAL_CB.OFFSTCB ;Put address of cur. buffer
        MOV     ES,LOCAL_CB.SEGADCB ; into ES:DI registers
;
        PAGE
;-----
; Check for process to initiate
;-----
;
                                ;IF Data is first in chain
        TEST    LOCAL_CB.CHAINID,FIRSTINC
        JZ      CK_LAST
;
        CALL    PRCFIC           ; Then Process the block for
                                ; possible stored format
;
CK_LAST:                                ; If data is last in chain
        TEST    LOCAL_CB.CHAINID,LASTINC
        JZ      CK_LCLCM
;
        CALL    PRCLIC           ; Then Process the block for
        JMP     MOVE_LCL         ; possible stored format
;
CK_LCLCM:                                ;Else If local channel command
        TEST    LOCAL_CB.CHAINID,LCLCMD
        JZ      MOVE_LCL
;
        CALL    PRCLCLCM         ; Process local command
;
;-----
; Move local copy of parameters to calling routine parameters
;-----
;
MOVE_LCL:
        CLD                    ;Clear direction flag (auto inc.
        MOV     CX,TYPE LOCAL_CB ;Number of bytes in Control Block
        LEA     SI,LOCAL_CB.OFFSTCB ;OFFSET of current buffer
        POP     DI                ;ES:DI from original call
        POP     ES                ;
    REPE MOVSB                    ;Parameter values to Caller's buf.
        SUB     DI,TYPE LOCAL_CB ; adjust DI after move
;
MAIN_RET:                                ;Return point for main procedure
        RET                     ; FAR return to procedure
;
MAIN                                     ENDP
;
        PAGE
;-----
;=====> Sub-Procedures <=====
;-----
;
;-----
;=====> Process Current Buffer First in Chain Procedure
;-----
PRCFIC    PROC    NEAR                ;Process stored format
        TEST    LCLFLAGS,LCLCMD ;

```


ODSP

```

        JNZ      PRCB_GO
;
        INC      DI                ; Increment past command for
PRCB_GO:                ; remote NON-SNA or for SNA
        INC      DI                ; Increment past WCC
        CMP      ES:BYTE PTR [DI],1EH ; If prefixing of format
        JE       PRC_PREFIX        ; is flagged Then process
;                                ; Else
        CMP      ES:BYTE PTR [DI],1CH ; If post fixing of format
        JE       PRC_POSTFIX       ; is flagged Then set up
;
        RET                        ; Else Return
;
;-----
; Move requested screen format name to local storage
;-----
PRC_POSTFIX:                ;Post fix processing set up
        OR       LCLFLAGS,POSTFLAG ;Turn on postfix flag
PRC_PREFIX:                ;Prefix processing and Post fix
        MOV      ESCAPE_LOC, DI    ;Save offset of escape code
        INC      DI                ;Increment past escape code
        MOV      CX,NAM_SIZE       ;Number of bytes to transfer
        LEA      SI,REQFNAME       ;Offset of local copy
        PUSH     DS                ;Save current DS
        PUSH     ES                ;Swap ES and DS around
        POP      DS                ;
        PUSH     CS                ;
        POP      ES                ;
;
        XCHG     DI,SI             ;Put contents of name field
        REP      MOVSB             ; into local storage
        POP      DS               ;Restore DS
;
;-----
; Compare name in control bytes with names in SFI table
;-----
;
        STD      DI                ;Set for auto-decrement
        LEA      DI,SFI + NAM_SIZE - 1 ;Addr. of end of 1st name
        LEA      SI,REQFNAME + NAM_SIZE - 1 ;Address of current
        MOV      CX,COUNT          ;Number of names to check
COMPARE:                ;
        PUSH     CX                ;Save count of names
        PUSH     DI                ;Save offset into SFT
        PUSH     SI                ;Save offset of current name
        MOV      CX,NAM_SIZE       ;Screen name length
;
        REPE     CMPSB             ;Compare screen names in SFI
;                                ; with name in control bytes
        POP      SI               ; Restore offset to cur name
        POP      DI               ; Restore offset to end
        POP      CX               ; Restore count of names
        JE       FOUND2           ;
;                                ; If names are not same Then
        ADD      DI,TYPE DEFSFI    ; increment to next in table
        LOOP     COMPARE          ;
;                                ; If all names are considered
;                                ; we are pointed at err msg.
;
;-----
; Move and update the current buffer values

```

```

;-----
FOUND2:                                ;When the names match or overrun
        SUB     DI,OFFSET SFI + NAM_SIZE - 1 ;Displ. in SFI
        MOV     AX,SFI.SCREENLN [DI] ;Length of stored format
        MOV     SF_COUNT,AX
        MOV     AX,SFI.SCREENAD [DI] ;Address of stored format
        MOV     SF_LOC,AX
;
        MOV     CX, NAM_SIZE + 1 ; Nop the escape code and name
        MOV     ES, LOCAL_CB.SEGADCB
        MOV     DI, ESCAPE_LOC ; in the current buffer
NOP_NAME:
        MOV     ES:BYTE PTR [DI], DATA_NOP
        INC     DI
        LOOP    NOP_NAME
;
        TEST    LCLFLAGS,POSTFLAG ;When post fix requested
        JNZ     FOUND3 ; Skip Buffer update
;
        LEA     SI,REQFNAME + NAM_SIZE - 1 ; When the last char
        CMP     BYTE PTR [SI],0D9H ; of SF name is R Then
        JE      UPD_OUTP ; Skip Supplemental update
;
        MOV     DX,LOCAL_CB.OFFSTCB ;Move current buffer address
        MOV     LOCAL_CB.OFFSTSB,DX ; to supplemental position
;
        MOV     DX,LOCAL_CB.SEGADCB ;Same segment address
        MOV     LOCAL_CB.SEGADSB,DX ;
;
        MOV     DX,LOCAL_CB.LENCURB ;Move current buffer length
        MOV     LOCAL_CB.LENSUPB,DX ; to suppl position
;
        MOV     AX,ESCP_INC ;Set up modifying constant
        TEST    LCLFLAGS,LCLCMD
        JNZ     PRCB_MIN ; either from local channel
;
        INC     AX ; or from bisynch
PRCB_MIN:
        SUB     LOCAL_CB.LENSUPB,AX ;Decrement length
        ADD     LOCAL_CB.OFFSTSB,AX ;Increment offset
;
;-----
; Update the output parameter list
;-----
;
UPD_OUTP:
;
        MOV     AX,SF_COUNT
        MOV     LOCAL_CB.LENCURB,AX
        MOV     AX,SF_LOC
        MOV     LOCAL_CB.OFFSTCB,AX
        MOV     LOCAL_CB.SEGADCB,DS ;data segment has segment
; address of SFT and SFI
        TEST    LCLFLAGS,LCLCMD ;When dealing with the
        JZ      FOUND3
; local channel,
        INC     LOCAL_CB.OFFSTCB ; increment the offset address
        DEC     LOCAL_CB.LENCURB ; decrement the length
FOUND3:
        RET
;
PRCFIC      ENDP ;Return from subprocedure
;-----

```

ODSP

```

                                PAGE
;-----
;=====> Last in Chain Procedure
;-----
PRCLIC      PROC      NEAR
;-----
;
;          TEST      LCLFLAGS,POSTFLAG ;If post fixing requested
;          JZ        PRCLIC_END
;
;                      THEN DO
;          LEA        SI,REQFNAME + NAM_SIZE - 1 ; When the last char
;          CMP        BYTE PTR [SI],0D9H      ; of SF name is R Then
;          JE         PRCLIC_END              ; Skip Supplemental update
;
;          MOV        DX,SF_LOC                ; move saved buffer address
;          MOV        LOCAL_CB.OFFSTSB,DX      ; to supplemental position
;
;          MOV        LOCAL_CB.SEGADSB,DS      ; using DS for segment addr.
;
;          MOV        DX,SF_COUNT              ;move saved buffer length
;          MOV        LOCAL_CB.LENSUPB,DX      ; to suppl position
;
PRCLIC_END:  MOV        LCLFLAGS,0              ;Reset local flag regardless
;          RET
PRCLIC      ENDP
;          PAGE
;-----
;=====> Local Channel Command Procedure
;-----
PRCLCLCM    PROC      NEAR
;-----
;
;          MOV        AL,LOCAL_CB.CHAINID
;          MOV        LCLFLAGS,AL              ;Save local chnl flag
;
;          RET
PRCLCLCM    ENDP
;          ;Return from subprocedure
;-----
                                PAGE
;-----
;=====> Init Procedure <=====
;-----
;
; Define the Init storage to be used
;-----
;
SERVNAME     DB      'INDODSP '                ;Name of User Exit Table which
;                                                ; is resolved into a UET ID
;
UET_ID       DW      ?                          ;User exit table resolved ID
;
NUMUETE      DW      0                          ;parameter
ENTRY1       DW      0                          ;list
OFENTRY1     DW      0                          ;to
SGENTRY1     DW      0                          ;install
ENTRY2       DW      0                          ;user
OFENTRY2     DW      0                          ;exit
SGENTRY2     DW      0                          ;table
ENTRY3       DW      0                          ;entries
OFENTRY3     DW      0
;

```

