

Within the PICK community there are many host applications that have been developed to run on text based terminals. These are not PC-based applications, but are designed, built and running on tens of thousands of hosts everywhere.

HOSTACCESS is the leading terminal emulator for PICK systems, covering more emulations and networks than any other terminal emulator. With HOSTACCESS, you can keep your reliable, efficient code, whilst your user community moves onto PC systems.

More than this, HOSTACCESS provides a wide range of features to allow you to develop a full Graphical User Interface (GUI) to your PICK host applications. This GUI-izing process is unique to HOSTACCESS.

Click here [\[icon\]](#) for information on installing the host programs.

If you are new to using PICK with HOSTACCESS, click here [\[icon\]](#) for some advice on using PICK to create a GUI Interface to your legacy applications.

The following topics are covered:

- [\[icon\]](#) File transfer.
- [\[icon\]](#) Applications Programs.
- [\[icon\]](#) Additional TCL Utilities.
- [\[icon\]](#) Programmer 's TOOLKIT.
- [\[icon\]](#) GUI TOOLKIT.
- [\[icon\]](#) HOSTACCESS Control Files.

The following topics describe how to install HOSTACCESS 's host utilities on PICK systems.

Once you have installed all of the host AiF and file transfer routines, you can type **TERMITE.DEMO** at TCL to review a demonstration of some of the facilities available.

The host programs and subroutines available are documented in the main body of this guide.

HOSTACCESS 's host utilities are available for and supported on all PICK systems, including UniData, UniVerse and all other Unix/VMS PICK implementations.




Uploading the PICK Programs.

Updating PICK Accounts.

Porting between different PICK Systems.

You should install the PICK programs in an Account called HOST.X (X is the version number). You should use this Account as the master HOSTACCESS Utilities Account and you should back it up on a regular basis. Other Accounts on your PICK system that require the HOSTACCESS utilities may be updated.

Click here  for details on updating PICK accounts.

The PICK programs are provided on a separate disk - labeled 'HOST INSTALLATION DISK 1 '. There may be more than one Host Programs Disk and these will be numbered as appropriate.

The elapsed time for a complete upload of all of the PICK host routines will vary depending upon the speed of the host system, the PC and the communications link. Experience has shown that this time will range from between 1 minute on a powerful host to over 45 minutes on a slow host.

**Reminder:** Prime INFORMATION and some UniVerse (Prime/Ideal flavors) users should read VOC for MD wherever MD is mentioned in our documentation.



Prerequisites.


Uploading procedure.

Uploading Unidata Programs on VMS or UNIX.

It is assumed that you are running the correct version of HOSTACCESS on your PC and have established a connection with the host PICK system. You must be using a PC with access to at least 1Mb of free hard disk space.

Before starting, please ensure that the emulation you are using in HOSTACCESS corresponds to the terminal type set on the host machine. If you are using a serial connection, you should also ensure that flow control is set correctly both within HOSTACCESS and on the host.

You are recommended to run this procedure from a PC directly connected to the Host PICK system, i.e. not attached by modems or a wide area network.

The following instructions are for all PICK systems *except* UniData. Click here  for instructions for UniData.

1. On the host PICK system, create an Account called HOSTACCESS.6.2 with at least SYS2 privileges. LOGTO this Account.

**UniVerse users:** ensure that the PTERM -CASE NOINVERT verb has been executed in this account so that no case conversions take place.

**General Automation (C-ITOH) users:** ensure that the "CONTROL-CHARS ON" verb has been executed in this account so that control characters will be accepted on input.

2. Create a file called PIX.PROGS.F as follows :

**CREATE-FILE PIX.PROGS.F 1,1 97,1**

**Prime INFORMATION** and **UniVerse** users should create this as a **type 1** file as follows:

**CREATE.FILE PIX.PROGS.F 1**

3. Ensure that this file is suitable for use as a PICK/BASIC programs file. In other words, modify the MD entry to reflect this as dictated by your PICK system.

For example, on C-ITOH PICK use **:SWC MD PIX.PROGS.F**.

On generic PICK carefully **ED**it the MD entry and change attribute 1 from 'D ' to 'DC '

McDonnell Douglas users need make no changes. Prime INFORMATION and UniVerse users will not need to make any such changes, as this file is already a type 1 file.

4. Insert the disk labeled 'HOST INSTALLATION DISK 1 ' in the PC 's floppy disk drive
5. Invoke HOSTACCESS 's File Transfer facility by selecting the appropriate button. This displays **install.key** - the name of the Keyfile used to control the installation. Select the key protocol
6. Press <ENTER> to proceed with the upload (or press <ESCAPE> to exit). After some small PICK programs are transferred and compiled, you see a menu of PICK machine types.
7. Select the entry that corresponds to your PICK machine and press <ENTER>. You will then be prompted for the drive letter from which the host programs will be uploaded.
8. Enter the letter of the drive containing the host installation disk and then press <ENTER>. The next message asks you to load HOST INSTALLATION DISK 1. If you have not already done this, do so now and press <ENTER>.

The program PIX.DOS.PICK will now be uploaded and compiled. The PIX.INSTALL.HOST program is then uploaded and compiled. When this is complete you will be presented with a controlling menu for the host programs upload.

9. Select the installation option and then press <ENTER>.

The upload procedure is completely automatic from this point onwards.

After following the instructions in the previous section, the following automatic uploading process occurs. The PICK programs are uploaded in the host PICK file PIX.PROGS.F and these programs are also compiled and catalogued. The appropriate MD (VOC) entries are created in the HOSTACCESS.6.0 account to run the appropriate TCL commands.

Once the host programs have been installed you may select the "**Update another Account**" option. This moves relevant TCL pointers, File pointers and Catalogued program entries into the MD (or VOC) of the target account. The target account must be a valid "LOGTO" account name.

You may exit this program by selecting the "**Exit**" option. You may re-run it any time by entering **PIX.INSTALL.HOST** from the TCL (PERFORM) command line in the "HOSTACCESS.6.2" account.

If for some reason the upload should fail, or some items not be transferred, you may re-start the installation procedure by running PIX.INSTALL.HOST and choosing the appropriate option from the menu displayed. However, if you require programs already loaded to be updated, existing items in the file PIX.PROGS.F must be deleted before running PIX.INSTALL.HOST.

**Note:** There is no need to repeat the file upload procedure.

To install HOSTACCESS 's host programs on UniData on either VMS or UNIX, follow the steps below.

1. On the host, create a UniData account into which you will install HOSTACCESS 's host programs. Name this account as appropriate. e.g. **hostacc6**. We recommend that you follow Unidata 's instructions for this. For UNIX, these may be similar to those below:

```
cd /usr
mkdir hostacc6
cd hostacc6
newacct
```

(and answer the Unidata prompts as appropriate)

**udt**

(to invoke Unidata)

Make sure that the environment variables for Unidata have been set.

2. Login to this account using HOSTACCESS and create the programs file needed for the host programs with the following commands (please note the underscores in the filename, and substitute /usr/hostacc6 with the path appropriate to your system):

Execute the verb BASICTYPE 'P ' and modify the Line Editor verb as follows:

```
AE VOC ED
G2
R/ED/AE
FI
```

**CREATE.FILE DIR PIX\_PROGS\_F**

**SETFILE /usr/thostacc6/PIX\_PROGS\_F PIX.PROGS.F  
OVERWRITING**

**Note:** VMS users *only* should use SETFILE as follows:

**SETFILE UDT\_HOME: [HOSTACC6.PIX.PROGS.F] PIX.PROGS.F  
OVERWRITING**

If control characters are turned off, remember to activate the CONTROLCHARS ON verb.

3. Start HOSTACCESS 's File Transfer facility. Change the default keyfile name from INSTALL.KEY to INSTALL.UNI and press <ENTER>. The host installation bootstrap programs will be uploaded.
4. Choose UNIDATA as the host machine. You will be prompted for the DOS drive letter for the host programs disk. Enter **A** or **B** as appropriate.

You will then be prompted for the full UNIX or VMS path to this account. Enter the appropriate path, in the correct case, such as **/usr/hostacc6** (on UNIX), or **UDT\_HOME : [HOSTACC6]** (on VMS). The PIX\_INSTALL\_HOST routine will be uploaded and compiled.

When this is complete you will be presented with a controlling menu for the host programs upload. Once you have selected the appropriate option, the upload procedure is completely automatic.



Hints and Guidelines

Asynchronous users should ensure that flow control is on at both ends, i.e. within HOSTACCESS (both in and out) and at the host.

Host **Prime** users should note the following:

INFORMATION strips the 8th bit on any I/O "data". Most users will set HOSTACCESS 's Port configuration for Serial sessions as below :

Data bits7  
ParityMark  
Stop bits1

Most INFORMATION applications will need the Echo set to Delay and will already have done this. If Echo is not set to Delay the Prime host will intersperse the echo of input data with output data from the host. You may set this parameter as below :

#### **PTERM -ECHO DELAY**

You may need to assign "reverse" flow-control for the host port. This controls input (to the host) flow control. PRIMOS rev 19.4 and later should support the SET\_ASYNC command

#### **SET\_ASYNC -LINE nn -REV\_XOFF**

where **nn** is the Port Number.

If you are unsure about using this command, please consult your Prime systems administrator.

Please also remember to ensure that the "Kill" character is suitably configured to, say, Control X. If the kill character is left as the Prime default, a question mark, one of the initial file transfers will fail.



The master HOSTACCESS account installed and loaded is ready for use with all the host utilities uploaded. To make these host utilities available in other PICK Accounts simply run the program PIX.INSTALL.HOST and choose the **Update Account** option. This prompts you for the name of the PICK account to be updated and transfers the appropriate MD (VOC) entries for the host utilities and files. The target account must be a valid LOGTO account.

Prime INFORMATION, UniVerse, and Unidata users should enter the appropriate path to the target account, according to each system 's operating conventions.

If MD entries for host utilities commands (such as EASY.ACCESS) already exist in the target account 's MD, the original entries are duplicated into items suffixed ".ORIGINAL" in the MD file.

All the other MD entries required are prefixed by PIX. for subroutines and are written directly into the target MD file.

Entries for files are created as Q-pointers, prefixed with PIX. and suffixed with .F.

If you need to use the HOSTACCESS host utilities on another PICK system you may use PICK 's own backup and restore facilities as long as the other PICK system is the *same* type as the original one.

If you need these utilities on another PICK system with a *different* flavour of PICK, you *must* follow the install procedure described in the section headed PICK Installation, above. The reason for this is that the installation process decides which of the various programs to upload depending upon the host system 's PICK flavour.

Today, users are expecting increasing sophistication from their applications, in terms of a true Graphical User Interface, with dialog boxes, menus and windows.

We have developed a comprehensive set of facilities to allow you to enhance and develop host applications using the full GUI effects normally only available to PC application developers. Using these facilities with HOSTACCESS, you have access to a wide range of GUI effects. We call these facilities the [GUI TOOLKIT](#).

Whether you want to simply put a red box on the screen or provide a full Graphical User Interface for the end users of your application, HOSTACCESS provides all of the necessary links within the [Programmer's TOOLKIT](#) and GUI TOOLKIT.

You can start enhancing applications immediately, dramatically improving the look and feel of host applications. A major advantage is the 'phased' enhancement approach of HOSTACCESS. You can add any or all of the following to your applications:

- DDE links.
- Automatic data transfer.
- Color, graphics and pictures.
- Windows.
- Toolbars and toolboxes.
- Menus and commands
- List boxes and combo boxes.
- Radio buttons and check boxes.
- Sculpted "3-D" dialog boxes.
- Edit boxes

 Available subroutines.

If you are new to using PICK with HOSTACCESS, click here  for some advice on using PICK to create a GUI Interface to your legacy applications.

There are a number of approaches to creating a GUI interface to your legacy applications, the approach you take should be determined by what you are trying to achieve and what application you have to start off with.

You may wish to give your menus a GUI look and feel and let your application screens run as normal in shades of gray with sculpted boxes surrounding the input fields or you may wish to add more options like Windows Edit Controls and buttons.

The following topics take you step by step through the necessary tasks to create a GUI interface using simple examples.

1. [Windows controls.](#)
2. [Events.](#)
3. [Colours.](#)
4. [Capturing events.](#)
5. [Reading controls.](#)
6. [Unloading controls.](#)

A control is a Windows object that can be placed on to the terminal screen at a given x & y coordinate. It will be sized to terminal character sizes, i.e. number of characters long by number of rows deep. This control will be placed onto the screen in the given coordinates using the default configuration parameters, that is the colour will be white text on black background.

For example we could load a text button on the screen at:

Row 23, Col. 10, 9 Chars wide by 2 rows deep:

```
001 CALL PIX.GUI.LOAD.TEXTBUTTON("", "BUTTON1,TEXTBUTTON,23,10,9,2,&Help", "")
```

From the example above you will notice that the string passed to the subroutine include the coordinates and dimensions of the button but it also include a unique name (CONTROL.ID) and a label. The label also has an '&' in front of the letter 'H' . This will highlight the 'H' of Help underlined. This will become a hot key which means that keying Alt H is the same as clicking on the button.

If you click this button, nothing will happen. This is because for a control to return actions back to the host, events must be assigned to them first.



Events

An event is an action that you must assign to a control in order for an action to be returned to the host. For example:

```
001 CALL PIX.GUI.LOAD.TEXTBUTTON("", "BUTTON1,TEXTBUTTON,23,10,9,2,&Help", "")
```

loads a text button control. You will probably want most buttons to have an event of clicked assigned against them. This means that if the user clicks on the button, the button will send a response to the host to say that it has been clicked.

You can assign multiple events to controls, for example, you might want an edit control to have enter and tab events assigned against it so that if the user tabs out of the control or hits enter a response is sent to the host.

On the other hand you may not wish to assign any events to a control, for example if a multi line scrolling edit box is on the screen for users to enter comments like a notepad, you (the Host) do not need to be informed if they tab out or hit enter. You will only need to know if they click on the File button when you can then read the contents of the control.

```
001 CALL PIX.GUI.LOAD.TEXTBUTTON("", "BUTTON1,TEXTBUTTON,23,18,9,2,&Help", "")
002 CALL PIX.GUI.LOAD.EDIT("", "EDIT1,EDIT,18,10,20,1", "")
003 CALL PIX.GUI.CONTROL.EVENT("", "BUTTON1,EVENT,ON,CLICKED", "")
004 CALL PIX.GUI.CONTROL.EVENT("", "EDIT1,EVENT,ON,ENTER,TAB", "")
```

If you run the example above you will probably not see the edit control as it is white on black and the standard default screen in a black background. You will need to set the colours of the screen and the controls to see it and get the best effect.



Colours.

There are two types of colours that you should be aware of. First, there is the background colour of the session. i.e. the normal test colour. This can be set via the Attribute mapping from the HOSTACCESS Configure Menu but for application control you will want to set it from your application. Confusion in colours can arise from the old style ASCII colours of DOS and the new Windows' Palette colours. You set the background colours using the old ASCII style. However controls use the Windows Palette e.g. the colour GREY in ASCII is the same as LIGHTGREY for the Windows Palette.

```
001 CALL PIX.SET.COLOUR("BLACK,GREY","D")
002 CRT @(-1):
003 CALL PIX.GUI.LOAD.COLOUR("", "ALLCONTROLS, COLOUR, BLACK, LIGHTGREY", "")
004 CALL PIX.GUI.LOAD.TEXTBUTTON("", "BUTTON1, TEXTBUTTON, 23, 18, 9, 2, &Help", "")
005 CALL PIX.GUI.LOAD.EDIT("", "EDIT1, EDIT, 18, 10, 20, 1", "")
006 CALL PIX.GUI.LOAD.COLOUR("", "EDIT1, COLOUR, BLACK, WHITE", "")
007 CALL PIX.GUI.CONTROL.EVENT("", "BUTTON1, EVENT, ON, CLICKED", "")
008 CALL PIX.GUI.CONTROL.EVENT("", "EDIT1, EVENT, ON, ENTER, TAB", "")
```

This example sets the default background colours to black on grey and clears the screen, sets the default colour of all controls that follow to black on light grey, until the default is changed, loads the text button and the edit control and changes the colour of the edit control to black on white. Note the default colour is still black on light grey.

Run the above program and click on the button or tab/enter from the edit control, you will see what appears to be garbage on the screen. This 'garbage' is events which are being sent back to the host.



Capturing events.

You need to capture events returned to the host so that your program can perform actions based on what the user is doing. The events returned to the host are sent back in two line inputs. The first input is the event identifier and the second event is the action.

The Basic code below will return to your host the three variables of

**CONTROL.ID** Set to the unique Control ID of the control that was actioned.

**EVENT** Set to the type of action the user did (if event reporting was enabled for that control).

**PARAMS** Not discussed here.

```
CALL PIX.GUI.GET.EVENT("", CONTROL.ID, EVENT, PARAMS, "")
BEGIN CASE
CASE CONTROL.ID = "BUTTON1"
PRINT CONTROL.ID : "Was " : EVENT
CASE CONTROL.ID = "EDIT1"
PRINT CONTROL.ID : "Was " : EVENT
CASE 1
NULL
END CASE
```

This logic is fine if you want to perform one action in a program and exit, but usually you will want to loop around waiting for the user to click on the File (OK) or Exit (Cancel) buttons. This leads us into event driven programming as in the following example:

```
CALL PIX.SET.COLOUR("BLACK,GREY","D")
CRT @(-1):
CALL PIX.GUI.LOAD.COLOUR("", "ALLCONTROLS, COLOUR, BLACK, LIGHTGREY", "")
CALL PIX.GUI.LOAD.TEXTBUTTON("", "BUTTON1, TEXTBUTTON, 23, 18, 9, 2, &Help", "")
CALL PIX.GUI.LOAD.TEXTBUTTON("", "EXIT, TEXTBUTTON, 23, 22, 9, 2, &Cancel", "")
CALL PIX.GUI.LOAD.EDIT(EDIT1, EDIT, 18, 10, 20, 1, "")
CALL PIX.GUI.LOAD.COLOUR("", "EDIT1, COLOUR, BLACK, WHITE", "")
CALL PIX.GUI.CONTROL.EVENT("", "BUTTON1, EVENT, ON, CLICKED", "")
CALL PIX.GUI.CONTROL.EVENT("", "EXIT, EVENT, ON, CLICKED", "")
CALL PIX.GUI.CONTROL.EVENT("", "EDIT1, EVENT, ON, ENTER, TAB", "")
LOOP
CALL PIX.GUI.GET.EVENT("", CONTROL.ID, EVENT, PARAMS, "")
UNTIL CONTROL.ID = "EXIT" DO
BEGIN CASE
CASE CONTROL.ID = "BUTTON1"
PRINT CONTROL.ID : "Was " : EVENT
CASE CONTROL.ID = "EDIT1"
PRINT CONTROL.ID : "Was " : EVENT
CASE 1
NULL
END CASE
REPEAT
```

The above example shows a method for validating events as they occur. If you want to do some processing before you enter a data field then you will need to use FOCUS Event. Please note that FOCUS means 'left' focus, not 'gained'. This means if you have one control with FOCUS then every control must have the FOCUS control, even those that do not require an event.



Reading controls.



The next step is retrieving the data that the user has typed in. For example, we want to read the edit controls, radio buttons, list boxes, check boxes and grids. You can even load your own VBX controls and read and write to them also.

**CALL PIX.GUI.CONTROL.READ.VALUE("", "EDIT1,EDIT", VALUE, "")**

The value of the edit control EDIT1 has been read and the contents returned into the variable VALUE. The only thing you must be careful of is the second parameter passed to this subroutine, eg. 'EDIT1,EDIT'. Because of the nature of the code behind the PIX.GUI.READ.VALUE program, you need to pass the name of the control and the type of control, ie. EDIT1 is the name of the control and it is an EDIT type control. You would never want to read the contents of a button as these are actions only.



Unloading controls.

The next step is removing controls from the screen (unloading controls).

That could not be simpler. Remember, if a control is LOADED then to remove a control you must UNLOAD it.

```
024 CALL PIX.GUI.CONTROL.UNLOAD ("", "BUTTON1", "")
025 CALL PIX.GUI.CONTROL.UNLOAD ("", "EXIT", "")
026 CALL PIX.GUI.CONTROL.UNLOAD ("", "EDIT1", "")
```

It is simpler to group the controls together before deleting them. This can be done in two ways:

You can define a Base Control Group before you start loading your control.

```
000 CALL PIX.GUI.LOAD.DEFGROUP ("", "TESTAPP, DEFGROUP", "")
```

Or after the controls are loaded, create a group of controls and delete the GROUP Control.

```
024 CALL PIX.GUI.LOAD.GROUP ("", "TESTAPP, GROUP, BUTTON1, EXIT, EDIT1", "")
025 CALL PIX.GUI.CONTROL.UNLOAD ("", "TESTAPP", "")
```

**Related topics:**



- Secondary Windows.
- PUSH Slots
- POP Slots.
- Controls.
- Pushbuttons.
- Images.

The following topics describe the PICK File Transfer commands that are called from the TCL (or PERFORM) command line.








All PICK file transfer commands:

- Can be used from the TCL command line.
- Take command line options and prompt for missing parameters if appropriate.
- Can be used by applications software as [PICK/BASIC subroutine calls](#).

All the applications and utility programs provided are invoked from commands which are processed by one program, **PIX.DRIVER**. This evaluates what the user typed from the command line, then converts the command line into the required subroutine parameters, then calls the appropriate subroutine. The Host Installation procedure loads the MD (or VOC) file with the appropriate commands for these programs and utilities.

The File Transfer verbs are documented in a standard format for ease of use and quick reference.

The following areas are covered:

-  File transfer conventions.
-  File transfer overview.
-  Transferring from DOS to PICK.
-  Transferring from PICK to DOS.
-  Dumping PICK Items to DOS.
-  Uploading PICK files from DOS.
-  Inter-Session File Transfer.

Mandatory options for a File Transfer command are shown in upper-case text after the command and in the position in which they should be specified.

Optional parameters for a File Services verb are shown within { } braces and are usually in lower-case text.

Options are specified at the end of the command line, after a leading open round bracket. Options are always single uppercase letters. Several options may be specified on the same line together - spaces or commas may be used to separate each option, if required.

Six PICK commands are provided to transfer data between PICK and DOS.

<b>Command</b>	<b>Function</b>
<u><a href="#">DOS.PICK</a></u>	Transfers a DOS file to a PICK item or to the PICK Spooler.
<u><a href="#">PICK.DOS</a></u>	Transfers a PICK item to a DOS file.
<u><a href="#">DOS.PICK.ALL</a></u>	Transfers a DOS file into multiple items in a PICK file.
<u><a href="#">PICK.DOS.ALL</a></u>	Transfers multiple PICK items into a single DOS file.
<u><a href="#">TDUMP</a></u>	Transfers multiple PICK items to multiple DOS files.
<u><a href="#">TLOAD</a></u>	Transfers multiple DOS files to multiple PICK items.

These core file transfer facilities have been integrated into several utilities supplied in addition to the PICK.DOS and DOS.PICK commands. The DOS.PICK.ALL, PICK.DOS.ALL, TDUMP and TLOAD commands enable transfers of multiple PICK items to and from DOS. They provide a very useful method of moving source code (programs) and data between different PICK systems, without the user ever having to worry about tape incompatibility problems.

During the transfer of data, a window is displayed on the user screen showing the current state of the transfer and information such as the size of the file, the number of bytes transferred and the estimated time for the transfer.

At the end of the transfer, the user is requested to press a key before the window is closed to allow time to read the final display.



Command options



Integrating file transfer with PICK applications.

By default the data is assumed to be textual and DOS records are mapped to PICK attributes and vice versa. To override this mapping, use the **Binary** option which performs no translations on the data.

If data is being transferred between the PICK host and a directly connected PC, use the **L** (Local) option to speed up the transfer.

A number of optional parameters may be specified to control all aspects of the file transfer, such as suppressing the progress window, removing the need to acknowledge completion, etc.

The following commands are all verbs interpreted by the PIX.DRIVER routine. They are processed by the PICK/BASIC subroutines shown, which you can call from any PICK program.

<b>Command</b>	<b>PICK BASIC Subroutine</b>
<u>DOS.PICK</u>	<u>PIX.CALL.DOS.PICK</u> <u>PIX.DOS.PICK</u>
<u>PICK.DOS</u>	<u>PIX.CALL.PICK.DOS</u> <u>PIX.PICK.DOS</u>
<u>DOS.PICK.ALL</u>	<u>PIX.DOS.PICK.ALL</u>
<u>PICK.DOS.ALL</u>	<u>PIX.PICK.DOS.ALL</u>
<u>TDUMP</u>	<u>PIX.TDUMP</u>
<u>TLOAD</u>	<u>PIX.TLOAD</u>

Use the **DOS.PICK** command to transfer DOS files up to the PICK host, using HOSTACCESS 's own protocol which provides fast error checking and correcting transfers with built in data compression. You can also use DOS.PICK to quickly transfer DOS files into the PICK spooler.

You can use DOS.PICK as a TCL command or a PICK/BASIC subroutine call to allow seamless integration with Host Applications.

Each DOS file is converted directly into a PICK item. By default, HOSTACCESS converts Carriage Return, Line Feeds (CRLF) into Attribute Marks. To manipulate data being transferred, use the PICK BASIC subroutine [PIX.DOS.PICK](#)

**Note:** DOS.PICK automatically deals with the systems that can and cannot handle unlimited item sizes. Those systems that do not, will find that large DOS records are split automatically and cleanly during the upload.

Invoke DOS.PICK from the TCL command line as follows:

**DOS.PICK {DICT} {filename itemname} FROM dosname {(options)}**

Where:

<b>DICT</b>	Use this literal if itemname is to be created in the dictionary of the file filename.
<b>filename</b>	Is the PICK filename where itemname is to be created. Should be null if 'S' or 'T' options used.
<b>itemname</b>	Is the name of the item to be created in the specified PICK file. Should be null if 'S' option specified.
<b>dosname</b>	Is the DOS file name to be uploaded to PICK, optionally containing a drive letter and DOS path. By default, <b>dosname</b> is uploaded from the current HOSTACCESS runtime directory.

To determine the status of the file transfer, the status record (as described for PIX.PICK.DOS and PIX.DOS.PICK) is written to the file [PIX.CONTROL.F](#) with a key of **nn.FT.LOG**, where **nn** is the port number.

Click here  for options available with the DOS.PICK command.

Click here  for examples.



Anything following the open left bracket in the DOS.PICK command is treated as an option. Multiple options can be specified with or without a delimiter. These options are:

- O** Overwrite the PICK item if it already exists.
- S** Sends the DOS file directly to the PICK spooler.
- F** Used with (and after) the S option, carries out a form feed.
- T** Upload data into the PIX.TRANSFER.F file in HOSTACCESS 's own transmit format. The record is created in this file using the name of itemname. You can use this to hold documents and graphics on the PICK host. To send this data back to DOS, use the DOS.PICK command (with T option again). File transfers are faster, there is no limit to the size of DOS files that can be transferred between PICK and DOS, and there is no problem storing these files if your PICK system does not support the character value 255.
- 2** Splits the data.
- H** Initiated by host. Transfer status window will not wait for user acknowledgment before closing and no status message is printed to the screen. This option is useful for automated multiple transfers carried out under program control. The file transfer activity/status window is still shown before closing.
- Z** Suppress all file transfer output including the status/activity window. It is generally used from applications so that the file transfer takes place completely transparently and without the user needing to press a key to acknowledge completion.
- B** Store a DOS binary file containing control characters. HOSTACCESS will not convert DOS CRLF 's to attribute marks (CHAR(254)). Characters with ASCII values of 255 (FF) are converted to a '~' (tilde), unless the PICK system, such as McDonnell Douglas 7.0, supports CHAR(255) in items. Use the HOSTACCESS command ENVIRONMENT to determine what action DOS.PICK will take on receiving a character of decimal value 255.
- L** Local: use on locally connected PCs to speed up the transfer by performing simpler error checking. If HOSTACCESS detects that you are running over one of its supported resilient networks, it uses this option by default.
- R** Remote - use this to override the default use of the L option. Use this option if you know that your network may be non-resilient at any point (e.g. if you are using non-error checking modems). Use this option if you are experiencing file transfer errors when communicating over a network.
- X** To write or spool in HEX. Each byte in the DOS file will be converted to its equivalent hexadecimal representation in two characters.
- N** Nationality - enables correct file transfer of data containing other language character sets over all types of link (i.e. 7 or 8 bit).



- Examples.
- Maximum record sizes
- Abnormal character conversions.
- Storing CHAR (255)<SEGMENT DELIMITER> on PICK Hosts.

The following examples highlight the power of this file transfer from TCL. To copy a DOS file C:\MAILSHOT.TXT into a new item MAIL1 in the PICK file DOCS, use the following command:

**DOS.PICK DOCS MAIL1 FROM C:\WP\MAILSHOT.DOC**

To copy a DOS file from the PCs floppy disk into an existing item PROG1 (i.e. to overwrite it) in the PICK file BP use the following command:

**DOS.PICK BP PROG1 FROM A:\SOURCE\PROG1.PIX (O**

To spool a DOS file C:\AUTOEXEC.BAT to the Host PICK system's spooler with NO user output use the following command :

**DOS.PICK FROM C:\DATA.DMP (SZ**

To store a WordPerfect document of unlimited size on the host in transmit format that can be sent back to other DOS WordPerfect users later (creating the item **MASTER.LETTER** in the file [PIX.TRANSFER.F](#)), enter:

**DOS.PICK MASTER.LETTER FROM C:\WP\LETTER.DOC (T**

Some PICK systems have a maximum record/workspace size limit of around 32000 bytes. DOS files can be considerably larger than this. If HOSTACCESS detects that it is transferring a large DOS file into a PICK host that does not support large items, it will split the original file into records smaller than the maximum record size limit.

