IBM

Personal Computer

3270-PC High Level Language Application Program Interface

continued on inside back cover

IBM

Personal Computer

3270-PC High Level Language Application Program Interface

First Edition (November 1984)

Changes are made periodically to the information herein; these changes will be incorporated in new editions of this publication.

Products are not stocked at the address given below. Requests for copies of this product and for technical information about this product should be directed to your IBM Marketing representative.

A Program Comment Form is provided at the back of this publication. If this form has been removed, address comments to:

IBM Corporation
Department 68Y
220 Las Colinas Boulevard
Irving, Texas 75039-5513

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

To The Reader

This book provides the information needed to use the IBM 3270 Personal Computer High Level Language Application Program Interface. This program gives the application programmer a set of functions which can be called from an application program running in the personal computer session to access other presentation spaces in the 3270 Personal Computer.

How This Manual Is Organized

Chapter 1. Introduction introduces the 3270 PC High Level API and tells you things you need to know before you start.

Chapter 2. Getting Started describes what you do to install the 3270 PC High Level API.

Chapter 3. The Function Calls describes your interface to the 3270 PC High Level API and provides you with samples to help in your development effort.

Chapter 4. Using the 3270 PC High Level API explains how to use the 3270 PC High Level API from the various high level languages that are supported.

Chapter 5. Application Development Tips has some helpful hints for the development of your applications.

Appendix A. Messages documents the messages that you may encounter.

Appendix B. Key Mnemonics explains the mnemonics provided that allow you to use ASCII characters to represent the special function keys of the 3270 Personal Computer keyboard.

Appendix C. Writing Your Own Language Interface Module shows how to build a "bridge" between any language and the main interface program.

Appendix D. Quick Reference Summary gives a handy reference guide for functions available with the 3270 PC High Level API.

Appendix E. System Modifications explains some system modifications which could be made to customize the system to specific environments.

Glossary defines terms that are used with this product.

Who Should Read This Book

- This manual is intended for programmers who will write applications that use the services of the 3270 PC High Level Language API. A working knowledge of the 3270 Personal Computer and IBM PC DOS is assumed. Users desiring more information on the 3270 Personal Computer should refer to the Control Program User's Guide and Reference - Part Number 1837434. For more information on IBM PC DOS, refer to the IBM PC DOS Manual - Part Number 6024061.

- This manual assumes you are already familiar with the language and compiler you will be using. If you need more information on how to write, compile or linkedit programs, you should refer to the appropriate reference manuals for the specific language.

- This program offering supports multiple high level languages and examples are provided for each of these in the appropriate chapters. In order to provide examples throughout the text, Interpretive BASIC has been used to show sample coding techniques.

References

Although it is presumed that you are familiar with the operation of the IBM 3270 Personal Computer and the related languages, you may wish to refer to the following publications:

3270 PC Online Tutorial SA23-0163

3270 PC Control Program User's Guide & Reference 1837434

3270 PC Guide to Operations 1837432

IBM PC Disk Operating System Manual 6024061

IBM PC BASIC Manual 6025010

IBM PC BASIC Compiler Manual 6024003

IBM PC COBOL Reference 6024011

IBM PC PASCAL Manual 6024010

IBM PC Macro Assembler Manual 6024002

Contents

Statement of Service

How to Obtain Service for the 3270 PC High Level Language Application Program Interface

Product service is available to registered users of the IBM 3270 Personal Computer High Level Language API (1753180). In order to register for this service, complete the enclosed postage paid Service Registration Card and return to:

IPS Product Support Center
IBM Corporation
P.O. Box 152560
Irving, TX 75015-2560

This product is provided " AS IS " without warranty of any kind, either expressed or implied.

Description of the Service Available

If you believe that you have a problem which is caused by a defect in an unmodified portion of the program or documentation during the period of product service, you may call IBM with a description of the problem with any related information. Postage paid Program Response Forms and Readers' Comment Forms are provided at the back of this publication for your use. The Toll Free number is 800-527-5068.

On a best effort basis, your documentation will be used to determine whether a program correction is necessary and:

1. You will be sent a notice of availability or an update containing the correction if the problem is an error and has been fixed;

OR

2. A fix will be developed and sent to you, if the problem is an error but has previously been reported

OR

3. You will be advised that the problem is not caused by a product defect.

A response to the defect will be made only if the defect occurs when the product is used in the operating environment described in this document.

IBM does not guarantee service results or that the program will be error free or that all program defects will be corrected.

Duration of Service

Service will be available until December 31, 1986. During the service period if you transfer this license to another party you must provide IBM with information about the new user by submitting a Program Response Form with the new user's name, company, address, telephone number and the Program Service Registration Number.

Chapter 1. Introduction

Why would I use this Program Offering?

- To generate host transactions from a personal computer program.
- To read data from a host session or notepad into a personal computer program.
- To simplify operator interactions with 3270 Personal Computer workstation control functions.
- To simplify host interaction and setup.

In other words, the 3270 PC High Level Language API can be used in conjunction with the capabilities of the 3270 Personal Computer to enhance the PC application developer's ability to conduct interactive sessions with an existing 3270 based host application.

Things You Need

Hardware

The 3270 PC High Level Language API is designed to operate on the IBM 3270 Personal Computer machine configuration 5271 standard models 2, 4, and 6. It requires approximately 12,000 bytes of memory in addition to the requirements of PC DOS and the 3270 Personal Computer Control Program.

NOTE: The 3270 PC High Level Language API is not supported on the 3270 PC/G or 3270 PC/GX.

Software

- PC DOS 2.0 or 2.1 is required.
- Only release 1.2 of the 3270 Personal Computer Control Program is supported.
- Interrupts 44-45 (hex) are reserved.
- The IBM PC Macro Assembler is required if you will be creating your own Language Interface Modules.

NOTE: The use of Programmed Symbols is not supported through the application interface.

Features

The 3270 Personal Computer High Level Language Application Programming Interface is a set of programs which allows you to perform the following functions on the 3270 Personal Computer:

- Connect to a target presentation space.
- Send keystrokes to the presentation space.
- Wait for a host response.
- Read the presentation space (or a selected string from the presentation space.
- Query the number and types of presentation spaces.
- Reserve a presentation space for exclusive use by your program.
- Release a reserved presentation space.
- Search a presentation space for a specific string.

The use of these functions will allow you to generate host transactions, check the transaction status and then copy data from the presentation space into your personal computer program. These calls allow you to develop a "programmed operator" to carry out host interactions. This simple set of building blocks will allow the programming of a variety of tasks extending from simple operator assistance to sophisticated "cooperative processing" applications.

Host Requirements

- There are no special host requirements other than those required for the installation of the 3270 Personal Computer.

Overview of System

The 3270 PC High Level Language API is implemented in two parts. The system is composed of a Language Interface Module (LIM) that is specific to a given language (eg: BASIC, COBOL, PASCAL) and a resident module (PCIRES) that handles the actual interface functions between your program and the 3270 PC Control Program.

The Language Interface Module

Each implementation of a programming language has unique methods of storing data and calling subroutines. In order to support multiple languages, the system is designed to use a Language Interface Module (LIM) as a "bridge" between your application and the main interface program. Your application calls the appropriate Language Interface Module which then takes your program's parameters and passes them to the main program in a standard format.

Language interfaces are provided for both Compiled and Interpretive BASIC, IBM COBOL and IBM PASCAL. Programs written using the MACRO Assembler can invoke PCIRES directly. If you prefer to use another language, see Appendix C for information on writing a Language Interface Module tailored to your needs.