```

SGENTRY3      DW      0          ;four      ! four
ENTRY4        DW      0          ;logical   ! table
OFENTRY4      DW      0          ;terminals ! entries
SGENTRY4      DW      0          ;
;
SFT           DB      7800H DUP(' ') ;Screen Format Table
;
PATHNME       DB      'C:\ODSPTEST\ODSPSFCF.CTL',0 ;pathname of SFCF
;
SFCFBUFF      DB      SFCFSIZE DUP(0) ;buffer for SFCF records
;
SFCFHDL       DW      ?          ;file handle for SFCF
;
SCRENHDL      DW      ?          ;file handle for screen format
;
PATHBUFF      DB      'C:\ODSPTEST\          ',structure for pathn
;           to get screen files in SFCF
;
EXTASCZ       DB      '.SF',0    ;extension put in PATHBUFF to get
;           the screen file
;
NXTHSTNM      DW      0          ;OFFSET of next position to fill
;           (HOSTID) in host/chain table
;
NXTSCR        DW      0          ;addr of next position in SFT
;
ERROR1        DB      0DH,0AH,'Error opening SFCF $',0DH,0AH
;
ERROR2        DB      0DH,0AH,'Error reading from SFCF $',0DH,0AH
;
ERROR3        DB      0DH,0AH,'Error opening SFT file $',0DH,0AH
;
ERROR4        DB      0DH,0AH,'Error reading SFT file $',0DH,0AH
;
ERROR5        DB      0DH,0AH,'Error closing SFT file $',0DH,0AH
;
ERROR6        DB      0DH,0AH,'INDODSP name not resolved $',0DH,0AH
;
ERROR7        DB      0DH,0AH,'PUT_UTE function failed $',0DH,0AH
;
PAGE
;-----
INIT          PROC          NEAR
;-----
;
MOV          AX,CS          ;Put initial values into
MOV          DS,AX          ; DS and ES
MOV          ES,AX          ;
MOV          NXTSCR,OFFSET SFT ;Set up to store Screen Data
MOV          NXTHSTNM,0      ;Set up to store Information
MOV          COUNT,0        ;Set up count of screen formats
CLD                    ;Set up for automatic increment
;
;-----
; Read in the pathname of the file to be opened
;-----
;
INITPROC:
LEA          DX, PATHNME    ;address of pathname
MOV          AL,0           ;set access code to 'OPEN TO READ'
MOV          AH,DOSOPEN     ;open this file (ASCIIZ format)
INT          DOSCALL        ;call DOS
;

```

```

        MOV     SFCFHDL,AX      ;save file handle
        JNC     READREC        ;If there is no error read record
;
        LEA     DX, PATHNME     ;address of pathname
        MOV     PATHNME, A      ;address of pathname for drive A:
        MOV     PATHBUFF, A     ;address of pathname for drive A:
        MOV     AL,0            ;set access code to 'OPEN TO READ'
        MOV     AH,DOSOPEN      ;open this file (ASCIIZ format)
        INT     DOSCALL         ;call DOS
;
        MOV     SFCFHDL,AX      ;save file handle
        JNC     READREC        ;If there is no error read record
;
        LEA     DX,ERROR1       ;Else call error routine
        CALL    ERROR           ; and return to DOS
        RET
;
;-----
; Read a single record from SFCF into buffer and check for EOF
;-----
;
READREC:
        MOV     CX,SFCFSIZE     ;read in first 12 bytes of SFCF
        LEA     DX,SFCFBUFF     ;address of buffer
        MOV     BX,SFCFHDL      ;put file handle in BX
        MOV     AH,3FH          ;read from file function
        INT     DOSCALL         ;call DOS
;
        JNC     CHECKEND        ;If there was no error check EOF
;
        LEA     DX,ERROR2       ;Else Do error routine
        CALL    ERROR           ; and return to DOS
        RET
;
CHECKEND:
        CMP     COUNT,MAX_NTRS  ;If at max capacity
        JE      MAX_CAP         ; Then exit file read
;
        CMP     AX,0DH          ;If last access to file was EOF
        JNL     STORENME
;
MAX_CAP:
        ;Set up error message
        MOV     DI,NXTHSTNM     ;displacement in SFI for entry
        MOV     AX,OFFSET NO_FIND ; Location of error message
        MOV     SFI.SCREENAD[DI],AX ;move in addr of screen start
        MOV     AX,NO_FINDL     ; Length of error message
        MOV     SFI.SCREENLN [DI],AX ;Length of screen
        JMP     NAME_RES        ;Then complete initialization
;
;-----
; Put the name from the SFCF buffer into the SFI
;-----
;
STORENME:
        ;
        MOV     DI,NXTHSTNM     ;displacement in SFI for entry
        MOV     AX,NXTSCR       ;put buffer OFFSET in SFI
        MOV     SFI.SCREENAD[DI],AX ;move in addr of screen start
;
        MOV     CX,NAM_SIZE     ;the size of screen name
        LEA     SI,SFCFBUFF     ;addr of buffer (SOURCE)
        ADD     DI,OFFSET SFI    ;calculate OFFSET of destination

```

```

        REPE    MOVSB                ;
;
;-----
; Put file name from SFCF into buffer which holds path of screen fill
;-----
;
        LEA     DI,PATHBUFF + 12    ;pathname (destination)
        MOV     CX,8                 ;store maximum of 8 chars
        INC     SI                   ;first SFCFBUFF location
;
STRPATH:
        MOV     DL,[SI]              ;put char into DL
        CMP     DL,0DH               ;check for CR
        JE      ENDPH                ;if so, end storing
;
        CMP     DL,BYTE PTR BLANK    ;is it a blank
        JE      ENDPH                ;if so, end storing
;
        MOV     [DI],DL              ;put char into pathbuff
        INC     SI                   ;next SFCFBUFF location
        INC     DI                   ;next PATHBUFF location
        LOOP    STRPATH              ;store up to 8 characters
;
ENDPH:
        LEA     SI,EXTASCZ            ;addr of extension to file
        MOV     CX,4                 ;4 characters long (.SF0)
;
        REPE    MOVSB                ;Put extension into name
;
;-----
; Open file in path buffer and save the handle
;-----
;
        MOV     DX,OFFSET PATHBUFF    ;addr of file name
        MOV     AL,0                 ;access code 'OPEN TO READ'
        MOV     AH,DOSOPEN            ;open file (ASCIIZ format)
        INT     DOSCALL               ;call DOS
;
        MOV     SCRENHDL,AX           ;save handle
        JNC     READSFT               ;If no error read SFT
;
        LEA     DX,ERROR3              ;Else do error routine
        CALL    ERROR                 ; and exit to DOS
        RET
;
;-----
; Read screen formats into SFT
;-----
;
READSFT:
;
        MOV     CX,MAX_SIZE           ;try to read 1920 characters
        MOV     DX,NXTSCR             ;addr to put file in
        MOV     BX,SCRENHDL           ;put handle into BX
        MOV     AH,3FH                ;read from file function
        INT     DOSCALL               ;call DOS
;
        JNC     STORESFT              ;If no error store SFT info
;
        LEA     DX,ERROR4              ;Else do error routine
        CALL    ERROR                 ; and then return to DOS

```

```

                RET                ;
;
;-----
; Store length and offset into SFI and update offsets
;-----
;
STORESFT:                ;
;
                MOV     SI,NXTHSTNM    ;addr to store
                MOV     SFI.SCREENLN [SI],AX ;Length of screen
;
                ADD     NXTSCR,AX      ;OFFSET of next screen table entry
                ADD     NXTHSTNM,TYPE DEFSFI ;OFFSET on next info. entry
;
;-----
; Close old screen file, reinit buffers and ctrs., return to read SFCF
;-----
;
                MOV     BX,SCREENHDL    ;Put screen handle into BX
                MOV     AH,3EH          ;Close handle function
                INT     DOSCALL         ;Call DOS
;
                JNC     REINIT          ;If no error REINIT
;
                LEA     DX,ERROR5       ;Else do error routine
                CALL    ERROR           ; and return to DOS
                RET
;
REINIT:                ;
;
                INC     COUNT           ;Increment cnt of formats
                JMP     READREC         ;Read the next file
;
;-----
; Do name resolution and install entries into user exit table
;-----
;
NAME_RES:                ;
                MOV     AH,81H          ;set up registers
                MOV     DI,DS           ;for a call to
                MOV     ES,DI           ;name resolution
                LEA     DI,SERVNAME     ;interrupt
;
                INT     7AH             ;Call 3270 Workstation Program
;
                TEST    CL,OFFH         ;If name resolution fails, Then
                JZ      PUT_ENTRY
;
                LEA     DX,ERROR6       ; Do error routine
                CALL    ERROR           ; and exit to DOS
                RET
;
PUT_ENTRY:                ;
                MOV     UET_ID,DX       ;store values
;
                MOV     NUMUETE,2       ;in
;
                MOV     AX,DS           ;parameter
                LEA     DX,MAIN         ;
;
                MOV     ENTRY1,0        ;list

```

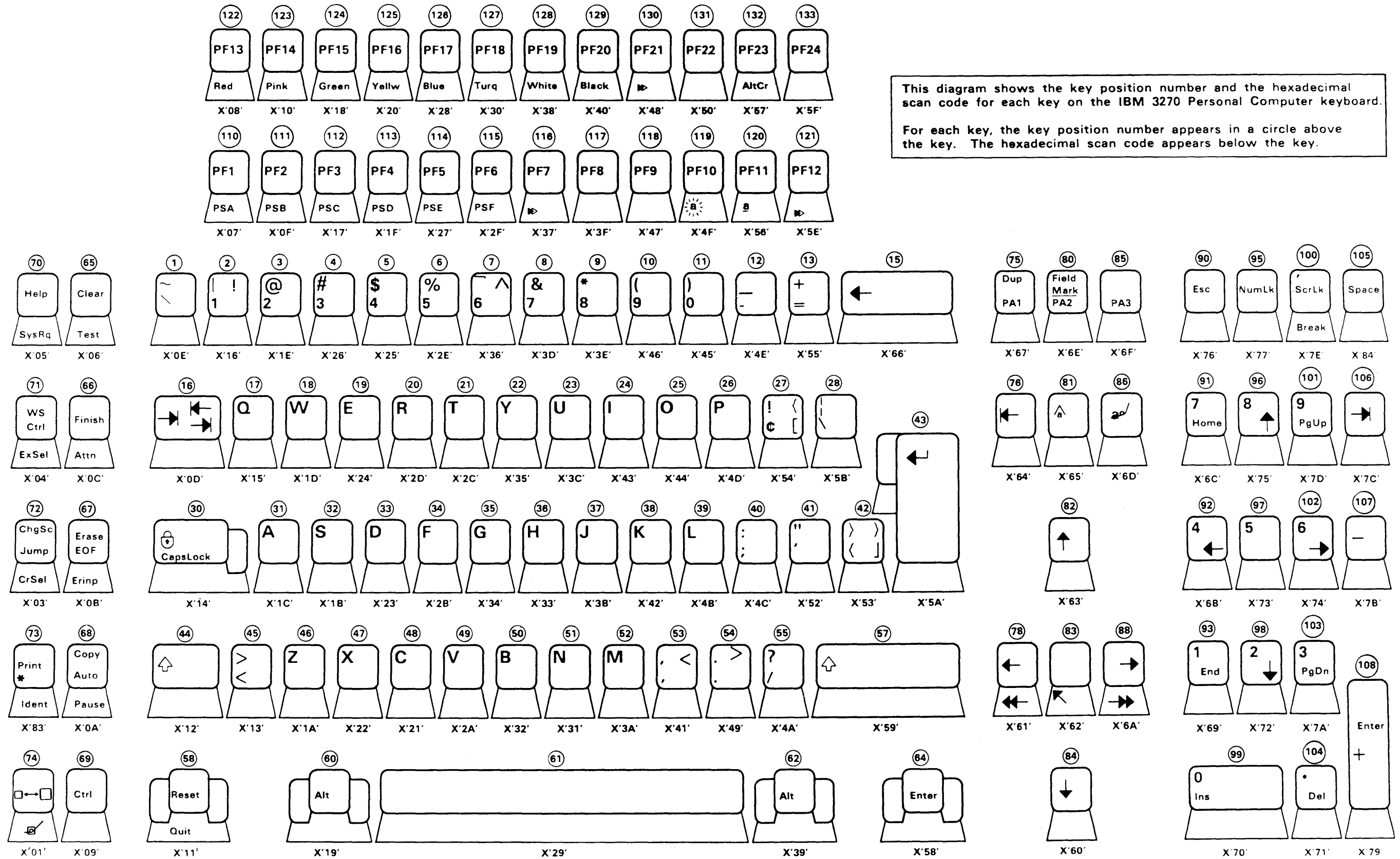
```