The split items will be created in the PIX.TRANSFER.F file keyed in numeric splitting order and uniquely identified by the user's port number. For example, if a DOS file is split 3 times during transfer to the PICK host by a user on Port 24, these records are stored with the following keys :

FT.SPLIT.24.1  
FT.SPLIT.24.2  
FT.SPLIT.24.3

The maximum record size limit for HOSTACCESS's DOS to PICK file transfer is a parameter held as attribute 4 in the ENVIRONMENT item in the [PIX.CONTROL.F](#) file.

When HOSTACCESS's host utilities are installed on to your PICK system, the maximum item size is determined and written into this record. On Prime, UniVerse, UniData and McDonnell Douglas 7.0+ systems this parameter is set to 5 megabytes as there is no limit on the maximum record size on these systems. You may reset this parameter as required, but you should be aware that this parameter is common to all users making use of HOSTACCESS's file transfer on the host PICK system.

UniVerse and Prime INFORMATION PICK systems do their own character conversions when writing records into certain file types: therefore, file transfers of DOS files into Type 1 files may create unpredictable results. INFORMATION treats Type 1 files as sub-UFD (SAM or DAM files) and UniVerse treats them as a UNIX directory. If DOS files containing binary data are written into these files, both systems make character conversions such as changing all CHAR(10)s (line feeds) into CHAR(254)s (field/attribute marks). You should only transfer DOS ASCII text files into Type 1 files.

HOSTACCESS converts CHAR(255) characters (reserved on many PICK hosts) into tildes during file transfer. Attribute 5 of the PIX.CONTROL.F ENVIRONMENT record contains a true/false flag to show if the host system can support this character. You should set this attribute to 1 **only** if you are sure that your PICK host can store this character and you need to transfer DOS files containing it.

During installation of the Host Utilities this attribute is automatically set to 1 if the host system is: Prime INFORMATION, UniVerse, UniData or McDonnell Douglas REALITY release 7.0 or later

If your host system is not one of these and you still need to store DOS files containing CHAR (255) on the PICK system, you should consider holding these in transmit format (via the **T** option). These files will not be easily accessible by PICK users nor applications (as the data will be in HOSTACCESS 's own internal file transfer format) but may be sent back to DOS as and when required.


The [DOS.PICK.ALL](#) command uploads multiple PICK items from a DOS file. This powerful command can upload records stored in DOS in a number of formats, including that created by the [PICK.DOS.ALL](#) routine. You can call this routine as a subroutine, passing user-specific options. If you are uploading a file that was previously created with the PICK.DOS.ALL routine, the same keys from the original file will be used automatically to write these items.

Call DOS.PICK.ALL from the TCL command line, as follows:

**DOS.PICK.ALL {DICT} PICKfilename {itemname1... itemnamen} FROM dosname {(options)}**

Where:

<b>DICT</b>	Use this literal <b>if itemname1 ... itemnamen</b> are to be created in the dictionary of the file <b>PICKfilename</b> .
<b>PICKfilename</b>	Is the PICK filename where the items are to be created.
<b>itemname1 ... itemnamen</b>	The names of the items to be created in the specified PICK file. If no items are specified, all items will be uploaded from <b>dosname</b> .
<b>dosname</b>	Is the DOS file name to be uploaded to PICK, optionally containing a drive letter and DOS path. By default, <b>dosname</b> is uploaded from the current HOSTACCESS runtime directory.

Click here  for options available with DOS.PICK.ALL.

Click here  for information on using DOS.PICK.ALL.

Anything following the open left bracket in the DOS.PICK.ALL command is treated as an option. Multiple options can be specified with or without a delimiter. These options are:

- O** Overwrite the PICK item if it already exists.
- Z** Suppress all file transfer output including the status/activity window. It is generally used from applications so that the file transfer takes place completely transparently and without the user needing to press a key to acknowledge completion.
- B** Store a DOS binary file containing control characters. HOSTACCESS will not convert DOS CRLF 's to attribute marks (CHAR(254)). Characters with ASCII values of 255 (FF) are converted to a '~' (tilde), **unless** the PICK system, such as McDonnell Douglas 7.0, supports CHAR(255) in items. Use the HOSTACCESS command **ENVIRONMENT** to determine what action DOS.PICK will take on receiving a character of decimal value 255.
- L** **Local:** use on locally connected PCs to speed up the transfer by performing simpler error checking. If HOSTACCESS detects that you are running over one of its supported resilient networks, it uses this option by default.
- R** **Remote** - use this to override the default use of the L option. Use this option if you know that your network may be non-resilient at any point (e.g. if you are using non-error checking modems). Use this option if you are experiencing file transfer errors when communicating over a network.
- N** **Nationality** - enables correct file transfer of data containing other language character sets over all types of link (i.e. 7 or 8 bit).

Click here  for options available to override the default format of records uploaded.



By default, DOS.PICK.ALL assumes that you are uploading records in the format created by the PICK.DOS.ALL routine. Using any of the following options overrides this default and automatically generates sequential numeric keys. Also, using these options allows any comma-delimited DOS file to be uploaded to the PICK file as multiple items, converting each comma into an attribute mark.

- 1 Use <CRLF> as the inter-record delimiter and use commas as the inter-field delimiter.
- 9 Use first field in delimited file as the key to the PICK file, instead of using the automated key.
- 0 Strip any double quotes from the data.

The DOS.PICK.ALL upload will stop when all the specified items have been found, regardless of whether the entire DOS file has been uploaded.

To determine the status of the file transfer, the status record (as described for [PIX.PICK.DOS](#) and [PIX.DOS.PICK](#)) is written to the file [PIX.CONTROL.F](#) with a key of **nn.FT.LOG**, where **nn** is the port number.



Using DOS.PICK.ALL

To upload all PICK items into the PICK file STAFF from a single DOS file called STAFFILE.ALL (that was created using the PICK.DOS.ALL routine), and overwrite any existing records, use the following command:

**DOS.PICK.ALL STAFF FROM C:\FILES\STAFF\STAFFILE.ALL (O**

To upload all PICK items into the DEMO file from a single comma-delimited DOS file using sequential keys starting from 1, and remove any double quotes, use the following command:

**DOS.PICK.ALL DEMO FROM DEMO.TXT (10**

Any records found in the file with the same key as the automatically generated key will not be overwritten without the O option.

Use the [PICK.DOS](#) routine to transfer PICK items to DOS. It uses HOSTACCESS 's own protocol which provides fast, error checking and correcting transfers with built in data compression.

PICK.DOS is available as a TCL command or as a PICK/BASIC subroutine call to allow seamless integration with Host Applications.

PICK.DOS is designed to transfer single PICK items into a DOS file. HOSTACCESS also provides powerful facilities for passing data directly into most DOS packages. See [GET](#) and [PASS.TO](#) for more information.

If the DOS file exists it will be overwritten, otherwise it will be created. HOSTACCESS assumes the data is textual and attribute marks will be converted to Carriage Return/Line Feeds (CRLF), value marks and sub-value marks will be converted to spaces and a DOS end of file character will be written to the end of the DOS file. Options are provided to override this translation.

**Note:** PICK data will be downloaded in the PICK internal format.


PICK.DOS may be invoked from the TCL command line with a number of parameters and options as described below :

**PICK.DOS {DICT} {filename itemname} TO dosname {(options)}**

Where:

<b>DICT</b>	Use this literal if itemname is to be retrieved from the dictionary of filename.
<b>filename</b>	Is the PICK filename where itemname is to be retrieved from. Should be null if T option used.
<b>itemname</b>	Is the name of the item to be sent to DOS.
<b>dosname</b>	Is the DOS file name to be created or overwritten, optionally containing a drive letter and DOS path. If no path is specified, <b>dosname</b> is created in the current HOSTACCESS runtime directory.

To determine the status of the file transfer, the status record (as described for PIX.PICK.DOS and PIX.DOS.PICK) is written to the file [PIX.CONTROL.E](#) with a key of **nn.FT.LOG**, where **nn** is the port number.

Click here  for options available with the PICK.DOS command.

Anything following the open left bracket is treated as an option. Multiple options can be specified with or without a delimiter. These options have the following effect:

- A** Append to the existing DOS file. This is useful for building up one large DOS file from a number of PICK items. If this option is not specified, the DOS file will be overwritten, or created if it does not already exist.
- T** Data is downloaded from the [PIX.TRANSFER.F](#) file using the record **itemname**. This record is created by DOS.PICK with the T option and is a very powerful way of holding DOS files, such as documents, graphics, spreadsheets etc., on the PICK host. By holding the records in this format, file transfers are faster, there is no limit to the size of DOS files that can be transferred between PICK and DOS, and there is no problem of storing these files if your PICK system does not support the character value 255.
- 2** Splits the data.
- H** Initiated by host. Transfer status window will not wait for user acknowledgment before closing and no status message is printed to the screen. This is useful for automated multiple transfers carried out under program control. The file transfer status window is still shown so that the user is aware that something is happening.
- Z** Suppress all file transfer output including the status window. It is generally used from applications so that the file transfer takes place completely transparently and without the user needing to press a key to acknowledge completion.
- B** Binary file - override default. i.e. no conversion of Attribute Marks CHAR(254) to CRLF and no conversion of value and sub-value marks to spaces.
- L** **Local** option, for a locally connected PC - this will speed up the transfer by performing simpler error checking. If HOSTACCESS detects that you are running over one of its supported resilient networks, it will use the "L" option by default.
- R** **Remote** option - use this to tell HOSTACCESS not to automatically use the L option when running file transfer over non resilient networks. You should specify this option if you know that your network may be non-resilient at any point (e.g. if you are using non-error checking modems). Use this option if you are experiencing file transfer errors when communicating over a network.
- N** **Nationality** - enables correct file transfer of data containing other language character sets over all types of link (i.e. 7 or 8 bit).

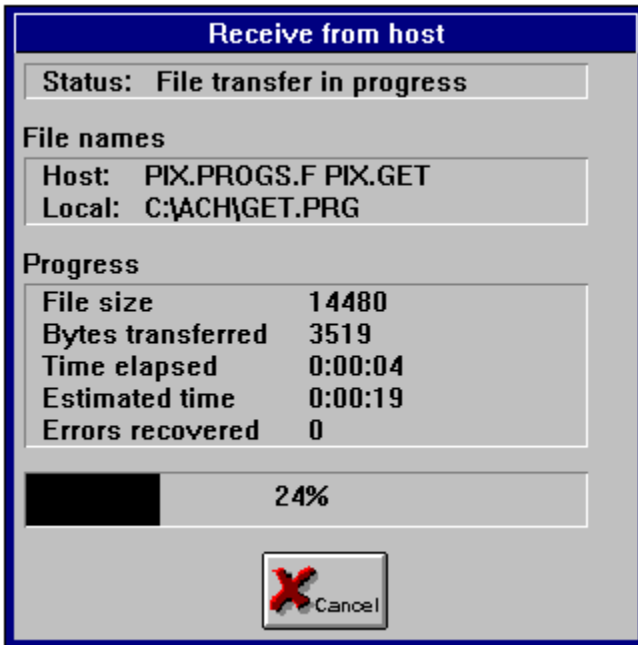


Diagram of File transfer Status Windows.

Examples.

Downloading multiple PICK items.

Below is a typical example of the status window displayed during file transfer.



All of HOSTACCESS 's file transfer capabilities are very easy to use. For direct integration of PICK data with DOS packages, refer to [GET](#) and [PASS.TO](#).

The following examples highlight the power of file transfer from TCL:

To transfer the host PICK item PROG1 from the file BP to the DOS **C:\source\prog1.prg**, enter:

**PICK.DOS BP PROG1 TO C:\source\prog1.prg**

To transfer a WordPerfect document of unlimited size, stored on the host in HOSTACCESS 's transmit format named MASTER.LETTER into the DOS file **C:\WP\LETTER.DOC**, enter:

**PICK.DOS MASTER.LETTER TO C:\WP\LETTER.DOC (T**

You may also directly transfer a PICK item to a DOS print service, for example:

**PICK.DOS BP PROG2 TO LPT1**

The [PICK.DOS.ALL](#) command creates one DOS file from an active select list or from an entire PICK file. The DOS file contains all the selected PICK items and their real keys in an internal format specific to HOSTACCESS. To upload the records in the file, use the [DOS.PICK.ALL](#) routines.

[TDUMP](#) can also download multiple PICK items, but will create a DOS file for each record. However, TDUMP has the advantage that it is capable of quickly uploading specific records.

PICK data is downloaded in the PICK internal format. Dates and numerics are stored as integers; for example, “24 AUG 94” becomes “9733”. See the [PASS.TO...](#) sections for details of this format.



PICK.DOS.ALL may be invoked from the TCL command line as follows:

**PICK.DOS.ALL {DICT} PICKfilename {itemname1... itemnamen} TO dosname {(options)}**

Where:

<b>DICT</b>	Use this literal to specify the dictionary of <b>PICKfilename</b> .
<b>PICKfilename</b>	Is the name of the PICK file to transfer from
<b>itemname1 ... itemnamen</b>	Are the names of the items to be sent to DOS. If not specified, either the active select list or the entire PICK file will be downloaded.
<b>dosname</b>	Is the DOS file name to be created or overwritten, optionally containing a drive letter and DOS path. If no path is specified, <b>dosname</b> is created in the current HOSTACCESS runtime directory.

To find the status of the file transfer, the status record (as described for PIX.PICK.DOS) is written to the file [PIX.CONTROL.F](#) with a key of **nn.FT.LOG**, where **nn** is the port number.



Options available.

Using PICK.DOS.ALL.

Anything following the open left bracket in the PICK.DOS.ALL command is treated as an option. Multiple options can be specified with or without a delimiter. These options are:

- A** Append to the existing DOS file. This is useful for building up one large DOS file from a number of PICK items. If this option is not specified, the DOS file will be overwritten, or created if it does not already exist.
- Z** Suppress all file transfer output including the status window. It is generally used from applications so that the file transfer takes place completely transparently and without the user needing to press a key to acknowledge completion.
- B** Binary file - override default. i.e. no conversion of Attribute Marks CHAR(254) to CRLF and no conversion of value and sub-value marks to spaces.
- L** **Local** option, for a locally connected PC - this will speed up the transfer by performing simpler error checking. If HOSTACCESS detects that you are running over one of its supported resilient networks, it will use the "L" option by default.
- R** **Remote** option - use this to tell HOSTACCESS not to automatically use the L option when running file transfer over non resilient networks. You should specify this option if you know that your network may be non-resilient at any point (e.g. if you are using non-error checking modems). Use this option if you are experiencing file transfer errors when communicating over a network.

**Note** If the records contain any control characters or any multivalued or subvalues, you should use the **B** (binary) option for download and upload.

To download selected records from the PICK file STAFF into a single DOS file called STAFFFILE.ALL, use the following commands:

```
SELECT STAFF WITH AGE > "65"  
PICK.DOS.ALL STAFF TO C:\FILES\STAFF\STAFFFILE.ALL
```

To download all programs in the BP file into a single DOS file called PROGS.ALL, enter:

```
PICK.DOS.ALL BP TO PROGS.ALL
```

Use the [TDUMP](#) command to dump a specified file or items to a DOS disk, in a similar way to PICK 's TDUMP utility dumping to tape. You can then use the [TLOAD](#) command to load them up again (for example on another host), removing any tape incompatibility problems.

This routine is very useful for program files and small data files, and also gives you a cost-effective way of shipping your data to different hosts as well as backing up your important programs. Simply dial up your customer site (if they have HOSTACCESS host programs loaded), TLOAD your latest code, compile, delete and disconnect.

This program exploits the [PICK.DOS](#) command.

If a PICK select list is active, i.e. you have executed a SELECT, SSELECT or GET-LIST immediately prior to invoking [TDUMP](#), then this list will be applied to the file to be TDUMPed.

This program is not designed for exporting PICK data into different DOS formats for down-loading into DOS packages. This is achieved using the [GET](#) and [PASS.TO](#) routines.

Click here  for details of downloading multiple PICK items.

**Note:** TDUMP can dump a maximum of 999 records at a time. This is because DOS only allows a maximum of 3 characters as a file extension.

You may also notice that the more DOS files created, the slower the transfer. This is due to the DOS file management system.

TDUMP may be invoked as a TCL command and follows a very similar format to the standard PICK T-DUMP command. The syntax is as follows:

**TDUMP {DICT} {filename} {itemname} {(options)}**

Where:

**DICT** Use this literal to specify the dictionary of filename.  
**filename** Is the PICK data file from which items will be dumped.  
**itemname** Is the optional name of any single item to be dumped. If you specify the item name as an asterisk "\*" then all records in the file will be dumped.

You are then prompted for the DOS 'tape' name. This must be no longer than 8 characters and should not include a DOS extension, as this will be appended by TDUMP. Lastly, you are prompted for the path to the DOS directory where the 'tape' is to be stored. This must be the FULL path.

- PICK.DOS.ALL.
- Options available for the TDUMP command.
- TDUMP Example.
- INTER.SEND.
- INTER.RECEIVE.

Any characters following the open round bracket are treated as options. Multiple options may be specified as follows:

- O** Overwrite existing 'tape' DOS files if they already exist.
- Z** Suppress display of file transfer status windows which would normally appear for each item in the file.
- S** Suppress record id 's being displayed as they are downloaded.

Any of the [PICK.DOS](#) file transfer options are also valid. The B binary option is forced ON by default to allow multi-values etc. to be passed between hosts.

**Note:** If a file transfer error occurs, TDUMP automatically restarts.

Say you want to TDUMP selected records in the BP file down to DOS. At TCL, enter:

**SELECT BP = "[GAME]"**

**TDUMP BP (ZS**

At the tape name prompt type : **PROGRAMS**. At the directory prompt type : **C:\TAPES**. This creates the following DOS files in the C:\TAPES directory:

**PROGRAMS.PIX** An internal index used to identify what 's in the TAPE to enable TLOAD to load the records back with the correct keys.

**PROGRAMS.001** The first record

**PROGRAMS.002** The second record

**PROGRAMS.xxx** etc.

**PROGRAMS.XRF** Index to identify DOS name given to each PICK name.

You can then backup all of these DOS files say to drive B by typing at DOS, **COPY C:\TAPES\PROGRAMS.\* B:** This diskette could then be sent to another site, perhaps, where they could run a corresponding TLOAD to upload the programs.

TLOAD performs similar functions to PICK 's T-LOAD utility but instead of loading the specified file or records from tape, they are loaded from a DOS disk. Please TDUMP prior to using this command. TLOAD uploads records from DOS created by TDUMP.



TLOAD may be invoked as a TCL command and follows a very similar format to the standard PICK T-LOAD command. The syntax is as follows:

**TLOAD {DICT} {filename} {(options)}**

Where:

**DICT** Use this literal to specify the dictionary of **filename**  
**filename** Is the PICK data file into which items will be loaded.

You will then be prompted for the DOS tape name. This must be the name used when the data was dumped by the TDUMP command.

The second and final prompt is for the path to the DOS directory where the 'tape' filenames are stored. This must be the full path e.g. C:\HOST

**Note** If a file transfer error occurs, TLOAD automatically restarts.



Uploading multiple PICK items using DOS.PICK.ALL.



Options available with TLOAD.



TLOAD Example.



Passing PICK data into DOS or Windows Packages.



INTER.SEND.



INTER.RECEIVE.

Any characters following the left brackets are treated as options. Multiple options may be specified as follows:

- O** Overwrite any existing PICK items if they already exist.
- S** Suppress item id 's being displayed as they are uploaded.
- Z** Suppress display of file transfer status window which would normally appear for each record uploaded.
- ?** The inclusion of this option will cause TDUMP to display a list of ALL the items found and then allow you to either enter selected item names or enter an '\* ' to upload all items.

Say you want to TLOAD the DOS files into the BP file from the 'tape ' PROGRAMS. (i.e. to upload using the data that would have been created following the example in TDUMP). At TCL, enter the following command :

**TLOAD BP (OZ**


At the tape name prompt type : **PROGRAMS**

At the directory prompt type : **C:\TAPES**

This will upload any records found in DOS and overwrite the corresponding PICK items. The Z option will suppress the file transfer windows.

HOSTACCESS 's Host Utilities have automated almost every aspect of extracting data from PICK databases and passing all or any portion of this into DOS or Windows packages, in PICK 's external format.

Host utility drivers are provided for PICK users to enable them to easily and quickly pass PICK data into virtually every DOS and Windows package available today, from WordPerfect to Lotus amongst others. Options are even provided for the users to specify how they would like the data to be presented by the DOS or Windows package, e.g. as a pie-chart within SuperCalc.

Click here  for further information.

The following sections describe how you can use the routines documented to perform file transfer between HOSTACCESS sessions.

- DOS.PICK
- PICK.DOS
- DOS.PICK.ALL
- PICK.DOS.ALL
- INTER.SEND
- INTER.RECEIVE

DOS.PICK may be invoked from TCL in inter-session mode using the following syntax:

**DOS.PICK {DICT} {PICKfilename itemname } FROM sessionname {(C){ options}**

Where:

<b>DICT</b>	Use this literal if itemname is to be created in the dictionary of the file <b>PICKfilename</b>
<b>PICKfilename</b>	Is the PICK file name where itemname is to be created.
<b>FROM sessionname</b>	Is the literal text FROM followed by the name given to this transfer session. The session name may be any name which the user wants to allocate. This name then identifies this particular session to all other sessions. Any other HOSTACCESS session which is given a <a href="#">PICK.DOS</a> command using this session name and the (C option will then establish an inter-session communications link. Data will be then be transferred between the 2 sessions.
<b>(C</b>	This is mandatory as it signifies to HOSTACCESS that the user wishes to invoke an inter-session data transfer as opposed to an ordinary DOS to PICK transfer. When DOS.PICK is invoked with a (C option HOSTACCESS will attempt to establish a communications link with another HOSTACCESS session which is using the same session name. As soon as this link is established, data transfer will take place provided that the commands are used in a complementary fashion. e.g. a DOS.PICK transfer as described here should always attempt to establish a link with another HOSTACCESS session running a PICK.DOS command.
<b>options</b>	All other normal DOS.PICK options are available. Options should be used sensibly. Always bear in mind that this session will interface with another session running a PICK.DOS command.

PICK.DOS may be invoked from TCL in inter-session mode using the following syntax :

**PICK.DOS {DICT} {PICKfilename itemname } TO sessionname {(C){ options}**

Where:

<b>DICT</b>	Use this literal if itemname is to be retrieved from the dictionary of the file <b>filename</b>
<b>PICKfilename</b>	Is the PICK file name where itemname is to be retrieved from. Should be null if T option used.
<b>TO sessionname</b>	Is the literal text TO followed by the name given to this transfer session. The session name may be any name which the user wants to allocate. This name then identifies this particular session to all other sessions. Any other HOSTACCESS session which is given a <a href="#">DOS.PICK</a> command using this session name and the (C option will then establish an inter-session communications link. Data will be then be transferred between the 2 sessions.
<b>(C</b>	This is mandatory as it signifies to HOSTACCESS that the user wishes to invoke an inter-session data transfer as opposed to an ordinary PICK to DOS transfer. When PICK.DOS is invoked with a (C option HOSTACCESS will attempt to establish a communications link with another HOSTACCESS session which is using the same session name. As soon as this link is established, data transfer will take place providing the commands are used in a complementary fashion. e.g. a PICK.DOS transfer as described here should always attempt to establish a link with another HOSTACCESS session running a DOS.PICK

command.

**options** All other normal PICK.DOS options are available with the exception of the A - append operation. Options should be used sensibly. Always bear in mind that this session will interface with another session running a DOS.PICK command.

DOS.PICK.ALL may be invoked in inter-session mode from the TCL command line as follows :

**DOS.PICK.ALL {DICT} {PICKfilename} FROM sessionname {(C){options}**

Where:

**DICT** Use this literal if the records are to be created in the dictionary of the file **PICKfilename**

**PICKfilename** Is the PICK filename where the records are to be created

**FROM sessionname** Is the literal text FROM followed by the name given to this transfer session. The session name may be any name which the user wants to allocate. This name then identifies this particular session to all other sessions. Any other HOSTACCESS session which is given a [PICK.DOS.ALL](#) command using this session name and the (C option will then establish an inter-session communications link. Data will be then be transferred between the 2 sessions.

**(C** This is mandatory as it signifies to HOSTACCESS that the user wishes to invoke an inter-session data transfer as opposed to an ordinary DOS to PICK transfer. When DOS.PICK.ALL is invoked with a (C option HOSTACCESS will attempt to establish a communications link with another HOSTACCESS session which is using the same session name. As soon as this link is established, data transfer will take place providing the commands are used in a complementary fashion. e.g. a DOS.PICK.ALL transfer as described here should always attempt to establish a link with another HOSTACCESS session running a PICK.DOS.ALL command.

**options** All other normal DOS.PICK.ALL options are available. Options should be used sensibly. Always bear in mind that this session will interface with another session running a PICK.DOS.ALL command.

PICK.DOS.ALL may be invoked in inter-session mode from the TCL command line as follows:

**PICK.DOS.ALL {DICT} {PICKfilename { itemname1 ... itemnamen } TO sessionname {(C){options}**

Where:

**DICT** Use this literal if the records are to be retrieved from the dictionary of the file **PICKfilename**

**PICKfilename** Is the PICK filename where the records are contained

**itemname1 ... itemnamen** Are the names of the items to be retrieved from the specified PICK file. If no itemnames are specified, all records will be transferred or if a SELECT command is active then the items selected will be transferred.

**TO sessionname** Is the literal text TO followed by the name given to this transfer session. The session name may be any name which the user wants to allocate. This name then identifies this particular session to all other sessions. Any other HOSTACCESS session which is given a [DOS.PICK.ALL](#) command using this session name and the (C option will then establish an inter-session communications link. Data

will be then be transferred between the 2 sessions.

**(C)** This is mandatory as it signifies to HOSTACCESS that the user wishes to invoke an inter-session data transfer as opposed to an ordinary DOS to PICK transfer. When PICK.DOS.ALL is invoked with a (C option HOSTACCESS will attempt to establish a communications link with another HOSTACCESS session which is using the same session name. As soon as this link is established, data transfer will take place providing the commands are used in a complementary fashion. e.g. a PICK.DOS.ALL transfer as described here should always attempt to establish a link with another HOSTACCESS session running a [DOS.PICK.ALL](#) command.

**options** All other normal PICK.DOS.ALL options are available with the exception of the A - append operation. Options should be used sensibly. Always bear in mind that this session will interface with another session running a DOS.PICK.ALL command.

INTER.SEND and [INTER.RECEIVE](#) are provided for all those who regularly use HOSTACCESS 's TDUMP and TLOAD commands. These 2 commands invoke HOSTACCESS 's inter-session mode. Consequently, instead of records being dumped to DOS by TDUMP and then subsequently being uploaded to PICK, data is transferred directly between 2 sessions of HOSTACCESS.

**Note:** one session must be running the INTER.SEND command and the other session must be running the INTER.RECEIVE command.

INTER.SEND may be run from TCL using the following syntax:

```
INTERSEND {DICT} {PICKfilename} {itemname}TO sessionname {(options)}
```

Where:

<b>DICT</b>	Use this literal if itemname is to be retrieved from the dictionary of the file <b>PICKfilename</b>
<b>PICKfilename</b>	Is the PICK file name from which items will be transmitted.
<b>itemname</b>	Is the optional name of any single item to be dumped. If itemname is set to * then all items will be dumped. If itemname is not specified then any items in an active SELECT list will be dumped.
<b>TO sessionname</b>	Is the literal text TO followed by the name given to this transfer session. The session name may be any name which the user wants to allocate. This name then identifies this particular session to all other sessions. Any other HOSTACCESS session which is given a <a href="#">INTER.RECEIVE</a> command using this session name and the (C option will then establish an inter-session communications link. Data will then be transferred between the 2 sessions.
<b>options</b>	Other options are the same as for <a href="#">TDUMP</a>

INTER.RECEIVE and [INTER.SEND](#) are provided for all those who regularly use HOSTACCESS 's [TDUMP](#) and [TLOAD](#) commands. However, these 2 commands invoke HOSTACCESS 's inter-session mode. Consequently, instead of records being dumped to DOS by TDUMP and then being subsequently uploaded to PICK, data is transferred directly between 2 sessions of HOSTACCESS.

**Note:** One session must be running the INTER.SEND command and the other session must be running the INTER.RECEIVE command.