You will invoke the program's services in one of two ways. If you are using a compiled language (such as COBOL or the BASIC Compiler) you call the appropriate language interface as an external subroutine. When you linkage edit your program (using the DOS LINK command), you will include the appropriate language interface object module.

Examples are provided for each language in the section describing that language's interface.

If you program in Interpretive BASIC, you will have to use another technique for calling the interface. Since Interpretive BASIC has no provision for using the linkage editor to include an external subroutine, a technique is provided for the user to access a special Interpretive BASIC Language Interface Module. This language interface is available whenever PCIRES has been made resident. An example of how to access this interface is provided in Chapter 4.

The Resident Module

After building a Parameter Control Block, the language interface invokes the resident module (PCIRES) with a software interrupt. PCIRES must be loaded prior to its use and remains in memory as a resident extension of PC DOS. PCIRES performs the requested function based on a function code and other parameter data passed in the control block built by the Language Interface Module. After completing (or attempting) the requested task, a return code is placed in the control block and control is returned to the calling Language Interface Module. The interface module then returns control to your calling program.

Chapter 2. Getting Started

Diskette Contents

The distribution diskette contains the following set of source, object and executable modules:

- PCIRES.EXE - The primary executable function module.
- PCIRES.ASM - The source to the main function module.
- COBLIM.ASM - The IBM COBOL language interface module's source.
- COBLIM.OBJ - The COBOL interface object module to use with LINK.
- CBASLIM.ASM - The BASIC Compiler language interface module source.
- CBASLIM.OBJ - The BASIC Compiler's interface object module.
- PASLIM.ASM - The IBM PASCAL language interface module source.
- PASLIM.OBJ - The PASCAL interface object module.
- SAMPLE1.BAS - A sample program in BASIC.

The distribution diskette should be copied to a backup diskette using the standard DISKCOPY program provided with PC DOS.

To assemble the source modules provided, the IBM PC Macro Assembler is required. However, unless you are developing your own Language Interface Module there is no requirement to reassemble any of the program modules. If you do wish to develop your own language interface module, you may find the distributed source programs to the standard language interfaces useful as sample programs.

Loading the Program

Prior to invoking the interface functions from an application, you must load the resident program as an extension to PC DOS. To do this, key in the command:

```
     PCIRES
```

and press ENTER. The program will load and display several identification and copyright messages. It should then produce the message:

```
PCI-001  PCIRES is now resident
```

If you receive any other message refer to Appendix A - Messages. Once this module is loaded it does not need to be reloaded unless PC DOS is rebooted. Once loaded the only way to remove the PCIRES module is to reboot PC DOS.

With PCIRES now resident you are ready to invoke the system's function from your program. The details for each language are outlined in Chapter 4.

Installing and Running the Sample Program

After you have successfully loaded the PCIRES program you are ready to use the interface. The distribution diskette contains an Interpretive BASIC sample program (SAMPLE1.BAS) that lets you experiment with the various function calls in a "real time" mode. The sample program allows you to specify a function and its parameters and observe the results. To run the sample you should use BASICA to run the SAMPLE1 program. If you are unfamiliar with running programs under BASICA refer to the IBM Personal Computer Hardware Reference Library BASIC Manual (6025010). The sample program will prompt you for the necessary input.

Using The Program

The 3270 PC High Level Language API is a "function code" driven system. For each service that you require you will call the interface with a function code requesting that service. The service functions may require other parameter data in support of your request. The function codes and the requisite parameters are described in Chapter 3 - "The Function Calls". After completing your request any appropriate data and a return code will be passed back to your program. Specific programming conventions and requirements for the supported languages are described in Chapter 4 - "Using the Functions from your Application"

How to Call the Functions

In order to use the program, your application must call a Language Interface Module (LIM) passing it the necessary parameters for that call. The interface will perform the requested function and will then return a status code. The requirements for each language are documented separately but, in general, each call passes the following data:

- A function code
- A character string
- The string's length
- A return code

For calls that don't require each of these elements, the LIMs still require four parameters - this allows your program to contain a single call format. (If you are calling a function that doesn't require all four parameters, the unused parameters do not need to be initialized). Likewise, many return codes are designed to conform to a standard structure so that they can be analyzed by a common error handling subroutine in your program.

Chapter 3. The Function Calls

Func Code	Name	Reqd Parms	Description
01	CONNECT	Sess-id Retcode	Access the requested Presentation Space as the active session.
02	DISCONNECT	Retcode	Drop the connection to a presentation space.
03	SEND KEY	Length Keystring Retcode	Send keystrokes to the active session.
04	WAIT	Retcode	Wait for X clock/X system to clear and return status.
05	COPYPS	String Retcode	Translate the presentation space to ASCII and move it to the user's buffer.
06	SEARCH	Length Srch Str Retcode	Search the Presentation Space for a given string.
08	COPY STRING	Length String Offset	Translate a string to ASCII and move it to the user's string.
09	SET SESSION PARMS	Length Parm Str Retcode	Sets a variety of program parms.

Func Code	Name	Reqd Parms	Description
10	QUERY SESS	Data Retcode	Get a list of sessions and parms. (Name, screen size, type, etc.)
11	RESERVE	Retcode	Locks the current session to inhibit user input.
12	RELEASE	Retcode	Unlocks the current session.

CONNECT

Function code - 01

Parameters: Session-id.
Return code.

Purpose:

Connect puts the PC program in logical session with a specified presentation space. Any subsequent calls will use the selected presentation space as the active session.

Remarks:

Prior to any call to a logical session the PC program must CONNECT to that session. To establish a connection, the 1 byte "short name" of the presentation space is passed during the call to CONNECT. It is not necessary to set a length parameter since the session name is a 1 byte field. The short name is the single character identifier you have defined to the Control Program during customization "naming" that session.

CONNECT sets the return code to indicate the status of the connection attempt and, if successful, the current status of the selected session.

Connect operates in two modes - physical or logical connect. If the session parameters specify logical connect mode (which is the default) the connection to the requested session occurs internally and the personal computer session continues to be displayed as the active window. On the other hand, if a physical connection is requested, the 3270 Personal Computer "jumps" to the requested presentation space just as though the operator pressed the JUMP key. In this mode all keystrokes from the operator will then go to the active session rather than to the personal computer session.(See the SET SESSION PARMS function description for information on how you set the connection mode).

The use of CONNECT only directs the internal keystroke input from your PC program. Input from the actual keyboard remains directed to the presentation space currently active on the display.

You can connect to a presentation space that is already connected. It is not necessary to DISCONNECT prior to CONNECTing to another presentation space; however you are only in session with one presentation space at a time.

You may connect to workstation control (WSCTRL) by using the symbol: # ("pound sign") as the session-id. SENDKEY is the only function you may use with WSCtrl. When you send keystrokes to WSCTRL, an entire workstation control function must be completed in a single keystroke string. For complex tasks such as COPY, this may involve some lengthy and complex keystroke strings.

CONNECT Return Codes

0	Connection was successful, selected presentation space is unlocked and ready for input.
1	Unable to connect to selected session name.

2	Not used.
3	Connection was successful, connection is to WSCTRL.
4	Connection was successful, presentation space is busy.
5	Connection was successful, presentation space is locked (input inhibited).
9	System error.

Example:

```
1000  FUNC% = 1
1010  DATASTR$ = "B"
1020  CALL BLIM(FUNC%,DATASTR$,DLEN%,RETC%)
1030  IF RETC% < > 0 THEN GOSUB 9000
```

This would connect to the 'B' presentation space. Error analysis is done (if necessary) in a subroutine at 9000.