        MOV     OFENTRY1,DX      ;
        MOV     SGENTRY1,AX      ;for
;
        MOV     ENTRY2,1         ;install
        MOV     OFENTRY2,DX      ;
        MOV     SGENTRY2,AX      ;user
        ;
        MOV     ENTRY3,2         ;exit
        MOV     OFENTRY3,DX      ;
        MOV     SGENTRY3,AX      ;table
;
        MOV     ENTRY4,3         ;entries
        MOV     OFENTRY4,DX      ;
        MOV     SGENTRY4,AX      ;interrupt
;
        MOV     AH,0EH           ;put values in
        MOV     DX, UET_ID       ;registers for
        MOV     DI,DS            ;install user exit
        MOV     ES,DI            ;table entries
        LEA     DI,NUMUETE       ;interrupt
;
        INT     7AH              ;Call 3270 Workstation Program
;
        TEST    CL,OFFH          ;If put table entry fails, Then
        JZ      EXIT_AND
;
        LEA     DX,ERROR7        ;      Do error routine
        CALL    ERROR            ;      and exit to DOS
        RET
;
EXIT_AND
;
        MOV     DX,OFFSET PATHNME ;OFFSET of next screen table entry
        MOV     CL,4
        SHR     DX,CL
        INC     DX
        MOV     AX,3100H
        INT     DOSCALL          ;exit and remain resident
;
INIT      ENDP                  ;end INIT procedure
;
-----
ERROR      PROC      NEAR
;
        MOV     AH,DISPLAY       ;display character function
        INT     DOSCALL          ;Call to DOS routine
        RET                    ;Return to caller
;
ERROR      ENDP
;
-----
;
PROGRAM    ENDS                  ;end of program segment
;
        END      START

```


Notes:

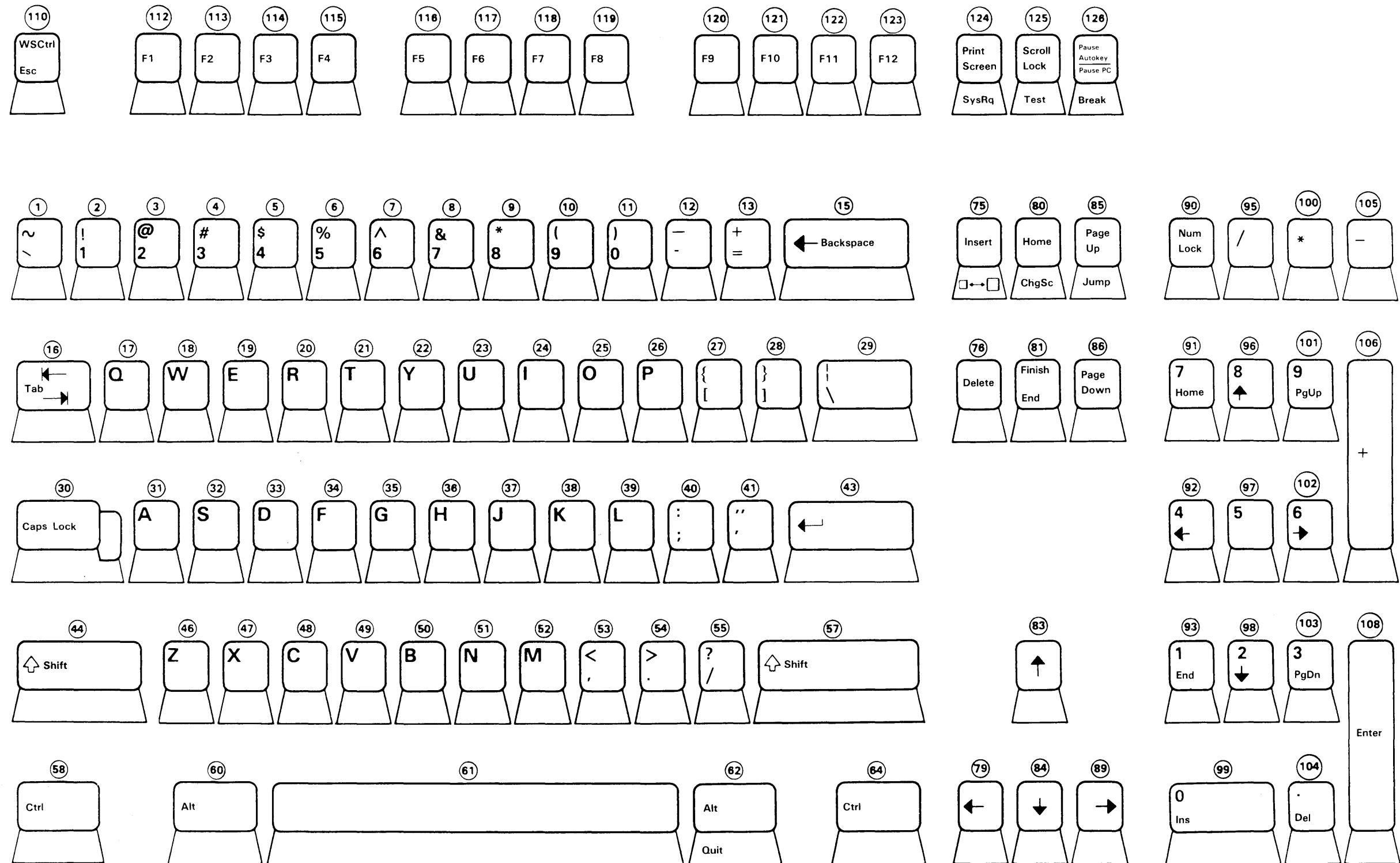
Notes:



IBM 3270 Personal Computer U.S. English Keyboard

This diagram shows the key position number for each key on the IBM 3270 Enhanced Personal Computer keyboard.

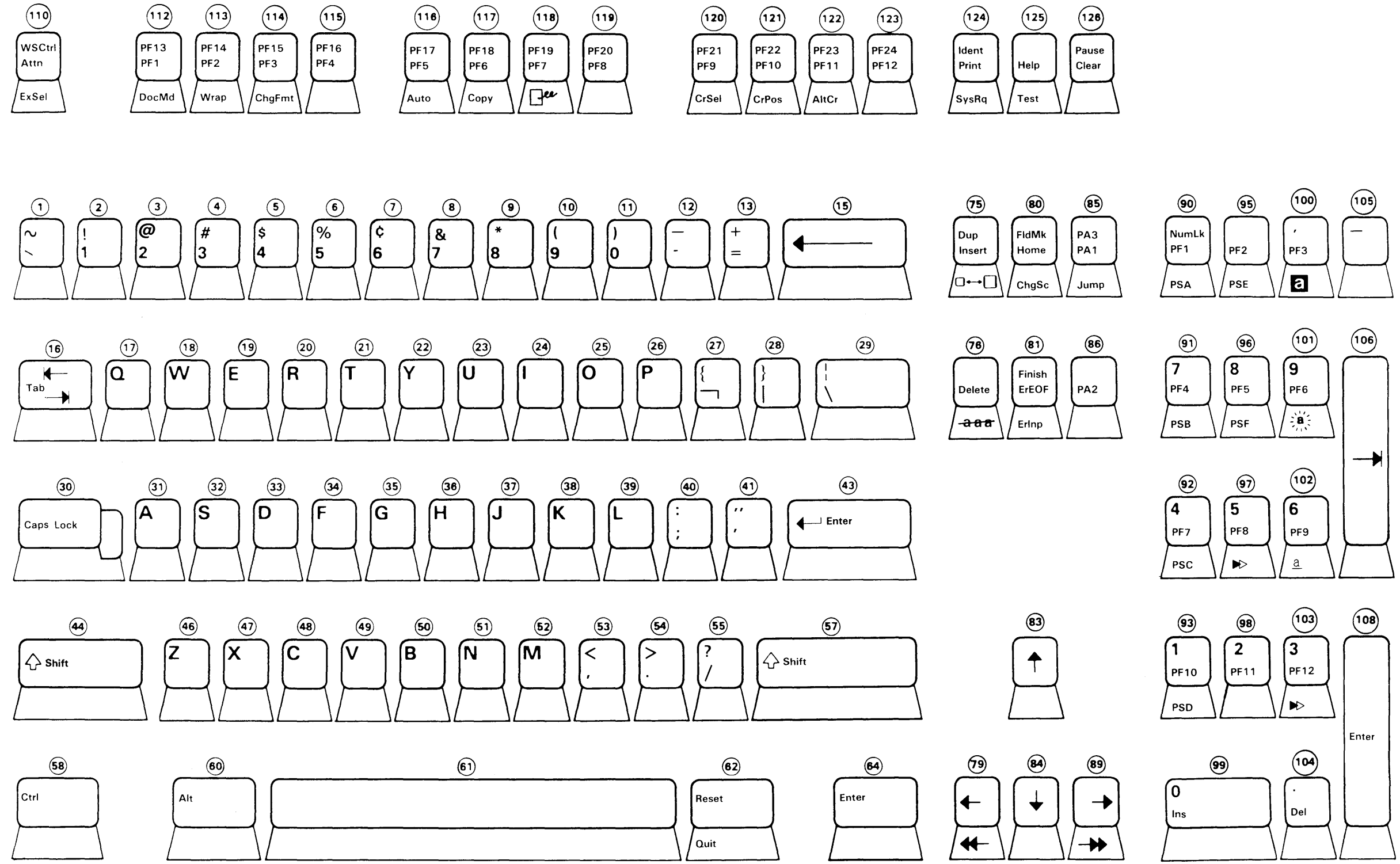
For each key, the key position number appears in a circle above the key.



IBM 3270 Enhanced Personal Computer U. S. English Keyboard, PC Mode

This diagram shows the key position number for each key on the IBM 3270 Enhanced Personal Computer keyboard.

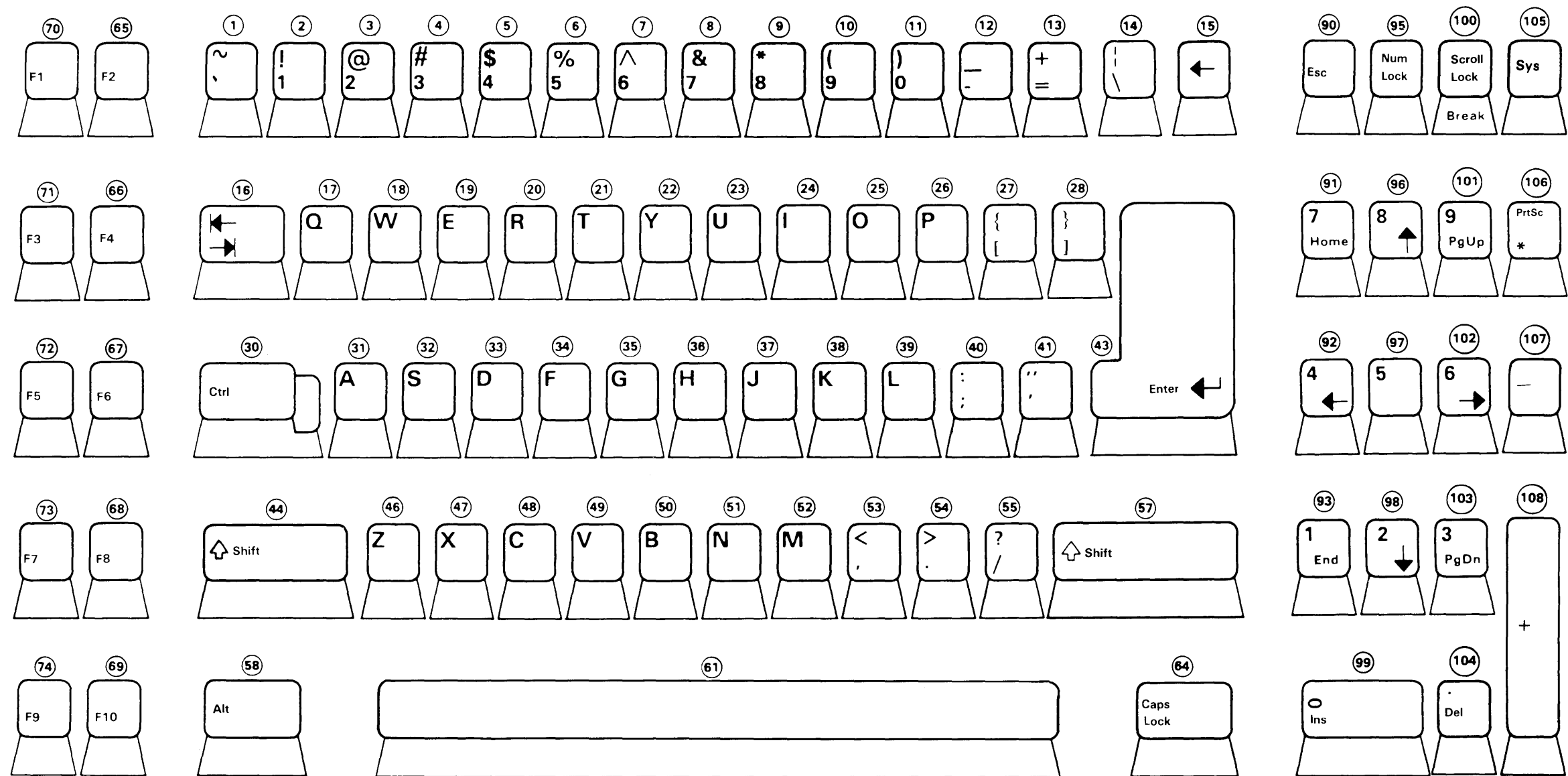
For each key, the key position number appears in a circle above the key.



IBM 3270 Enhanced Personal Computer U. S. English Keyboard, MFI Mode

This diagram shows the key position number for each key on the IBM Personal Computer AT keyboard.

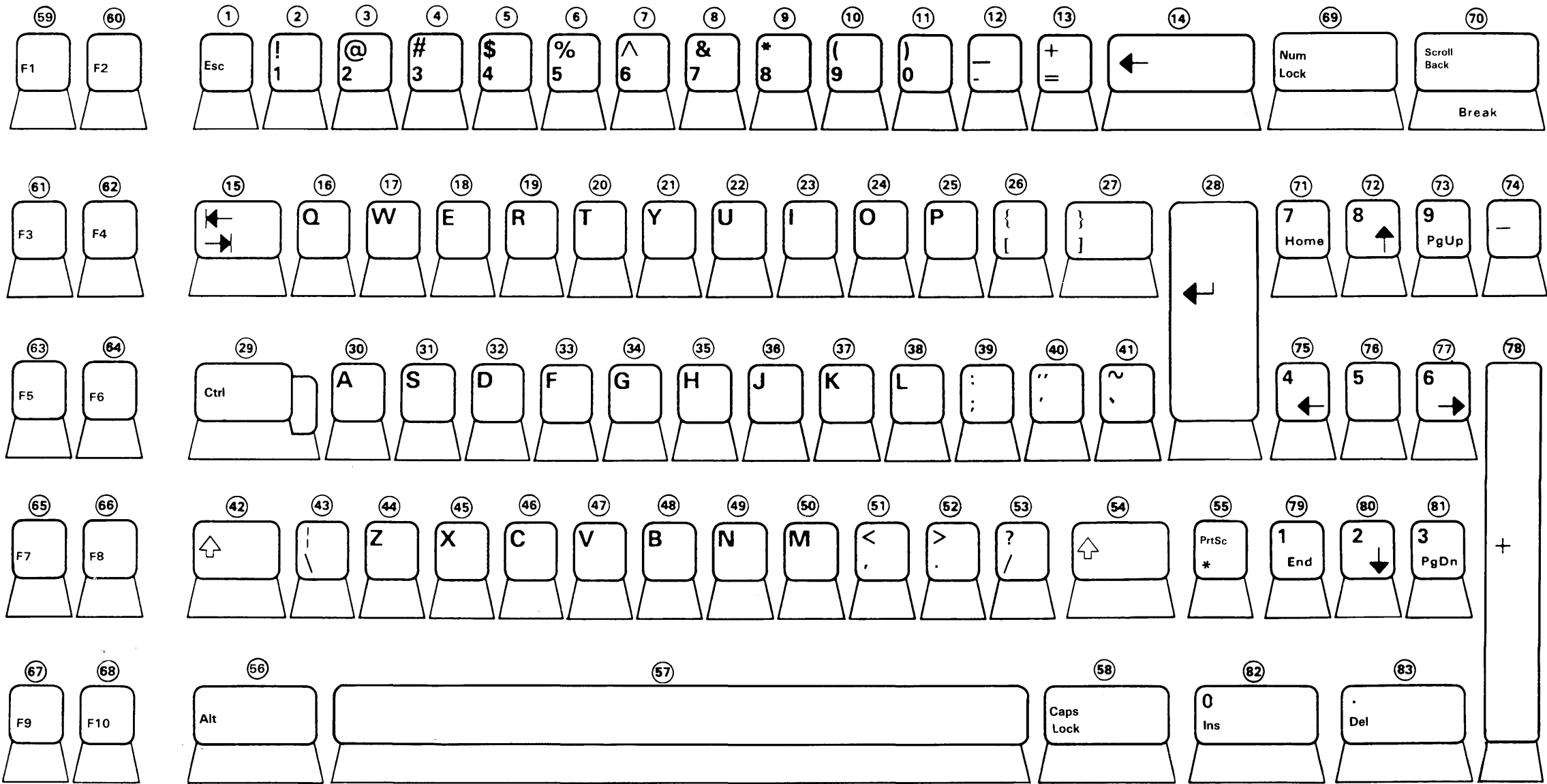
For each key, the key position number appears in a circle above the key.



IBM Personal Computer AT U.S. English Keyboard

This diagram shows the key position number for each key on the IBM 3270 Personal Computer XT keyboard.

For each key, the key position number appears in a circle above the key.



IBM Personal Computer XT U.S. English Keyboard

Index

A

- Add Resource service, coding information 23-8
- Add Window service, coding information 6-14
- APA graphics, changes or limitations on personal computer sessions D-37
- API services
 - functions of 1-4
 - overview 1-2
 - overview, diagram of 1-2
 - using the interface 1-9
- application program
 - call from for Save, Restore, Send and Receive G-2
 - exception condition structured field B-10
 - Interrupt Handler Management services 14-20
 - services, list of 1-5
 - type PC running on 2-13
- ASCII
 - ASCII/ASCII Mnemonics 5-7, A-4
 - characters common to all countries A-4
 - characters used by U.S. English A-4
 - mnemonics common to all countries A-4
 - Read Input API 5-7
 - Write Keystroke API 5-7
- asynchronous API services processing 3-3
- Attach Session ID service, coding information 4-17
- Attention Identifier (AID) keys, defined 5-5
- attributes, session F-2
- AUTOEXEC.BAT file, using C-3

B

- background session 2-10
- background, definition 2-10
- BAT files, using C-2
- batch files, using C-2
- bit numbering conventions used in this manual, described 2-16, 13-22
- bits, meaning of
 - write control character (WCC) D-9
- buffer addresses D-31

C

- calling Save, Restore, Send, and Receive from your application program G-2
- Change Enlarge State service, coding information 6-36
- Change Hidden State service, coding information 6-33
- Change Screen Background service, coding information 6-38
- Change Task's Priority service, coding information 17-14
- Change Window Attributes service, coding information 6-92
- Change Window Color service, coding information 6-25
- Change Window Position on Presentation Space service, coding information 6-29
- Change Window Position on Screen service, coding information 6-17
- Change Window Size service, coding information 6-21
- character attributes D-7, F-4, F-5
- Claim a Semaphore service, coding information 18-3
- Clear Screen service, coding information 6-66
- Close X'D0' structured field
 - sending from host to 3270 PC
 - format B-18
 - overview B-12
 - sending from 3270 PC to host
 - format B-28
 - overview B-20
- coding descriptions
 - copy services 10-2
 - environment manager services 23-2
 - fixed-length queue management services 20-2
 - host interactive services 7-2
 - interrupt handler management services 21-2
 - keyboard services 5-2
 - logical timer management services 19-2
 - Multi-DOS services 13-2-13-20
 - operator information area services 12-2
 - presentation space services 8-2
 - request services 16-2
 - semaphore management services 18-2
 - session information services 4-2
 - supervisor services 3-2
 - supervisory object services 15-2
 - system extension message service 24-16
 - system extensions 24-2
 - task state modifier services 17-2
 - translate service 11-2
 - window management services 6-2

-
- 3270 keystroke emulation services 9-2
 - coding examples
 - Add Resource service 23-11
 - Add Window service 6-16
 - Attach Session ID service 4-19
 - Change Enlarge State service 6-37
 - Change Hidden State service 6-35
 - Change Screen Background service 6-40
 - Change Task's Priority service 17-15
 - Change Window Attributes service 6-96
 - Change Window Color service 6-28
 - Change Window Position on Presentation Space service 6-32
 - Change Window Position on Screen service 6-20
 - Change Window Size service 6-24
 - Claim a Semaphore service 18-5
 - Clear Screen service 6-68
 - Connect for Copy to PC Session service 10-21
 - Connect for 3270 Keystroke Emulation service 9-9
 - Connect to Host Session service 7-10
 - Connect to Keyboard service 5-12
 - Connect to Work Station Control service 6-10
 - Copy Block service 10-17
 - Copy String service 10-10
 - Create Component Entry service 15-10