INTER.RECEIVE may be run from TCL using the following syntax :

```
INTER.RECEIVE {DICT} {PICKfilename}FROM sessionname {(options)}
```

Where:

<b>DICT</b>	Use this literal if items are to be created in the dictionary of the file <b>PICKfilename</b>
<b>PICKfilename</b>	Is the PICK file name to which items will be transferred.
<b>FROM sessionname</b>	Is the literal text FROM followed by the name given to this transfer session. The session name may be any name which the user wants to allocate. This name then identifies this particular session to all other sessions. Any other HOSTACCESS session which is given a <a href="#">INTER.SEND</a> command using this session name and the (C option will then establish an inter-session communications link. Data will be then be transferred between the 2 sessions.
<b>options</b>	Other options are the same as for <a href="#">TLOAD</a> except for the ? option which is not supported.

This section describes EASY.ACCESS and related functions - GET and PASS.TO. These applications make extensive use of the PICK AiF Programs library and are excellent examples of what can be achieved by combining the power of PICK/BASIC with AiF and the DOS environment.

<a href="#">EASY.ACCESS</a>	A user-friendly database inquiry front-end which automatically builds database inquiry and extraction commands using the PICK inquiry language (INFORM, ENGLISH, Retrieve, RECALL, etc.).
<a href="#">GET</a>	A powerful database extraction program combining the power of PICK 's inquiry language with full support for all dictionary correlatives, conversions and I-types.
<a href="#">PASS.TO</a>	A number of "PASS.TO" routines that will take data formatted by GET and pass this to popular DOS or Windows products, such as LOTUS, SuperCalc, WordPerfect, WORD for Windows, etc..

EASY.ACCESS is a powerful user-friendly front-end for all versions of PICK 's inbuilt inquiry language. It may be used to create and execute inquiry sentences for:

INFORM	Prime INFORMATION
ENGLISH	McDonnell Douglas REALITY
ACCESS	PICK R83
Retrieve	VMark UniVerse
RECALL	Ultimate
Uniquery	UniData

EASY.ACCESS supports all of the inquiry commands in ALL the PICK flavours. Developers have full control of which commands are presented to the user and may modify (or remove) the command descriptions available to the user to construct sentences.

Specific tables of commands and modifiers have been prepared especially for UniVerse and Prime INFORMATION users and these are selected automatically by EASY.ACCESS to match the machine type you are running on.

Help is available for certain commands and options by calling TCL HELP (only if standard HELP is supported on that system). If help is requested for a Dictionary Definition, the definition is displayed within a pop-up window.

EASY.ACCESS has been integrated with HOSTACCESS 's [PASS.TO DOS](#) and Windows facilities, to allow the result of virtually any inquiry to be automatically transferred into DOS or Windows applications including databases, spreadsheet products, and word processors.

The data extraction and transfer is completely automatic and includes:

- Taking the user into the DOS or Windows product.
- Importing the transferred data.
- Presenting the user with the result (for example, a LOTUS 3-D pie-chart).

When using the [PASS.TO](#) facilities, the relevant DOS or Windows product will be called and the data imported, e.g. PASS.TO.LOTUS.

Some "PASS.TO" routines only create a DOS file and do not call a DOS or Windows product. For example, [PASS.TO.DIF](#) will create a DIF file under DOS in the file called PIXELnnn.TMP (where 'nnn' is the PICK user 's port number) in the current



HOSTACCESS directory.

- Starting EASY.ACCESS.
- Using EASY.ACCESS.
- EASY.ACCESS Applications Integration.
- EASY.ACCESS diagram.

Info	Command	Sort	Selection	Output	Aliases	Profile	Special
<b>EASY ACCESS</b>							
Get List	Command	Sort					
Save List	File	Dictionary					
<b>Sort Criteria</b>							
<b>Selection Criteria</b>							
<b>Output Criteria</b>							
<b>Aliases</b>							
<b>Profile</b>							
<b>Special</b>							

To start EASY.ACCESS, invoke it from the TCL command line as follows:

**EASY.ACCESS {filename} {group} {(options)}**

Where:

- filename** Optional name of data file for which inquiries will be constructed. This can also be specified from within EASY.ACCESS.
- group** Dictionary groups can be set up as records in the dictionary of filename. These records allow you to logically group together dictionaries. i.e. SALES, COSTS. And, more importantly, groups can be used to restrict access to certain areas of the dictionary.

Using the word **ALL**, overrides any groups and will present the user with ALL of the dictionaries found in the file.

- (options)** Anything following the open left round bracket is treated as an option. Multiple options can be specified with or without a comma delimiter. Most of the options are designed to be set when EASY.ACCESS is CALLED as a subroutine from other applications. These options can be as follows
- C** Clear last sentence, as by default, EASY.ACCESS displays the last sentence for this port.
- D** Used with the **C** option, clears back to default sentences held in the PIX.DEFAULTS record in the **PIX.CONTROL.E** file. The format of this record is the same as the other records in this file.
- 132** Switch EASY.ACCESS into 132 column mode.
- A** The user will not be able to select ALL of the dictionaries. Only groups are allowed. This option is ignored if NO groups are found in the file so ensure you have at least one.
- M** Multiple Groups, this option tells EASY.ACCESS to prompt for dictionary names by first prompting for the group name. Some users may find this a more friendly way of presenting dictionaries, some users may find this restrictive. Applications may FORCE the user to use multiple groups by setting both the 'F' and 'M' options.
- F** Security feature - If a group name or 'M' option is used with the 'F' option, the user is FORCED to use them. This option is useful if applications wish to limit users to only certain parts of the dictionary. The user will also NOT be able to change files or load other sentences. To FORCE the user to use a previously saved sentence, the application can copy this saved record from the PIX.EASY.ACCESS.F file into the 'LAST.SENTENCE.PORT.n' record.

All of EASY.ACCESS's facilities are presented to the user in friendly pop-down menus and selection boxes. Users are guided through these menus in a logical but efficient manner when constructing inquiry sentences. The ease of use of the menu system is one of EASY.ACCESS's strongest points. Use of just the arrow keys, <RETURN> and <ESCAPE>, allows the user to quickly access and select the menu options required.

A number of Function keys are provided for general use throughout the menus, which may be used instead of moving to the appropriate menu option. At the top most level of EASY.ACCESS main menus, these are:

- F5 Process sentence.
- F10 Help on currently highlighted option or dictionary definition (if system supports HELP command from TCL).

Other function keys available while reviewing (amending) parts of the constructed sentence are:

- F1 Amend text/option.
- F2 Delete text/option.
- F3 Move text/option.
- F8 Restore text/options.

The <ESCAPE> key should be used to return from lower menu levels or to exit from EASY.ACCESS when at the top menu level.

You may review the various component parts of the final sentence at any point. For example, you may wish to change the Selection Criteria - simply select **Review** from the **Selection** menu.

Dictionary groups may be used to 'break-up' perhaps a very large list of dictionary items into smaller and more manageable groups. Options are available to the user to select to work with a different group on the fly (unless the F FORCE option has been used).

Dictionary Groups are records defined in the Dictionary of the file being used by EASY.ACCESS. They have the following format:

Key	PIX.GROUP. <b>groupname</b>
Field 1	<b>Group description</b>
Field 2	Multi-valued list of Dictionary Definition keys belonging to this group, these should really exist in the dictionary of the file to ensure valid sentences are constructed.

Where:

<b>Groupname</b>	User defined name for this group (e.g. SALES, DEPARTMENTS) - the file containing the group name must be prefixed by "PIX.GROUP." and thus is selected automatically by EASY.ACCESS for use by the users.
<b>Group description</b>	Up to 60 characters describing this group - this is the text that will be presented to the user when working with groups.

These Dictionary Groups should be entered into the appropriate Dictionary files.

Sentences may be saved and recalled by selecting the relevant options from the CONFIG menu. Any saved sentences are stored in the file [PIX.EASY.ACCESS.F](#) which normally points back to the main HOSTACCESS account. This file can be created in the local account if preferred.

You may also specify that the data file be inquired upon by using another Dictionary definitions file (uses generic PICK USING construct).

Selecting and processing one of the 'pass to PC' application options tells EASY.ACCESS to call HOSTACCESS's data extraction command [GET](#) and then call the relevant [PASS.TO](#) routine.

EASY.ACCESS is processed by the subroutine PIX.EASY.ACCESS which may be called by any PICK program, as follows:

**CALL PIX.EASY.ACCESS(TCL.INPUT,TCL.OPTIONS)**

Where:

**TCL.INPUT**        May be null or any of the command line parameters described above (i.e. PRICES SALES) for the SALES group from the PRICES file.

**TCL.OPTIONS**     May be null or any of the command line options described above i.e. C132F to FORCE only the SALES group from the PRICES file, 'C' clearing the last sentence and switching HOSTACCESS into 132 column mode first.

**Program Example**

```
*
* Program to call EASY.ACCESS using the file PRICES and
* force the user to only look at the SALES group of
* dictionaries. Where PIX.GROUP.SALES exists in the
* dictionary of the file PRICES containing the list of
* dictionaries in that group as a multi-value array etc.
*
* The 'C' option tells EASY.ACCESS to clear the previous
* sentence first and start with an empty statement.
*
CALL PIX.EASY.ACCESS ("PRICES SALES", "C")
*
```

GET is a powerful facility that follows a similar syntax to the LIST and SORT commands from the standard PICK inquiry language. But, instead of sending the result of the inquiry to the screen or printer, the GET command stores the result in a temporary file, ready to send to any one of the many DOS or Windows products supported by HOSTACCESS. For instance the TCL statement:

**GET STAFF NAME AGE SALARY // WITH RETIREMENT = "1991"**

could be used to extract the data from the STAFF file that can be subsequently transferred to WordPerfect in mailmerge format and print the letters to all people retiring in 1991. Also:

**GET STAFF BREAK-ON DEPT TOTAL SALARY // BY DEPT**

could be used to extract the data from the STAFF file that can be subsequently transferred to a LOTUS 123 spreadsheet and plot a piechart of the salary breakdown by department.

The GET command evaluates and processes the inquiry statement and stores the data in a specific port dependent file ready for subsequent transfer to a DOS file.

GET is not involved in passing the data to DOS or to Windows. To pass the data to a DOS or Windows product in the format you require, you only have to run one of the [PASS.TO](#) routines provided.



Starting GET.

Using GET.

GET Applications Integration.

GET may be invoked from the TCL command line and if a select list is active, it will be used. At first sight, the GET syntax may look confusing. However, the command differs only slightly from the standard PICK LIST and SORT commands. The main difference is that GET requires the syntax to be in a specific order with some additional delimiters (the `//` or `%%` ). At the end of this section, there are some examples that show just how easy it is to use GET. The GET command may be entered at TCL as follows:

```
GET file {USING dictfile} {output} {suppression} {// sort_selection} {%% listname} {(options)}
```

Where:

<b>file</b>	The name of the PICK Data file.
<b>USING dictfile</b>	Optional if separate dictionary required on data file.
<b>output</b>	Output list of dictionary definitions in order required. This may also include the BREAK-ON, TOTAL and GRAND-TOTAL modifiers. The following modifiers may also be used if your host system normally supports them from the LIST/SORT commands, AVERAGE, PERCENT etc.
<b>suppression</b>	Optional suppression modifiers including ID-SUPP, DET-SUPP, HDR-SUPP, COL-HDR-SUPP.
<b>//</b>	The // is used to help GET identify the start of the sort/selection criteria. If not used and there is no active select list GET will prompt for the sort/select the start of the sort/selection criteria. The // can be used on its own to suppress the prompt if no sort/selection criteria is required.
<b>sort_selection</b>	If you have specified the // delimiter, the SORT and/or SELECTION criteria should be specified in the normal ACCESS format, e.g. WITH field EQ "100", BY, BY-DSND, etc.
<b>%%</b>	The %% is used to help GET find the specified `listname` in the command. Even though GET will work from the currently active select list, it is useful, for instance when calling GET as a subroutine, to be able to pass a select list name. GET will then do a GET-LIST of `listname` before processing the GET command.
<b>listname</b>	The name of the previously saved list should be specified after the %% delimiter.
<b>(options)</b>	Everything following the `( ` delimiter is treated as an option. Options can be combined and may be space or comma delimited. They may be: <ul style="list-style-type: none"> <li><b>B</b> Suppresses the blank lines between breaks and totals.</li> <li><b>C</b> Same as COL-HDR-SUPP</li> <li><b>I</b> Same as ID-SUPP</li> <li><b>D</b> Same as DET-SUPP</li> <li><b>H</b> Same as HDR-SUPP</li> <li><b>P</b> Send the result to the printer instead of the file.</li> <li><b>S</b> Suppresses the display of all progress messages.</li> <li><b>T</b> Send the result to the terminal screen instead of the file.</li> <li>. If a full-stop is in the options, this suppresses the dots used as column separators on output reports.</li> </ul>

**Note:** The `T` and `P` options are normally used only to check that GET is producing the results you expect, if you are experiencing difficulties.



Using GET.

GET Applications Integration.

GET is very simple to use, just enter the TCL command as described above and you should have no problems. When the GET routine has finished you are returned direct to TCL. If any errors occurred, these will be reported.

Some sample TCL commands using GET:

1. GET SALES BREAK-ON AREA TOTAL COST TOTAL SALES ID-SUPP DET-SUPP // BY AREA WITH AREA NE "NORTH" %% LIST.A  
  
Will generate a summary 3 column output file, area, cost and sales using a previously saved list LIST.A
2. GET SALES AREA DESC PROFIT LY.PROFIT //  
  
Will generate an output file containing 4 columns, area, desc, profit and last year 's profit. Since '// ' is used, you will not be prompted for any sort or selection criteria.
3. GET SALES AREA DESC PROFIT LY.PROFIT %% LIST.A  
  
Will generate an output file containing 4 columns, area, desc, profit, last year 's profit using a previously saved list called LIST.A.
4. GET-LIST LIST.A at TCL followed by the above GET command without '%% LIST.A ' will have the same effect.

In each of the above cases, the resulting information is stored in the file [PIX.OUTPUT.nn.F](#) (where **nn** is the port number). If the file does not already exist it is created, otherwise it is cleared before use.

This file will normally contain one header record and one or more detail records with sequential keys. These records contain the result of the GET inquiry with an attribute mark delimiting each line of output and a multi-value delimiting each column. Most users will have no interest in the format of the file since they are only interested in passing the result of their inquiry on to their required DOS or Windows product.

After running GET you can run any of the [PASS.TO](#) routines to transfer the data to Windows or DOS spreadsheets, wordprocessors, etc. You may run any of the "PASS.TO" routines over and over on the same inquiry without having to re-run GET, because the PIX.OUTPUT.nn.F file remains intact until the next GET.



GET Applications Integration.



GET is processed by the subroutine PIX.GET which may be called by any PICK program as follows:

**CALL PIX.GET(OUTPUT, SORT, OPTIONS, LIST.NAME, ERROR)**

Where

**OUTPUT** Should start with the file name followed by the output list of dictionaries in the order required. This may also include the BREAK-ON, TOTAL and GRAND-TOTAL modifiers as well as ID-SUPP, DET-SUPP etc. The following modifiers are also supported if your host system normally supports them from the LIST/SORT commands, AVERAGE, PERCENT, CALC etc.

**SORT** This variable should be passed containing the SORT and SELECTION criteria that would normally follow the //documented earlier.

**OPTIONS** Should be the string containing any/all of the options, such as I,D,H,C, etc. documented earlier.

**LIST.NAME** If no '%%' is passed via TCL.INPUT, this variable may be passed as the name of the previously saved list to be used with the inquiry. This variable is optional.

**ERROR** Since GET relies on certain HOSTACCESS control files existing, if any of them are not found the user will see an error message on the screen and this variable will be set to TRUE. Your application should take action accordingly, i.e. do not call a "PASS.TO" if ERROR set since the output may be invalid.

#### Program Example

```
*
* Program to automate taking everybody from the STAFF file
* aged 65 and over using a GET-LIST name of COMPANY.A.LIST
* and creating a Wordperfect mailmerge file with
* ID, name, address, age, date of birth, and salary
* information.
*
OUTPUT = "STAFF NAME ADD1 ADD2 ADD3 AGE DOB SALARY"
*
SELECTION = 'WITH AGE >= "65" BY SURNAME'
*
CALL PIX.GET(OUTPUT, SELECTION, "", "COMPANY.A.LIST", ERROR)
*
CALL PIX.PASS.TO.WORDPERFECT("WP5", "C:\WP\RETIRE.DOC", ERROR)
*
END
```

HOSTACCESS provides a powerful facility for PICK users to be able to pass data directly into DOS or Windows products using virtually any ACCESS/ENGLISH/INFORM etc., like statement. It is not necessary for a user to understand how to read or import data into the DOS or Windows software since HOSTACCESS, through the PASS.TO routines, can do everything automatically.

"PASS.TO" implies all of the HOSTACCESS pass to DOS or Windows routines, for example PASS.TO.EXCEL, PASS.TO.LOTUS, PASS.TO.SUPERCALC, PASS.TO.SYMPHONY, PASS.TO.QUATTRO, PASS.TO.WORD, or PASS.TO.WORDPERFECT. The PASS.TO routines fall into the following categories:

PASS.TO spreadsheet products - PASS.TO.EXCEL, PASS.TO.LOTUS, PASS.TO.SUPERCALC, PASS.TO.SYMPHONY, and PASS.TO.QUATTRO.

PASS.TO word processing products - PASS.TO.WORD and PASS.TO.WORDPERFECT.

PASS.TO DOS files - PASS.TO.DIF and PASS.TO.DOS.

PASS.To Other PC Products and Formats.

Click here  for notes on PASS.TO.

Click here  for information on using PASS.TO.

Each of these [PASS.TO](#) categories is described in its own section, showing general syntax rules and examples for that category. Any details that are specific to a particular routine are then documented separately in a subsection.

All these routines can be driven directly as TCL commands. They can also be driven from PICK/BASIC as subroutine calls or are available as command interfaces from [EASY.ACCESS](#).

It is recommended that you read the topics relating to the [GET](#) process before using any of the PASS.TO routines.

All the PASS.TO routines will pass the data from the last GET inquiry into the particular DOS or Windows software product or file format such as LOTUS, SUPERCALC, DIF etc.

Each PASS.TO application has a control record, and any application that supports both DOS and Windows has 2 records, the Windows record having a suffix of .WINDOWS. The PASS.TO control records also hold information on graph types (where relevant) and how to start and run the DOS or Windows application.

Most of the PASS.TO routine names that explicitly imply a DOS or Windows product, for example PASS.TO.LOTUS (as opposed to those that imply a DOS file format such as [PASS.TO.DIF](#)) not only file transfer the data to the PC but also take the user directly into the product as well. In most cases, PASS.TO routines can continue by importing the data and even plotting graphs automatically, without any user intervention.

**Note 1** For backwards compatibility, the COMMAND parameter in all the PASS.TO application subroutines still exists. However, we recommend that you leave this null and that the DOS or Windows command is stored in attribute 3 of the relevant control record in the PIX.CONTROL.F file. For example, with PASS.TO.LOTUS the stored command would read:

```
003 123
```

and with PASS.TO.LOTUS.WINDOWS the stored command would read:

```
003 123W.EXE
```

This is based on the assumption that your DOS PATH variable has been correctly configured to point to this command. We recommend that you use this standard for calling all DOS or Windows applications through HOSTACCESS.

The command used to start the spreadsheet application is retrieved by reading attribute 3 of the PASS.TO. application record in the PIX.CONTROL.F file. If the 'nth' multi-value ('n' being port number plus 2) is not null then the command stored there is used. If this multi-value is null then the command stored in multi-value 1 is used. We recommend that everyone on the host calls the particular DOS or Windows product in the same way. That is, they should use multi-value position 1 only. This is easier on networks, for example, where the port number is not always known.

**Note 2** When you use PASS.TO routines, temporary DOS files are created. If data is being passed to a DOS product, the temporary files are subsequently removed automatically. However, if data is being passed to a Windows product, the temporary files cannot be deleted owing to Windows multi-tasking environment. Your application may want to delete this temporary DOS file or just leave it to be overwritten by the next PASS.TO routine. The temporary DOS file is called PIXELnn.TMP (in PASS.TO.QUATTRO it is called PIXELnn.DIF) where nn is the host port number.

**Note 3** EXCEL, and the Windows versions of WORD, QUATTRO PRO, and LOTUS, can be DDE servers. Therefore, data that is passed to or merged into these products using the PASS.TO routines is done using the DDE calls and subroutines built into HOSTACCESS for Windows.

An example of the way a "PASS.TO" routine works in conjunction with GET is briefly described here.

If you require a spreadsheet containing sales information for each area, the following TCL statement:

**GET SALES BREAK-ON AREA TOTAL SALES ID-SUPP // BY AREA**


followed by:

**PASS.TO.LOTUS (P**

will take the result of the GET statement, transfer this data into a DOS file, call LOTUS (for Windows or DOS), import the DOS file, change the column widths, and draw a piechart. All with no user intervention.

The PASS.TO facilities can easily be integrated into PICK applications so that a PICK user could, perhaps, select a menu option such as 'Histogram total Costs by Area '. Underneath, your application will first call the GET subroutine and then call the PASS.TO.LOTUS subroutine. The user will see this facility as an integral part of your own application and will be completely unaware that the GET and PASS.TO are doing most of the work.

```
*
* Program to automate taking last year's COST figures
* out of PICK using ACCESS statements via the GET
* process and then to plot a LOTUS histogram.
*
* GET called with D and I options, DET-SUPP and ID-SUPP
*
CALL PIX.GET ("COST.FILE BREAK-ON AREA TOTAL COSTS",
"BY AREA", "DI", "", ERROR)
*
CALL PIX.PASS.TO.LOTUS("", "H", ERROR) ; * "H"istogram
*
END
```

These routines include PASS.TO.EXCEL, PASS.TO.LOTUS, PASS.TO.QUATTRO, PASS.TO.SUPERCALC and PASS.TO.SYMPHONY. Click here  for more information.

The PASS.TO routines that relate to spreadsheet products take the data produced by the last [GET](#) statement and pass this directly into a new spreadsheet. The data will be imported automatically into the new spreadsheet. Numbers will be imported into the spreadsheet as numerics. The column widths will be changed to correctly match those in each output dictionary specified in the GET command.

Because a new spreadsheet is created, the user can save this to a spreadsheet file as required. If the user exits from the spreadsheet, say after viewing a graph, no data is saved.



Starting the PASS.TO Spreadsheet Routine.



PASS.TO Spreadsheet Examples.



PASS.TO Spreadsheet Applications Integration.



PASS.TO Spreadsheet Example Program.

The PASS.TO spreadsheet routine may be called from the TCL command line as follows:

**PASS.TO.programname {(graph\_type)}**

Where:

**programname** Is EXCEL, LOTUS, SUPERCALC, SYMPHONY, or QUATTRO.

**graph\_type** If null, the data is just imported and the user is taken into the spreadsheet.

You may specify the graph type either by using the name of the graph or by using a number that corresponds to the required graph type. A list of graph type names is held in attribute 2 of the control record

PASS.TO.**programname**(.WINDOWS) in the PIX.CONTROL.F file. The graph type number should be the number of the multi-value in attribute 2 of this record.

Specifying one of the graph options tells the PASS.TO routine that, after transferring and importing the data and changing the column widths, it should also plot the ranges and draw the relevant graph for the user automatically. Once plotted, the user can then select other graph types, change the data for 'what if' analysis etc., as per normal in the spreadsheet.

The spreadsheet product need not already be started on your desktop. HOSTACCESS will determine if the spreadsheet program is already running and, if not, it will start it accordingly. Data will then be transferred and imported automatically whilst the spreadsheet is running (in the foreground if HOSTACCESS started it, or in the background if it was already running).

If you are using [EASY.ACCESS](#), the full list of graphs is shown after selecting to pass data to the spreadsheet product.



Let 's say that you want to produce an EXCEL spreadsheet containing sales information from the file SALES, with total sales per area, then plot a piechart. This can be done by just typing at TCL:

**GET SALES BREAK-ON AREA TOTAL SALES // BY AREA (ID**

to extract the data from the SALES file and then typing at TCL:

**PASS.TO.EXCEL (P**

This will take the user into a new EXCEL spreadsheet. [Using DDE](#), the data will be automatically imported and a piechart drawn. Because a new spreadsheet is created, the user can save this to any normal spreadsheet file as required.

Although this example uses EXCEL, it is equally applicable to all other supported spreadsheet products.

PASS.TO.programname is a verb interpreted by the PIX.DRIVER routine. It is processed by the subroutine PIX.PASS.TO.programname which may be called by any PICK program.

PICK developers who wish to integrate their applications with PASS.TO.programname should call the PIX.PASS.TO.programname subroutine directly as follows:

**CALL PIX.PASS.TO.programname (COMMAND,GRAPH.TYPE, ERROR)**

Where:

<b>programname</b>	Is EXCEL, LOTUS, SUPERCALC, SYMPHONY, or QUATTRO.
<b>COMMAND</b>	Should be null. The command is read from attribute 3 of the control record PASS.TO.programname(.WINDOWS) in the <a href="#">PIX.CONTROL.F</a> file.
<b>GRAPH.TYPE</b>	As above (graph type)
<b>ERROR</b>	Because this routine relies on the existence of certain internal control files, any of these not found will result in the user seeing an error message and this variable will be set to TRUE. This can be used to determine if the PASS.TO.programname was unsuccessful. Your application should take appropriate actions. Under most circumstances this error will not be set.

The following example uses PIX.PASS.TO.EXCEL but is equally applicable to all other supported spreadsheet products.

```
*
* Program to automate taking sales figures from the SALES
* file out of PICK using ACCESS statements via the GET
* process and then to plot an EXCEL Stacked graph.
*
* GET called with D and I options, DET-SUPP and ID-SUPP
*
CALL PIX.GET("SALES BREAK-ON AREA TOTAL SALES","BY AREA","DI","",ERROR)
*
CALL PIX.PASS.TO.EXCEL("", "S",ERROR) ; * "S"tacked graph
*
END
```

**Note** QUATTRO PRO has 2 anomalies (at time of press), 1 of which is the 128 byte limit on DDE commands. This means that each DDE instruction causes the QUATTRO PRO screen to flash. The second is that FILE OPEN forces QUATTRO PRO to the foreground application which shows the user everything that is happening.

These routines include PASS.TO.WORD and PASS.TO.WORDPERFECT.

The [PASS.TO](#) routines that relate to word processing products pass data created as a result of the last GET command down to a temporary DOS file. The word processing product is automatically started or called and the data is merged into the WORD or WordPerfect document **doc\_name**. The user will be left looking at the final merged documents on the screen and can then choose to PRINT it after verifying that it is correct.

PIXELnnn.TMP (where 'nnn' is the PICK user 's port number) is the name of the DOS file created in the HOSTACCESS run time DOS directory containing the mailmerge data.

- Starting the PASS.TO Word Processor Routine.
- PASS.TO Word Processor Examples.
- PASS.TO Word Processor Application Integration.
- PASS.TO Word Processor Example Program.

The PASS.TO word processor routine may be invoked from the TCL command line with the following syntax:

**PASS.TO.programname doc\_name**

Where:

<b>PASS.TO.programname</b>	Is the TCL command PASS.TO.WORD or PASS.TO.WORDPERFECT.
<b>doc_name</b>	A document name that must exist as a <b>programname</b> mailmerge format document. HOSTACCESS will call <b>programname</b> , retrieve the document <b>doc_name</b> and pass the necessary keystrokes to merge that document with the mailmerge file created by PASS.TO. <b>programname</b> . The user can then Save, Print or modify this document as required. If <b>doc_name</b> is not specified, it is prompted for.  The word processing product need not already be started on your desktop, HOSTACCESS will determine if it is already running and, if not, it will start it accordingly. Data will then be transferred and imported automatically in the background.

Let 's say that you want to create a WORD mailmerge file containing the key, name, address and ages of everybody in the STAFF file who has retired. This can be done by just typing at TCL:

**GET STAFF NAME AGE ADD1 ADD2 TOWN COUNTY %% RETIRED**

to extract the data from the STAFF file using a previously saved list called RETIRED and then typing at TCL:

**PASS.TO.WORD RETIRE.DOC**

which will create a mailmerge format DOS file called PIXELnnn.TMP containing the fields NAME, AGE, ADD1, ADD2, TOWN and COUNTY from the records extracted by the previous GET statement. It will then take the user into WORD. The document RETIRE.DOC will be retrieved and automatically mailmerged with the file PIXEL.TMP.

The user can then Save, Print or modify this document as required.

The above example uses PASS.TO.WORD, but it is equally applicable to WordPerfect.

**Note:** the Word document must have the field names in this document that match the dictionary names that are going to be used. Please refer to your Word User Guide for details of field names and Mailmerge.


PASS.TO.programname is a verb interpreted by the PIX.DRIVER routine. It is processed by the subroutine PIX.PASS.TO.programname which may be called by any PICK program.

PICK developers who wish to integrate their applications with PASS.TO.programname should call the PIX.PASS.TO.programname subroutine directly as follows:

**CALL PIX.PASS.TO.programname ("",DOC.NAME,ERROR)**

Where:

- |                    |   |
|--------------------|---|
| <b>programname</b> | Is either WORD or WORDPERFECT.  |
| <b>DOC.NAME</b>    | As `doc_name` above, should be a valid mailmerge document and must be specified.  |
| <b>ERROR</b>       | Because this routine relies on the existence of certain internal control files, any of these not found will result in the user seeing an error message and this variable will be set to TRUE. This can be used to determine if the PASS.TO.programname was unsuccessful. Your application should take appropriate actions. Under most circumstances this error will not be set. |

Click here  for help setting up WORD mailmerge documents.