DISCONNECT

Function code - 02

Parameters:	None.
	Return code.

Purpose:

DISCONNECT drops the connection between the PC and the last connected Presentation Space.

Remarks:

Even after disconnecting, routines that reference the presentation space (COPYPS, SEARCH and COPYSTRING) still reference the last connected presentation space.

Once a disconnect has occurred, calls that must interact with an active presentation space are no longer valid (SEND KEY, WAIT, RESERVE, RELEASE).

It is not necessary to DISCONNECT after each call to the interface. The PC program may stay connected to a presentation space for a series of calls. However, you should always issue a DISCONNECT at the completion of a set of programs. Failure to disconnect can inhibit the use of other 3270PC functions (such as file transfer). If you wish to leave the operator in session with a given presentation space at the completion of your program, you should "physically" connect to the presentation space, change the session parameters to "logical" connect mode (see the SET SESSION PARMS function) and then DISCONNECT.

If the program is in "physical connect" mode, disconnect jumps back to the PC window.

DISCONNECT return codes

0	DISCONNECT was successful.
1	Program was not connected.
9	System error.

Example:

```
1000  FUNC% = 2
1010  CALL BLIM(FUNC%,DATASTR$,DLEN%,RETC%)
1020  IF RETC% < > 0 THEN GOSUB 9000
```

SEND KEY

Function code - 03

Parameters:	String of keystrokes.
	Length of string of keystrokes.
	Return code.

Purpose:

SEND KEY sends a string of keystrokes to the connected presentation space.

Remarks:

You must issue CONNECT to a logical session (active presentation space) prior to sending keystrokes. This session can be either a 3270 session, a notepad, or workstation control (WSCTRL). The session must have the keyboard unlocked for input prior to accepting keystrokes. Autokey should not be active.

The string of keystrokes is handled just as though it was entered by an operator at the keyboard; all fields that are protected for input or are numeric only must be treated accordingly.

Keystroke input is no longer accepted after the first AID character is received (Clear, PA, PF or Enter).

You may only send 255 keystrokes in a single call to SENDKEY. If you need to key in more data than this you may do several calls to SENDKEY before you send the ENTER key.

If you are sending keystrokes to Workstation Control (WsCtrl), you MUST complete an entire WsCtrl "task" in a single keystroke string. You CANNOT do a portion of a function, such as copy, and then attempt to send another set of keystrokes to complete the task.

In order to send the special 3270 control keys, a compound character coding scheme is used. This coding technique uses 2 ASCII characters to indicate 1 keystroke and is comprised of the @ sign followed by a key code.

As an example, to key the sequence: LOGON 12345 followed by an Enter key; the following string would be coded: LOGON 12345@E . The purpose of this compound coding scheme is to allow an ASCII string representation of all necessary keystroke codes without requiring the use of complex "hex" key codes. A complete list of the keycodes is found in Appendix B. The string length passed to the program should count these compound characters as being 2 bytes long.

SEND KEY Return Codes

0	Keystrokes were sent, normal status.
1	Not connected to a valid session.
2	Bad parameter was passed to interface.
3	Keystrokes were sent, you are in WSCTRL mode.
4	3270 session was busy, keystrokes could not be sent.
5	3270 session was locked, all keystrokes could not be sent.
9	System error.

Example:

```
1000  FUNC% = 3
1010  DATASTR$ = "XREF 12345@E"
1020  DLEN% = 12
1030  CALL BLIM(FUNC%,DATASTR$,DLEN%,RETC%)
1040  IF RETC% < > 0 THEN GOSUB 9000
```

This example keys in XREF 12345 and hits ENTER in the selected presentation space.

WAIT

Function code - 04

Parameters: Return code.

Purpose:

WAIT checks the status of the logical session; if the 3270 session is waiting on a host response (X clock or X SYSTEM) the interface waits several seconds to see if the condition will clear.

Remarks:

The interface must be connected to a valid presentation space prior to issuing this call. The PC application program should always analyze the return code issued by wait in case of an error.

The SET SESSION PARMS function can be used to change the "timeout" option for WAIT. When waiting for an "X Clock or X SYSTEM" to reset, the system normally waits a short period of time before reporting a host busy condition. You can specify this wait to always wait for completion (the LWAIT option) or to not wait at all (NWAIT). You should use caution in requesting the LWAIT option since the system will not return control to your program for error recovery in the event the host never responds.

WAIT Return Codes

0	Keyboard unlocked and ready for input.
1	Not connected to a valid session.
2	not used
3	Connected to WSCTRL.

4 Timeout while still in "X clock" or "X SYSTEM" mode.

5 Keyboard is inhibited.

9 System error.

Example:

```
1000  FUNC% = 4
1020  COUNT = 0
1030  CALL BLIM(FUNC%,DATASTR$,DLEN%,RETC%)
1040  IF RETC% = 0 THEN GOTO 2000
1050  IF RETC% < > 4 THEN GOSUB 9000
1060  COUNT = COUNT + 1
1070  IF COUNT < 10 THEN GOTO 1030
1080  PRINT "THE HOST HAS NOT RESPONDED"
1090  END
2000  . . . . . .
```

This routine waits for a host response. If an error is detected a routine at line 9000 is invoked. If the host remains busy (X Clock or X System) for a predetermined length of time, the program will end.

COPYPS

Function code - 05

Parameters: Data String.
Return code.

Purpose:

COPYPS copies the contents of the last selected presentation space into the user's data area.

Remarks:

COPYPS translates the presentation space into ASCII. Attribute bytes and other characters not represented in ASCII are translated to blanks. Extended attributes are not copied to the PC session.

This call is not valid from interpretive BASIC since it would exceed the maximum string size allowed. Users of other languages should ensure their data areas can handle the largest possible presentation space size. If you want only a portion of the presentation space use the COPY STRING function. You can use the QUERY SESSIONS call to obtain the presentation space sizes.

This call does not require the PC session to still be connected to the presentation space; it will pass back the contents of the last connected session.

If you don't want the attribute bytes translated to blanks, use the ATTRB option of SET SESSION PARMS.

COPYPS Return Codes

0	PS Contents transferred, session was active and keyboard unlocked.
1	PS Contents transferred from last active session, however the session is no longer actively connected.
2	not used.
3	PS contents transferred, was in WSCTRL mode.
4	PS contents transferred, active session was waiting for host response.
5	PS contents transferred, keyboard locked.
9	System error.

Example:

This function is not available to Interpretive BASIC users.The following example could be used in a compiled Basic program:

```
1000  FUNC% = 05
1005  DATASTR$ = SPACE$(1920) 'Preallocate target string
1010  CALL BLIM(FUNC%,DATASTR$,DLEN%,RETC%)
```

This example assumes that the receiving data string is at least as large as the selected presentation space.

SEARCH

Function Code - 06

Parameters:	String to search for.
	Length of string to search for.
	Return code.

Purpose:

SEARCH allows the program to examine the presentation space for the occurrence of a specified string.

Remarks:

SEARCH scans the selected presentation space for the first occurrence of the specified string. If the string is not located, the return code is set to 0. If the string is found, the return code is set to the string's beginning location in the presentation space. This location represents an offset into the presentation space based on a layout where the upper left corner (home position) is location 1 and the bottom right (for a "3270 Model 2") is location 1920.

The SEARCH function is useful in determining when a specific 3270 screen is available. If you are expecting a specific prompt or message before keying in data, SEARCH allows you to check for the message before continuing. If the expected prompt has not yet been sent, you can continue calling SEARCH until a non-zero return code is received.