 - Create Fixed-Length Queue Entry service 3-11
 - Create Fixed-Length Queue service 15-16
 - Create Gate Entry service 15-20
 - Create Semaphore Entry service 15-13
 - Create Task Entry service 15-7
 - Create User Exit Table Entry service 15-23
 - Define Buffer service 7-29
 - Define Presentation Space service 8-9
 - Delete Entry service 15-33
 - Delete Entry service, coding information 3-16
 - Delete Presentation Space service 8-13
 - Delete Resource service 23-14
 - Delete Window service 6-80
 - Dequeue Data service 20-7
 - Dequeue Data service, coding information 3-14
 - Detach Session ID service 4-16
 - Disable Input service 5-32
 - Disconnect for Copy to PC Session service 10-24
 - Disconnect for 3270 Keystroke Emulation 9-12
 - Disconnect From Host Session service 7-14
 - Disconnect from Keyboard service 5-15
 - Disconnect from Work Station Control service 6-13
 - Display Presentation Space service 8-16
 - Enable Input service. 5-35
 - Enqueue Data service 20-4
 - Get a Request service 16-10
 - Get Logical Timer service 19-4
 - Get Request Completion service 3-8, 16-16
 - ID Resolution service 15-31
 - Identify Resource Manager service 23-7
 - Install a Hardware Interrupt Handler service 21-6
 - Install an Interrupt Handler service 21-9
 - Install User Exit Table Entries service, coding information 15-26
 - Make a Request service 16-7
 - Name Resolution service 3-6, 15-29
 - Post Status Code service 5-38
 - Purge Queue Data service 20-9
 - Query a Semaphore service 18-9
 - Query Active Screen service 6-86
 - Query Active Task service 17-3
 - Query Active Window service 6-83
 - Query Base Window service 4-32
 - Query Enlarge State service 6-59
 - Query Environment Characteristics service 23-38
 - Query Environment of Window service 4-25
 - Query Hidden State service 6-56
 - Query Interrupt Vector Contents service 21-11
 - Query PC Session PIF Information service 4-29
 - Query Resource service 23-16
 - Query Screen Background Color service 6-62
 - Query Session Cursor service 4-35
 - Query Session ID service 4-9
 - Query Session Parameter service 4-13
 - Query Task's Environment ID service 23-35
 - Query Window Attributes service 6-91
 - Query Window Colors service 6-50
 - Query Window Names service 6-65
 - Query Window Position on Presentation Space service 6-53
 - Query Window service 6-43
 - Query Window Size service 6-46
 - Query Windows in Environment service 4-22
 - Read AID Key service 9-19
 - Read Input service 5-21
 - Read Operator Information Area Group service 12-13
 - Read Operator Information Area Image service 12-6
 - Read Structured Field service 7-19
 - Redraw Screen service 6-74
 - Redraw Window service 6-77
 - Release a Semaphore service 18-7
 - Release Logical Timer service 19-9
 - Remove an Interrupt Handler service 21-13
 - Reply to a Request service 16-13
 - Return to Dispatcher service 17-17
 - Select Active Screen service 6-100
 - Select Active Window service 6-71
 - Send a Signal to a Task service 16-18
 - Set Cursor Position service 8-20
 - Set Logical Timer service 19-7
 - Set Task "Nonpreemptable" service 17-13
 - Set Task "Preemptable" service 17-11
 - Set Task "Ready" service 17-6
 - Set Task "Unready" service 17-9
 - Stop/Reset Environment service 23-33
-

- Suspend/Resume Environment service 23-22
- Switch Presentation Space service 8-22
- System Extension Message service 24-19, 24-21, 24-24
- Translate Data service 11-8
- Write Keystroke service 5-28
- Write Structured Field service 7-24
- color, changes or limitations on personal computer session D-36
- command
 - line G-4
 - procedures
 - file transfer (Send and Receive) C-6
 - Save and Restore C-2, C-4
- command line, DOS EXEC G-4
- communication status information
 - listed 7-18
 - use of with the Read Structured Field service 7-18
- completion queue signal 14-8
- completion signal 14-8
- components, defined 14-4
- Connect for Copy to PC Session service, coding information 10-19
- Connect for 3270 Keystroke Emulation service, coding information 9-7
- Connect to Host Session service, coding information 7-4
- Connect to Keyboard services, coding information 5-9
- Connect to Work Station Control service, coding information 6-7
- Control unit communication session termination on personal computer session D-36
- conventions used in the API service
 - descriptions 2-16, 13-22
- Copy Block service, coding information 10-12
- copy services
 - Connect for Copy to PC Session 10-19
 - Copy Block 10-12
 - Copy String 10-5
 - defined 1-7
 - Disconnect for Copy to PC Session 10-22
- copy services:X'64': H-41
- Copy String service, coding information 10-5
- Create Component Entry service, coding information 15-8
- Create Fixed-Length Queue Entry service, coding information 3-9, 15-14
- Create Gate Entry service, coding information 15-17
- Create Semaphore Entry service, coding information 15-11
- Create Task Entry service, coding information 15-4
- Create User Exit Table Entry service, coding information 15-21
- creating a BAT file
 - Save and Restore C-2
 - Send and Receive C-2, C-6

- creating a batch file
 - Save and Restore C-2
 - Send and Receive C-2, C-6
- creating an AUTOEXEC.BAT file C-3
- creating objects with names 14-6
- cursor, physical: changes or limitations on personal computer sessions D-35
- customization 24-5
 - home panel 24-5
 - system extension loading 24-5
- CUT hardware initialization:X'43': H-33
- CUT host sessions, presentation space size F-8

D

- data available signal 14-8
- data stream, manual to use for information, listed vii
- debugging a personal computer application
 - program E-10
- decimal numbers used in this manual,
 - described 2-16, 13-22
- Define Buffer service, coding information 7-25
- Define Presentation Space service, coding information 8-4
- Delete Entry service, coding information 3-15, 15-32
- Delete Presentation Space service, coding information 8-11
- Delete Resource service, coding information 23-12
- Delete Window service, coding information 6-78
- Dequeue Data service, coding information 3-12, 20-5
- Detach Session ID service, coding information 4-14
- DFT host session presentation space size F-7
- DFT operations:X'30': H-26
- Disable Input service, coding information 5-30
- Disconnect for Copy to PC Session service, coding information 10-22
- Disconnect for 3270 Keystroke Emulation service, coding information 9-10
- Disconnect from Host Session service, coding information 7-11
- Disconnect from Keyboard service, coding information 5-13