The following example program uses PIX.PASS.TO.WORD but is equally applicable to PIX.PASS.TO.WORDPERFECT.

```
*
* Program to automate taking everybody from the STAFF file
* who is MALE aged between 30 and 40 and creating a
* WORD mailmerge file with ID, name, address and age.
* WORD will then be called and the mailmerge file
* 'PIXELnnn.TMP' will be automatically mailmerged with the
* document REVIEW.DOC.
*
OUTPUT = "STAFF NAME ADD1 ADD2 ADD3 AGE"
*
SELECTION = 'WITH AGE >= "30" AND WITH AGE <= "40"'
*
CALL PIX.GET(OUTPUT,SELECTION,"","",ERROR)
*
CALL PIX.PASS.TO.WORD("","C:\WPDATA\REVIEW.DOC",ERROR)
*
END
```

**Note:** WordPerfect 5.1 does not support being a DDE server. HOSTACCESS therefore sends the keystrokes to WordPerfect to automate the MERGE. The only disadvantage is that:

- If WordPerfect is already started, it must have no documents already open and must have a window title of WordPerfect [Document1 - unmodified]
- The user will see every keystroke as it is replayed and the corresponding screen updates
- Wordperfect *must* remain as the *active* foreground application (not minimized) and *cannot* be interrupted. Unlike DDE, keystrokes can only be sent to the currently active application - if the user changes this, the mailmerge will fail.

WordPerfect 6 does support [DDE](#).

CREATE.WORD.DICT creates a DOS file containing a list of dictionaries for a given PICK file. The DOS file can then be used by WORD for Windows users to assist in the creation of mailmerge documents. The DOS file can be attached from Word's Print Merge menu. Clicking on the insert field button in WORD will then reveal, as a select list, all the given fields in this file

CREATE.WORD.DICT may be invoked from the TCL command line as follows:

**CREATE.WORD.DICT PICK.FILE.NAME, DOS.FILE.NAME**

Where:

**PICK.FILE.NAME** Is the actual PICK file name in this account to use for the list

**DOS.FILE.NAME** Is the full DOS path and DOS file name to store the result in. Normally this path will be where WORD has easy access to it.

Click here  for an example.

To create a DOS file called STAFF.LST in the directory D:\WINWORD, use:

**CREATE.WORD.DICT STAFF D:\WINWORD\STAFF.LST**

Using Print Merge from Words File menu and selecting this file as the ATTACH HEADER file you can use the Insert Field button to select any fields from the STAFF file you wish to use in the merge.

**Note** WORD does not support dots (.) in dictionary/field names. This routine automatically deals with this by replacing all dots with underscores, as appropriate.



These routines include [PASS.TO.DIF](#) and [PASS.TO.DOS](#).

The [PASS.TO](#) routines that relate to DOS files take the data produced by the last GET statement and create a DOS file, either in DIF format (PASS.TO.DIF), or in a specified format (PASS.TO.DOS).

Many of these common DOS formats can be used by many DOS products to import data. By using a combination of [GET](#) and PASS.TO.DIF or PASS.TO.DOS, ou can integrate PICK data with most DOS products.

**Note** If you find that you are using PASS.TO DOS file routines and are frequently having to manually import the file into your DOS software then please contact PIXEL. We may be able to provide you with the routines and tools to automate the import process for yourDOS software.

PASS.TO.DIF may be invoked from the TCL command line with the following syntax:

**PASS.TO.DIF {dos\_file}**

Where:

**PASS.TO.DIF** Is the TCL command.

**dos\_file** Is an optional DOS file name (including path and directory if required) where you wish the DIF file to be stored. If null, the DIF file will be created in the current HOSTACCESS directory with PIXELnnn.TMP (where nnn is the PICK user 's port number) as the file name.



PASS.TO DIF Examples.



Application Integration.



Example Program.

Let 's say you want to create a DIF file containing the name, address and ages of everybody in the STAFF file who has retired. This can be done by just typing at TCL: -

**GET STAFF NAME AGE ADD1 ADD2 TOWN COUNTY %% RETIRED**

To extract the data from the SALES file using a previously saved list called RETIRED and then typing at TCL:

**PASS.TO.DIF RETIRE.DIF**

This will create a DIF format DOS file called RETIRE.DIF (in the run-time HOSTACCESS directory) containing the fields NAME, AGE, ADD1, ADD2, TOWN and COUNTY from the records extracted by the previous GET statement.

PASS.TO.DIF is a verb interpreted by the PIX.DRIVER routine. It is processed by the subroutine PIX.PASS.TO.DIF which may be called by any PICK program.

PICK developers who wish to integrate their applications so that DIF files are created automatically should call the PIX.PASS.TO.DIF subroutine directly as follows:

**CALL PIX.PASS.TO.DIF(DOS.FILE,ERROR)**

Where:

**DOS.FILE** Is an optional DOS file name (including path and directory if required) where you wish the DIF file to be stored. If null, the DIF file will be created in the current HOSTACCESS directory with PIXELnnn.TMP (where nnn is the PICK user 's port number) as the file name.

**ERROR** Since this routine relies on the existence of certain internal control files, any of these not found will result in the user seeing an error message and this variable will be set to TRUE. This can be used to determine if the PASS.TO.DIF was unsuccessful. Your application should take appropriate actions. Under most circumstances this error will not be set.

```
*
* program to automatically create a DIF file STOCK.DIF
* containing the product code (key), description, buy price
* sell price, and current stock quantity for all stock
* with a buy price of greater than 100 pounds.
*
OUTPUT = "STOCK DESC BUY SELL QTY"
*
SELECTION = 'WITH BUY > "100.00"'
*
CALL PIX.GET(OUTPUT,SELECTION,"","",ERROR)
*
CALL PIX.PASS.TO.DIF("C:\DBASE\STOCK.DIF",ERROR)
*
END
```

The DOS file formats provided for with PASS.TO.DOS include:

- Comma delimited.
- Quote delimited.
- Space delimited.
- Fixed length.
- User definable.



Starting PASS.TO.DOS.



Examples.



Applications Integration.



Example Program.

PASS.TO.DOS may be invoked from the TCL command line with the following syntax:

**PASS.TO.DOS {dos\_file} {delim\_1} {delim\_2} {(option)}**

Where:

**PASS.TO.DOS** Is the TCL command. If no other parameters are specified then a file called PIXELnnn.TMP (where 'nnn' is the PICK user's port number) is created in the current HOSTACCESS directory as a comma delimited file using the data from the last [GET](#) statement.

**dos\_file** Is an optional DOS file name (including path and directory if required) where you wish the created file format to be stored. If null then the file will be created in the current HOSTACCESS directory with the name PIXELnnn.TMP (where 'nnn' is the PICK user's port number).

**delim\_1** If specified, delimiter 1 is the character(s) to put in front of every dictionary output value.

**delim\_2** If specified, delimiter 2 is the character(s) to put after every dictionary output value.

For instance, to obtain a quote delimited file, both **delim\_1** and **delim\_2** should be quotes. Setting the **option** parameter to 1 would achieve the same result.

If using **delim\_1** and **delim\_2** when using PASS.TO.DOS from TCL you *must* specify the **dos\_file** parameter.

**(option)** By setting this option to 1,2,3 or 4, the DOS file will be created in the format as follows:

1. Comma delimited (default)
2. Quote delimited
3. Space delimited
4. Fixed Length

Option 4 uses the length held in each dictionary to determine the fixed length of each value. Each value is padded out in a mask of spaces to that length, either to the right or the left, depending on the dictionary justification.

If any of the delimiters to be used are found in the actual data then those characters are converted to an exclamation mark "!". This is to ensure that the alignment of the values remain valid.

Let's say you want to create a comma delimited file containing the name, address and age of everybody in the STAFF file who has retired. This can be done by just typing at TCL:

**GET STAFF NAME AGE ADD1 ADD2 TOWN COUNTY %% RETIRED**

to extract the data from the STAFF file using a previously saved list called RETIRED and then typing at TCL:

**PASS.TO.DOS C:\TEMP\RETIRE.TXT**

This will create a comma delimited format DOS file called RETIRE.TXT containing NAME, AGE, ADD1, ADD2, TOWN and COUNTY from the previous GET statement.

PASS.TO.DOS is a verb interpreted by the PIX.DRIVER routine. It is processed by the subroutine PIX.PASS.TO.DOS which may be called by any PICK program.

PICK developers who wish to integrate their applications, so that these delimited or fixed length DOS files are created automatically, should call the PIX.PASS.TO.DOS subroutine directly as follows:

**CALL PIX.PASS.TO.DOS(DOS.FILE,DELIM1,DELIM2,OPTIONS,ERROR)**

Where:

**PASS.TO.DOS** Is the TCL command. If no other parameters are specified then a file called PIXELnnn.TMP (where 'nnn' is the PICK user's port number) is created in the current HOSTACCESS directory as a comma delimited file using the data from the last GET statement.

**dos\_file** Is an optional DOS file name (including path and directory if required) where you wish the created file format to be stored. If null then the file will be created in the current HOSTACCESS directory with the name PIXELnnn.TMP (where 'nnn' is the PICK user's port number).

**delim\_1** If specified, delimiter 1 is the character(s) to put in front of every dictionary output value.

**delim\_2** If specified, delimiter 2 is the character(s) to put after every dictionary output value.

For instance, to obtain a quote delimited file, both **delim\_1** and **delim\_2** should be quotes. Setting the **option** parameter to 1 would achieve the same result.

If using **delim\_1** and **delim\_2** when using PASS.TO.DOS from TCL you *must* specify the **dos\_file** parameter.

**(option)** By setting this option to 1,2,3 or 4, the DOS file will be created in the format as follows:

1. Comma delimited (default)
2. Quote delimited
3. Space delimited
4. Fixed Length

Option 4 uses the length held in each dictionary to determine the fixed length of each value. Each value is padded out in a mask of spaces to that length, either to the right or the left, depending on the dictionary justification.

If any of the delimiters to be used are found in the actual data then those characters are converted to an exclamation mark "!". This is to ensure that the alignment of the values remain valid.

**ERROR** Since this routine relies on the existence of certain internal control files, any of these not found will result in the user seeing an error message and this variable will be set to TRUE. This can be used to determine if the PASS.TO.DOS was unsuccessful. Your application should take appropriate actions. Under most circumstances this error will not be set.

```
*
* program to automatically create a fixed length format DOS
* file called STOCK.FIX containing the product code (key),
* description, buy price, sell price, and current stock
* quantity for all stock with a buy price greater than
* 100 pounds.
*
OUTPUT = "STOCK DESC BUY SELL QTY"
*
```

```
SELECTION = 'WITH BUY > "100.00"'
*
CALL PIX.GET(OUTPUT,SELECTION,"","",ERROR)
*
CALL PIX.PASS.TO.DOS("C:\DBASE\STOCK.FIX","","",4,ERROR)
*
END
```



HOSTACCESS supports a large number of DOS and Windows products and formats. The tools and facilities provided allow simple and yet powerful integration of PICK data with virtually any product in the DOS and Windows world.

With the ever changing number of software products available it is impossible for HOSTACCESS to cover every user's preference for every DOS and Windows product and format.

Pixel guarantee to seriously consider any user requests for a more seamless integration with other DOS and Windows products. We feel that we owe a lot of HOSTACCESS's success to our willingness to listen to our users and provide enhancements that can then be shared with other users of the product.

To that end, you may find that other PASS.TO routines are available but are not obvious from the standard documentation. These will be documented in either loose-leaf addenda or in a READ.ME file.

If you require any assistance in respect of the PASS.TO routines, or in fact any of the HOSTACCESS facilities then please do not hesitate to contact your HOSTACCESS dealer directly by phone or with detail, by fax.

**Note** Pixel are happy to provide the full source code for any of the PASS.TO routines, should you need to modify them. The source programs provided with HOSTACCESS are normally encrypted to save space and are unreadable.

You can use the features described in this section, on all PICK systems, to:

- Open and close windows from TCL.
- Change screen size.
- Display images in HOSTACCESS.
- Install and update account.
- View control parameters.

**DB** An invaluable PICK Programmers toolkit for fast editing, compiling and running of programs and much more.

**CALCIT** A pop-up calculator written in PICK/BASIC but running at DOS speeds.

All of these routines are written in PICK/BASIC - please feel free to copy and modify them to suit your own host applications requirements.

All these routines are verbs interpreted by the PIX.DRIVER routine, and are processed by their own subroutines which can be processed by any PICK program and called from anywhere within your own applications. Upon completion, HOSTACCESS restores your application screens instantly.

PICK applications can use HOSTACCESS to open coloured windows on the screen instantly. The window is painted on top of the current screen with optional effects such as shadowing. PICK applications can be dramatically enhanced with slight changes to enable them to run in a window.

Once opened, the window takes all cursor addressing and text output to that window only. The window effectively becomes a mini screen with the top left corner of the window being addressed as column 0 row 0.

Windows can be closed instantly by using the CLOSE.WINDOW command. The underlying screen is instantly restored giving the appearance - the window just disappears.

Normally, opening and closing windows will be done directly from applications. These commands are useful however for highlighting the power of HOSTACCESS without having to resort to any form of programming.



Opening a window.



Closing a window.



Applications integration.

To open a window, enter the following TCL command (keeping spaces as shown):

**OPEN.WINDOW x,y,x,y foreground,background effects**

In all cases the comma ', ' needs to be specified

- x,y,x,y** Are the screen column (x) and row (y) co-ordinates for the window position. The first pair of x,y are the top left hand corner of the screen position and the second pair are the bottom right corner of the position. (Any border specified in 'effects' will be painted around the window *after* it has been drawn at these co-ordinates).
- foreground, background** These 2 parameters specify the color in which you require the window to be drawn. E.g. WHITE,RED will draw a red window with white text (WHITE on RED). Colors should be entered as their normal names, and may be prefixed by "LIGHT" for intense colors. Color names are : BLACK RED GREEN BROWN BLUE MAGENTA CYAN WHITE GREY YELLOW
- effects** Multiple effects can be added to the window, and may be as follows:  
SINGLE or DOUBLE: or SINGLE DOUBLE: or DOUBLE  
SINGLE: add border around window as a single or double line to the top and sides.  
SHADOW: draw a shadow to the right and below the window. Any text in the shadow will be low intensified.  
EXPLODE: when drawing the window, HOSTACCESS will draw from the center outwards giving an exploding effect.

To close the last opened window, instantly refreshing the screen, use the TCL command :

**CLOSE.WINDOW**

To close the last opened window, without refreshing the screen (leaving the window as normal text on the screen), use:

**CLOSE.WINDOW leave**

where **leave** is a literal.

Click here  for an example.

To open a green window with yellow text and run a LISTU in that window use:

```
OPEN.WINDOW 1,5,78,18 YELLOW,GREEN SINGLE,EXPLODE  
LISTU  
CLOSE.WINDOW
```

To open a blue window with white text and run a listing of the MD (VOC) in that use:

```
OPEN.WINDOW 5,5,70,15 WHITE,BLUE SINGLE,EXPLODE,SHADOW  
TERM 65,11  
LIST MD *A1 *A2 *A3 WITH *A1 = "D]"  
CLOSE.WINDOW  
TERM 79,23
```

OPEN.WINDOW and CLOSE.WINDOW are processed by the subroutines [PIX.OPEN.WINDOW](#) and [PIX.CLOSE.WINDOW](#). To integrate windowing, PICK programs use:

**CALL PIX.OPEN.WINDOW(POSITION,COLOUR,EFFECTS)**

Where **POSITION** as 'x,y,x,y' above, **COLOUR** as 'foreground,background' above, and **EFFECTS** are as 'effects' above. This is shown in the following example:

```
* Program to open 2 windows on the screen, the first to do
* a LISTU, the second to do a LIST MD
CALL PIX.OPEN.WINDOW("1,5,78,18","WHITE,RED","SINGLE")
EXECUTE "LISTU" ; INPUT REPLY
CALL PIX.OPEN.WINDOW("5,5,70,15","RED,CYAN","SHADOW,DOUBLE")
EXECUTE "TERM 65,11"
EXECUTE 'SORT MD *A1 *A2 *A3 WITH *A1 = "D"'
INPUT REPLY
EXECUTE "TERM 79,23"
CALL PIX.CLOSE.WINDOW("") ;* close 2nd window
CALL PIX.CLOSE.WINDOW("") ;* close 1st window
END
```

You can change the current HOSTACCESS screen size directly from the TCL prompt, using the [SCREEN.MODE](#) command. The screen mode can also be changed from HOSTACCESS 's configuration menus for each session. The mode can also be changed directly from **procs** and **programs** by calling the relevant AiF sequence.

This will reset the current HOSTACCESS session, i.e. all windows, backpages etc. will be lost. Changing between 132 and 80 column mode does preserve the current screen text for compatibility with certain terminal types.

**Note:** All modes are supported under Windows, regardless of card type.

To change the screen size from the TCL command line, enter::

**SCREEN.MODE {mode}**

Where **mode** is a mode number which have the following effects:

- 0 Change to 132 x 24 screen. This is only valid if your PC and monitor can support 132 column mode on screen.
- 1 Change to 80 x 24 screen. This is the default mode.
- 2 Change to 80 x 42 screen (EGA cards only).
- 3 Change to 80 x 49 screen (VGA cards only).
- 5 Change to 132 x 25 screen (VGA cards only).
- 6 Change to 80 x 25 screen (VGA cards only).
- 7 Change to 80 x 43 screen (EGA cards only).
- 8 Change to 80 x 50 screen (VGA cards only).
- 9 Change to 40 x 24 screen (all cards).
- 10 Change to 40 x 25 screen (all cards).

For example, to set your screen size to 80 columns by 40 rows (if you have a PC that supports VGA), enter:

**SCREEN.MODE 4**



**SCREEN.MODE** is processed by the subroutine **PIX.SCREEN.MODE**.

Remember, this will reset the current session. To preserve the session when changing modes from an application, call the [PIX.PUSH.ENVIRONMENT](#) subroutine prior to changing modes. The [PIX.POP.ENVIRONMENT](#) subroutine can be called to restore the session screen mode, windows, menus etc., upon completion.

To integrate other screen modes, PICK programs can call the [PIX.SCREEN.MODE](#) subroutine as follows:

```
CALL PIX.SCREEN.MODE(MODE)
```

Where **MODE** is as **mode** above.

```
*
* Program to preserve the current session, change the screen
* mode to 50 lines (VGA only) and do a LIST MD. The program
* will prompt for input before restoring the session back
* to its original state.
*
CALL PIX.PUSH.ENVIRONMENT ; * save session
*
CALL PIX.SCREEN.MODE(3) ; * 50 rows x 80 columns
*
EXECUTE "TERM 79,49" ; * tell PICK to match screen size
*
EXECUTE 'LIST MD WITH *A1 = "D]" *A1 *A2 *A3'
*
INPUT WAIT: ; * give user a chance to read
*
EXECUTE "TERM 79,23" ; * set PICK
*
CALL PIX.POP.ENVIRONMENT ; * restore screen mode, text etc.,
*
END
```

**MOUSE.TEST** is a small PICK/BASIC utility that can be run from TCL, showing how the mouse can be used with HOSTACCESS on your PC. We recommend that you also look at the [PIX.MOUSE.TEST](#) program to see how the code is structured.

MOUSE.TEST allows the user to move and click the mouse pointer over the current screen. Pressing and releasing the mouse button(s) cause a small box to appear around the position where the mouse button is pressed. The status line is also used to indicate which button you have pressed along with the column,row co-ordinates it was pressed at.

This program will only run if your PC supports a mouse through a mouse driver loaded into memory, or if you have Windows with a mouse. MOUSE.TEST will report an error message if no mouse is found.

This facility demonstrates just how powerful the mouse can be when your own software applications are in control.

MOUSE.TEST may be invoked from the TCL command line as follows:

**MOUSE.TEST**

The user should see a block cursor on the screen and, by moving the mouse, this cursor will move accordingly indicating the position of the mouse. By clicking any of the buttons on the mouse, a small box is drawn and the status line will be updated to indicate what has happened.

By dragging the mouse whilst holding down a button, the status line is continuously updated. The co-ordinates are also displayed indicating where you released the mouse button.

One or more mouse buttons may be held down at the same time.

Press <RETURN> on the PC's keyboard to stop the MOUSE.TEST program and reset the screen.

MOUSE.TEST is a subroutine that calls the main mouse driving subroutines. The subroutines are [PIX.MOUSE.AVAILABLE](#), [PIX.MOUSE.ON](#), [PIX.MOUSE.RESPONSE](#) and [PIX.MOUSE.OFF](#).

Two programs are provided with HOSTACCESS that fully utilize the mouse. These are [PIX.MOUSE.TEST](#) and [PIX.CALC](#). It is recommended that you examine these programs.

```

*
* This program will activate the mouse and allow any of
* the mouse buttons to be used. An '*' will be printed
* anywhere on the screen that the mouse is pressed.
*
* Pressing <return> will cause BUTTON to return as false
* which causes this program to stop.
*
MOUSE.FOUND = 0
*
CALL PIX.MOUSE.AVAILABLE(MOUSE.FOUND)
*
IF NOT(MOUSE.FOUND) THEN
    PRINT "MOUSE NOT FOUND - ASK THE CAT!"
    STOP
END
*
CALL PIX.MOUSE.ON("ALL,CONTINUOUS")
*
OLDX = 0 ; OLDY = 0
*
LOOP
    CALL PIX.MOUSE.RESPONSE (KEYBOARD,BUTTON,XCORD,YCORD)
WHILE BUTTON DO
    PRINT @(OLDX,OLDY):" ":
    PRINT @(XCORD,YCORD):"*":
    OLDX = XCORD ; OLDY = YCORD
REPEAT
*
CALL PIX.MOUSE.OFF
*
END

```

To display images in HOSTACCESS for Windows, invoke WIN.IMAGE from TCL as follows:

**WIN.IMAGE imagename {title} {scale} {(options)}**

Where:

<b>imagename</b>	Is the DOS image file name (including drive and path where necessary) of the image to be displayed. The imagename must be a valid PCX image file.
<b>scale</b>	Is the percentage of the size of the original image to be displayed. By default, 100 percent is used.
<b>title</b>	Is an optional title that will be displayed in the Application Name bar. If omitted, "Image [imagename]" will be used.
<b>(options)</b>	May be: <b>F</b> : specifies that the image is to fit into the size of the image display window. If specified, this means that if the user changes the image display window size, the image will automatically be scaled to fit as best as possible. <b>C</b> : closes the displayed image window named in <b>title</b> .

For example, to display a full colour photograph called DJB.PCX at full size, with a title of DPick:

**WIN.IMAGE C:\IMAGES\DJB.PCX "DPick"**

and to close this image:

**WIN.IMAGE "DPick" (C**

WIN.IMAGE is processed by the subroutine [PIX.WIN.DISPLAY.IMAGE](#).

Remember, unlike Image for DOS, screen updates and normal input can be achieved whilst an image is being displayed. It is up to the application or user to close any open images.

Multiple images can be displayed on screen at the same time.

To integrate images, PICK programs can call the PIX.WIN.DISPLAY.IMAGE. subroutine as follows:

```
CALL PIX.WIN.DISPLAY.IMAGE.(IMAGE.NAME, TITLE, SCALE, STATE, OPTIONS, STATUS)
```

Where:

<b>IMAGE.NAME</b>	Is the DOS image file name (including drive and path where necessary) of the image to be displayed. The imagename must be a valid PCX image file.
<b>TITLE</b>	Is an optional title that will be displayed in the Application Name bar. If omitted, "Image [imagename]" will be used.
<b>SCALE</b>	Is the percentage of the size of the original image to be displayed. By default, 100 percent is used.
<b>OPTIONS</b>	May be: <b>F</b> : specifies that the image is to fit into the size of the image display window. If specified, this means that if the user changes the image display window size, the image will automatically be scaled to fit as best as possible. <b>C</b> : closes the displayed image window named in <b>title</b> .
<b>STATE</b>	Specifies how you want the image displayed, as follows: 1: Activates and displays window. 2: Activates and minimizes window. 3: Activates and maximizes window. 7: Displays and minimizes but does not make active.
<b>STATUS</b>	0: Image program started successfully 1: <b>IMAGE.NAME</b> not found in specified path. 2: IMAGE.EXE was not started. 3: <b>C</b> option specified but no <b>IMAGE.NAME</b> or <b>TITLE</b> specified.

```
* Display 2 images one after the other on the screen followed by  
* the same images together on the same screen, of different sizes.  
*
```

```
IMAGE1 = "C:\HOSTACCESS\DICKPICK"  
IMAGE2 = "C:\HOSTACCESS\LIBERTY"
```

```
CALL PIX.WIN.DISPLAY.IMAGE (IMAGE1, "", "", "", "", STATUS)  
CALL PIX.WIN.DISPLAY.IMAGE (IMAGE2, "", "", "", "", STATUS)
```

```
*
```

```
* Next image at 50 percent of size  
CALL PIX.WIN.DISPLAY.IMAGE (IMAGE1, "", 50, "", "", STATUS)  
CALL PIX.WIN.DISPLAY.IMAGE (IMAGE2, "", 50, "", "", STATUS)  
END
```



PIX.INSTALL.HOST is a TCL routine. This routine is the same as that used at Host Programs Installation.

It should only be used by the systems administrator to upload Host programs. It can also be used to update other accounts on the system with the HOSTACCESS commands and files.

You can also use this program to re-upload any or all of the Host Programs and utilities (for example, if you want to restore an original program), as follows:

1. Delete any program, file, record or TCL command that you require to be re-uploaded.
2. Type PIX.INSTALL.HOST at TCL from the original account HOSTACCESS was installed into. Load the floppy disk when requested.
3. Select the option that will upload the program(s) or commands you require.

HOSTACCESS will not upload any records in files if they already exist on the host, unless requested. HOSTACCESS will then upload all or any missing records.

Any MD (VOC) entries that exist will be overwritten but the original will be copied to mdname.ORIGINAL.

You can view the control parameters that HOSTACCESS is currently using, with the TCL **ENVIRONMENT** command (calling the subroutine [PIX.ENVIRONMENT](#)).

The information displayed is actually retrieved from the file [PIX.CONTROL.F](#) from the record called ENVIRONMENT.

ENVIRONMENT only displays the information from this record. We have deliberately not provided a program to update this record since it should only be updated with extreme care. It can however be changed using the standard editor if required.

The technical information regarding the display from the ENVIRONMENT routine is as follows.

- Machine Type.
- Account Name.
- Reformat Type.
- Max Item Size.
- System support for char(255).

This is the identifier indicating the machine type that the HOSTACCESS Host Programs were originally installed onto. This should not be changed. If you wish to move the programs between different machine types, you should re-install the Host Programs directly from the PC.

You can do a full account save of the master account containing the HOSTACCESS Host Programs from one machine and restore it with the same name to another machine of the SAME type.

This is the account name that the HOSTACCESS Host Programs were originally installed into. This should not be changed. Use either PIX.INSTALL.HOST to update another account or re-install the Host Programs into a new account.

This is an internal flag to tell HOSTACCESS 's `GET` routine how to function on a multiple range of machines. This is set automatically and should *never* be changed by users.

This is used to tell HOSTACCESS 's file transfer facility when to split large DOS records being uploaded onto PICK. HOSTACCESS will automatically split around this size but will split logically on the nearest attribute. This is a global parameter for all users and is set at install time to 30000 bytes for standard PICK systems. This is set to 5 megabytes for systems that are known to have an unlimited item size.

This size may be increased or decreased as required. Setting it too high may cause the PICK workspace to be exceeded even if the item size is within the limit of the system.

This value should be either 0 or 1. It is set automatically at install time to match host "machine type" and will be :

- 0 Host system does NOT support char(255).
- 1 Host does support char(255).

Many standard PICK stems do not allow character 255 to be used at all since it is used as an internal segment mark. Since many DOS files that you may wish to store or print on the PICK host may contain char(255), HOSTACCESS has to strip them out if it is not supported. This means that some graphics files may not print correctly from DOS to the PICK spooler. Systems that do support character(255) can both print and store all characters from DOS.

**Warning:** This value should *never* be changed unless it is known how the system treats character(255). Setting it to 1 incorrectly may corrupt your system 's database.

**TERMITE.DEMO** is a TCL command that will demonstrate some of the features available for PICK users using HOSTACCESS.

Invoke this command to see these features (presented within a menu) and we are sure that you will be pleasantly surprised by the power of PICK BASIC programs combined with HOSTACCESS.



DB is a utility designed to make life easier for any PICK/BASIC programmer. It uses a powerful pull-down menu style to guide the programmer through most of the facilities needed when developing programs. The menuing system uses a format similar to that found in powerful DOS application development tools.

The idea behind DB, in addition to being a utility, is to show just what is possible when you combine the 2 worlds of PICK and HOSTACCESS. You will find that DB is written entirely in PICK/BASIC!

If a mouse driver is loaded on the PC, or if Windows is running, the user can move around the menus with either the mouse or keyboard.