SEARCH normally checks the entire presentation space, however using the SET SESSION PARMS function you can specify "SRCHFROM" mode. In this mode you may specify a starting offset for SEARCH, which will then look for the string from that offset thru the end of the presentation space. As an exception to the normal call usage, the starting offset is passed to the interface in the return code parameter. The return code is still returned in the normal way. This option is useful if you are looking for a keyword that may have multiple occurrences in the presentation space since SEARCH only reports the first location.

SEARCH examines the last connected presentation space if the PC program is not currently connected.

SEARCH Return Codes

0	The string was not found.
>0	The string was found at this offset in the PS. (The presentation space contains offsets from 1 thru the max screen size)

Example:

```
1000  FUNC% = 6
1020  DATASTR$ = "ENTER PART #:"
1030  DLEN% = 13
1040  COUNT = 0
1050  CALL BLIM(FUNC%,DATASTR$,DLEN%,RETC%)
1060  IF RET% < > 0 THEN GOTO 2000
1070  COUNT = COUNT + 1
1080  IF COUNT < 5000 THEN GOTO 1050
1090  PRINT "THE CORRECT HOST SCREEN HAS NOT
      BEEN RECEIVED"
1095  END
2000  . . . . .
```

This example looks for a host prompt to be sent. If, after a predetermined period of time, the message is not received the program ends. As an alternative to ending the program, the program could be designed to search for several alternative responses. For many applications, designing a comprehensive set of SEARCH calls for possible host responses will be a key element of your application design.

COPY STRING

Function code - 08

Parameters:	String variable.
	String length.
	Offset in PS to the string.

Purpose:

COPY STRING moves a string from the last connected presentation space into the user's string variable.

Remarks:

The offset into the PS is based on a 1's origin (ie 1 through 1920 for a 3270 model 2 session). The value of offset + length may not exceed the maximum screen size for that presentation space. As an exception to the normal calling conventions, the return code is also used to pass the beginning offset to the interface.

The string is returned as an ASCII string. The requested length must not exceed the length of the receiving user buffer.

COPY STRING Return Codes

0	The copy was successful.
2	Parameter error.

Example:

```
1000  FUNC% = 8
1010  DATASTR$ = SPACE$(80)
1020  DLEN% = 80
1030  RETC% = 1
1040  CALL BLIM(FUNC%,DATASTR$,DLEN%,RETC%)
```

This example copies the first line of the presentation space into a personal computer program variable called DATASTR$. Note that DATASTR$ had to be pre-allocated. Also notice that the return code was used as an input parameter.

SET SESSION PARMS

Function code - 09

Parameters:	Parameter string.
	String length.
	Return code.

Purpose:

SET SESSION PARMS allows you to change certain default options of the PCIRES module.

Remarks:

The parameters can be separated by any delimiter you prefer. You must NOT imbed blanks in the keywords. Misspelled keywords are not flagged in error.

The new session parameters go into effect at the conclusion of the function call and remain in effect until changed again or the resident program (PCIRES) is reloaded.

You must always specify an explicit string length even if you are in EOT mode. (See the STREOT parameter below).

The following is a list of the specifiable options:

NOTE: DEFAULTS ARE IDENTIFIED BY * PRECEDING OPTION.

ATTRB	Pass back all codes that do not have an ASCII equivalent as their original value.
***NOATTRB**	Convert to blanks all unknown values.
CONPHYS	During CONNECT, Jump to the requested Presentation Space. (do a physical connect)
***CONLOG**	During CONNECT do not jump to the Presentation Space. (do a logical connect)
ESC=n	Specify the escape character for keystroke mnemonics (@ is the default). Do not leave a blank after the equal sign.
***STRLEN**	An explicit length will be passed for all strings.
STREOT	String lengths are not explicitly coded, strings are terminated with an EOT (End Of Text) character.
EOT=	Allows you to specify the EOT character for string terminators (in STREOT mode). Binary zero is the default. Do not leave a blank after the equal sign.
***SRCHALL**	SEARCH will scan the entire presentation space.
SRCHFROM	SEARCH will start from a specified beginning offset.

***AUTORESET**	The program will attempt to reset inhibited conditions by prefixing all SENDKEYS with a reset.
NORESET	Do not AUTORESET.
TRON	Turns on a trace of interface calls.
***TROFF**	Turns the trace off. NOTE: The Trace function may conflict with messages on the screen from languages or applications that manage their own displays.
***TWAIT**	Wait will time out on a long X Clock/X SYSTEM.
LWAIT	Long Wait - will wait until X "clock"/ X SYSTEM clears. This option is NOT recommended since control will not return to your program until the host is available.
NWAIT	Wait checks status and returns immediately (No Wait).

SET SESSION PARMS Return Codes

0 The session parameters have been set.

2 The length of the parameter list is invalid.

Example:

```
1000  FUNC% = 9
1010  DATASTR$ = 'STREOT,EOT=!'
1020  DLEN% = 12
1030  CALL BLIM(FUNC%,DATASTR$,DLEN%,RETC%)
```

This example sets the session parameter to allow strings to be specified with an ending delimiter. In this mode, you do not use the string length parameter. Instead you use an EOT character (in this example an exclamation point is used).

QUERY SESSIONS

Function code - 10

Parameters:	Empty string for session data. Return code.

Purpose:

QUERY SESSIONS sets the return code to the number of Host and Notepad sessions defined to the 3270PC Control Program. A data string is returned describing each of these presentation spaces.

Remarks:

The string MUST always be large enough to contain 6 session descriptors of 12 characters each (i.e. 72 bytes). The format of each descriptor record is as follows:

Short name	1 byte ASCII character.
Long name	8 bytes ASCII character.
Session type	1 byte ASCII character. (H = Host, N = Notepad)
PS size	2 bytes binary number.

Note that the presentation space size is a binary number and is not in display format.

QUERY SESSIONS Return Codes

The return code is set to the number of configured host and notepad sessions.

Example:

```
1000 FUNC% = 10
1010 DATASTR$ = SPACE$(80) 'PREALLOCATE THE STRING
1020 DLEN% = 72
1030 CALL BLIM (FUNC%,DATASTR$,DLEN%,RETC%)
```

RESERVE

Function code - 11

Parameters: Return code.

Purpose:

RESERVE locks the active presentation space to prevent all user keyed input.

Remarks:

The presentation space remains locked until a RELEASE is issued. Multiple presentation spaces can be locked. The RESERVE function does NOT prevent the user from accessing the WSCTRL functions.

This function can be used to prevent the terminal operator from keying into a selected presentation space. If your program is sending a series of transactions to a host, you may need to prevent the user from gaining access to that session until your program completes. Without issuing a RESERVE, the interface only inhibits keyboard input during the processing of the SENDKEY keystrokes.

You may not lock the workstation control session. The session being reserved should not be in an inhibited status when RESERVE is issued.

RESERVE Return Codes

0	The function was successful.
1	Not connected.
5	Inhibited.
9	System error.

Example:

```
1000  FUNC% = 11
1010  CALL BLIM(FUNC%,DATASTR$,DLEN%,RETC%)
1020  IF RETC% < > 0 THEN GOSUB 9000
```

If an error is detected, a routine at "9000" will analyze the condition.

RELEASE

Function Code - 12

Parameters: Return code.

Purpose:

RELEASE unlocks a previously RESERVED presentation space.

Remarks:

The presentation space to be released must be the currently connected session.

If you forget to release a reserved presentation space before ending your program, the user will be locked out of that session until another program releases it or the Control Program is reloaded.