- Disconnect From Work Station Control service, coding information 6-11
- dispatching tasks 14-11
- Display Presentation Space service, coding information 8-14
- DOS EXEC function call G-3
 - command line G-4
 - environment string G-3
 - file control blocks G-4
 - take over hardware interrupts 14-17
 - take over software interrupts 14-18

DOS function calls 14-18
 EXEC G-3
 invoke Save, Restore, Send and Receive G-2
 SETBLOCK G-2
 DOS SETBLOCK function call G-2
 DOS subsystem services:X'22' or X'23': H-16
 DOS, levels supported by the Workstation
 Program iv
 draw service:X'67': H-43
 dump task:X'7F': H-54
 duplicate names 14-6

E

EBCDIC control character I/O codes between host
 and IBM 3270 Personal computer D-5
 Enable Input service, coding information 5-33
 enhanced graphics adapter (EGA)
 defined 1-3
 extended field bit assignment F-5
 Enqueue Data service, coding information 20-3
 environment
 defined 1-3
 string G-3
 environment manager services
 Add Resource 23-8
 coding information 23-2
 Delete Resource 23-12
 Identify Resource Manager 23-4
 Query Environment Characteristics 23-36
 Query Resource 23-15
 Query Task's Environment ID 23-34
 requesting the 23-2
 Stop/Reset Environment 23-23
 Suspend/Resume Environment 23-17
 environment manager services:X'13': H-11
 Erase/Reset structured field
 format D-14
 ID code D-12
 error handler:X'72': H-53
 error service, system extension 24-16
 error steps H-57
 events, signals 14-8
 exception handling B-8
 EXEC function call G-3
 extended field attributes D-7, F-4, F-5

F

failure, changes or limitations on personal computer
 sessions D-36
 field attributes D-6, F-3
 file control blocks G-4
 file control blocks, DOS EXEC G-4

file transfer commands (Send and Receive)
 BAT file to invoke, using C-2
 DOS function calls to invoke, using G-2
 programmed command procedure to invoke,
 using C-6
 fixed-length queue
 creating 3-3
 deleting 3-4
 obtaining data from 3-3
 fixed-length queue management services
 Dequeue Data 20-5
 Enqueue Data 20-3
 introduction 14-16
 Purge Queue Data 20-8
 fixed-length queues, definition 14-5
 foreground, definition 2-10
 full screen with APA mode, changes or limitations
 on personal computer session D-37
 functions the API provides 1-4

G

gate names 3-2
 gate, defined 3-2
 gates, definition 14-6
 generic signal 14-8
 Get a Request service, coding information 16-8
 Get Logical Timer service, coding information 19-3
 Get Request Completion service, coding
 information 3-7, 16-14
 global software interrupt handlers 14-19

H

hardware interrupt handlers 14-16
 hardware interrupts
 DOS function calls 14-17
 Install a Hardware Interrupt Handler
 service 14-17
 Install an Interrupt Handler service 14-18
 interrupt handler considerations 14-18
 hexadecimal numbers used in this manual,
 described 2-16, 13-22
 host interactive services
 Connect to Host Session 7-4
 Define Buffer 7-25
 defined 1-6
 Disconnect from Host Session 7-11
 Read Structured Field 7-15
 Write Structured Field 7-20
 host interactive services:X'32': H-32

I

- IBM Macro Assembler manual, listed vii
- IBM 3270 data stream manual, listed vii
- ID Resolution service, coding information 15-30
- Identify Resource Manager service, coding information 23-4
- inbound 3270 data stream for partition 0
 - Read Buffer format D-19
 - Read Buffer format in extended field and character mode D-19
 - Read Buffer format in field reply mode D-19
 - Read Modified All format D-18
 - Read Modified format D-18
 - Short Read format D-18
- inbound 3270 data stream structured field
 - input control B-6
- inbound 3270 data stream structured fields
 - Auxiliary Device Query Reply structured field format D-25
 - character set descriptors D-24
 - Character Sets Query Reply structured field D-23
 - Color Query Reply structured field format D-27
 - DDM Query Reply structured field format D-25
 - defined B-3
 - Direct Access self-defining parameter D-26
 - Document Interchange Architecture Query Reply structured field format D-26
 - Highlight Query Reply structured field format D-28
 - Implicit Partition Query Reply structured field format D-29, D-30
 - character cell dimensions D-29
 - implicit partition default and alternate screen size D-29
 - self-defining parameters D-30
 - Reply Modes Query Reply structured field format D-24
 - Usable Area Query structured field D-22
- INCTRL B-6
- input control B-6
 - See also INCTRL
- input queue size 5-11
- Insert and Insert Data
 - sending X'D0' from host to 3270 PC format B-15
- Insert and Insert Data, structured fields
 - sending X'D0' from host to 3270 PC overview B-12
- Install a Hardware Interrupt Handler service
 - take over hardware interrupt 14-17
- Install an Interrupt Handler service
 - take over hardware interrupts 14-18
 - take over software interrupts 14-19
- Install User Exit Table Entries service, coding information 15-24

- interface
 - codes, host and 3270 PC D-3
 - structured field
 - exception handling B-8
 - query reply format B-4
 - read partition query format B-4
 - verifying operational B-4
- interrupt handler management services
 - application program uses 14-20
 - Install a Hardware Interrupt Handler 21-4
 - Install a Hardware Interrupt Handler service, coding information 21-4
 - Install an Interrupt Handler 21-7
 - Install an Interrupt Handler service, coding information 21-7
 - introduction 14-16
 - Query Interrupt Vector Contents 21-10
 - Remove an Interrupt Handler 21-12
- interrupt handlers 14-16
 - DOS EXEC function call 14-17
 - DOS function calls 14-18
 - global software 14-19
 - hardware 14-17
 - hardware interrupt considerations 14-18
 - hardware interrupts 14-17
 - Install a Hardware Interrupt Handler service 14-17
 - Install an Interrupt Handler service 14-18, 14-19
 - local software 14-19
 - software interrupt handler considerations 14-19
 - software interrupts 14-18
- Interrupt X'10' 2-8

K

- keyboard services
 - Connect to Keyboard 5-9
 - defined 1-6
 - Disable Input 5-30
 - Disconnect from Keyboard 5-13
 - Enable Input 5-33
 - how to request 5-8
 - Post Status Code 5-36
 - Read Input 5-16
 - return codes 5-8
 - use of 5-7
 - Write Keystroke 5-22
- keyboard services:X'62': H-35
- keystroke definition services:X'6E': H-52
- Keystroke Emulation
 - connect for 3270 keystroke emulation 9-7
 - disconnect for 3270 keystroke emulation 9-10
 - field attribute definition 9-2
 - presentation space format 9-4
 - Read AID key 9-13

- requesting 9-5
- return codes 9-5
- return codes :X'6Exx' H-51