The menuing system used by DB is fully re-configurable by the user. The text and commands used by DB are held in the [PIX.CONTROL.F](#) file in the records DB.TEXT and DB.COMMANDS. All you have to do when altering these records is to ensure that the multi-values in the 2 records remain in alignment. Existing commands may be replaced with the user's preference. For instance, where DB calls the standard TCL 'ED' verb, the user may want to call their own preferred editor.

- Starting DB.
- Using DB.
- Applications Integration.

DB may be invoked from the TCL command line as follows:

**DB {file} {program}**

Where:

**file** Is the optional PICK file name where the program you want to work on resides. This can be entered/altered from within DB [Program>Change File] option.

**program** Is the optional PICK item name that you want to work with. This can be entered/altered from within DB [Program>Change Program] option.



Using DB.

Using DB is very easy. Use the arrow keys to highlight the menu option you want to select and activate by pressing <RETURN>. Pressing <ESCAPE> from lower level menus returns you up one level. Press <ESCAPE> from the top level menu to EXIT DB. Using a mouse, if available, is a simple matter of clicking on the required menu and option. To exit DB with a mouse, simply click outside of the menu area.

Upon exiting DB, the original TCL/application screen is instantly restored allowing DB to be called from virtually anywhere.

The status line normally shows the currently active file and program name. This status display can be disabled/enabled from the CONFIG menu.

DB remembers the last 20 files and programs the user has been working with. The number 20 can be increased/decreased from the CONFIG menu. Any one of the last 20 programs can be reselected to become the currently active one by selecting 'choose from stack' on the Program menu.

DB is supplied with 6 top level menus and these are basically used for the following purposes:

- TOOLKIT - Menu 1.
- Program - Menu 2.
- Printing - Menu 3.
- Special - Menu 4.
- DOS - Menu 5.
- Config - Menu 6.

Click here  for information on applications integration.

From here the user can select what process is required to run on the currently active program, i.e. edit, compile, format etc.

Any one of the options from the TOOLKIT menu can be selected by pressing the appropriate function key from any other menu whilst within DB.

JET is normally provided as one of the editing facilities from DB and can be used if JET is installed on that system. If the user prefers, the JET command in the DB.COMMANDS record ([file PIX.CONTROL.F](#)) can be modified to another editor of your choice.

This menu is used if the user wishes to change the current program being worked on. Three main options are provided to change file, change program or select from the stack of the last 20 programs. A fourth option on the menu allows you to view information on the current program, i.e. size, last compiled etc.

By default this menu has only one option that processes a BLIST file program (P). Other options may be added/amended to suit the user/system requirements by modifying the [PIX.CONTROL.F](#) file records, DB.TEXT and DB.COMMANDS accordingly.

As per menu 3, the options here may be added to/amended to suit the user/system requirements. Any items added to this list are EXECUTEd by DB when selected. Upon exiting the EXECUTEd routine, the user is returned to DB.

As per menus 3 and 4, the options here may be added to/amended to suit the user/DOS requirements. Any items added to this list are passed down to DOS to be processed. Upon exiting the DOS application, the user is returned to DB.



From here, the user can configure DB to their own requirements. The size of the stack can be changed (the number of programs to remember), the status line display can be toggled on/off.

The user can also turn WINDOWing on/off from this menu. Selecting windows to ON tells DB to attempt to run the commands from TOOLKIT menu in a window. DB sets the TERM width and depth of the port to match the window size, so the user has to be sure that the commands selected can be processed through a WINDOW.

The user can also optionally change the colours used by DB for menus and windows.

The user can also select to view the PC fonts table, which shows which characters are available through HOSTACCESS on this PC.

DB is a verb interpreted by the PIX.DRIVER routine. It is processed by the subroutine PIX.DB which may be called by any PICK program.

To call DB from a program use a program statement as follows:

**CALL PIX.DB**

CALCIT is a useful pop up calculator written in PICK/BASIC that can be called from TCL at any time. It can also be called as a subroutine from any PICK/BASIC program. When called, the calculator is drawn almost instantly on top of the current application/TCL screen. When the user exits the calculator, the underlying application/TCL screen is restored instantly to it's original state.

CALCIT is written entirely in PICK/BASIC!

CALCIT has built in support for a MOUSE. The BASIC program, through HOSTACCESS will detect the existence of the mouse and allow it to be used in addition to the keyboard.

- Starting CALCIT.
- Using CALCIT.
- Applications Integration.

CALCIT may be invoked from the TCL command line as follows:

**CALCIT {startvalue}**

Where:

**startvalue** Is an optional starting value to add into the calculator.

Using CALCIT is fairly straightforward. When using the keyboard, you just type the digits you require along with the relevant +, -, \*, /, % keys. If you have a mouse you can also move the mouse pointer to the relevant digit and click to select.

To obtain the result of a calculation, select '=' or <RETURN>.

In addition to the numeric and operand keys, the following keys are available :

- C** - Clear Value.
- M** - Save Value into Memory.
- R** - Recall value from memory.
- ?** - Help (shows current precision).

To exit the calculator press Q or click outside of the calculator window with the mouse.

CALCIT is a verb interpreted by the PIX.DRIVER routine. It is processed by the subroutine PIX.CALC which may be called by any PICK program.

Remember, CALCIT will restore your application screen so you can CALL PIX.CALC from virtually anywhere in your code.

PICK developers who wish to present their users with the CALCIT pop up calculator should call the PIX.CALCIT subroutine directly as follows:

**CALL PIX.CALC(START.VALUE,RETURN.VALUE)**

Where:


**START.VALUE** Optional starting value your application can send to feed into the calculator. Set this variable to null if no start value required.

**RETURN.VALUE** The result of the user 's last calculation, which your application may use, as required.

You can use HOSTACCESS's Programmer's TOOLKIT to exploit the power of library routines and screen manipulation features.

Using HOSTACCESS you can create a sophisticated, application driven workstation. While continuing to support existing applications unchanged through industry standard terminal emulations, you can also introduce a PC style user interface to host applications with colour, windows, pop-down menus and many more features.

The Programmer's TOOLKIT is implemented via a set of [PICK BASIC](#) subroutines.

Click here  to find out how to produce a more sophisticated user interface, giving a true GUI (Windows) look and feel to your applications.

Click here  for Programmer Notes.

All routines are supported on all PICK systems. All routines are prefixed with 'PIX. ' and they are all CATALOGed on the host system.

All of our filenames are prefixed with PIX. and suffixed with .F and are either created by the Host Installation procedure or "on the fly" as and when required by the programs.

All of the applications and utility programs provided are invoked from commands which are processed by one program PIX.DRIVER. This evaluates what the user typed from the command line and then calls the appropriate subroutine after converting the command line into the required subroutine parameters. The Host Installation procedure loads the MD (or VOC) file with the appropriate commands for these programs and utilities.



Executing commands from PROCs.

If you are familiar with PROCs and are used to using these to run commands, you should note that you cannot directly execute HOSTACCESS host commands from within a PROC. This is because all of these HOSTACCESS host commands are processed by one program, PIX.DRIVER, which expects to be able to interpret the TCL command line.

To execute the HOSTACCESS host commands from within PROCs, you should use PROCs that build the command line (user input) for each verb, as in the following examples :

1. PROC to extract data from the STAFF file, in readiness to pass into a DOS product (say, LOTUS spreadsheet).

```
PQ  
S1  
IHGET STAFF BREAK-ON DEPT TOTAL SALARY // BY  
DEPT  
HPIX.DRIVER  
P
```


2. PROC to pass data (say, that extracted by the GET statement above) to LOTUS and draw a Stacked graph.

```
PQ  
S1  
IHPASS.TO.LOTUS (S)  
HPIX.DRIVER  
P
```












If you use PROCs in the above manner, you should consider using BASIC subroutine calls to the appropriate PIX.PROGS.F subroutine. All of the HOSTACCESS host commands are processed by subroutines that can be called with the relevant parameters. Click here [\[ \]](#) for a list of available subroutines.

The following topics cover all the PICK BASIC subroutines used in the Programmer 's TOOLKIT. Each subroutine name is prefixed with PIX.


Each subroutine description explains the syntax components, and provides an example of the subroutine's use.

Click here  for an alphabetical list and links to all these subroutines.

The subroutines are divided into the following areas:

-  Screen display optimization.
-  Menus.
-  Selection boxes.
-  Colours.
-  Graphics.
-  Mouse control.
-  PC Utilities and file transfer.
-  EASY.ACCESS, PASS.TO and GET.
-  Miscellaneous.
-  Windows integration.
-  DDE.



Click here  to return to the overview.

<a href="#">PIX.BOX.INPUT</a>	Create an input box
<a href="#">PIX.CALC</a>	Activate pop-up calculator.
<a href="#">PIX.CALL.DOS</a>	Call a DOS application from PICK.
<a href="#">PIX.CALL.DOS.PICK</a>	Transfer data from DOS to PICK.
<a href="#">PIX.CALL.PICK.DOS</a>	Transfer data from PICK to DOS.
<a href="#">PIX.CENTRE.TEXT</a>	Centre text.
<a href="#">PIX.CLOSE.MENU</a>	Close menus.
<a href="#">PIX.CLOSE.SELECTION.BOX</a>	Close a selection box.
<a href="#">PIX.CLOSE.WINDOW</a>	Close a window.
<a href="#">PIX.COLOUR.CONFIG</a>	Change colours.
<a href="#">PIX.COMMAND.STACK</a>	Activate command stacker.
<a href="#">PIX.CREATE.WORD.DICT</a>	Create a list of dictionaries.
<a href="#">PIX.CURSOR</a>	Select a cursor.
<a href="#">PIX.DB</a>	Activate database utilities.
<a href="#">PIX.DDE.CLOSE</a>	Close DDE link.
<a href="#">PIX.DDE.EXECUTE.MACRO</a>	Send commands to the server.
<a href="#">PIX.DDE.INITIALISE</a>	Initiate a DDE channel.
<a href="#">PIX.DDE.POKE</a>	Pass data to another Windows application.
<a href="#">PIX.DDE.REQUEST</a>	Retrieve data from another Windows application.
<a href="#">PIX.DISPLAY.IMAGE</a>	Display a PCX image file.
<a href="#">PIX.DOS.PICK</a>	Transfer a DOS file to a PICK item/PICK spooler.
<a href="#">PIX.DOS.PICK.ALL</a>	Transfer a DOS file into multiple items in a PICK file.
<a href="#">PIX.DRAW.BOX</a>	Draw a box.
<a href="#">PIX.DRAW.LINE</a>	Draw a line.
<a href="#">PIX.EASY.ACCESS</a>	EASY ACCESS
<a href="#">PIX.ENVIRONMENT</a>	Display environment.
<a href="#">PIX.ERASE.DOS.FILE</a>	Erase a DOS file.
<a href="#">PIX.EXISTS</a>	Check DOS for the existence of a file or directory.
<a href="#">PIX.EXIT.KEYS</a>	Load exit keys.
<a href="#">PIX.FREEZE.OFF</a>	FREEZE OFF.
<a href="#">PIX.FREEZE.ON</a>	FREEZE ON.
<a href="#">PIX.GET</a>	Extract data from a PICK file.
<a href="#">PIX.GET.COLOUR.CODES</a>	Set ANSI colours for printing
<a href="#">PIX.GET.MENU.RESPONSE</a>	Activate pull down menus.
<a href="#">PIX.GET.SBOX.RESPONSE</a>	Activate a selection box.
<a href="#">PIX.GET.SERIAL.NO</a>	Returns HOSTACCESS's serial number to the Host application.
<a href="#">PIX.GET.SLOT.NUMBER</a>	SLOT number.
<a href="#">PIX.LINE.INPUT</a>	Create a single line input.

<a href="#"><u>PIX.LOAD.MENU</u></a>	Create and load a menu.
<a href="#"><u>PIX.LOAD.SELECTION.BOX</u></a>	Load a selection box.
<a href="#"><u>PIX.LOCAL.PRINT.DEVICE</u></a>	Change print device.
<a href="#"><u>PIX.MESSAGE.LINE</u></a>	Display a message line.
<a href="#"><u>PIX.MESSAGE.WINDOW</u></a>	Display a message window.
<a href="#"><u>PIX.MOUSE.AVAILABLE</u></a>	Detect a mouse driver.
<a href="#"><u>PIX.MOUSE.OFF</u></a>	Turn off mouse event reporting.
<a href="#"><u>PIX.MOUSE.ON</u></a>	Activates the mouse.
<a href="#"><u>PIX.MOUSE.RESPONSE</u></a>	Turns on mouse reporting.
<a href="#"><u>PIX.MOUSE.TEST</u></a>	Mouse test program.
<a href="#"><u>PIX.OPEN.WINDOW</u></a>	Open a window.
<a href="#"><u>PIX.PASS.TO</u></a>	PASS.To routines
<a href="#"><u>PIX.PASS.TO.DIF</u></a>	Create a DOS file in DIF format.
<a href="#"><u>PIX.PASS.TO.DOS</u></a>	Create a DOS file in a specified format.
<a href="#"><u>PIX.PASS.TO.Spreadsheet.Product</u></a>	PASS.TO spreadsheet.
<a href="#"><u>PIX.PASS.TO Word Processing Product</u></a>	PASS.To Word Processing product.
<a href="#"><u>PIX.PICK.DOS</u></a>	Transfer a PICK item to a DOS file
<a href="#"><u>PIX.PICK.DOS.ALL</u></a>	Transfer multiple PICK items into a single DOS file.
<a href="#"><u>PIX.POP.ENVIRONMENT</u></a>	Restore session details.
<a href="#"><u>PIX.POP.SLOT</u></a>	Retrieve a screen of data.
<a href="#"><u>PIX.PROGRAM.FKEYS</u></a>	Program function keys.
<a href="#"><u>PIX.PUSH.ENVIRONMENT</u></a>	Save session details and start new session.
<a href="#"><u>PIX.PUSH.SLOT</u></a>	Save screen .
<a href="#"><u>PIX.SCAN.KEYS</u></a>	Turn on all keys on the DOS keyboard.
<a href="#"><u>PIX.SCREEN.DUMP</u></a>	Send the screen contents to DOS.
<a href="#"><u>PIX.SCREEN.MODE</u></a>	Sets the screen mode.
<a href="#"><u>PIX.SET.COLOUR</u></a>	Sets colours
<a href="#"><u>PIX.SHOW.PC.FONT</u></a>	Displays fonts.
<a href="#"><u>PIX.TCL</u></a>	Gives a TCL environment.
<a href="#"><u>PIX.TDUMP</u></a>	Transfer multiple PICK items to multiple DOS files.
<a href="#"><u>PIX.TERMITE.DIRECTORY</u></a>	
<a href="#"><u>PIX.TERMITE.INFO</u></a>	Return information about HOSTACCESS.
<a href="#"><u>PIX.TLOAD</u></a>	Transfer multiple DOS files to multiple PICK items.
<a href="#"><u>PIX.WIN.CHANGE.STATE</u></a>	Control the Windows state.
<a href="#"><u>PIX.WIN.DISPLAY.IMAGE</u></a>	Display a Windows image.
<a href="#"><u>PIX.WIN.RUN</u></a>	Start a Windows Program.
<a href="#"><u>PIX.WIN.RUNNING</u></a>	Determines whether a Windows application is running.
<a href="#"><u>PIX.WIN.SEND.KEYS</u></a>	Sends keys in DOS Keyboard stacker format to specified Windows application.
<a href="#"><u>PIX.WINDOW.EDITOR</u></a>	Creates a window for text editing.
<a href="#"><u>PIX.WINDOW.TITLE</u></a>	Displays title text.

The following BASIC subroutines allow you to optimise your screen display:

<a href="#"><u>Save session details and start new session.</u></a>	PIX.PUSH.ENVIRONMENT
<a href="#"><u>Restore session details.</u></a>	PIX.POP.ENVIRONMENT
<a href="#"><u>Save screen.</u></a>	PIX.PUSH.SLOT
<a href="#"><u>Retrieve a screen of data.</u></a>	PIX.POP.SLOT
<a href="#"><u>Open a window.</u></a>	PIX.OPEN.WINDOW
<a href="#"><u>Close a window.</u></a>	PIX.CLOSE.WINDOW
<a href="#"><u>Send the screen contents to DOS.</u></a>	PIX.SCREEN.DUMP
<a href="#"><u>PIX.FREEZE.ON</u></a>	PIX.FREEZE.ON
<a href="#"><u>FREEZE.OFF</u></a>	PIX.FREEZE.OFF
<a href="#"><u>Sets the screen mode.</u></a>	PIX.SCREEN.MODE

PIX.PUSH.ENVIRONMENT is a BASIC subroutine that can be called from your applications as follows:

#### **CALL PIX.PUSH.ENVIRONMENT**

This is a very powerful routine that tells HOSTACCESS to save everything to do with the current session onto a stack in memory and effectively start a new session with all facilities set back to HOSTACCESS 's defaults, i.e. colours, status line, new set of 16 slots, new set of 50 menus and selection boxes.

The reason for this feature is important and some **examples** where it might be used are as follows:

If your application wants to change from 80 to 132 column mode to run a report then a call to PIX.PUSH.ENVIRONMENT before changing modes would preserve the session onto a memory stack. When you want to go back to 80 column mode with your screen intact just call PIX.POP.ENVIRONMENT.

Applications that provide gateways out to other applications may need to be able to save and restore their own environment, especially if the called/gateway applications make use of other or the same HOSTACCESS AiF features.

PIX.POP.ENVIRONMENT is a BASIC subroutine that can be called from your applications as follows:

**CALLPIX.POP.ENVIRONMENT**

This routine is used to restore all session details that have previously been stored on the stack by [PIX.PUSH.ENVIRONMENT](#).

PIX.PUSH.SLOT is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.PUSH.SLOT**

This is a very powerful routine that tells HOSTACCESS to push the current screen and everything associated with it (cursor position, colour etc..) onto a stack in HOSTACCESS 's memory.

Your application can then do whatever it likes to the main screen confident in the knowledge that the original screen will be literally INSTANTLY restored as was, when PIX.POP.SLOT is called.

You can now call any other host application perhaps, you can allow the user to call TCL commands and much more. To restore the original screen, just pop it back, instantly. By stacking, this routine can be called many times, each time placing the current screen image on the top of the stack. [PIX.POP.SLOT](#) just takes the screen from the top of the stack.

Try making things instantly appear and disappear, try this :

**CALL FREEZE.ON**

**CALL PIX.PUSH.SLOT**

**EXECUTE "BLOCK-PRINT HELLO WORLD"**

**CALL PIX.FREEZE.OFF**

**INPUT WAIT**

**CALL PIX.POP.SLOT**

PIX.POP.SLOT is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.POP.SLOT**

This routine is used to retrieve a screen of data that has previously been stored on the stack by [PIX.PUSH.SLOT](#).

PIX.OPEN.WINDOW is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.OPEN.WINDOW(POSITION, COLOURS, EFFECTS)**

This program will open a window on screen at the position specified by the calling program with any colour combination and with any of the HOSTACCESS window effects. Windows are opened instantly no matter what the line speed. They disappear instantly as well by calling PIX.CLOSE.WINDOW to restore the screen image underneath.

All subsequent output from the host will be displayed in that window as if the top left of the window is seen by the host as PICK position 0,0. You can run help in windows and if you set your TERM depth and width to match the window size you can run ACCESS and other VERBS in the WINDOW.

If you specify a border then that border is put around a window of the size you specify.

The following subroutine parameters need to be passed or will be returned accordingly:

- POSITION** X,Y,X,Y where the first XY is top left and the second XY is the bottom RIGHT. These XY co-ordinates are PICK specific, i.e. 0,0 is the top left hand corner of the screen.If -1,-1 is passed as first X,Y co-ordinates, window will be centered on the screen.
- COLOURS** The names of the colours required for foreground and/or background e.g. 'WHITE,RED '.
- EFFECTS** All parameters as follows are optional and comma separated :
- 'SHADOW ' Shadow effect.
  - 'EXPLODE ' Bring window up with exploding effect.
  - 'NO CLEAR ' The text behind the window stays within the window.
  - 'SINGLE' Single line border around window.
  - 'DOUBLE' Double line border around window.
  - 'SINGLE DOUBLE' Single line border at top and bot, Double sides.
  - 'DOUBLE SINGLE' Double line border at top and bot, Single sides.

Click here  for an example.

**CALL PIX.OPEN.WINDOW ("5,5,40,10","RED,BLUE","SINGLE,SHADOW")**

Will open a window on the screen, red text on a blue background 36 chars wide and 6 rows deep. The window will have a SINGLE border around it and will have a shadow effect as well.

See also: [PIX.CLOSE.WINDOW](#)



PIX.CLOSE.WINDOW is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.CLOSE.WINDOW(OPTION)**

Subroutine to close currently open window.

This subroutine will close the currently open window on the screen. If **OPTION** is not null, the window is still closed but is left behind on the screen as text.

PIX.SCREEN.DUMP is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.SCREEN.DUMP(OUTPUT.DEVICE)**

HOSTACCESS can manipulate DOS printers and devices in many ways. This facility lets you send the current screen contents to any DOS device or DOS file name.

**OUTPUT.DEVICE** Can be any name you want the screen dump to be sent to, it could be LPT1, LPT2, COM1, COM2 if say you have a printer on one of those devices or it could be a DOS file name like, TEST.DOC, C:\DUMPS\SCREEN.DMP etc. If null the device is left unchanged and the screen dump will go to the current print device or file.

PIX.FREEZE.ON is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.FREEZE.ON**

This subroutine is a clever little way of making screens, data etc. appear *instantly* to users.

This subroutine will suspend all updates to the current screen until the next **FREEZE.OFF**. The hot-key ALT/U from HOSTACCESS will actually show the screen so far and turn freeze off. This is useful if you are debugging code. There are no performance benefits with this facility but the users perception of increased speed is amazing. Text is restored instantly when PIX.FREEZE.OFF is called.

If you call this routine to stop the user seeing things like compiles or batch jobs, it is recommended you periodically inform the user that something is going on, or else he might become suspicious and suspect that the PICK system has hung, or worse still, reboot his PC and go and make himself a cup of coffee!

PIX.FREEZE.OFF is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.FREEZE.OFF**

This program *instantly* reveals the information to the current screen sent since the last FREEZE.ON.

PIX.SCREEN.MODE is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.SCREEN.MODE(CODE)**

This program, when passed a parameter will set the current screen mode to the specified requirements. i.e. 132 columns or 50 lines. Some modes are only supported by certain VGA screens and or monitors. You need to ensure that you have specified the correct VGA card/mode type from the HOSTACCESS Configure menu before attempting to change modes to/from 132 columns.

Changing modes only affects the current session, but also causes the current session to be reset as if you had just come into HOSTACCESS, i.e. Colours, menus, selection boxes, etc., are all cleared from memory.

To preserve the current session before changing modes you should call PIX.PUSH.ENVIRONMENT before calling this routine. In that way, your current session including the current screen mode (80/24 etc.) and colours, menus, etc., will be restored instantly with a call to [PIX.POP.ENVIRONMENT](#).

The following subroutine parameters need to be passed or will be returned accordingly:

**CODE**

<b>Code</b>	<b>Card</b>	<b>Screen Display</b>
0	VGA	For 132/24 column screen. ONLY if your current PC supports this screen mode will this work correctly.
1	ALL	80 x 24 screen
2	EGA	80 x 42 screen
3	VGA	80 x 49 screen
5	VGA	132 x 25 screen (see 132 note under mode 0)
6	ALL	80 x 25 screen
7	EGA	80 x 43 screen
8	VGA	80 x 50 screen
9	ALL	40 x 24 screen
10	ALL	40 x 25 screen

**Note** All modes are supported under Windows, regardless of card type.

The following subroutines allow you to create and control menus.

<a href="#">Create and load a menu.</a>	PIX.LOAD.MENU
<a href="#">Activate pull down menus.</a>	PIX.GET.MENU.RESPONSE
<a href="#">Close menus.</a>	PIX.CLOSE.MENU
<a href="#">Program function keys.</a>	PIX.PROGRAM.FKEYS
<a href="#">SLOT number.</a>	PIX.GET.SLOT.NUMBER

PIX.LOAD.MENU is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.LOAD.MENU(MENU.NAME, MENU.TEXT, SINGLE.MENU.NO, MAIN.COLOURS, CHARACTER.COLOURS, HIGHLIGHT.BAR.COLOURS, ERROR)**

Menus are loaded into HOSTACCESS's memory by this program. Once loaded, the text need never be sent again. They remain permanently in HOSTACCESS's memory for as long as HOSTACCESS is running. By activating and deactivating menus via [PIX.GET.MENU.RESPONSE](#) and [PIX.CLOSE.MENU](#), the menus appear and disappear instantly. Up to 50 menu sets can be loaded into HOSTACCESS at any one time, containing up to 200 menus; each menu can contain up to 20 elements each.

Menus can now be cascaded from other menus, i.e. similar to HOSTACCESS's own configuration menus. The first 8 menus passed are loaded across the top of HOSTACCESS's main screen. All other menus (9-200) automatically cascade that menu and its options when that element is selected.

Call the [PIX.GET.MENU.RESPONSE](#) program to display the menu on the screen. To write to the host application screen after getting a response back from menus, call PIX.CLOSE.MENU. You cannot update the host screen when menus are displayed. MENUS are not removed automatically from the screen, allowing you to open a selection box, input boxes or help windows, etc., on top of the menu. The following subroutine parameters need to be passed or will be returned accordingly:

<b>MENU.NAME</b>	Name of the menu you want to load into, this name is used to generate a valid menu no. between 1 and 50 by searching <a href="#">PIX.CONTROL.F</a> file item PIX.MENU.NAMES for the name.
<b>MENU.TEXT</b>	Menu text. Each menu as one dynamic array, each element as a multi-value. Any element ending with "n" will, when selected by the user, cause HOSTACCESS to display menu 'n' as a cascaded menu from that option. Cascaded menus can be called from more than one higher level menu, and cascades can call any of the 200 menus as well. If SINGLE.MENU.NO used to load one menu only then MENU.TEXT should only be a single dynamic array.
<b>SINGLE.MENU.NO</b>	If numeric then MENU.TEXT is loaded into the menus specified by SINGLE.MENU.NO only, otherwise entire menu will be reloaded. This is useful if you only want to change one menu's text out of the 200 rather than reload the lot. Normally this will be null.
<b>MAIN.COLOURS, CHARACTER.COLOURS and HIGHLIGHT.BAR.COLOURS</b>	Can for example be set to WHITE, BLACK for WHITE foreground on BLACK background etc. Main is the main box color, character is the highlight char and bar is the selection/highlight bar.
<b>ERROR</b>	Returns 1 if fatal errors.

PIX.GET.MENU.RESPONSE is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.GET.MENU.RESPONSE(MENU.NAME, MENU.TEXT, MENU.PATH, PATH.TEXT, ELEMENT.TEXT, EXIT.KEY, ERROR)**

This program will activate one of the 50 menu sets loaded by [PIX.LOAD.MENU](#) and display the pull down menus on the screen and wait for a user response.

The Home and End keys will move the highlight bar to the top and bottom of a Pop-Down Menu list. To select a menu option, the user presses the Enter key when the highlighted element is the required option. Pressing the ESCape key at any point allows the user to exit from the Pop-Down Menus or move back up a level if the current menu is a cascade from a previous one.

HOSTACCESS 's AiF handles all of the menu movement and underlying screen refreshes without any intervention or additional code on the Host system. Once a user selects an option, AiF will send the full menu path for the selected Menu number and the Element number to the Host system. It is a simple matter for the Host application to interrogate this response and determine which option the user has chosen.

The following subroutine parameters need to be passed returned accordingly:

<b>MENU.TEXT</b>	If passed with the data that was used to build current menu then ELEMENT.TEXT will contain the text from MENU.TEXT for the element number selected. PATH.TEXT is also returned as a multi valued string for the entire path of menus selected including any cascades. It is not necessary for MENU.TEXT to be the same text as loaded by <a href="#">PIX.LOAD.MENU</a> . In fact, the text you load may be variable to the users choice whilst that returned from RESPONSE may be the fixed text which your program checks for. This is how <a href="#">PIX.DB</a> and <a href="#">PIX.EASY.ACCESS</a> work to allow you to change text for foreign users.
<b>MENU.NAME</b>	Name of the menu you want to load into. This name is used to generate a valid menu number between 1 and 50 by searching <a href="#">PIX.CONTROL.F</a> file item PIX.MENU.NAMES for the name.  This name should be one that you have previously used when calling <a href="#">PIX.LOAD.MENU</a> .
<b>MENU.PATH</b>	Full menu path on which element was selected, multi-valued as follows:  Menu no]option no {]menu no]option no...} etc.  Where the 3rd and subsequent multi-values represent the path down any cascading menus.
<b>ELEMENT.TEXT</b>	The text of the very last menu element that was selected.
<b>PATH.TEXT</b>	Using MENU.PATH and MENU.TEXT, this variable is returned as a multi-value of the strings of text relative to the path the user went through to get through the menus. i.e. it could be SORT]DICT]BY... etc.
<b>EXIT.KEY</b>	A 2 character mnemonic is returned indicating which key was depressed to exit the selection box, e.g. ES for escape, CR for return, F1 for function key 1. Only exit keys set by PIX.EXIT.KEYS will actually allow that exit key to exit menus. A list of the 2 character mnemonics can be found documented under HOSTACCESS's DOS Keyboard Stacker.
<b>ERROR</b>	Returned as 1 if fatal error occurs.