RELEASE Return Codes

0	Release was successful.
1	Not connected.
9	System error.

Example:

```
1000   FUNC% = 12
1010   CALL BLIM(FUNC%,DATASTR$,DLEN%,RETC%)
```

Chapter 4. Using The Functions From Your Application

Using BASIC

Users of BASIC have a choice of two programming environments: Interpretive BASIC and Compiled BASIC. While these environments are similar, differences exist in subroutine linkage and string handling. If you use interpretive BASIC you will need special initialization programming to access the interpretive BASIC language interface that is in the PCIRES module. This programming locates the interface from a fixed pointer location and uses this address to set the SEG and OFFSET values for the subroutine call. If you don't understand the routine involved, don't worry; just copy the provided program statements into your program's initialization routine.

```
10   REM Sample Interpretive BASIC Initialization
20   DEF SEG = 0
30   INTR = &H45
40   SEGM = 256*PEEK(INTR*4+3) + PEEK(INTR*4+2)
50   BLIM = 256*PEEK(INTR*4+1) + PEEK(INTR*4)
60   IF SEGM <> 0 THEN 80
70   PRINT "PCIRES IS NOT AVAILABLE": END
80   DEF SEG = SEGM
```

Users of compiled BASIC can just call the Compiled Basic Language Interface Module (BLIM) as an external call and then at linkage edit time, using LINK, include CBASLIM.OBJ. The LINK command will be: LINK YOURPROG+CBASLIM;

To call the interface from BASIC, you use a fixed format call with four parameters. These parameters must occur in a fixed sequence and all four parameters must be in the call list - even if they aren't used. If a parameter isn't needed for a given call, you don't need to value it. The call format is as follows:

```
100 CALL BLIM (FUNC%,SDATA$,DLEN%,RETC%)
```

Where:

```
FUNC% is the function code.
SDATA$ is a string of data.
DLEN% is the data length.
RETC% is the return code.
```

The variables can have any names, however they MUST be of the proper type. The function code, data length and return code MUST be fullword integer variables. The data string MUST be a string variable.

Users of BASIC need to carefully review the restrictions for string processing. Since the maximum string length for interpretive BASIC is 255 bytes, the COPYPS function is not available and COPYSTRING must be used to extract segments of the presentation space. Furthermore, the string variables are managed dynamically in BASIC's data segment; this means that all strings MUST be pre-allocated before they can receive data from the interface. While this sounds complicated, all it means is that the subroutine package CANNOT change the size of a string so you must pre-format your string to the required size. For example:

```
1000  SDATA$=SPACES$(100) 'PREALLOCATE TO 100 BYTES
1010  FUNC%=08    'COPYSTRING FUNCTION
1020  DLEN%=80    'COPY 80 BYTES
1030  RETC%=1     'LOCATIONS 1-80 FOR COPY
1035  REM NOTE USE OF RETC% TO PASS OFFSET FOR COPYSTRING
1040  CALL BLIM(FUNC%,SDATA$,DLEN%,RETC%)
```

If you forget to preallocate your strings, the language interface attempts to detect this condition and will return a parameter specification error.

An issue related to string preallocation for interpretive BASIC is string performance. BASIC manages strings dynamically. As you assign and reassign data to a string, BASIC reacquires space in its dynamic storage pool. While this is not a problem for many programs, if you write a complex application that uses strings frequently, BASIC may go into "garbage collection" resulting in a lengthy pause in your program's execution while BASIC compresses strings. If you are writing complex or long running applications you should understand the implications of interpretive BASIC's string usage and how to avoid these performance issues. As an alternative, Compiled BASIC avoids many of these problems.

Using COBOL

The COBOL program should declare the parameter variables in the Data Division. The following is an example of the parameter declaration.

```
77  FUNCODE      PIC 99        COMP-0.
77  RETCODE      PIC 99        COMP-0.
77  STRLEN       PIC 99        COMP-0.
01  DATA-STRING  PIC X(1920)   VALUE SPACES.
```

You have the option of changing the data names to fit your own programming conventions. The string variable should always be large enough to receive the largest presentation space size you will request.

The call to the COBOL Language Interface would be coded as follows:

```
CALL 'COBLIM' USING FUNCODE DATA-STRING
              STRLEN RETCODE.
```

Once you have compiled your program, you will LINK it. To do this you need the LINK command from PC DOS and the language interface COBLIM.OBJ module. The LINK command may be issued as:

```
LINK   COBPROG + COBLIM;
```

This will then produce an executable module called COBPROG.EXE. After loading PCIRES you may execute this COBOL program in the normal manner.

Using PASCAL

The PASCAL program needs to declare the required variables and the PASCAL language interface module. The following is an example of the necessary definitions:

```
VAR
   API_FUNC, API_LEN, API_RETC: INTEGER;
   API_STRING:                  STRING (255);
   API_SCREEN:                  STRING (1920);

   PROCEDURE PASLIM (VAR TST_FUNC: INTEGER;
                     VAR TST_STR: STRING;
                     VAR TST_LEN, TST_RETC:
                     INTEGER); EXTERNAL;
```

The call to the language interface would then be coded:

```
PASLIM (API_FUNC, API_STR,
        API_LEN, API_RETC);
```

Once you have compiled your program, you will need to linkage edit it. Using the PC DOS "LINK" command you will link the PASLIM.OBJ module into your program. Assuming PASLIM.OBJ is on the same disk as your object module, you should issue the following command:

```
LINK YOURPROG+PASLIM;
```

On completing this step, your program is ready to execute.

Using 8088 Assembler

If you program in the Assembler language you do not need a Language Interface Module. You may directly invoke the PCIRES module by issuing an Interrupt 44H. At the time of the interrupt DS:SI should point to a Parameter Control Block specifying your parameters. For more detail on how to access the interface from the Assembler language, see Appendix C - Writing Your Own Language Interface Module.

Chapter 5. Application Development Tips

Defining A Programmed Operator

There are several techniques that you may find useful in developing your applications. To write an effective program you must think like a terminal operator. When an operator sits down at a terminal, they typically look at the screen, determine "where they are" in the application and take the necessary actions to get to the desired starting point. This may involve an action as simple as "hitting Clear" or may require routing through a VTAM network to the proper application. The SEARCH function is useful for determining which keyword messages or prompting messages are on the terminal.

Once you have gotten to the correct starting point you will then begin creating host transactions and reading the results. After you create a host "transaction" and enter it, getting the data you want is a three part process. If you think about what an operator does after entering a transaction, you can break it down into the three phases: 1) Waiting for the host to respond, 2) Analyzing the response to see if it is the expected response and 3) Extracting and using the data from the response. The elements of the application interface allow you to carry out these same activities. Using SENDKEY you can key in and enter a host transaction. The WAIT function waits for the "X Clock or X System" to go away (or returns a keyboard locked condition if the terminal has locked up). The SEARCH function allows you to look for an expected keyword to validate that you received the proper response. Finally, if you have the expected data, you can use COPYPS and COPY STRING to extract the data you want.

The SEARCH function is also key to simulating another task of the terminal operator. Some host systems do not stay locked in "X Clock/X System" mode until they respond; instead they quickly unlock the keyboard and allow the operator to stack other requests. In this environment the operator depends on some other visual cue to know that the data has returned (perhaps a screen title or label). SEARCH allows you to do the same thing since you can loop doing SEARCHes while waiting for the expected cue (or checking for multiple possible cues). As a good programming practice you should keep a counter during this loop and after some appropriate number of iterations have the application timeout and produce an error indicating the host has not responded as expected. Also, change control becomes very important in this environment since the "programmed operator" must be retrained (reprogrammed) for even minor changes in the display dialogues. As an example, if a person expects the message:

ENTER PART NUMBER:

as a prompt, they will probably be able to respond properly to an application change that produces the message:

ENTER COMPONENT NUMBER:

However, if your "programmed operator" is looking for a keyword string, subtle changes in message syntax, even one as trivial as upper versus lower case, can completely fool the program.