- keytop characteristics 5-4

L

- levels of DOS supported by the Workstation Program iv
- limitations
 - 3270 PC D-2
- local software interrupt handlers 14-19
- logical timer management services
 - Get Logical Timer 19-3
 - introduction 14-15
 - Release Logical Timer 19-8
 - Set Logical Timer 19-5

M

- macro assembler, manual to use for information, listed vii
- Make a Request service, coding information 16-3
- make only keys
 - defined A-3
 - scan codes A-3
- make/break keys
 - defined 5-4
 - scan codes A-3
- managing resources, system extensions 22-5, 24-26
- moderately well-behaved program 2-10
- Multi-DOS
 - application program performance 2-6, 2-7
 - Free Storage service 13-15
 - guidelines for running 2-6
 - support services defined 1-7
 - Writing Applications 2-9
- multi-host, simplifying setup and control 1-4

N

- Name Resolution service, coding information 3-5
- Name Resolution services, coding information 15-27
- names, creating objects 14-6
- non-SNA channel commands D-8
- non-3270 PC hardware
 - defined 1-3
 - restrictions 2-5, 2-11, D-34
- nonstoppable environment, defined 1-3
- notepad operations:X'51' H-34

- notepad sessions
 - presentation space size F-8
 - restoring, using programmed command procedure C-4

O

- object ID 14-6
- objects
 - components 14-4
 - creation 14-3
 - deletion 14-3
 - fixed-length queues 14-5
 - gates 14-6
 - names, creating objects 14-6
 - semaphores 14-5, 14-14
 - SVC table 14-6
 - tasks 14-3
 - user exit tables 14-6
- ODSP See Outbound Data Stream Preprocessor Option
- OIA services:X'6D' H-50
- Open X'D0' structured field
 - sending from host to 3270 PC
 - format B-12
 - overview B-11
 - sending from 3270 PC to host
 - format B-21
 - overview B-20
 - successful transmission response, format B-14
 - unsuccessful transmission response, format B-14
- operational interface B-4
- operator information area services
 - defined 1-7
 - Read Operator Information Area Group 12-7
 - Read Operator Information Area Image 12-4
- order of bit numbering used in this manual, described 2-16, 13-22
- Outbound Data Stream Preprocessor Option
 - customizing for ODSP I-2
 - entry parameters I-4
 - initializing ODSP I-2
 - restrictions and recommendations I-5
 - return parameters I-4
 - sample program I-5
 - using ODSP I-3
- outbound 3270 data stream structured fields
 - defined B-3
 - Erase All Unprotected
 - format D-14
 - ID code D-12
 - Erase/Write
 - format D-14
 - ID code D-12
 - Erase/Write Alternate

- format D-14
- ID code D-12
- listed D-12
- Write
 - format D-14
 - ID code D-12
- overview of API services 1-2
- diagram of 1-2

P

- PC application program
 - debugging E-10
 - display interaction B-7
 - exception handling B-8
- personal computer session changes or limitations D-34
- physical cursor, changes or limitations on personal computer session D-35
- PIF
 - See Program Information Files
- poorly behaved program 2-10
- Post Status Code service, coding information 5-36
- prerequisite knowledge needed to use the API services v
- presentation space
 - character table F-6
 - considerations F-2
 - size
 - CUT host F-8
 - DFT host F-7
- presentation space services
 - Define Presentation Space 8-4
 - defined 1-6
 - Delete Presentation Space 8-11
 - Display Presentation Space 8-14
 - Set Cursor Position 8-17
 - Switch Presentation Space 8-21
- presentation space services: X'69': H-44
- presentation space, defined 1-3
- print spooling, changes or limitations on personal computer session D-35
- priorities of tasks 14-3
- priorities, Create Task Entry service 15-4
- problem determination
 - procedures E-2-E-9
 - system error E-2
 - using the trace command E-8
- procedures, command
 - file transfer C-2
 - Save and Restore C-2
- Program Information Files
 - creating and modifying 2-4
 - defined 2-3

- programmed command procedures
 - for file transfer (Send and Receive) C-6
 - for Save and Restore C-4
- Purge Queue Data service, coding information 20-8

Q

- Query a Semaphore service, coding information 18-8
- Query Active Screen service, coding information 6-84
- Query Active Task service, coding information 17-3
- Query Active Window service, coding information 6-81
- Query Base Window service, coding information 4-30
- Query Enlarge State service, coding information 6-57
- Query Environment Characteristics service, coding information 23-36
- Query Environment of Window service, coding information 4-23
- Query Hidden State service, coding information 6-54
- Query Interrupt Vector Contents service, coding information 21-10
- Query PC Session PIF Information, coding information 4-26
- query reply B-6
- query reply structured field B-4
- Query Resource service, coding information 23-15
- Query Screen Background Color service, coding information 6-60
- Query Session Cursor, coding information 4-33
- Query Session ID service, coding information 4-5
- Query Session Parameters service, coding information 4-10
- Query Task's Environment ID service, coding information 23-34
- Query Window Attributes service, coding information 6-87
- Query Window Colors service, coding information 6-47
- Query Window Names service, coding information 6-63
- Query Window Position on Presentation Space service, coding information 6-51
- Query Window Position on Screen service, coding information 6-41
- Query Window Size service, coding information 6-44
- Query Windows in Environment service, coding information 4-20

R

Read AID Key service, coding information 9-13
Read Input service, coding information 5-16
Read Operator Information Area Group service, coding information 12-7
Read Operator Information Area Image service, coding information 12-4
read partition query structured field B-4
Read Partition structured field
 Query
 format D-15
 ID code D-12
Read Buffer
 format D-15
 ID code D-12
Read Modified
 format D-15
 ID code D-12
Read Modified All
 format D-15
 ID code D-12
Read Structured Field service, coding information 7-15
Receive command
 BAT file to invoke, using C-2
 DOS function calls to invoke, using G-2
 programmed command procedure to invoke, using C-6
Redraw Screen service, coding information 6-72
Redraw Window service, coding information 6-75
Release a Semaphore service, coding information 18-6
Release Logical Timer service, coding information 19-8
Remove an Interrupt Handler service, coding information 21-12
Reply to a Request service, coding information 16-11
request queue signal 14-8
request services
 defined 1-8
 Get a Request 16-8
 Get Request Completion 16-14