Click here  for an example.

The BASIC code below is an **example** to how to load a simple menu and process the user's option.

```
* assign menu colors
MENU.COLOURS = 'BLUE,WHITE'
CHARACTER.COLOURS = 'RED,WHITE'
HIGHLIGHT.BAR.COLOURS = 'WHITE,RED'
*
* assign user text for menus, the ] represents a multi-value
MENU.TEXT = ''
MENU.TEXT<1> = 'SELECT]Reports]Enquiry]Data Entry'
MENU.TEXT<2> = 'INFO]Company]Network]Users'
MENU.TEXT<3> = 'HELP]Access Commands]Verbs'
PATH.TEXT = ''
CALL PIX.LOAD.MENU("DEMO", MENU.TEXT,"",MENU.COLOURS, CHARACTER.COLOURS, HIGHLIGHT.BAR.COLOURS,
ERROR)
*
LOOP
  CALL PIX.GET.MENU.RESPONSE ("DEMO",MENU.TEXT,"",PATH.TEXT,
ELEMENT.TEXT,EXIT.KEY,ERROR)
*
UNTIL EXIT.KEY EQ "ES" DO
*
* process each menu's option (you may need to close the menu to update the screen) *
  MENU.NAME = PATH.TEXT<1,1>
  BEGIN CASE
*
  CASE MENU.NAME EQ "SELECT"
    BEGIN CASE
      CASE ELEMENT.TEXT EQ "Reports"
        GOSUB PROCESS.REPORTS
      CASE ELEMENT.TEXT EQ "Enquiry"
        CALL PIX.EASY.ACCESS("", "")
      CASE ELEMENT.TEXT EQ "Data Entry"
        GOSUB GET.DATA
    END CASE
  CASE MENU.NAME EQ "INFO"
    GOSUB SHOW.INFO
  CASE MENU.NAME EQ "HELP"
    GOSUB SHOW.HELP
  END CASE
REPEAT
CALL PIX.CLOSE.MENU
```

PIX.CLOSE.MENU is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.CLOSE.MENU**

This routine closes the currently open menus. It only removes the window and menus from the screen, which can easily be reactivated/displayed again by calling [PIX.GET.MENU.RESPONSE](#).

You should always call this routine if you have asked for a response from PIX.GET.MENU.RESPONSE and then wish to update the host application screen. Otherwise HOSTACCESS will attempt to write any host text in current menu window!



PIX.PROGRAM.FKEYS is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.PROGRAM.FKEYS(KEY.DATA, FKEY.NO)**

As well as supporting programmable function keys for any emulation that supports them, HOSTACCESS supports its own AiF sequence to allow you to program them as well. This way, if you know a HOSTACCESS user is active, you can load the function keys no matter what the emulation.

By default, if you have not loaded F keys from the application then the particular emulation's default function key codes will be active. Programming a function key then causes the programmed code to be sent to the host. Emptying the key then reverts the key back to the emulation default.

Any one or all of the 48 Function Keys available in HOSTACCESS may be programmed by a host application to send character sequences to the host as if they were entered from the keyboard.

These character sequences may consist of any ASCII character including control codes.

Control characters are entered as ^A, ^B, or ^001, ^002 etc.

Use ^^ for the character '^'.

For characters in the range 128 to 255 enter the three digit decimal value after a '^', e.g. ^128. (Any character may be entered in this manner.)

Since PC keyboards do not always match terminal keyboards the following rules apply: -

1 - 10    F1 to F10  
11 - 20    Shifted F1 to F10  
21 - 30    Control F1 to F10  
31 - 40    Alt F1 to F10

On AT compatible PCs, the 8 additional function key F11 and F12 combinations can be used as you wish. You may also program additional keys, such as the arrow and edit keys.

For a full description of which keys can be programmed, please refer to the Developer's Guide and the User Guide.

The following subroutine parameters need to be passed or will be returned accordingly:

**KEY.DATA**    This may be a dynamic array where each array element will be loaded into that key, i.e. KEY.DATA<3> is loaded into Fkey 3.

**FKEY.NO**    If not null, it is assumed that KEY.DATA only contains one string, this is then loaded into Function Key FKEY.NO. If null, each function key will be loaded with the data corresponding to its attribute number in the KEY.DATA dynamic array.

Click here [PIX.PROGRAM.FKEYS Example](#) for an example.

**CALL PIX.PROGRAM.FKEYS( "WHO^M" :CHAR(254): "TIME^M", "")**

Will program functions keys F1 and F2 with WHO and TIME respectively.

PIX.GET.SLOT.NUMBER is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX . GET.SLOT.NUMBER(SLOT.NAME, SLOT.NUMBER, SLOT.TYPE, ERROR)**

This program is normally only called by MENUS and SELECTION BOX programs.

This program should be passed a menu name that you wish to use as your currently active menu. Since HOSTACCESS only allows menu numbers, this program will return the next available number in SLOT.NUMBER. This number is derived from the [PIX.CONTROL.F](#) file using a single item PIX.SLOT.NAMES. Complex applications may need this control record to be maintained on a per port basis to allow for more than 50 menu/selection box environments per system.

The following subroutine parameters need to be passed or will be returned accordingly:

<b>SLOT.NAME</b>	Any meaningful menu name that represents the applications menu or selection box environment.
<b>SLOT.NUMBER</b>	Returned value giving the number currently allocated to this SLOT.NAME or if not already used, the next available number is returned.
<b>SLOT.TYPE</b>	The environment set that needs to be used, defined as ONE of the following :  MENU  SELECTION.BOX  This name is used as a lookup in the PIX.CONTROL.F file for the number.
<b>ERROR</b>	Set to 0 if no error, 1 if error occurs.

The following subroutines allow you to load and control selection boxes. A selection box is a scrolling window with a list of elements. The user may scroll through this list and select any element within the list.

<a href="#">Load a selection box.</a>	PIX.LOAD.SELECTION.BOX
<a href="#">Activate a selection box.</a>	PIX.GET.SBOX.RESPONSE
<a href="#">Close a selection box.</a>	PIX.CLOSE.SELECTION.BOX
<a href="#">Load exit keys.</a>	PIX.EXIT.KEYS

PIX.LOAD.SELECTION.BOX is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.LOAD.SELECTION.BOX(SBOX.NAME, SBOX.TEXT, SBOX.TYPE, SBOX.HEADING, POSITION, WIDTH.DEPTH, MAIN.COLOURS, CHARACTER.COLOURS, HIGHLIGHT.BAR.COLOURS, EFFECTS, ERROR)**

Selection boxes are loaded into HOSTACCESS 's memory by this program. Once loaded, the text need never be sent again, they remain permanently in HOSTACCESS 's memory for the duration HOSTACCESS is loaded in the PC. By activating and deactivating selection boxes, by calling [PIX.GET.SBOX.RESPONSE](#) and [PIX.CLOSE.SELECTION.BOX](#) the user sees the selection boxes instantly.

Selection boxes are a very powerful way of presenting multiple items/options to a user. HOSTACCESS will automatically create a box around the selection if you require it, with any of the available HOSTACCESS effects.

Up to 50 selection boxes (of up to 32K each in size) can be sent to HOSTACCESS and then by simply calling the RESPONSE program, the selection box is displayed on the screen in the chosen window.

The following subroutine parameters need to be passed or will be returned accordingly:

<b>SBOX.NAME</b>	Name of the selection box you want to load into, this name is used to generate a valid selection box number between 1 and 50 by searching the <a href="#">PIX.CONTROL.F</a> file, item SELECTION.BOX.port.account.SLOT (where port is a port number such as 14, and account is an account ID such as ACCT) for the name.
<b>SBOX.TEXT</b>	Array list of items to load into the selection box. This can be dynamic or multi-valued but not both

<b>SBOX.TYPE</b>	If value is 1 then selection box type will be NOVELL style. If 0 or null then a normal selection box is used.
<b>SBOX.HEADING</b>	The heading text to appear at the top of the selection box.
<b>POSITION</b>	X,Y screen co-ordinates, from 0,0, of the top left hand corner of the window to contain the selection box. If passed as -1,-1 then the selection box will be displayed below the last HOSTACCESS menu/selection box option selection. This gives the effect of cascading down below previous menus/boxes.
<b>WIDTH.DEPTH</b>	e.g. 20,10 gives a selection box size to display a max. of 10 elements at any one time within a width of 20 bytes. If both or either are 0 then HOSTACCESS will calculate the maximum necessary or that will fit on the screen.
<b>MAIN.COLOURS, CHARACTER.COLOURS and HIGHLIGHT.BAR.COLOURS</b>	The names of the colors required for each part of the selection box. May be null (in which case current screen defaults used). Can for example be set to WHITE,BLACK for WHITE foreground on BLACK background etc. Main is the main box color, character is the highlight char and bar is the selection/highlight bar.
<b>EFFECTS</b>	SINGLE,DOUBLE,SHADOW,EXPLODE, comma delimited as one string.
<b>ERROR</b>	Returns 1 if fatal errors.

PIX.GET.SBOX.RESPONSE can be called from your applications as follows:

**CALL PIX.GET.SBOX.RESPONSE (SBOX.NAME, SBOX.TEXT, USE.WINDOW, SBOX.ELEMENT.NO, SBOX.ELEMENT.TEXT, EXIT.KEY, ERROR)**

This activates a previously loaded selection box loaded by [PIX.LOAD.SELECTION.BOX](#). If not already on the screen, this program will put it there. When a user chooses an option, this program returns the option number chosen as well as the text from SBOX.TEXT for that array position.

You should always call [PIX.CLOSE.SELECTION.BOX](#) if you wish to write to the screen after getting a response from the selection box. Selection boxes are *not* removed automatically.

The following subroutine parameters need to be passed or will be returned accordingly:

<b>SBOX.NAME</b>	Name of the selection box you want to use, this name is used to generate a valid selection box number between 1 and 50 by searching <a href="#">PIX.CONTROL.F</a> file item PIX.SELECTION.BOX.NAMES for the name. This name should be one that you have previously used when calling <a href="#">PIX.LOAD.SELECTION.BOX</a> .
<b>SBOX.TEXT</b>	Can be null but if used, ELEMENT.TEXT will be returned with the value in array position ELEMENT.NO. This saves your calling program having to work out what selection was made. If passed with the data used to build current menu, SBOX.ELEMENT.TEXT contains the text from MENU.TEXT for the element selected. It is not necessary for MENU.TEXT to be the same text as loaded by <a href="#">PIX.LOAD.MENU</a> . The text you load may be variable to the users choice while the return from RESPONSE may be the fixed text your program checks for. This is how <a href="#">PIX.DB</a> and <a href="#">PIX.EASY.ACCESS</a> work to allow you to change text for foreign users.
<b>USE.WINDOW</b>	If true, then display selection box in currently open window, otherwise let HOSTACCESS open one for you.
<b>SBOX.ELEMENT.NO</b>	Returned as the number of the element selected
<b>SBOX.ELEMENT.TEXT</b>	Returned as the text contained within the selected element.
<b>EXIT.KEY</b>	A 2 character mnemonic is returned indicating which key was depressed to exit the selection box, e.g. ES for escape, CR for return, F1 for function key 1. Only exit keys set by <a href="#">PIX.EXIT.KEYS</a> will actually allow that exit key to exit selection boxes. A list of the 2 character mnemonics can be found documented under HOSTACCESS's DOS Keyboard Stacker.
<b>ERROR</b>	Returned as 1 if fatal error.

PIX.CLOSE.SELECTION.BOX is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.CLOSE.SELECTION.BOX**

This routine closes the currently open selection box. It only removes the window and text from the screen, which can easily be reactivated and displayed again by calling the subroutine [PIX.GET.SBOX.RESPONSE](#).

You should always call this routine if you have asked for a response from PIX.GET.SBOX.RESPONSE and then wish to update the host application screen. Otherwise HOSTACCESS will attempt to write any host text in this selection box window.

PIX.EXIT.KEYS is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.EXIT.KEYS(EXIT.KEYS)**

The EXIT.KEYS subroutine is used to notify HOSTACCESS which additional keys should cause an exit from any one of the following AiF functions:

WINDOW.EDITOR

FIELD/BOX INPUT

PULL DOWN MENUS

SELECTION BOXES

Any previous exit keys will be disabled, i.e. they are not additive.

This routine should be called before getting a response from a menu or selection x. You could make the 'F1' function key cause an exit from the menu and then perhaps assume the user wants help on the menu option that was highlighted. It could be 'F10' to delete the current highlighted option and so on.

Virtually any key can be set as an exit key, including single characters or special PC keys like 'HM' home etc.

The following subroutine parameters need to be passed or will be returned accordingly:

**EXIT.KEYS** Can contain a string of 2 character mnemonics as used in and documented under HOSTACCESS's DOS Keyboard Stacker. E.g. CR is carriage return, F1 for F1, F0 for F10, DA for down arrow. If you want any ASCII characters to be exit keys then enclose these in quotes, e.g. F1F2 'A?' will give F1,F2,A and ? exit keys.  
ES, escape is always an EXIT key and cannot be disabled, as is CR, carriage return.  
Note that alphanumeric characters as exit keys will be ignored by the box and window editors for obvious reasons, they are OK for windows and menus though.

Click here  for an example.

**CALL PIX.EXIT.KEYS("F1F2F3")**

Will allow functions keys F1,F2,F3 to exit from MENUS, LINE EDITOR, BOX EDITOR, WINDOW EDITOR and SELECTION boxes.  
Any other exit keys are cleared.



The following subroutines allow you to set and configure colours.

<a href="#">Set ANSI colours for printing</a>	PIX.GET.COLOUR.CODES
<a href="#">Sets colours</a>	PIX.SET.COLOUR
<a href="#">Change colours.</a>	PIX.COLOUR.CONFIG

PIX.GET.COLOUR.CODES can be called from your applications as follows:

**CALL PIX.GET.COLOUR.CODES(COLOURS, OPTIONS, COLOUR.CODE)**

Pass this routine the colours you want and set OPTIONS to D for default and/or F for flashing and COLOUR.CODE will return the colour codes that can be printed to achieve the desired effect. i.e. 1;37;44;5 for flashing lightwhite on blue.

The following parameters will need to be passed or will be returned accordingly:

<b>COLOURS</b>	Should be set to {LIGHT}FOREGROUND,BACKGROUND. If either color is invalid it will be set to white.  If COLOURS is null then COLOUR.CODE will be set to null.
<b>OPTIONS</b>	Can be set to null or 'D' and/or 'F', where 'D' default screen colors and 'F' flashing attribute on.
<b>COLOUR.CODE</b>	Is returned as the ANSI code required to generate the specified colors. To activate these colors within the calling program, simply wrap COLOUR.CODE with the ANSI start and end sequences and PRINT it.

Click here  for an example.

To print text with the following colours, white on red, use the following:

```
TEXT = "Good Morning !"  
CALL PIX.GET.COLOUR.CODES ("WHITE,RED", "", COLOUR.CODE)  
PRINT CHAR(27): "[" : COLOUR.CODE : "m" : TEXT
```

To set the screen default foreground and background to blue on white (light gray), use the following:

```
CALL PIX.GET.COLOUR.CODES("BLUE,WHITE", "D", COLOUR.CODE)  
PRINT CHAR(27): "[" : COLOUR.CODE : "m"  
PRINT @(-1); ; * clear screen to reset colours.
```

PIX.SET.COLOUR is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.SET.COLOUR(COLOURS, OPTIONS)**

Setting colours could not be easier than with this little routine. Rather than have to work out what ANSI colour code number is given to BLUE, RED etc., just pass the text of the colour you require.

This program then sets the colour specified on the screen for you. If you like, you can specify an option to tell HOSTACCESS that the colour specified should be used as the default from now on.

The following subroutine parameters need to be passed or will be returned accordingly:

**COLOURS** The names of the colors required for foreground and/or background. For **example**, for a Yellow foreground on Black Background set COLOURS = 'YELLOW,BLACK '; for a Light red foreground on a green background set COLOURS = 'LIGHTRED,GREEN '.

**OPTIONS** Can be 'D ', 'F ' or both.

To set to current default, i.e. for all screen clears and clear to end of line, send a D in OPTIONS.

To set the foreground color to flashing, send an F in OPTIONS.

PIX.COLOUR.CONFIG is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.COLOUR.CONFIG(COLOUR.PART, COLOUR1, COLOUR2, COLOUR3, COLOUR4, ERROR)**

This subroutine enables programs to allow users to change the colours of application controlled AiF pop-down menus, selection boxes, windows and even the current screen colours.

To see this in action, you should select the **menu** or **window** colours option from the relevant menu in the [DB - A DATA BASIC Programmer 's Utility](#) routine (programmer 's toolkit facility).

The following subroutine parameters need to be passed or returned accordingly:

- COLOUR.PART** Tells the subroutine which "object" should have its colors changed. Specify this "object" as one of the following :
- MENU
  - SELECTION.BOX
  - WINDOW
  - SCREEN
- Free format text (no more than 20 characters)
- A simulated screen together with the object, say a pop-down menu, will be displayed showing the colours as passed into the subroutine. If you specify this parameter as "free format text", a simulated window containing this text will be presented. Only the colours of this text will be changed.
- COLOUR1** Specify the foreground and background colour for the menu or selection box text and box (including frame). May be null, click here  for defaults.
- COLOUR2** Specify the foreground and background colour for the selection character, as used within a menu or selection box. May be null, click here  for defaults.
- COLOUR3** Specify the foreground and background colour for the highlight bar as used within a menu or selection box. May be null, click here  for defaults.
- COLOUR4** Specify the foreground and background colour for the underlying screen. May be null, for defaults see below.
- ERROR** On return, will be set to 1 if an error occurs, 0 if no errors.

Click here  for further information.

## 1. Defaults

All the colours specified in the COLOUR1-4 parameters may be null. The appropriate default will be taken from a multi-value 1 to 4 in attribute 7 in the record [PIX.COLOUR.CONFIG](#) in the [PIX.TEXT.F](#). You may modify these default colours as required, but please remember to keep all of the colour names in capitals and in ENGLISH. (The same applies to attribute 4 of this record). All of the other text, that appears when PIX.COLOUR.CONFIG is run, may be modified by translating the text in the PIX.TEXT.F file record, as appropriate to the language you require.

## 2. Using PIX.COLOUR.CONFIG

This subroutine has been written to be as easy to use as possible. It may even be presented in other languages by translating the text in the PIX.TEXT.F file.

Help is available to users by pressing the function key F10. The display of the "object" (menu, selection box, etc..) being colour configured changes as the user selects a different colour.

## 3. Colour Parameters

The parameters COLOUR1, COLOUR2, COLOUR3 and COLOUR4 should be specified as pairs of foreground and background colours in their ENGLISH notation. For example, to specify a highlight bar colour of white text on a red background, specify the COLOUR3 parameter as :

WHITE,RED

This subroutine will return the colours selected by the user (or the defaults, if null parameters passed in) in each of these parameters in the same notation.

Names of colours may be specified as any of the following :

**BLACK, RED, GREEN, BROWN, BLUE, MAGENTA, CYAN, WHITE, GREY (GRAY), YELLOW**

For intense (light) foreground colours, prefix the colour name with "LIGHT", e.g. LIGHTBLUE. Note that LIGHTBROWN is the same as YELLOW and that you may specify YELLOW directly.

The following subroutines enable you to use graphical features:

<a href="#"><u>Create an input box</u></a>	PIX.BOX.INPUT
<a href="#"><u>Centre text</u></a>	PIX.CENTRE.TEXT
<a href="#"><u>Draw a box</u></a>	PIX.DRAW.BOX
<a href="#"><u>Draw a line</u></a>	PIX.DRAW.LINE
<a href="#"><u>Create a single line input</u></a>	PIX.LINE.INPUT
<a href="#"><u>Display a message line</u></a>	PIX.MESSAGE.LINE
<a href="#"><u>Display a message window</u></a>	PIX.MESSAGE.WINDOW
<a href="#"><u>Creates a window for text editing</u></a>	PIX.WINDOW.EDITOR
<a href="#"><u>Displays title text</u></a>	PIX.WINDOW.TITLE

PIX.BOX.INPUT is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.BOX.INPUT(STRING, POSITION, SIZE.BOX, SIZE.INPUT, VALIDATION, CLEAR.MODE, EFFECTS, TITLE, EXIT.KEY)**

Box Input enables host applications to pop up an input box anywhere on the screen and request user input. This input may be restricted to only certain types of input and this saves unnecessary validation by the host.

This very powerful routine effectively gives ALL of the benefits of single character inputs, more features and of course with NO burden on host 's performance. Hence it is excellent across slow links.

User input is NOT constrained to the box 's width. Text input may exceed the width of the box and HOSTACCESS will indicate that additional text exists by displaying appropriate arrow symbols. The maximum length for the text that can be input should be specified by the host application.

The input box itself may be enhanced with selected frame styles and/or title. Once the user has completed input, the box will disappear and the underlying screen be instantly restored.

The following subroutine parameters need to be passed or will be returned accordingly:

<b>STRING</b>	Data to be input can be null or an existing string. On return, it contains the string input (or modified) by the user.
<b>POSITION</b>	COL,ROW where to put box, PICK co-ordinates i.e. 0,0 are valid.
<b>SIZE.BOX</b>	Width of box, may be smaller than SIZE.INPUT
<b>SIZE.INPUT</b>	Max length of data allowed to be input, may exceed SIZE.BOX
<b>VALIDATION</b>	Restrict input to certain type of data as follows: - 'INTEGERS ' 'NUMERIC ' 'ALPHA ' 'ALPHANUMERIC ' 'HEX ', only 0-9 and A-F allowed 'HIDDEN ', useful for passwords etc. Any other value will result in no validation being applied.
<b>CLEAR.MODE</b>	If CLEAR.MODE is 0 then any existing string that is passed will be displayed and insert mode will be enabled. The only difference with CLEAR.MODE set to 1 is that if the user starts typing text then the first character he inputs replaces the string. If the user uses arrow keys etc. as the first input, the string is left for insert mode. CLEAR.MODE as 1 is similar to most DOS applications input.

**EFFECTS**

All the following parameters are optional and comma separated e.g. 'SHADOW,EXPLODE,SINGLE '.


'SHADOW '	Shadow effect
'EXPLODE '	Bring window up with exploding effect
'SINGLE '	Single line border around window
'DOUBLE '	Double line border around window
'SINGLE DOUBLE '	Single line border at top and bot, Double sides
'DOUBLE SINGLE '	Double line border at top and bot, Single sides

**TITLE**

Optional text to appear as heading in input box

**EXIT.KEY**

A 2 character mnemonic is returned indicating which key was depressed to exit the input, e.g. ES for escape, CR for return, F1 for Function Key 1. Only exit keys set by PIX.EXIT.KEYS will actually allow that exit key to exit input boxes. A list of the 2 character mnemonics is found documented under HOSTACCESS 's DOS Keyboard Stacker.

Click  here for an example.



PIX.EXIT.KEYS.



PIX.LINE.INPUT.



PIX.WINDOW.EDITOR.

**CALL PIX.BOX.INPUT(TXT,"5,5",20,50,"ALPHA",0,"SINGLE",DESC,XKEY)**

Will draw a box at X,Y pos 5,5 and 20 characters wide with a SINGLE border. DESC will be displayed in the heading of the box. Any text in variable TXT will be displayed in the box and can be amended to include only ALPHA characters up to 50 characters in length. The exit key, normally 'CR' is returned in XKEY and TXT is returned containing the final text that was in the box.



PIX.CENTRE.TEXT is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.CENTRE.TEXT(START.ROW, CENTRE.TEXT)**

This program tells HOSTACCESS to center **CENTRE.TEXT** on **START.ROW** line in the current window, or if there is no window open, then in the center of the screen.

If **CENTRE.TEXT** is a dynamic array then each line will be centered on subsequent lines, starting at

For PICK compatibility, **START.ROW** starts at 0.

The following subroutine parameters need to be passed or will be returned accordingly:

**START.ROW**        The start row number on which the text will be centered.  
**CENTRE.TEXT**     The string(s) you want centered on the screen. This can be a dynamic array of strings which will be centered on each subsequent row starting from **START.ROW**.

PIX.LINE.INPUT is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.LINE.INPUT(STRING, LENGTH, EXIT.KEY)**

This subroutine provides single line input facilities for existing host applications. This routine takes, from the current cursor position for a length of **LENGTH** all characters on the screen. The user can change the content of the text using the local (PC) edit keys, e.g. ins/del/home/end/arrow keys etc. When the user selects an **EXIT.KEY** (see [PIX.EXIT.KEYS](#)) or presses ESCcape or RETURN, **STRING** is returned with the contents of the specified part of the screen at that point.

This program differs from the program [PIX.BOX.INPUT](#) in the sense that all an application effectively has to do to use this facility is replace something like this:

```
PRINT @(10,10): ; INPUT ADDRESS1,30:
```

With:

```
PRINT @(10,10):  
IF ON.TERMITE THEN  
  CALL PIX.LINE.INPUT (ADDRESS1, 30, EXIT.KEY)  
END ELSE  
  INPUT ADDRESS1,30:  
END
```

This gives full line editing facilities. You can modify or clear **ADDRESS1** which may already be on the screen. For even more powerful inputs, you should look at [PIX.BOX.INPUT](#). This allows you to put flashing boxes around inputs and to pass it the string to amend etc.

The following subroutine parameters need to be passed or will be returned accordingly:

**STRING**        Returned string with the value of exactly what was on the screen from current cursor position for length **LENGTH**.  
**LENGTH**        From the current cursor position, start editing **LENGTH** characters and only allow that max. **LENGTH** to be entered.  
**EXIT.KEY**     Key pressed, e.g. 'CR ', 'ES ', 'F1 ' to exit input. See [PIX.EXIT.KEYS](#) for more information

**Example:**

**CALL PIX.LINE.INPUT(STRING,10,EXIT.KEY)**

From the current cursor position, take the 10 characters on the screen and allow full line editing input on them. When the user presses an **EXIT.KEY**, say <ENTER>, then **STRING** is returned with the amended 10 characters. **EXIT.KEY** is returned to the caller as well to indicate the key that key was pressed. Trailing spaces are *not* returned.

PIX.DRAW.BOX is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.DRAW.BOX(POSITION, COLOURS, EFFECTS)**

This program will draw any specified box on screen at the position specified by the calling program with any colour combination and with any of the box effects.

If you specify a SINGLE or DOUBLE border then the border is put around the box and the box itself is the size you specify.

The following subroutine parameters need to be passed or will be returned accordingly:

**POSITION** X,Y,X,Y where the first XY, is top left and the second XY is the bottom right. These XY co-ordinates are PICK specific i.e., the top left-hand corner is co-ordinate 0,0.

If -1 passed as the first XY co-ordinates, box will be centered in the currently open window or screen.

**COLOURS** 'foreground,background ', e.g. 'WHITE,RED '.

**EFFECTS** All parameters as follows are optional and comma separated:

'SHADOW' Shadow effect.

'EXPLODE' Bring box up with exploding effect.

'SINGLE' Single line border around box.

'DOUBLE' Double line border around box.

'SINGLE DOUBLE' Single line border at top and bottom,  
Double sides.

'DOUBLE SINGLE' Double line border at top and bottom,  
Single sides.

**Example:**

**CALL PIX.DRAW.BOX("5,5,40,10","RED,WHITE","SINGLE,SHADOW")**

Will draw a box on the screen, red text on a white background 35 chars wide and 5 deep. The box will have a SINGLE border around it and will have a shadow effect as well.

PIX.DRAW.LINE is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.DRAW.LINE(POSITION, COLOURS, EFFECTS)**

Draws a line or frame in the specified colour and type between the two points specified by POSITION. A line can either be horizontal or vertical. HOSTACCESS also provides intelligent merging of lines to insert the correct tees and crosses etc. If the co-ordinates passed in position do not match to draw a line, a frame will be drawn. E.g. 10,10,20,20 will draw a frame 11 deep by 10 wide from 10,10. This differs from drawing a box with [PIX.DRAW.BOX](#) in the sense that you cannot fill or have effects on a frame. Equally you cannot merge lines when using PIX.DRAW.BOX, as you can here.

The following subroutine parameters need to be passed or will be returned accordingly:

- POSITION** X,Y,X,Y where the first XY, is the starting co-ordinates for the line or frame & the 2nd, the end co-ordinates. These XY co-ordinates are PICK specific, i.e. 0,0 is the top left hand corner of the currently open window or screen.
- If either pair of XY are -1 then the line co-ordinates will match the dimensions of the currently active window/screen. For example, to draw a box from 10,10 to the bottom right on a full 80,24 screen send '10,10,-1,-1'.
- COLOURS** 'foreground,background ', e.g. 'WHITE,RED'.
- EFFECTS** Comma delimited with no spaces. Line type :
- 'SINGLE' Single line
  - 'DOUBLE' Double line
  - 'MERGE' Merge the line with any other on screen, great for joining lines, boxes etc., without having to worry about tees and crosses etc.
- Let HOSTACCESS do the hard bit for you, e.g. just specify 'DOUBLE,MERGE'.