Sending Keystrokes

There are several techniques you should consider when sending keystrokes. Again, it is necessary to mimic the actions of a terminal operator. Some application environments are very simple and you need merely to key in a command or transaction code and hit the ENTER key. Other applications involve more complex 3270 formatted screens. In this environment you must understand the keystrokes required to "fill in" the display panel. The Tab key mnemonic (@T) can be used to skip between fields. Care must be used, however, when using tabs with autoskip attribute bytes on the 3270 session. If your keystrokes exactly fill a field the autoskip will take you to the next field. If you then issue a tab you will skip again to still another field. Also, you need to anticipate any "left over" data in the field you will be keying into and possible issue an "Erase EOF" to clear the remainder of the field.

String Specification

The program, as a default, requires you to pass the length of all strings in the length parameter. If you wish to avoid calculating your string lengths, you can use an "End of Text" (EOT) character at the end of every string. You specify the EOT character and the use of this mode with the SET SESSION PARMS function. Thus, after a SET SESSION PARMS command of:

```
STREOT, EOT=!
```

the string "XREF 12345@E!" would key in the transaction:

```
XREF 12345
```

and then hit the ENTER key. The exclamation point would serve as the end of string delimiter and, therefore, no length declaration would be required. In this mode, all strings you send (keystrokes and search arguments) must have the EOT character. The SET SESSION PARMS command ALWAYS requires an explicit length. Also, remember that the session parameters remain set for the duration that the PCIRES module is loaded.

Performance

The interface is designed to simulate interactive terminal activity. As a result, the performance of keystroke entry is similar to that of a typist. For applications requiring the input of large quantities of data, you should also evaluate the use of the various batch file transfer programs.

Appendix A. Messages

Messages from the 3270 Personal Computer High Level Language Application Program Interface can be identified by the prefix "PCI" and a message code followed by the message text. The message codes have the following format:

- PCI-NNN 'Message Text'

Where 'NNN' is a message ID.

Common Messages

PCI - 001 PCIRES is now resident
The program is successfully loaded and available for your use.

PCI - 009 Return Code = XXXX
If you have specified TRON (Trace On) as a session parameter this message will indicate ending status from each function call. Note that this message may overlap with messages from programs that track their own screen management (ie, BASIC).The value is displayed in hex.

PCI - 901 Internal Error XXX
The program has detected an error in its internal processing. XXX is a component identifier. You should retry the operation after reloading the Control Program and PCIRES.

PCI - 902 Internal Error during Initialization
A program error has prevented the program from initializing properly. This is an internal system error.

PCI - 903 Unable to install PCIRES

The program was not able to make itself resident.

PCI - 907 PCIRES is already resident

The program is already resident. If you have not already loaded PCIRES, another program is using the interrupt vectors needed by PCIRES.

PCI - 933 Program Damage has been detected

The PCIRES module has detected a problem with the internal stack or possible stack or overlay damage. Be sure your program is not altering memory in an unexpected fashion.

PCI - NN0 IN XXXXX

If you have specified TRON (Trace On) as a session parameter, this message will indicate which function you have invoked. NN will be the function number and XXXXX will be the function name. The trace messages may conflict (overlap) with your application messages if you are using a language like BASIC that manages its own display.

Appendix B. Key Mnemonics

A set of keyboard mnemonics is provided to allow you to use ASCII characters to represent the special function keys of the 3270 Personal Computer keyboard. The use of an escape character signals the SENDKEY function that the next character will represent a special key function. (The default escape character is "@" but you can change this with the SET SESSION PARMS function.) As an example of this technique, code an ENTER key by using "@E".

To make these special keys easier to remember, a mnemonic abbreviation scheme has been used. In trying to keep it easy to use, most common keys have been provided - for example, the CLEAR KEY is "C", the TAB key is "T", etc. Please note that the upper and lower case alpha characters are mnemonic abbreviations for different keys. To send a key that is in ALT shift, you simply send the @A preceding the appropriate key. For example, ALT PF1 would be encoded as @A@1 where @A indicates the ALT key and @1 is the mnemonic for PF1. The shift key works the same way (using @S). The Change Screen function, which is the SHIFT and JUMP keys, would be coded as @S@J.Notice that the key code actually indicates the key pressed and that ALT and SHIFT may change the "meaning" of the mnemonic (i.e. @A@C (ALT+Clear) is the Test key). Your regular text characters do not require any special shift codes - they will be entered in upper or lower case as required.

NOTE: To execute certain workstation control functions (for example JUMP) you must first connect to Workstation Control.

The following chart lists the valid codes for the special keystrokes. These mnemonics must follow the ESCAPE character which has a default of @ (At sign). To send the escape character as a keystroke, code the character twice (i.e. @@ sends a single "at sign").

A-ALT	I-Insert	Q-Finish(Quit)
B-Backtab	J-Jump	R-Reset
C-Clear	K-Copy	S-Shift
D-Delete Char	L-Left Cursor Move	T-Tab
E-Enter	M-Enlarge(Magnify)	U-Up Cursor Move
F-Erase EOF	N-New Line	V-Down Cursor Move
G-Not Used	O-Not Used	W-Not Used
H-Help	P-Print	X-Not Used
		Y-Not Used
		Z-Right Cursor Move

0	Home	7	PF7	e	PF14	l	PF21
1	PF1	8	PF8	f	PF15	m	PF22
2	PF2	9	PF9	g	PF16	n	PF23
3	PF3	a	PF10	h	PF17	o	PF24
4	PF4	b	PF11	i	PF18	x	PA1
5	PF5	c	PF12	j	PF19	y	PA2
6	PF6	d	PF13	k	PF20	z	PA3

Appendix C. Writing Your Own Language Interface Module

The 3270 Personal Computer High Level Language Application Program Interface is designed to allow you to develop support for languages and features that may be unique to your environment. By using a Language Interface Module as a bridge between the application program and the primary interface program, you can:

- Write a Language Interface Module for any language you may wish to support.
- Use an existing Language Interface Module as a gateway to other external functions.
- Change the format of the function calls or parameter lists to better suit your environment.

The primary function of a Language Interface Module is as follows:

- Handle linkage from the calling application
- Obtain data parameter pointers or values
- Convert any language unique data formats
- Construct a control block for the primary program
- Call (using an interrupt) the Resident Module
- Receive control back from the Resident Module
- Pass back necessary parameters and return control to the calling application

If you are writing support for a language you only need the above functions. If you want to extend these functions, there are two primary areas for interface extension: 1) Support for additional external functions and 2) Modifications or extensions of user calls to the primary program.

Frequently, users of Personal Computer languages develop multiple subroutine packages to provide utility functions such as display managers and data managers. Many of these packages require the application developers to call external subroutines. If you would like to consolidate several of your subroutines into one function package you may write extensions to the Language Interface Module for this purpose. Two different techniques are available. The easiest extension is to reserve a range of function codes for each subroutine package. The Language Interface then examines the function code being invoked and calls the appropriate subroutine package. You may need to make adjustments in the number and type of parameters passed to the interface . A second technique for interface extensions is more complex. Function Code 0 has been reserved for a SIGNAL GATEWAY function. In this mode, you can pass a "gatename" to your language interface. The interface can then use this function to establish routing for all subsequent function codes to a given subroutine. This technique is useful when existing subroutine packages have overlapping function codes. By switching all function requests through a specified gateway until a new "gateway" is requested you can access a variety of external functions through a common call architecture.