 Make a Request 16-3
 Reply to a Request 16-11
 Send a Signal to a Task 16-17
requests, tasks 14-7
resource manager
 defined 22-5, 24-26
Restore command
 AUTOEXEC.BAT file to invoke, using C-3
 BAT file to invoke, using C-2
 command procedure to invoke, using C-2
 DOS function calls to invoke, using G-2
Restrictions
 non-3270 PC hardware 2-11, D-34

semaphores 14-14
workstation program 2-5
return codes H-2
 X'Dx through Fx': user system extension H-56
 X'12': system services H-3
 X'13': environment manager services H-11
 X'22' or X'23': DOS Subsystem Services H-16
 X'24': DOS system loader H-22
 X'25': DOS system loader H-22
 X'30': DFT operations H-26
 X'32': host interactive services H-32
 X'43': CUT hardware initialization H-33
 X'51': notepad operations H-34
 X'6B': session information services H-47
 X'6C': translate services H-49
 X'6D': OIA services H-50
 X'6E': keystroke emulation services H-51
 X'6F': keystroke definition services H-52
 X'62': keyboard services H-35
 X'63': window management services H-38
 X'64': copy services H-41
 X'67': draw service H-43
 X'69': presentation space services H-44
 X'7F': dump task H-54
 X'72': error handler H-53
 X'81': enhanced connectivity router H-55
Return to Dispatcher service, coding information 17-16

S

Save and Restore
 BAT file to invoke, using C-2
 command procedure to invoke, using C-2
 DOS function calls to invoke, using G-2
Save command
 BAT file to invoke, using C-2
 command procedure to invoke, using C-2
 DOS function calls to invoke, using G-2
scan codes
 described A-2
 IBM Enhanced PC Keyboard (MFI Mode) A-16
 IBM Enhanced PC Keyboard (PC Mode) A-13
 IBM PC XT Keyboard (MFI Mode) A-22
 IBM PC XT Keyboard (PC Mode) A-19
 IBM Personal Computer AT Keyboard (MFI Mode) A-28
 IBM Personal Computer AT Keyboard (PC Mode) A-25
 IBM 3270 PC Keyboard (MFI Mode) A-9
 IBM 3270 PC Keyboard (PC Mode) A-5
 in list of keystrokes, format 5-25
 introduction 5-3
 special A-3
 table of A-2
scheduling tasks 14-11

-
- Select Active Screen service, coding information 6-98
 - Select Active Window service, coding information 6-69
 - semaphore
 - code serialization 14-14
 - management 14-14
 - restrictions 14-14
 - signal 14-8
 - semaphore management services
 - Claim a Semaphore 18-3
 - Query a Semaphore 18-8
 - Release a Semaphore 18-6
 - semaphore signal 14-8
 - semaphores
 - definition 14-5
 - Send a Signal to a Task service, coding information 16-17
 - Send and Receive
 - BAT file to invoke, using C-2
 - DOS function calls to invoke, using G-2
 - programmed command procedure to invoke, using C-6
 - Send command
 - BAT file to invoke, using C-2
 - DOS function calls to invoke, using G-2
 - programmed command procedure to invoke, using C-6
 - services and gate names 3-2
 - session information services
 - Attach Session ID 4-17
 - defined 1-6
 - Detach Session ID 4-14
 - Query Base Window 4-30
 - Query Environment of Window 4-23
 - Query PC Session PIF Information 4-26
 - Query Session Cursor 4-33
 - Query Session ID 4-5
 - Query Session Parameter 4-10
 - Query Windows in Environment 4-20
 - session information services:X'6B': H-47
 - session, defined 1-3
 - sessions
 - CUT host, presentation space size for F-8
 - DFT host, presentation space for F-7
 - notepad, presentation spaces size for F-8
 - personal computer, changes or limitations to D-34
 - suspended PC sessions 2-10
 - Set Cursor and Get structured fields
 - sending from 3270 PC to host overview B-20
 - Set Cursor and Get X'D0' structured fields
 - sending from 3270 PC to host format B-24
 - Set Cursor Position, coding information 8-17
 - Set Logical Timer service, coding information 19-5
 - Set Reply Mode structured field
 - format D-13
 - ID code D-12
 - Set Task "Nonpreemptable" service, coding information 17-12
 - Set Task "Preemptable" service, coding information 17-10
 - Set Task "Ready" service, coding information 17-4
 - Set Task "Unready" service, coding information 17-7
 - SETBLOCK function call G-2
 - shift state
 - format of A-4
 - in list of keystrokes, format 5-25
 - introduction 5-3
 - SIF
 - See System Information Files
 - SIFs
 - creating and modifying 24-9
 - software interrupts
 - defined 14-18
 - DOS EXEC function call 14-18
 - DOS function calls 14-18
 - global interrupt handlers 14-19
 - Install an Interrupt Handler service 14-19
 - interrupt handler considerations 14-19
 - local interrupt handlers 14-19
 - software needed to write programs that use the API services iv
 - special 5-6
 - Stop/Reset Environment service, coding information 23-23
 - stoppable environment, defined 1-3
 - structured field
 - exception handling B-8
 - PC application program and display interaction B-7
 - query reply B-6
 - query reply format B-4
 - read partition query format B-4
 - sending X'D0' from host to 3270 PC
 - Open X'D0' structured field, format B-12
 - sending X'D0' from the host to 3270 PC
 - Insert and Insert Data B-12
 - successful transmission response, format B-14
 - unsuccessful transmission response B-14
 - structured fields
 - destination/origin B-9
 - exception condition B-10
 - application program, self defining parm. B-10
 - format B-10
 - sending X'D0' from host to 3270 PC
 - Close X'D0' structured field, format B-18
 - Close X'D0' structured field, overview B-12
 - Insert and Insert Data, format B-15
 - sending X'D0' from 3270 PC
 - Set Cursor and Get X'D0' structured fields, format B-24
 - sending X'D0' from 3270 PC to host
 - Close X'D0' structured field, format B-28
-

-
- Close X'D0' structured field, overview B-20
 - Open X'D0' structured field, format B-21
 - Set Cursor and Get structured fields, overview B-20
 - successful transmission response, format B-14
 - supervisor
 - components 14-4
 - creating objects with names 14-6
 - fixed-length queues 14-5
 - gates 14-6
 - object creation 14-3
 - object deletion 14-3
 - semaphores 14-5, 14-14
 - SVC table 14-6
 - tasks 14-3
 - user exit table 14-6
 - supervisor services
 - Create Fixed-Length Queue Entry 3-9
 - Delete Entry 3-15
 - Dequeue Data 3-12
 - Get Request Completion 3-7
 - introduction 14-3
 - list of 1-7
 - supervisor services, Name Resolution 3-5
 - supervisory object services
 - Create Component Entry 15-8
 - Create Fixed-Length Queue Entry 15-14
 - Create Gate Entry 15-17
 - Create Semaphore Entry 15-11
 - Create Task Entry 15-4
 - Create User Exit Table Entry 15-21
 - defined 1-8
 - Delete Entry 15-32
 - ID Resolution 15-30
 - Install User Exit Table Entries 15-24