**Example:**

**CALL PIX.DRAW.LINE("5,5,75,5","RED, WHITE","SINGLE,MERGE")**

Will draw a single line on the screen at row 5, 70 characters long. If any other lines cross this line on the screen then the relevant tee and cross bars are generated to make the line MERGE.

PIX.MESSAGE.LINE is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.MESSAGE.LINE(MESSAGE,COLUMN,COLOURS,OPTIONS)**

This subroutine allows you to print to the system message line regardless of what emulation you are running.

This program will display a given message on the system message line in any given colour at a specified column. Options are available to clear the line first (otherwise it will print on top of any existing text), and also to reset the status line back to the current status.

The following subroutine parameters need to be passed or will be returned accordingly:

<b>MESSAGE</b>	Message text, maximum length should be SCREEN WIDTH-COLUMN.
<b>COLUMN</b>	If not null specifies starting column for message
<b>COLOURS</b>	The names of the colors required for foreground and/or background, e.g. WHITE,BLUE or RED,WHITE, etc.
<b>OPTIONS</b>	F,C,R or even all 3 together, no commas needed where: <b>F</b> for flashing foreground. <b>C</b> to clear any message on the status line first. <b>R</b> will restore status line with main HOSTACCESS status line if not previously switched off from HOSTACCESS 's Configure menu.

PIX.MESSAGE.WINDOW is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.MESSAGE.WINDOW(COLOURS, EFFECTS, TEXT)**

This subroutine is used by some of the PICK HOSTACCESS programs. If you pass a dynamic array of TEXT and specify the COLOURS and EFFECTS, this routine will display that text, centered on the screen in a window.

The window will be left OPEN on the current screen, your application may want to close it with the [PIX.CLOSE.WINDOW](#) subroutine. You might want to copy this program and use it as a template to build your own help programs etc.

The following subroutine parameters need to be passed or will be returned accordingly:

<b>COLOURS</b>	The names of the colors required for foreground and/or background e.g. GREEN,BLACK or WHITE,BLUE or YELLOW,WHITE etc.
<b>EFFECTS</b>	Any permutation of SHADOW, EXPLODE, SINGLE, DOUBLE, SINGLE DOUBLE etc. Full details in PIX.OPEN.WINDOW subroutine.
<b>TEXT</b>	Dynamic array containing text you want in the WINDOW.

PIX.WINDOW.EDITOR is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.WINDOW.EDITOR(POSITION, COLOURS, EFFECTS, STRING, OPTIONS, EXIT.KEY)**

The AiF Window Editor enables host applications to pop up an input window (of one or more lines) anywhere on the screen and request user input. The user may maneuver around the text within the window using all of the available PC based keys to insert, delete and add text at fast PC speeds. User input is constrained by the window 's width and depth, i.e. no scrolling is available.

This facility acts like a mini wordprocessor, all of the hard work is done by HOSTACCESS. It is very useful for address line inputs, additional information, statements, etc.

The following subroutine parameters need to be passed or will be returned accordingly:

<b>POSITION</b>	X,Y,X,Y where the first XY is top left and the second XY is the bottom RIGHT. These XY co-ordinates are PICK specific, i.e. 0,0 is the top left hand corner of the screen.  If -1 passed as first co-ordinates, window will be centered in the screen.
<b>COLOURS</b>	The names of the colors required for the foreground and/or

background, e.g. 'WHITE,RED'.

#### **EFFECTS**

All parameters as follows are optional and comma separated :

'SHADOW '	Shadow effect
'EXPLODE '	Bring window up with exploding effect
'NO CLEAR '	The text behind the window stays within the window
'SINGLE '	Single line border around window
'DOUBLE '	Double line border around window
'SINGLE DOUBLE '	Single line border at top and bottom, Double sides
'DOUBLE SINGLE '	Double line border at top and bottom, Single sides

#### **STRING**

Dynamic array containing start string to store in window, if any of any dynamic arrays exceed the size of the window then these will be truncated.

STRING will be returned equally as a dynamic array for each line from the window, irrespective of what exit key that was used. It is up to the application 's responsibility to examine EXIT.KEY and take appropriate action.

#### **OPTIONS**

The following options are available:

- 0 When leaving window editor, close window automatically and refresh user screen.
- 1 Close window but leave the window on the screen. Maybe useful to save redisplaying address lines etc.
- 2 Leave window on screen and DO NOT close, it is up to the caller to issue call to [PIX.CLOSE.WINDOW](#).

#### **EXIT.KEY**

A 2 character mnemonic is returned indicating which key was depressed to exit the editor e.g. ES for escape, CR for return, F1 for function key, etc. Only exit keys set by [PIX.EXIT.KEYS](#) will actually allow that exit key to exit window editor. A list of the 2 character mnemonics can be found documented under HOSTACCESS 's DOS Keyboard Stacker.

Click here  for an example.

**CALL PIX.WINDOW.EDITOR("5,5,75,10","RED,WHITE","SINGLE",  
REC,2,EXIT.KEY)**

will open a window on the screen 6 lines deep and 71 columns wide. A single border will be put around the window and the colour will be RED text on a WHITE background. The dynamic array REC will be displayed in the window, 1 array element = 1 line of text. The user will then be able to 'mini wordprocess' the text. When the user presses an exit key then the '2' tells this routine to close the window. REC is returned containing the amended text.

PIX.WINDOW.TITLE is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.WINDOW.TITLE(TITLE.TEXT, COLUMN, COLOURS, OPTIONS)**

This program will display any given text in either the heading or footing of the currently open window, as long as the window has a border. Any existing title in the top or bottom of the window will be replaced by TITLE.TEXT.

The following subroutine parameters need to be passed or will be returned accordingly:

**TITLE.TEXT**    Heading or footing text depending on    text depending on  
                  option setting.

**COLUMN**        If null text is centered otherwise starts at COLUMN.

**COLOURS**      The names of the colors required for foreground and/or  
                  background, e.g. 'WHITE,RED'.

**OPTIONS**       'F' for footing or 'H' for heading.


PIX.DISPLAY.IMAGE is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.DISPLAY.IMAGE(IMAGE.ID, POSITION, POSITION2, SCALE, OPTIONS, EXIT.KEY)**

This program will display a PCX image file on top of, or alongside the host application screen at the specified position. The image may also be displayed reduced in size. It is possible to display multiple images on the screen at the same time by calling this routine consecutively, without displaying anything else on the screen in between calls. HOSTACCESS will display the image and wait for user input. The calling application should wait on input and when the user presses a key the images will disappear and the keystroke sent to the host program waiting on input.

You should not write to the screen or use HOSTACCESS menus, DOS gateway, etc., whilst images are on the screen.

If you specify the S option, users can pan through documents such as contracts, invoices, legal forms, poll tax forms and so on. Use HOSTACCESS to display incredibly high quality colour or monochrome images alongside any of your host applications by simply calling this subroutine.

Click here  for a description of the parameters.



The following are the parameters for the PIX.DISPLAY.IMAGE subroutine:

**CALL PIX.DISPLAY.IMAGE(IMAGE.ID, POSITION, POSITION2, SCALE, OPTIONS, EXIT.KEY)**

<b>IMAGE.ID</b>	HOSTACCESS will display the PCX file image with the DOS name IMAGE.ID, the DOS extension does not have to be specified. i.e. if IMAGE.ID is C:\IMAGES\HOUSE or C:\IMAGES\PEOPLE\000192.PCX will both display the image in the relevant DOS file, HOUSE.PCX or 000192.PCX.
<b>POSITION</b>	X,Y position from (0,0) to (nn, nn) pixels which determines the position to display top left hand corner of image relative to the chosen screen resolution.
<b>POSITION2</b>	X2,Y2 position which determines the pixel position to display bottom right hand corner of image.
<b>SCALE</b>	The size of the image being displayed as a percentage of the original image size. E.g. normal size is 100, half size is 50, quarter size is 25, double size is 200, etc.
<b>OPTIONS</b>	May be one of the following :
<b>'M '</b>	Tells this routine that MORE images are going to be displayed on the same screen. This stops this routine waiting on INPUT and EXIT.KEY will be returned as null. You should immediately call this routine again with the next image information that you want display. The last image should not use the 'M ' option and HOSTACCESS will then wait for user INPUT and return EXIT key with the text pressed.
<b>'C '</b>	Tells this routine to CLEAR the host screen before displaying the image first. i.e. do not overlay the image on top of the host screen. Useful and faster for instance if you are displaying a full screen image. (early HOSTACCESS versions will always clear the screen before displaying the first image).
<b>'B '</b>	For backwards compatibility. Uses the older HOSTACCESS 2.0+FSI "_95" escape sequence to display images. SCALE should then be on a number 1-5 where 2 is half the size etc.
<b>'S '</b>	Enables scrolling (or panning) of an image within a window.
<b>'R '</b>	Prevents resetting of the image screen - i.e. leaves the image on the screen.
<b>EXIT.KEY</b>	Will return a two character mnemonic exit key which the user used to exit the display of the LAST image. i.e. 'ES ' or 'CR ' if the user pressed ESCAPE or <RETURN> respectively.  These mnemonics are defined under HOSTACCESS 's DOS Keyboard stacker. You can easily and quickly load Exit Keys by calling the PIX.EXIT.KEYS subroutine. Note if you have defined single character exit keys, such as 'X ' for <b>example</b> , this will be returned as space followed by the exit key (i.e. as ' X ').  EXIT.KEY will be returned as NULL if the 'M ' (more) option has been used.

#### Example

**CALL PIX.DISPLAY.IMAGE("E:\CARS\BMW", "0,0", "50", "", EXIT.KEY)**

This example displays a colour photograph stored in the PCX file BMW.PCX in top left of the screen at half the size it is stored at.

<a href="#">Detect a mouse driver.</a>	PIX.MOUSE.AVAILABLE
<a href="#">Activates the mouse.</a>	PIX.MOUSE.ON
<a href="#">Turns on mouse reporting.</a>	PIX.MOUSE.RESPONSE
<a href="#">Turn off mouse event reporting.</a>	PIX.MOUSE.OFF
<a href="#">Mouse test program.</a>	PIX.MOUSE.TEST

PIX.MOUSE.AVAILABLE is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.MOUSE.AVAILABLE(AVAILABLE)**

This program returns a 1 or 0 value in the variable AVAILABLE, depending upon whether HOSTACCESS can detect that a MOUSE driver has been loaded on the PC. 1 is always returned in HOSTACCESS for Windows.

PIX.MOUSE.ON is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.MOUSE.ON(MONITOR.MOUSE.EVENTS)**

This program activates the MOUSE on the PC for use with HOST applications. Turning on the mouse still allows keyboard input through. Applications can detect if the mouse has been pressed by checking for an STX followed by the text 'MS' followed by carriage return. If any other input comes through the input buffer then it can be processed in the normal way.

This program does not return the response type from the mouse, it only activates it. There is a program called [PIX.MOUSE.RESPONSE](#) that can be called but you may wish to integrate the logic around your own INPUT statements. It is nice and easy, so you should have no problems.

MONITOR.MOUSE.EVENTS, should be a string containing the words, LEFT, RIGHT, CENTRE, ALL and/or CONTINUOUS. This tells HOSTACCESS that when any of the specified MOUSE buttons have been depressed, a response should be sent to the HOST. You can specify any or all of the options in the string using comma delimiters, e.g. if MONITOR.MOUSE.EVENTS contains "LEFT,CENTRE" then HOSTACCESS will not send a response if the RIGHT button is clicked but will if the left or center ones are pressed.

Passing the CONTINUOUS parameter causes HOSTACCESS to send a response back to the host *every* time the MOUSE is moved whilst one of the specified mouse buttons is depressed. This can be used by applications that need to know where the MOUSE is at any given time. This should be used with caution since *lots* and *lots* of inputs will occur while the mouse button is held down.

PIX.MOUSE.RESPONSE is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.MOUSE.RESPONSE(RESPONSE, BUTTON, XCORD, YCORD)**

This program will return the response from the mouse when one of the selected buttons has been pressed or if the button is being continuously held down. These options are selectable when turning on the MOUSE, see [PIX.MOUSE.ON](#).

To match most PICK systems, this routine will subtract 1 from the X and Y co-ordinates, i.e. assuming start row/col is 0,0.

The following subroutine parameters need to be passed or will be returned accordingly:

- RESPONSE** Contains the text string as input by the user, if the mouse was not used. RESPONSE will be NULL if MOUSE used, otherwise keyboard input.
- BUTTON** Button will return a code from 0 to 7 which have the following meaning:
- 0 Button(s) released.
  - Only if continuously monitoring the mouse button, does this value get returned. It is only returned when any or all of the buttons are released.
  - 1 Left Button depressed.
  - 2 Right Button depressed.
  - 3 Left and Right Buttons depressed.
  - 4 Centre button depressed.
  - 5 Left and Centre buttons depressed.
  - 6 Right and Centre buttons depressed.
  - 7 Left and Right and Centre buttons depressed.
- In most cases, your calling routine just needs to check for a negative BUTTON variable to ensure detecting that the MOUSE has not been used.
- If BUTTON is -1 then the user must have pressed return from the keyboard and thus no MOUSE response will be valid.
- XCORD, YCORD** Will be set to X,Y both minus one to match PICK assuming top left is 0,0 instead of the ANSI standard 1,1.
- Both of these will be -1 if user has used the keyboard instead of the MOUSE.

Click here  for an example.

**CALL PIX.MOUSE.RESPONSE(RESPONSE,BUTTON,XCORD,YCORD)**

Make sure you have used [PIX.MOUSE.ON](#) first.

Will, if the mouse is active, wait until the mouse button has been pressed and return in XCORD and YCORD the X,Y position of the mouse cursor when a button was pressed. BUTTON is returned as a code for whichever mouse button has been pressed. If BUTTON is -1 then RESPONSE will contain the text string that the user must have typed in and pressed return to.

PIX.MOUSE.OFF is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.MOUSE.OFF**

This program will stop HOSTACCESS returning anything when the mouse is used. The mouse should be turned off when not it use to stop input being passed into non-mouse host applications.

PIX.MOUSE.TEST is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.MOUSE.TEST**

This program simply demonstrates how the MOUSE can be used within a host application.

It only runs if the PIX.MOUSE.ACTIVE returns true, i.e. if you have loaded a mouse driver into your PC. It will not work unless you have this driver loaded.

This program demonstrates using the MOUSE continuously. With a button down you can drag a box around the screen. EVERY event is monitored and displayed on the status line.

Try changing some of the parameters and experiment with other mouse detect buttons.

<a href="#">Call a DOS application from PICK.</a>	PIX.CALL.DOS
<a href="#">Transfer a DOS file to a PICK item/PICK spooler.</a>	PIX.DOS.PICK
<a href="#">Transfer a DOS file into multiple items in a PICK file.</a>	PIX.DOS.PICK.ALL
<a href="#">Transfer data from DOS to PICK.</a>	PIX.CALL.DOS.PICK
<a href="#">Transfer data from PICK to DOS.</a>	PIX.CALL.PICK.DOS
<a href="#">Transfer a PICK item to a DOS file</a>	PIX.PICK.DOS
<a href="#">Transfer multiple PICK items into a single DOS file.</a>	PIX.PICK.DOS.ALL
<a href="#">Transfer multiple PICK items to multiple DOS files.</a>	PIX.TDUMP
<a href="#">Transfer multiple DOS files to multiple PICK items.</a>	PIX.TLOAD
<a href="#">Erase a DOS file.</a>	PIX.ERASE.DOS.FILE
<a href="#">Check DOS for the existence of a file or directory.</a>	PIX.EXISTS

PIX.CALL.DOS is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.CALL.DOS(DOS.COMMANDS, OPTIONS)**

Subroutine to call a DOS application directly from PICK with a number of options.

The following subroutine parameters need to be passed or will be returned accordingly:

**DOS.COMMANDS** Command may include ";" and/or "%" as required, for **example**:

"123"	Go to DOS and run 123.
"CD\SCALC5;SC5"	Go to DOS and run SC5 from the SCALC5 directory.
"123% ' /FRTEMP.WK1 'CR"	Run 123 and File Retrieve (FR) the worksheet TEMP.WK1 automatically.

If **DOS.COMMANDS** is null the DOS Gateway is entered, type **EXIT** at DOS command prompt to return back to HOSTACCESS.

**OPTIONS**

<b>Null</b>	Open a DOS window normalized and activate.
<b>2</b>	Open a DOS window minimized and activate.

No other options are applicable in HOSTACCESS.

It is invalid to try to use SWAP with options 1 and 2 above since HOSTACCESS will not be in memory to allow any screens, backpages or windows, etc., to be updated.

SWAPPING will ONLY take place if S option is used AND if HOSTACCESS has been configured to SWAP to EMS or DISK.

Click here  for an example.



**CALL PIX.CALL.DOS("CD\LOTUS;123","S")**

Will tell HOSTACCESS to go to DOS, change the directory to LOTUS and call the DOS application 123. If HOSTACCESS has been configured to SWAP out of memory then this will also take place before calling 123. The "S" option is there for backwards compatibility only.

PIX.DOS.PICK is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.DOS.PICK(DOS.FILE.NAME, TRANSFER.DATA, OPTIONS, DESCRIPTION, STATUS.REC, ERROR)**

Transfers data from a DOS file to variable TRANSFER.DATA or to the PICK Spooler. By default it is assumed that the data is textual. Each DOS record is converted into a PICK attribute by converting Carriage Return and Line Feed (CRLF) into an Attribute Mark and the transfer will stop when the DOS end of file marker is found. Options are provided to override this translation and to transfer Binary files that may contain end of file markers.

Click here  to display parameters.

Click here  for an example.

Click here  for information on processing data during transfer to PICK.

The following subroutine parameters need to be passed or will be returned accordingly:

**DOS.FILE.NAME** Name of DOS file to be transferred up to PICK.

**TRANSFER.DATA** This variable returns the string containing the data uploaded. If the data uploaded is greater than MAX.ITEM.SIZE then TRANSFER.DATA contains the word SPLIT followed by space and then a number for the number of times the data has been split. Each SPLIT is held in the PIX.TRANSFER.F file under the names FT.SPLIT.p.s where 'p' is the port number and 's' is the split number, starting with one.

The caller should read these split records in and then delete them as appropriate.

**OPTIONS** Click here  for information on Options.

The I option (which enables programmers to process data as it is transferred to PICK) see [Processing Data during Transfer to PICK](#).

**DESCRIPTION** Any text you want to appear in the file transfer window might as well be null if Z option used.

**STATUS.REC** Dynamic array returned back to calling program containing:

- <1> Finish code with the numeric value of:
  - 0 Transfer successful
  - 1 Aborted at user request
  - 2 Unable to open DOS file
  - 3 DOS file read error
  - 4 Protocol error
  - 5 Retry limit reached
  - 6 DOS file write error
  - 7 File transfer not supported by this copy of HOSTACCESS
  - 8 Exists on file
- <2> Finish message (as above for relevant finish code)
- <3> Reserved for future use
- <4> Number of bytes transferred so far or all
- <5> Number of times data has been split and therefore the number of SPLIT.p.n records in [PIX.TRANSFER.F](#) file.

**ERROR** Returns with a value of 1 if fatal error occurs.

**Note** To detect the success of the file transfer, check the 1st attribute of STATUS.REC, not ERROR. The ERROR variable is reserved for fatal errors such as unable to open the control file [PIX.CONTROL.F](#) and it will not be set if a file transfer error occurs.

**CALL PIX.DOS.PICK("C:\123.BAT",REC,"Z","Description",STATUS,ERR)**

Will transfer the DOS file 123.BAT into variable REC and return to the calling program. Since no 'B' option has been specified, the REC variable will contain attribute marks instead of CRLF's. The Z option means all output is suppressed to the user.

**STATUS** is returned to allow the caller to verify the success of the transfer.

One of the options that you can specify with PIX.DOS.PICK is the I option. This option enables programmers to write user subroutines to process data as it is transferred from DOS to PICK.

### **Warning**

We strongly recommend that the I option is used only by experienced developers. Also, it is the developer's responsibility to validate all relevant parameters, files, etc. before invoking [PIX.DOS.PICK](#) with the I option. We are interested in any comments or feedback from developers who use this option, and we will consider their suggestions for any subsequent implementations.

- I If specified, this option will call a specified user subroutine and suspend file transfer. File transfer is resumed on successful exit from the user subroutine. This option is used by the [PIX.DOS.PICK.ALL](#) and [PIX.PICK.DOS.ALL](#) routines, and developers might wish to refer to these as examples of how to implement dynamic processing of data as it is transferred from DOS to PICK.


To invoke the I option, initialize the parameter TRANSFER.DATA with the following attributes in a dynamic array:

<b>Attr</b>	<b>Description</b>
1	The name of the user subroutine to be called.
2	The number of bytes of transfer data to be sent to the user subroutine for processing.
3	Reserved - should not be used.
4 - 9	Reserved for future use.
10	Attributes 10 onwards may be specified as required by the routine that calls PIX.DOS.PICK and this can be used as a mechanism of passing additional parameters into the user subroutine.

When PIX.DOS.PICK first invokes the user subroutine, it copies the TRANSFER.DATA array into the first element of the dimensioned array called USER.VAR. USER.VAR is passed to the user subroutine and may be manipulated as described below. TRANSFER.DATA is then assigned to the data string transferred from DOS which will then be processed by the user subroutine. In all subsequent calls to the user subroutine, TRANSFER.DATA will contain the DOS data to be processed.

The user subroutine will be invoked with the following parameters.

**SUBROUTINE user.sub(TRANSFER.DATA, BLOCK.STATUS, MAT USER.VAR)**

Click here  for more information.

The following subroutine is invoked by the [PIX.DOS.PICK](#) I option.

**SUBROUTINE user.sub(TRANSFER.DATA, BLOCK.STATUS, MAT USER.VAR)**

Where:

- user.sub** Is the name of the user subroutine that will process the DOS data. This will have been compiled and catalogued as appropriate to your host PICK environment.
- TRANSFER.DATA** Is the data that is to be transferred from DOS and processed. The state of this data string will be determined by the file transfer options (B for Binary) assigned when [PIX.DOS.PICK](#) was called, and the initial TRANSFER.DATA parameters which assign a number of bytes to control when the DOS data is passed for processing by the user subroutine. Click here  to see the effect of this.
- BLOCK.STATUS** This is a flag indicating the status of the data transferred from DOS, set as follows:
- F**First block received from DOS (so may also be used for initialization during first call to the user subroutine.
  - L**The last block received from DOS, may also be the first, so always use INDEX to check the block status.
  - N**The next block received from DOS, i.e. Continue processing the data.
  - E**If the user subroutine returns BLOCK.STATUS as E, the DOS.PICK file transfer will be closed down. You should use this if your user subroutine needs to check for errors and stop processing.
- BLOCK.STATUS will be copied into [DOS.PICK](#) 's STATUS.REC variable prior to closing down the file transfer and returning to the controlling routine. The first attribute of BLOCK.STATUS must be the letter E, attribute 2 may be used to signal different error conditions that have occurred in the user subroutine. For example:

BLOCK.STATUS <1> = "E"

BLOCK.STATUS <2> = "Unable to open XYZ file".

If the user subroutine closes down the file transfer with an error, the controlling program (i.e. the one that called [DOS.PICK](#)) may determine the nature of the error by checking [DOS.PICK](#) 's STATUS.REC variable, Where:

STATUS.REC <1> = 10, the code for user program error.

STATUS.REC <2> = a message returned from the usersubroutine through attribute 2 of BLOCK.STATUS.

You should use INDEX to interrogate the contents of BLOCK.STATUS which may contain more than one flag, for **example** "FL" when the block is both the first and to be transferred from DOS.

**USER.VAR** This variable will be maintained untouched by the [PIX.DOS.PICK](#) routine, apart from the first call to the user subroutine, when it contains in the first element a copy of the initial parameters passed in as TRANSFER.DATA. Remember to use MAT in the parameter list as USER.VAR is a dimensioned array, which should be DIM(25) within the user subroutine.

It is useful, during the first call of the user subroutine, to analyze USER.VAR and to use other elements to, say, store the "open file" variable used for the target PICK file (which means that the file need only be opened once).

Click here  for an example.



**File  
Transfer  
Option Binary**

**Use Number of  
Bytes to chop DOS  
data**

**State of TRANSFER.DATA**

The exact number of bytes as specified by the Number of Bytes parameter.

The number of bytes up to the nearest whole attribute mark, i.e. the data string may be longer or shorter than the Number of Bytes parameter.



This program uses PIX.DOS.PICK with the I option. This option is used in [PIX.DOS.PICK](#) and [PIX.PICK.DOS.ALL](#) which may be used to upload a complete PICK file, previously sent to DOS using PIX.PICK.DOS.ALL. PIX.DOS.PICK.ALL may also be used to import a fixed length or carriage return/linefeed delimited DOS file into a PICK file, to satisfy most users' requirements.

The **example** application program is as follows:

```
USER.SUB = "USER.PROGRAM.NAME"
NBR.BYTES = 5000
TRANSFER.DATA = ""
TRANSFER.DATA<1>=USER.SUB
TRANSFER.DATA<2>=NBR.BYTES
TRANSFER.DATA<10>="TARGET.FILE.NAME"
OPTION = "I"
CALL PIX.DOS.PICK("TEST.DOC", TRANSFER.DATA, OPTION, "Description", STATUS.REC, ERROR)
IF NOT (ERROR) AND STATUS.REC<1> = 0 THEN
PRINT "TARGET.FILE.NAME updated with DOS data from TEST.DOC"
END
```

User subroutine that processes the data as it arrives from DOS:

```
SUBROUTINE USER.PROGRAM.NAME(TRANSFER.DATA, BLOCK.STATUS, MAT USER.VAR)
DIM USER.VAR(25)
FIRST.TIME = INDEX(BLOCK.STATUS,"F",1)
REC = TRANSFER.DATA
IF FIRST.TIME THEN
  TARGET.FILE.NAME = USER.VAR(1)<10>
  OPEN "",TARGET.FILE.NAME TO USER.VAR(2) ELSE
    BLOCK.STATUS<1> = "E"
    BLOCK.STATUS<2> = "Unable to open file"
    RETURN;*abort file transfer
  END
END ELSE
  READ REC FROM USER.VAR(2),"REC.ID" ELSE REC = ""
  IF REC # "" THEN REC=REC:CHAR(254):TRANSFER.DATA
END

WRITE REC ON USER.VAR(2),"REC.ID"
RETURN
END
```

PIX.DOS.PICK.ALL is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.DOS.PICK.ALL(DOS.FILE.NAME, FILENAME, OPTIONS, FORMAT, DESCRIPTION, STATUS.REC, ERROR)**

This subroutine uploads multiple PICK records from a DOS file. This is a very powerful subroutine and is capable of uploading records stored in DOS in a number of formats, including the format created by the PIX.PICK.DOS.ALL subroutine.

If you are uploading a file that was previously created with the [PIX.PICK.DOS.ALL](#) subroutine, the same keys from the original file will be used automatically to write these records.

The following subroutine parameters need to be passed or will be returned accordingly:

<b>DOS.FILE.NAME</b>	Name of DOS file to be transferred up to PICK.
<b>FILENAME</b>	Is the PICK filename where the records are to be created
<b>OPTIONS</b>	Numerous options are available. For a description of all options please refer to <a href="#">Options available DOS.PICK.ALL</a> .

By default, this routine assumes that you are uploading records in the format created by the [PICK.DOS.ALL](#) routine. Using any (or all) of the following options (as well as, or instead of, those mentioned above) overrides this default and automatically generates sequential numeric keys.

Click here  for an example.

Uploading comma delimited DOS files.

You can also use the following options to upload a standard comma-delimited DOS file into multiple PICK records.

- 1 Use <CRLF> as the inter-record delimiter and use commas as the inter-field delimiter.
- 9 Use first field in delimited file as the key instead of using the automated key.
- 0 Strip any double quotes from the data.

Using these options allows any comma-delimited DOS file to be uploaded to the PICK file as multiple records, converting each comma into an attribute mark.

**FORMAT This variable allows even greater control, as follows:**

- <1> Any string specifying inter-record delimiter (default is CR/LF).
- <2> Any string specifying inter-field delimiter.
- <3> Any character string(s). Each character string will be removed from the data. If you want multiple string removals, this variable should be multi-valued.
- <4> Numeric value to use for first key. (Default = 1).
- <5> Any string to use as a prefix when writing the numeric key.
- <6> Any string to use as a suffix when writing the numeric key.
- <7> Multi-valued variable containing a list of specific item ids that you want to upload.

**DESCRIPTION Any text you want to appear in the file transfer window (might be null if Z option used).**

**STATUS.REC Dynamic array returned back to calling program containing:**

- <1>= finish code with the numeric value of:
  - 0 Transfer successful.
  - 1 Aborted at user request.
  - 2 Unable to open DOS file.
  - 3 DOS file read error.
  - 4 Protocol error.
  - 5 Retry limit reached.
  - 6 DOS file write error.
  - 7 File transfer not supported by this copy of HOSTACCESS.
  - 8 exists on file.
  - 10 user program finished or aborted.
- <2> = finish message (as above for relevant finish code).
- <3> = reserved for future use.
- <4> = number of bytes transferred so far or all.
- <5> = number of times data has been split and therefore the number of SPLIT.p.n records in [PIX.TRANSFER.F](#) file.