Revised Call Support for the 3270 Personal Computer

The provided Language Interface Modules are designed to provide a fixed format, easy to use, call format. You may wish to tailor the number or format of the parameters to your specifications. For example, calls to the interface use offsets into the presentation space. As an ease of use feature, you may want to redesign the function calls to use a row/column format. In that case your language interface would convert the row/column data into the offset values required by the actual interface function module. Other revisions you might make to the call architecture could include:

- Restrict certain functions from your application programmers
- Automatically setup the session parameters
- Extended validity checking of function calls
- Revisions in the call structure to fit your requirements.
- Support for different data formats.

Details of Writing Your Own Language Interface Module

The distribution diskette contains the source code for the provided Language Interface Modules. You can use these as a starting point for your own development efforts. The language interface is responsible for building a Parameter Control Block with the requisite data and issuing an INT 44H to call the function module. As a good programming technique, you should verify that an interrupt vector has been established for the interface's PCIRES module. The Parameter Control Block has the following format:

```
PCB_HDR      DB     'PCB'  ;PCB Header (in Caps)
PCB_FUNC     DB     0      ;Function code VALUE
PCB_DSEG     DW     0      ;String seg addr
PCB_DADDR    DW     0      ;String offset addr
PCB_LENGTH   DW     0      ;Data length VALUE
PCB_FILLER   DB     0      ;Unused
PCB_RETCODE  DW     0      ;Return code VALUE
```

The Control Block Header "PCB" (in upper case) must appear at the start of the control block. The function code must be the one byte numeric (binary) function code. The pointer to the user's data string must contain the segment and offset address (Note that the segment address precedes the offset). The string pointer should point to the actual data string and not a length or pointer prefix. The length should be the actual binary length value passed by the user. Since some function calls pass data in the fourth parameter (normally the return code) this value should be passed in the Parameter Control Block.

Once the control block is built, you call the resident module with the INT 44H interrupt. The DS:SI registers should point to the control block. You should save any required non-segment registers. After executing the requested function, control is returned to your next sequential instruction after the INT44. The return code is in the Parameter Control Block. You should pass this value back to the application caller's data area. Note that the interface passes the string data back directly but the return code is passed back to the Parameter Control Block and you must retain the user's data address and pass back the return code. This allows you to examine and, if necessary, reformat the return code.

Appendix D. Quick Reference Summary

Func Code	Name	Reqd Parms	Description
01	CONNECT	Sess-id Retcode	Access the requested Presentation Space as the active session.
02	DISCONNECT	Retcode	Drop the connection to a presentation space.
03	SEND KEY	Length Keystring Retcode	Send keystrokes to the active session.
04	WAIT	Retcode	Wait for X clock/X system to clear and return status.
05	COPYPS	String Retcode	Translate the presentation space to ASCII and move it to the user's buffer.
06	SEARCH	Length Srch Str Retcode	Search the Presentation Space for a given string.
08	COPY STRING	Length String Offset	Translate a string to ASCII and move it to the user's string.
09	SET SESSION PARMS	Length Parm Str Retcode	Sets a variety of program parms.

Func Code	Name	Reqd Parms	Description
10	QUERY SESS	Data Retcode	Get a list of sessions and parms. (Name, screen size, type, etc.)
11	RESERVE	Retcode	Locks the current session to inhibit user input.
12	RELEASE	Retcode	Unlocks the current session.

A-ALT	I-Insert	Q-Finish(Quit)
B-Backtab	J-Jump	R-Reset
C-Clear	K-Copy	S-Shift
D-Delete Char	L-Left Cursor Move	T-Tab
E-Enter	M-Enlarge(Magnify)	U-Up Cursor Move
F-Erase EOF	N-New Line	V-Down Cursor Move
G-Not Used	O-Not Used	W-Not Used
H-Help	P-Print	X-Not Used
		Y-Not Used
		Z-Right Cursor Move

0	Home	7	PF7	e	PF14	l	PF21
1	PF1	8	PF8	f	PF15	m	PF22
2	PF2	9	PF9	g	PF16	n	PF23
3	PF3	a	PF10	h	PF17	o	PF24
4	PF4	b	PF11	i	PF18	x	PA1
5	PF5	c	PF12	j	PF19	y	PA2
6	PF6	d	PF13	k	PF20	z	PA3

Appendix E. System Modifications

The primary vehicle for system customization is the Language Interface Module. In most cases this is sufficient, however, some users may have a requirement for additional modification of the PCIRES module itself. While this is not recommended, there are four areas where the systems programmer might consider making alterations.

Changing the Session Parameter Defaults

If you wish to change the session parameters to a different set of default options, the PCIRES module must be changed. The values are hardcoded into PCIRES in a text string beginning at label PCI__PARMS. These codes are ASCII character codes and you may either change them and reassemble PCIRES or ZAP the constants in the executable module.

Changing the Keystroke Mnemonics

The keystroke mnemonic compound characters use a set of key codes that is driven from a translate table at label CMPNDTABLE. The ASCII key code is translated against this table to determine the 3270 Personal Computer key scan code to send to the keystroke interface. Values for A, S and X are hard coded into application logic and should NOT be altered. If you wish to change the mnemonic definitions you may alter this table.

Note that the scan codes used are 3270 PC scan codes and NOT Personal Computer scan codes. You may obtain these scan codes by running the keyboard diagnostic program for the 3270PC.

Saving Storage

The PCIRES module contains an internal buffer that can contain the largest possible presentation space size. If you are in a memory constrained environment and will never have a display size larger than a 3270 Model 2 (1920 bytes), you may reduce the buffer size at VBUFFER to a smaller value. If you do this and a user generates a larger presentation space to the Control Program, it could cause an unrecoverable error.

Changing the Interrupt Number

The system uses two interrupt vectors (Hex 44 and 45). The first vector is used for the actual PCIRES interrupt. The second vector is used to provide a fixed pointer to the integrated Interpretive BASIC Language interface. If these interrupts conflict with other software you are using, you may change the interrupt assignments. The PCIRES program establishes the interrupts based on the value at label MYINTNUM. You may either ZAP this or change the value and then reassemble/relink the PCIRES module. Also remember to change the value in your Language Interface Modules and the value for the INTNUM in the Interpretive BASIC source modules.

Glossary

active window. The window outlined with a double border and containing the cursor, indicating the session to which your keyboard is logically attached.

Alt key functions. Keyboard functions printed on the front of the keys that are selected when the Alternate (Alt) key is pressed in conjunction with the desired key.

American National Standard Code for Information Interchange (ASCII). One of the two standard codes used for exchanging information among data processing systems and associated equipment; the standard code used by the IBM Personal Computer and other microcomputers.

autokeying. A control program capability that lets you record frequently used groups of keystrokes and plays them back at designated locations on the screen.