 - Name Resolution 15-27
 - Suspend/Resume Environment service, coding information 23-17
 - suspended PC sessions 2-10
 - SVC table
 - definition 14-6
 - Switch Presentation Space service, coding information 8-21
 - synchronous API services processing 3-3
 - system errors problem determination E-2
 - system extension
 - defined 1-3
 - extending workstation program 1-5
 - System Extension Message service 24-16
 - coding 24-18, 24-20, 24-23
 - to identify return codes 24-18
 - to request informational messages 24-20, 24-23
 - identifying error return codes 24-16
 - requesting error messages 24-17
 - requesting informational messages 24-17
 - system extensions
 - coding 24-2
 - creating 24-3
 - error service 24-16
 - how to load 24-13
 - initialization code 24-4
 - introduction 24-2
 - loading 24-13
 - managing resources 22-5, 24-26
 - messages and codes 24-15
 - resident code 24-3
 - return codes 24-15
 - System Extension Message service 24-16
 - system information files creation 24-10
 - telling workstation program about 24-5
 - user supplied 24-5
 - user supplied options 24-7
 - System Information Files
 - creating 24-9
 - defined 2-2
 - determining the numbers to use 24-10
 - modifying 24-9
 - options panel 24-10
 - system loader:X'24': H-22
 - system services H-3
- T
- task creating, coding 15-27
 - task requests 14-7
 - task state modifier services
 - Change Task's Priority 17-14
 - Query Active Task 17-3
 - Return to Dispatcher 17-16
 - Set Task "Nonpreemptable" 17-12
 - Set Task "Preemptable" 17-10
 - Set Task "Ready" 17-4
 - Set Task "Unready" 17-7
 - tasks
 - definition 14-3
 - dispatch activity 14-12
 - dispatch cycles 14-11
 - dispatcher states 14-12
 - dispatching procedure 14-11
 - obtaining request completion 14-10
 - priorities 14-3
 - receiving a request 14-9
 - replying to a request 14-10
 - request from another task 14-9, 14-10
 - request to another 14-9
 - requests 14-7
 - sending requests 14-9
 - state modifiers 14-11
 - task state modifiers 14-11
 - timer signal 14-8
 - trace command, using E-8
 - Translate Data service, coding information 11-4
 - translate services
 - defined 1-7
-

- Translate Data 11-4
- translate services:X'6C': H-49
- transmission of buffer addresses
 - buffer addresses, described D-31
 - in 12/14-bit address mode D-32
 - in 16-bit address mode D-32
- typematic keys, defined 5-4
- typematic make/break keys, defined 5-4
- types of signals 14-8

U

- unsuccessful transmission response, format B-14
- user exit tables
 - definition 14-6
- User supplied system extensions
 - loading 24-5
 - options panel 24-7
- user system extension:X'Dx through Fx': H-56

W

- wait states 14-8
- WCC (write control character) D-9
- well-behaved program 2-10
- window management services
 - Add Window 6-14
 - Change Enlarge State 6-36
 - Change Hidden State 6-33
 - Change Screen Background 6-38
 - Change Window Attributes 6-92
 - Change Window Color 6-25
 - Change Window Position on Presentation
 - Space 6-29
 - Change Window Position on Screen 6-17
 - Change Window Size 6-21
 - Clear Screen 6-66
 - Connect to Work Station Control 6-7
 - defined 1-6
 - Delete Window 6-78
 - Disconnect From Work Station Control 6-11
 - Query Active Screen 6-84
 - Query Active Window 6-81
 - Query Enlarge State 6-57
 - Query Hidden State 6-54
 - Query Screen Background Color 6-60
 - Query Window Attributes 6-87
 - Query Window Colors 6-47
 - Query Window Names 6-63
 - Query Window Position on Presentation
 - Space 6-51
 - Query Window Position on Screen 6-41
 - Query Window Size 6-44
 - Redraw Screen 6-72

- Redraw Window 6-75
 - Select Active Screen 6-98
 - Select Active Window 6-69
- window management services:X'63': H-38
- window, defined 1-3
- work station control keys 5-6
- work station control, using 1-5
- Workstation Program
 - determining level 2-13
 - restrictions 2-5
- write control character (WCC) D-9
- write control character reset actions D-10
- Write Keystroke service, coding information 5-22
- Write Structured Field service, coding
 - information 7-20

X

- X'D0' structured fields
 - sending from host to 3270 PC
 - Open X'D0' structured field, overview B-11
 - sending from 3270 PC to host
 - Open X'D0' structured field, overview B-20
- X'D0' structured fields, host to 3270 PC
 - Close X'D0' structured field
 - format B-18
 - overview B-12
- Insert and Insert Data
 - format B-15
 - overview B-12
- Open X'D0' structured field
 - format B-12
 - overview B-11
- X'D0' structured fields, 3270 PC to host
 - Close X'D0' structured field
 - format B-28
 - overview B-20
 - Open X'D0' structured field
 - format B-21
 - overview B-20
- Set Cursor and Get X'D0' structured fields
 - format B-24
 - overview B-20
- XMA card, defined 1-3

Numerics

- 3270 data stream
 - attributes D-6
 - commands B-3, D-8
 - exception handling B-8
 - fields B-3
 - functions D-3
 - inbound stream B-3

interface codes D-3
manual to use for information, listed vii
orders D-11
outbound stream B-3
3270 keystroke emulation services

Connect for 3270 Keystroke Emulation 9-7
Disconnect for 3270 Keystroke Emulation 9-10
Read AID Key 9-13
3270 limitations D-2

IBM 3270 Workstation Program
Programming Guide

READER'S
COMMENT
FORM

Order No. 84X0390

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

How did you use this publication?

- | | |
|--|---|
| <input type="checkbox"/> As an introduction | <input type="checkbox"/> As a text (student) |
| <input type="checkbox"/> As a reference manual | <input type="checkbox"/> As a text (instructor) |
| <input type="checkbox"/> For another purpose (explain) _____ | |

Is there anything you especially like or dislike about the organization, presentation, or writing in this manual? Helpful comments include general usefulness of the book; possible additions, deletions, and clarifications; specific errors and omissions.

Page Number:

Comment:

What is your occupation? _____

Newsletter number of latest Technical Newsletter (if any) concerning this publication:

Note: Staples can cause problems with automatic mail-sorting equipment.
Please use pressure-sensitive or other gummed tape to seal this form.

84X0390

Reader's Comment Form

Fold and Tape

Please Do Not Staple

Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS

PERMIT NO. 40

ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department 95H / 998
11400 Burnet Rd.
Austin, TX 78758



Fold and Tape

Please Do Not Staple

Fold and Tape



PRINTED IN U.S.A. 84X0390

IBM

84X0390
SA23-0343-0



9084X03900001