BASIC. Beginner's All-Purpose Symbolic Instruction Code. A high-level, widely used computer programming language.

combination keys. Keys that must be used in conjunction with another key or keys to produce a desired function; these include the personal computer keys Control, Shift and Alternate, and the host computer keys Shift and Alternate

connect. The function code which allows the application program to access the requested presentation space as the active session.

control program functions. The set of functions in the IBM 3270 Personal Computer allowing you to create windows, change their size and position, and hide and enlarge them; use autokeying and copying; save and restore autokey recordings, notepad contents, and screen profiles; and transfer files between host computer and personal computer sessions.

copy. The work station function that allows you to mark source lines in one text mode window and move them to a target location within the same window or from one window to another on the same screen profile or a different screen profile.

copy ps. The function code which allows the application program to translate the presentation space to ASCII and move it to the user's buffer.

copy string. The function code which allows the application program to move a string of data from the presentation space to the user's data area.

disconnect. The function code which allows the application to make the personal computer session the active session.

hide. The work station function that allows you to remove a window temporarily from the screen profile in which it appears: it does not delete the window or affect its presence on other screen profiles; it merely "hides it.

jump. The work station function that allows you to move from window to window on a screen profile; initiated with the Jump key or pressing the WS Ctrl key and the letter name of the window.

logging on. The procedure by which you are linked to a multiple-user host computer system; the procedure requires a user identification and a password.

menu. A list of available operations. You select which operation you want from the list.

notepad. The application mode session that contains notes you make to yourself at your work station; its keyboard is similar to the host keyboard; its contents disappear at the end of a session unless stored with the Save utility.

operator information area (OIA). The bottommost line of your screen where you receive information about the status of your work station and your host, notepad, and personal computer sessions.

presentation space. A concept that represents the area that contains the data that goes to your display screen during a 3270 host session, an IBM Personal Computer session, or a notepad session.

query session. The function code which allows the application program to get a list of sessions and parameter.

release. The function code which allows the application program to unlock the current session.

reserve. The function code which allows the application program to lock the current session to inhibit user input.

restore command. The work station command that allows you to load autokey recordings, notepad information and/or screen profiles from personal computer diskette or fixed disk storage to work station temporary storage.

save command. The work station command that allows you to move autokey recordings, notepad information and/or screen profiles from work station temporary storage to personal computer diskette or fixed disk storage.

search. The function code which allows the application program to search the presentation space for a given string.

selected window. In work station control mode, the double-bordered window that will be affected by work station functions.

sendkey. The function code which allows the application program to send keystrokes to an active session.

session. A connection between your work station and a host computer, a personal computer, or a notepad.

set session parms. The function code which allows the application program to set a variety of program parameters.

setup. The work station control function from which aspects of screen layout may be changed

short name. A letter name (A through Z) of a window displayed in the upper left corner of a window border.

valid key. A key that is recognized by the session type (host, personal compute or notepad) or mode (WS Ctrl) to which your keyboard is logically attach

wait. The function code which allows the application program to wait for the X Clock/X System to clear from the OIA and returns the status code.

window. The opening on the screen through which you view your application data. A window can be the same size as your full IBM 3270 Personal Computer screen or as small as one character.

work station control mode (WsCtrl). The master control mode of the IBM 3270 Personal Computer from which the control program functions are initiated.

IBM Personal Computer
3270-PC High Level Language Application Program Interface
SH20-6544-0

READER'S COMMENT FORM

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.*

Possible topics for comment are:

Clarity   Accuracy   Completeness   Organization   Coding   Retrieval   Legibility

If you wish a reply, give your name, company, mailing address, and date:

____

____

____

____

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

What is your occupation? ______________________

Number of latest Newsletter associated with this publication: ______________________

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

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 68Y
220 Las Colinas Boulevard
Irving, Texas 75039-5513

Fold here

Tape Please do not staple Tape

Program Comment Form IBM Personal Computer

3270-PC High Level Language Application Program Interface (1753180) SH20-6544-0

Service Registration Number: ______________________________
(Required information—located on the first screen of program)

Please use this form only to identify errors or to suggest changes to programs. Defects in this program may be corrected or its facilities may be enhanced by your comments. Provide specific information whenever possible.

Comments:

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.

Do you want a reply? ☐ Yes ☐ No. If you do, give us your name and address:

Name ______________________________

Company ______________________ Customer No. __________

Address ______________________________

City ________________ State ________ Zip Code ________

IBM employees only: Branch office no. ________ City and State ____________

Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

FOLD ON LINES, SEAL AND MAIL

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

IPS Product Support Center
IBM Corporation
P. O. Box 152560
Irving, TX 75015-2560

Fold here

Tape Please do not staple Tape

SERVICE REGISTRATION CARD

Please fill in and return to IBM in order to register for service.

3270-PC HIGH LEVEL LANGUAGE APPLICATION
PROGRAM INTERFACE (1753180)

Name ______________________________

Company ______________________________

Title ______________ Dept. or P.O. Box ______________

Street ______________________________

City ______________ State __________ Zip __________

Telephone Number ______________________________

Date Product Received: Mo ________ Day ________ Year ____________

Where Product Was Acquired: ______________________________

Alternate Customer Service Contact ______________________________

Alternate's Telephone Number ______________________________

Service Registration Number: ______________________________
(Required information—located on the first screen of program)

ATTENTION

SEND THIS CARD IN TODAY

FOR

PROPER SERVICE AND COVERAGE

SH20-6544-0

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

IPS Product Support Center
IBM Corporation
P. O. Box 152560
Irving, TX 75015-2560

Fold here

Tape Please do not staple Tape

ALL OTHER PROGRAMS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU (AND NOT IBM OR AN IBM AUTHORIZED REPRESENTATIVE) ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IBM does not warrant that the functions contained in any program will meet your requirements or that the operation of the program will be uninterrupted or error free or that program defects will be corrected.

THE FOREGOING WARRANTIES ARE IN LIEU OF ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

SOME STATES DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

LIMITATION OF REMEDIES

IBM's entire liability and your exclusive remedy shall be as follows:

1. With respect to defective media during the warranty period:

   a. IBM will replace media not meeting IBM's "Limited Warranty" if returned to IBM or an IBM authorized representative with a copy of your receipt.

   b. In the alternative, if IBM or such IBM authorized representative is unable to deliver replacement media free of defects in materials and workmanship, you may terminate this Agreement by returning the program and your money will be refunded.

2. With respect to warranted programs, in all situations involving performance or nonperformance during the warranty period, your remedy is (a) the correction or bypass by IBM of program defects or (b) if, after repeated efforts, IBM is unable to make the program operate as warranted, you shall be entitled to a refund of the money paid or to recover actual damages to the limits set forth below.

   For any other claim concerning performance or nonperformance by IBM pursuant to, or in any other way related to, the warranted programs under this Agreement, you shall be entitled to recover actual damages to the limits set forth below.

   IN NO EVENT WILL IBM BE LIABLE TO YOU FOR ANY LOST PROFITS, LOST SAVINGS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF OR INABILITY TO USE ANY PROGRAM EVEN IF IBM OR AN IBM AUTHORIZED REPRESENTATIVE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

   SOME STATES DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.

IBM's liability to you for actual damages for any cause whatsoever, and regardless of the form of action, shall be limited to the greater of $5,000 or the money paid for the program that caused the damages or that is the subject matter of, or is directly related to, the cause of action.

SERVICE

Service from IBM, if any, will be described in program specifications or in the statement of service, supplied with the program, if there are no program specifications.

IBM may also offer separate services under separate agreement for a fee.

GENERAL

Any attempt to sublicense, rent or lease, or, except as expressly provided for in this Agreement, to transfer any of the rights, duties or obligations hereunder is void.

This Agreement will be construed under the Uniform Commercial Code of the State of New York.

International Business Machines Corporation

P.O. Box 152750
Irving, TX 75015-2750

SH20-6544-0

Printed in USA