**ERROR Returns with a value of 1 if fatal error occurs.**

The [DOS.PICK.ALL](#) uploads will stop when all the specified records have been found, regardless of whether the entire DOS file has been uploaded.

**Note** To detect the success of the file transfer, check the 1st attribute of STATUS.REC, not ERROR. The ERROR variable is reserved for fatal errors such as unable to open the control file [PIX.CONTROL.F](#) and it will not be set if a file transfer error occurs.

Click here  for an example.

To upload all PICK records into the DEMO file from a single comma-delimited DOS file called DEMO.TXT using sequential keys starting from 1, and remove any double quotes, use the following subroutine call:

**CALL PIX.DOS.PICK.ALL("DEMO.TXT","DEMO","10", 1, "Uploading demo recs",STATUS.REC,ERROR)**

Any records found in the PICK file with the same key as the autogenerated key will not be overwritten, unless the "O" option is specified.

PIX.CALL.DOS.PICK is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.CALL.DOS.PICK(TCL.INPUT, TCL.OPTIONS, ERROR)**

Transfers data from a DOS file to a PICK item or to the PICK Spooler. By default it is assumed that the data is textual. Each DOS record is converted into a PICK attribute by converting Carriage Return and Line Feed (CRLF) into an Attribute Mark and the transfer will stop when the DOS end of file marker is found. Options are provided to override this translation and to transfer Binary files that do not contain end of file markers.

This program actually calls the subroutine [PIX.DOS.PICK](#) to return the DOS record in a variable which this program then writes to the specified PICK file.

The following subroutine parameters need to be passed or will be returned accordingly:

**TCL.INPUT** Command line string without the VERB ([DOS.PICK](#)) and without the options . In other words, in the format:

**{DICT} FILENAME ITEMNAME FROM dosname**

**FILENAME** and **ITEMNAME** are the PICK file and item to be created with the data from the DOS file dosname.

If the **S** spooler option is used then FILENAME and ITEMNAME can be null. i.e. DOS.PICK FROM C:\AUTOEXEC.BAT (S) will send the DOS file AUTOEXEC.BAT to the PICK spooler.

If the **FROM dosname** parameter is missing it will be prompted for.

**TCL.OPTIONS** Click here  for details. No ` ` or commas required.

**ERROR** Returns 1 if fatal error occurs.

To determine the status of the file transfer, the status record (as described for [PIX.PICK.DOS](#) and [PIX.DOS.PICK](#)) is written to the file [PIX.TRANSFER.F](#) with a key of **nn.FT.LOG**, where **nn** is the port number.

Click here  for an example.

**CALL PIX.CALL.DOS.PICK("BP TEST FROM C:\123.BAT","ZO",ERROR)**

Will transfer the DOS file 123.BAT to the PICK file BP, item TEST. The 'Z' option suppresses ALL output to the user. The 'O' option forces overwrite of the TEST item if it exists in BP.

PIX.CALL.PICK.DOS is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.CALL.PICK.DOS(TCL.INPUT, TCL.OPTIONS, ERROR)**

Main PICK.DOS program that calls PIX.PICK.DOS to actually perform the transfer. This program verifies the item and file and passes the actual record as a dynamic array to the subroutine [PIX.PICK.DOS](#)

The following subroutine parameters need to be passed or will be returned accordingly:

**TCL.INPUT** Command line string without the VERB ([PICK.DOS](#)) and without the options. In other words, in the format :  
{DICT} FILENAME ITEMNAME TO dosname  
FILENAME and ITEMNAME are the PICK file and item to be sent to the DOS file dosname.

**TCL.OPTIONS** Click here  for details. No ' ' or commas required.

**ERROR** Returns 1 if fatal error occurs.

To determine the status of the file transfer, the status record (as described for [PIX.PICK.DOS](#) and [PIX.DOS.PICK](#)) is written to the file [PIX.TRANSFER.F](#) with a key of **nn.FT.LOG**, where **nn** is the port number.

Click here  for an example.

**CALL PIX.CALL.PICK.DOS("BP TEST TO C:\123.BAT", "H", ERROR)**

Will transfer the PICK item TEST from the file BP to the DOS file 123.BAT. The **H** option will still display HOSTACCESS 's file transfer status window but the window is closed automatically when the transfer has finished, i.e. no user intervention required.



PIX.PICK.DOS is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.PICK.DOS(TRANSFER.DATA, DOS.FILE.NAME, OPTIONS, DESCRIPTION, STATUS.REC, ERROR)**

Transfers data in variable TRANSFER.DATA to a DOS file. By default the data is assumed to be text and attribute marks will be converted to Carriage Return/Line Feeds (CRLF), value marks and sub-value marks will be converted to spaces and a DOS end of file character will be written to the end of the DOS file. Options are provided to override this translation.

The following subroutine parameters need to be passed or will be returned accordingly:

<b>TRANSFER.DATA</b>	The variable containing the data you want transferred to DOS.
<b>DOS.FILE.NAME</b>	The name of the DOS file you wish to transfer the PICK data to. This can optionally include the drive letter and/or full path. If no drive/path specified, DOS file will be created in the current HOSTACCESS directory.
<b>OPTIONS</b>	Numerous options are available, click here <input type="checkbox"/> for more information.
<b>DESCRIPTION</b>	Any text you want to appear against the "Host file name :." in the file transfer window, might as well be null if Z option used.
<b>STATUS.REC</b>	Dynamic array returned back to calling program containing :  <1> = finish code with the numeric value of: 0 Transfer successful. 1 Aborted at user request. 2 Unable to open DOS file. 3 DOS file read error. 4 Protocol error. 5 Retry limit reached. 6 DOS file write error. 7 File transfer not supported by this copy of HOSTACCESS.  <2> = finish message (as above for relevant finish code).  <3> = reserved for future use.  <4> = no. of bytes transferred.
<b>ERROR</b>	Returns 1 if fatal error occurs.

**Note:** To detect the success of the file transfer, check the 1st attribute of STATUS.REC, not ERROR. The ERROR variable is reserved for fatal errors such as unable to open the control file [PIX.CONTROL.F](#) and it will not be set if a file transfer error occurs.

Click here  for an example.

**CALL PIX.PICK.DOS(REC,"C:\123.BAT","Z","Xfer",STATUS,ERR)**

Will transfer the data in the variable REC into the DOS file 123.BAT and return to the calling program. Since no 'B' option has been specified, any attribute marks in REC will be converted to CRLF's in the DOS file. The 'Z' option means all output is suppressed to the user. STATUS is returned to allow the caller to verify the success of the transfer.

PIX.PICK.DOS.ALL is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.PICK.DOS.ALL(PICK.FILE.NAME, DOS.FILE.NAME, OPTIONS, DESCRIPTION, STATUS.REC, ERROR)**

This subroutine creates a single DOS file from an active select list or from an entire file. The DOS file will contain all the PICK selected records and their keys in an internal format specific to HOSTACCESS. The records in the file can then be uploaded again using PIX.DOS.PICK.ALL.

[PIX.TDUMP](#) can also be used to download multiple PICK records, but has the disadvantage of creating a DOS file for each record. When used with PIX.TDUMP, this has the advantage that it is capable of quickly uploading specific records.

The following subroutine parameters need to be passed or will be returned accordingly:

<b>PICK.FILE.NAME</b>	Is the PICK filename containing the records that you want transferred to DOS.
<b>DOS.FILE.NAME</b>	The name of the DOS file you wish to transfer the PICK data to. This can optionally include the drive letter and/or full path. If no drive/path specified, DOS file will be created in the current HOSTACCESS directory.
<b>OPTIONS</b>	See <a href="#">Options Available With DOS.PICK.ALL</a>
<b>DESCRIPTION</b>	Any text you want to appear against the "Host file name :." in the file transfer window, might as well be null if Z option used.
<b>STATUS.REC</b>	Dynamic array returned back to calling program containing :  <1> = finish code with the numeric value of: 0 Transfer successful 1 Aborted at user request 2 Unable to open DOS file 3 DOS file read error 4 Protocol error 5 Retry limit reached 6 DOS file write error 7 File transfer not supported by this copy of HOSTACCESS  <2> = finish message (as above for relevant finish code)  <3> = reserved for future use  <4> = no. of bytes transferred
<b>ERROR</b>	Returns 1 if fatal error occurs.

**Note:** to detect the success of the file transfer, check the 1st attribute of STATUS.REC, not ERROR. The ERROR variable is reserved for fatal errors such as unable to open the control file PIX.CONTROL.F and it will not be set if a file transfer error occurs.

Click here  [PIX.PICK.DOS.ALL](#) for an example.

**EXECUTE 'SELECT SALES WITH CODE = "ARCHIVE" '**

**CALL PIX.PICK.DOS.ALL("SALES","SALES.HIS","", "Sales Archive",STATUS.REC,ERR)**

This **example** will create a single DOS file called SALES.HIS in the HOSTACCESS run-time directory. SALES.HIS will contain all the PICK records selected from the SALES file. This DOS file is in the format that is readable by the PIX.DOS.PICK.ALL routine to bring the records back up to the PICK system when required. STATUS.REC is returned to allow the caller to verify the success of the transfer.

To download selected records from the PICK file STAFF into a single DOS file called STAFFFILE.ALL, use the following:

**EXECUTE 'SELECT STAFF WITH AGE > "65" '**

**CALL PIX.PICK.DOS.ALL("STAFF","C:\FILES\STAFFFILE.ALL","",  
"Downloading Staff recs",STATUS.REC,ERROR)**

This DOS file STAFFFILE.ALL may be subsequently uploaded to a PICK file using [DOS.PICK.ALL](#) (or a call to the subroutine PICK.DOS.PICK.ALL).

PIX.TDUMP is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.TDUMP(TCL.INPUT, TCL.OPTIONS, PATH.NAME, TAPE.NAME, ERROR)**

This routine performs the same way as PICK T-DUMP except that instead of writing to tape, the records are sent to DOS.

You will be asked for a 'PSUEDO TAPE' name of up to 8 chars in length. This name should be unique to allow you to identify this tape in the future and a valid DOS file name. You will then be asked for a DOS path name where you want the tape called 'name' to be stored.

For example, call your tape PROGRAMS and gave a directory of "\TAPES".

In the directory TAPES after a TDUMP will be the following:

**PROGRAMS.PIX** An internal index used to identify what 's in 'TAPE' .  
**PROGRAMS.001** The first record.  
**PROGRAMS.002** The second record.  
**PROGRAMS.xxx** Etc., Up to ".999".  
**PROGRAMS.XRF** Index to identify DOS name given to each PICK name so that the user can select individual items at DOS.

If a PICK SELECT list is active, i.e. you have executed a SELECT statement or a GET-LIST immediately prior to invoking TDUMP, then his list will be applied to the named file.

The following subroutine parameters need to be passed or will be returned accordingly:

**TCL.INPUT** Command line without the TCL command, i.e.  
{DICT} PROGRAMS {\*} {itemname}  
**TCL.OPTIONS** Valid options are basically the same as those for PICK.DOS but 'Z' is likely to be the most common one used to stop the display of each file transfer window for each record. The 'S' option will suppress the display of the record id 's to the screen. The 'O' option will overwrite the DOS files if they already exist.  
**PATH.NAME** DOS path to directory where 'tape' is to be stored. If null then 'tape' will be created in the HOSTACCESS run-time directory.  
**TAPE.NAME** DOS file name (without suffix) to be used as 'Tape' .  
**ERROR** Returned as 1 if error occurred.

PIX.TLOAD is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.TLOAD(TCL.INPUT, TCL.OPTIONS, PATH.NAME, TAPE.NAME, ERROR)**

This program uploads TDUMP format DOS files back to the host system. The following subroutine parameters need to be passed or will be returned accordingly:

**TCL.INPUT** Command line without the TCL command, i.e. {DICT} PROGRAMS  
**TCL.OPTIONS** The options available are:  
? Cause TDUMP to display a list of the items on the specified file, then allow you to enter either selected item names or '\*' to upload all items. The user should enter item names separated by space.  
O Overwrite existing PICK items, with the same record id.  
**PATH.NAME** DOS path to directory where 'Tape' is stored. If null then the HOSTACCESS directory is used.  
**TAPE.NAME** DOS file name (without suffix) holding 'Tape' records.  
**ERROR** Returned as 1 if error occurred.

PIX.ERASE.DOS.FILE is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.ERASE.DOS.FILE(DOS.FILE)**

Subroutine to erase a single DOS file. This routine is useful if you want to make sure the file is deleted, for **example**, before starting terminal printing to a DOS file.

No screen output occurs, so user is unaware of anything happening. If you want to check if the file exists first then CALL PIX.EXISTS(DOS.FILE,CHECK) and see that CHECK is not 0 or 2, i.e., does not exist or not a directory, respectively.

DOS.FILE can be the complete drive and path parameters. If no path is specified, it will delete that file from the current HOSTACCESS directory. If DOS.FILE evaluates to being a path name only then the erase will not take place.

Only the whole file name is valid and only a SINGLE DOS file is deleted i.e. NO wildcards \*.\* or \*.DOC etc.

The following subroutine parameters need to be passed or will be returned accordingly:

**DOS.FILE** Can be a DOS file with or without a drive and/or path destination. If no path is specified, the default current HOSTACCESS directory will be assumed, that is If DOS.FILE is C:\TEMP\TEST.DOC then the DOS file TEST.DOC will be deleted from the TEMP directory in PC drive 'C'.

Click here  for an example.

**CALL PIX.ERASE.DOS.FILE("C:\LOTUS\BACKUP.WK1")**

Will delete the DOS file BACKUP.WK1 from the LOTUS directory.

PIX.EXISTS is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.EXISTS(DOS.FILE, CODE)**

This program is extremely useful if you want to check DOS for the existence of a file or directory e.g. before deleting it or uploading it or validating user response.

The following subroutine parameters need to be passed or will be returned accordingly:

**DOS.FILE** DOS.FILE should be the FULL path including the drive if necessary.

**CODE** Returned as :

0 - DOS directory or file does not exist

1 - exists and is a DOS file

2 - exists and is a DOS directory

Click here  for an example.



**CALL PIX.EXISTS("C:\LAST.LOG",CODE)**

Will return **CODE** back to the calling program indicating whether **LAST.LOG** exists in the DOS root directory as either a file or a directory.

<a href="#"><u>EASY ACCESS</u></a>	PIX.EASY.ACCESS
<a href="#"><u>Extract data from a PICK file.</u></a>	PIX.GET
<a href="#"><u>PASS.To routines</u></a>	PIX.PASS.TO
<a href="#"><u>PASS.TO spreadsheet.</u></a>	PIX.PASS.TO.Spreadsheet.Product
<a href="#"><u>PASS.To Word Processing product.</u></a>	PIX.PASS.TO.Word.Processing.Product
<a href="#"><u>Create a list of dictionaries.</u></a>	PIX.CREATE.WORD.DICT
<a href="#"><u>Create a DOS file in DIF format.</u></a>	PIX.PASS.TO.DIF
<a href="#"><u>Create a DOS file in a specified format.</u></a>	PIX.PASS.TO.DOS

PIX.EASY.ACCESS is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.EASY.ACCESS(TCL.INPUT, TCL.OPTIONS)**

This routine provides a front end to all flavors of the PICK inquiry language, such as ACCESS, INFORM, ENGLISH, RECALL, etc.

This is a tremendously powerful yet very easy to use facility that should really be integrated in virtually every PICK database. It combines a sophisticated sentence builder with most of the data extraction and transfer routines available in HOSTACCESS. This means that, from just one routine, the user can now inquire on his PICK database, extract the relevant information and automatically pass this in to almost any desired DOS package ( even 3 dimensionally graph it) with only a few keystrokes!

The parameters TCL.INPUT and TCL.OPTIONS should be exactly as if specified by a user on the TCL command line.


The 'F' option is more relevant to applications calling PIX.EASY.ACCESS since specific options and facilities can be FORCED either on or off.

PIX.GET is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.GET (TCL.INPUT, SORT.CRITERIA, TCL.OPTIONS, LIST.NAME, ERROR)**

PIX.GET has been designed to extract data from any PICK file in preparation for transfer of this data to a DOS package or file. The data extracted by GET is used by one of the PASS.TO... routines.

PIX.GET will process virtually any inquiry and uses all of the facilities available to the user from PICK Dictionary definitions, including "A" and "F" correlatives and "I-types".

Click here  for full information on GET.

The parameters should be passed almost exactly as if the user had typed them in on the TCL command line.

<b>TCL.INPUT</b>	Command line without the TCL command GET.
<b>SORT.CRITERIA</b>	Optional sort/select criteria string, e.g. 'BY NAME WITH NAME EQ "SMIT]" '.
<b>TCL.OPTIONS</b>	T and/or P option string.
<b>LIST.NAME</b>	Optional PICK SELECT-LIST name.

...HOSTACCESS provides a powerful facility for PICK users to be able to pass data directly into DOS or Windows products using the PIX.PASS.TO subroutines. It is not necessary for a user to understand how to read or import data into the DOS or Windows software since HOSTACCESS, through the [pass](#) subroutines, can do everything automatically.


In this topic, the meaning "PASS.TO" implies all of the HOSTACCESS pass to DOS or Windows subroutines, for example PIX.PASS.TO.EXCEL, PIX.PASS.TO.LOTUS, PIX.PASS.TO.SUPERCALC, PIX.PASS.TO.SYMPHONY, PIX.PASS.TO.QUATTRO, PIX.PASS.TO.WORD, or PIX.PASS.TO.WORDPERFECT. The PIX.PASS.TO subroutines fall into three categories:

- [PIX.PASS.TO spreadsheet](#) packages - PIX.PASS.TO.EXCEL, PIX.PASS.TO.LOTUS, PIX.PASS.TO.SUPERCALC, PIX.PASS.TO.SYMPHONY, and PIX.PASS.TO.QUATTRO
- [PIX.PASS.TO word processing](#) packages - PIX.PASS.TO.WORD and PIX.PASS.TO.WORDPERFECT
- [PIX.PASS.TO.DIF](#) and [PIX.PASS.TO.DOS](#).

All the PASS.TO routines can be driven from PICK/BASIC as subroutine calls, or they can be driven directly as TCL commands. They can also be driven or are available as command interfaces from [EASY.ACCESS](#).

Each PASS.TO application has a control record in the [PIX.CONTROL.F](#) file, and any application that supports both DOS and Windows has 2 records, the Windows record having a suffix of .WINDOWS. The PASS.TO control records also hold information on graph types (where relevant) and how to start and run the DOS or Windows application.

All the PASS.TO routines will pass the data from the last [GET](#) inquiry into the particular DOS or Windows software package or file format such as LOTUS, SuperCalc, DIF etc.

Click here  for further information.

**Note 1** For backwards compatibility, the COMMAND parameter in all the PASS.TO application subroutines still exists. However, we recommend that you leave this null and that the DOS or Windows command is stored in attribute 3 of the relevant control record in the PIX.CONTROL.F file. For example, with PASS.TO.LOTUS the stored command would read:

```
003 123
```

and with PASS.TO.LOTUS.WINDOWS the stored command would read:

```
003 123W.EXE
```

This is based on the assumption that your DOS PATH variable has been correctly configured to point to this command. We recommend that you use this standard for calling all DOS or Windows applications through HOSTACCESS.

The command used to start the spreadsheet application is retrieved by reading attribute 3 of the PASS.TO. application record in the [PIX.CONTROL.F](#) file. If the 'nth' multi-value ('n' being port number plus 2) is not null then the command stored there is used. If this multi-value *is* null then the command stored in multi-value 1 is used. We recommend that everyone on the host calls the particular DOS or Windows product in the same way. That is, they should use multi-value position 1 only. This is easier on networks, for example, where the port number is not always known.

**Note 2** When you use PASS.TO routines, temporary DOS files are created. If data is being passed to a DOS package, the temporary files are subsequently removed automatically. However, if data is being passed to a Windows product, the temporary files cannot be deleted owing to Windows multi-tasking environment. Your application may want to delete this temporary DOS file or just leave it to be overwritten by the next PIX.PASS.TO routine. The temporary DOS file is called PIXEL $nn$ .TMP (in PIX.PASS.TO.QUATTRO it is called PIXEL $nn$ .DIF) where  $nn$  is the host port number.

**Note 3** EXCEL, WORD, and the Windows versions of QUATTRO PRO, and LOTUS, can be DDE servers. Therefore, data that is passed to or merged into these packages using the PIX.PASS.TO routines is done using the DDE calls and subroutines built into HOSTACCESS for Windows.

These subroutines include PIX.PASS.TO.EXCEL, PIX.PASS.TO.LOTUS, PIX.PASS.TO.QUATTRO, PIX.PASS.TO.SUPERCALC and PIX.PASS.TO.SYMPHONY.

PIX.PASS.TO.**productname** are BASIC subroutines that can be called from your applications as follows:

**CALL PIX.PASS.TO.productname(COMMAND,GRAPH.TYPE,ERROR)**

Where:

**productname** Is EXCEL, LOTUS, QUATTRO, SUPERCALC or SYMPHONY.

This program automatically calls and passes the result from the last GET statement into the relevant spreadsheet product.

The spreadsheet product need not already be started on your desktop, HOSTACCESS will determine if it is already running and, if not, it will start it accordingly. Data will then be transferred and imported automatically whilst the spreadsheet product is running in the foreground if HOSTACCESS started it, or in the background if it was already running.

The following subroutine parameters need to be passed or will be returned accordingly:

**COMMAND** This parameter should be null. (See Note 1 in [PIX.PASS.TO notes.](#))

**GRAPH.TYPE** if null, the data will be imported and the user will be taken into the spreadsheet.

You may specify the graph type either by using the name of the graph or by using a number that corresponds to the required graph type. A list of graph type names is held in attribute 2 of the control record PASS.TO.productname in the PIX.CONTROL.F file. The graph type number should be the number of the multi-value in attribute 2 of this record.

A list of graph type descriptions can be viewed easily in the appropriate menu option when using [EASY.ACCESS.](#)

**ERROR** Returns 0 if successful, 1 if error occurred.

**Note:** QUATTRO PRO has 2 anomalies (at time of going to press), one of which is the 128 byte limit on DDE commands. This means that each DDE instruction causes the QUATTRO PRO screen to flash. The second is that FILE OPEN forces QUATTRO PRO to the foreground application which shows the user everything that is happening.

These subroutines include PIX.PASS.TO.WORD and PIX.PASS.TO.WORDPERFECT.

PIX.PASS.TO.**productname** is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.PASS.TO.productname (COMMAND,DOC.NAME,ERROR)**

Where:

**productname** Is WORD or WORDPERFECT.

This program automatically calls and passes the result from the last GET statement into the relevant word processing product.

The spreadsheet product need not already be started on your desktop, HOSTACCESS will determine if it is already running and, if not, it will start it accordingly. Data will then be transferred and imported automatically whilst the spreadsheet product is running in the foreground if HOSTACCESS started it, or in the background if it was already running.

The following subroutine parameters need to be passed or will be returned accordingly:

**COMMAND** This parameter should be null. (See Note 1 in the [PIX.PASS.TO notes.](#))

**DOC.NAME** The name of the document that you want the data to be passed into. If no DOC.NAME is specified, it is prompted for.

**ERROR** Returns 0 if successful, 1 if error occurred.

**Note 1** Because (at time of going to press) WordPerfect does not support being a DDE server, HOSTACCESS sends the keystrokes to WordPerfect to automate the MERGE. The only disadvantage is that:

- a)WordPerfect if already started must have no documents already open and must have a window title of WordPerfect [Document1 - unmodified].
- b)The user will see every keystroke as it is replayed and the corresponding screen updates.
- c)Wordperfect *must* remain as the *active* foreground application (not minimized) and *cannot* be interrupted. Unlike DDE, keystrokes can only be sent to the currently active application - if the user changes this, the mailmerge will fail.

**Note 2** PIX.PASS.TO.WORD users can use the subroutine [PIX.CREATE.WORD.DICT](#) to help build WORD header files from PICK dictionaries.

PIX.CREATE.WORD.DICT is a BASIC subroutine that creates a DOS file containing a list of dictionaries for a given PICK file. The DOS file can then be used by WORD for Windows users to assist in the creation of mailmerge documents. (We hope you find this useful as we do when producing mailing lists for HOSTACCESS users).

PIX.CREATE.WORD.DICT can be called from your application as follows:

**CALL PIX.CREATE.WORD.DICT (PICK.FILE.NAME, DOS.FILE.NAME, ERROR)**

This program creates a DOS file that allows anyone using Word to call up a list of dictionaries of a given PICK file by using the Print Merge menu and attaching this file as the ATTACH HEADER file. Clicking on the insert field button in Word will then reveal, as a select list, all the given fields in this file.

The following parameters need to be passed or returned accordingly:

**PICK.FILE.NAME** The actual PICK file name in this account to use for the list.

**DOS.FILE.NAME** The full DOS path and DOS file name to store the result in. Normally this path will be where WORD has easy access to it.

**ERROR** Returns 1 if fatal error occurs.



PIX.PASS.TO.Word Processing Product.

PIX.PASS.TO.DIF is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.PASS.TO.DIF(DOS.FULL.PATH, ERROR)**

This program creates and sends a DIF file to DOS using the result of the last GET statement.

The following subroutine parameters need to be passed or will be returned accordingly:

**DOS.FULL.PATH** Should be set to full drive:\path\file where you wish to store the file. If null the file PIXELnnn.TMP (where 'nnn' is PICK user's port number) is created in the main HOSTACCESS run-time directory.

**ERROR** Returned as 0 if successful, 1 if error occurred.

PIX.PASS.TO.DOS is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.PASS.TO.DOS(DOS.FULL.PATH, FIRST.DELIMITER, SECOND.DELIMITER, OPTIONS, ERROR)**

This program passes the data from the last GET statement to the specified DOS file using delimiters passed in the variables FIRST.DELIMITER and SECOND.DELIMITER or by setting options 1 to 4 for some specific DOS defaults.

The following subroutine parameters need to be passed or will be returned accordingly:

**DOS.FULL.PATH** e.g. C:\HOST\TEXT.TXT  
Should be set to full drive:\path\file where you wish to store the file. If null the file PIXELnnn.TMP (where 'nnn' is PICK user's port number) is created in the main HOSTACCESS run-time directory.

**FIRST.DELIMITER** DOS delimiter to the LEFT of each field.

**SECOND.DELIMITER** DOS delimiter to the RIGHT of each field  
If FIRST and SECOND delimiters are BOTH null then a comma delimited file will be created as default.

**OPTIONS** The following options are available as defaults :  
1 Comma delimited file  
2 Quote delimited file  
3 Space delimited file  
4 Fixed Length delimited file where field dictionary width fixes the length.  
Others can be added very easily if you look at the option process code. Let Pixel know about any others you are adding and we will be happy to put relevant enhancements into the standard product and therefore fully support them.

**ERROR** Returned as 0 if successful, 1 if error occurred.

[Display environment](#)

[Return HOSTACCESS's serial number to the host.](#)

[Return information about HOSTACCESS](#)

[Return HOSTACCESS's file path](#)

[Activate command stacker](#)

[Turn on all keys on a DOS keyboard](#)

[Change print device](#)

[Display fonts](#)

PIX.ENVIRONMENT.

PIX.GET.SERIAL.NO.

PIX.TERMITE.INFO.

PIX.TERMITE.DIRECTORY.

PIX.COMMAND.STACK.

PIX.SCAN.KEYS.

PIX.LOCAL.PRINT.DEVICE.

PIX.SHOW.PC.FONT.S.



[TCL environment](#)

[Activate pop-up calculator](#)

[Activate database utilities](#)

[Select a cursor](#)

PIX.TCL.

PIX.CALC.

PIX.DB.

PIX.CURSOR.

PIX.ENVIRONMENT is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.ENVIRONMENT(ERROR)**

This subroutine displays the current HOSTACCESS ENVIRONMENT. The record ENVIRONMENT is held in the [PIX.CONTROL.F](#) file.

Do not change this record - if you do need to change any of the attributes, please contact PIXEL for advice first.

<b>Attribute</b>	<b>Description</b>
<b>1</b>	Machine type (set at install time)
<b>2</b>	Account name (main installed HOSTACCESS account)
<b>3</b>	Reformat type as one of the following :
<b>0</b>	Not supported
<b>1</b>	Generic pick,
<b>2</b>	ID-SUPP bug so GET has to fix it automatically.
<b>4</b>	Maximum item size, 30000 is the default except for Prime, UniVerse, UniData and McDonnell Douglas 7.0+ when this is set to 5120000 (5Mb).
<b>5</b>	System supports char(255) in data and spooler. This will be either :
<b>0</b>	Host does not support char(255) Host can support char(255) Prime, Universe, Unidata and McDonnell Douglas 7.0+ all support 255 so this is set to 1 at install time.

PIX.GET.SERIAL.NO is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.GET.SERIAL.NO(SERIAL.NO)**

Each copy of HOSTACCESS is given a unique serial number and when it is installed on a PC it is targeted to that PC 's hard disk.

This program returns HOSTACCESS 's unique serial number to the Host application.

The following subroutine parameters need to be passed or will be returned accordingly:

<b>SERIAL.NO</b>	Returned as the serial number of HOSTACCESS running in this PC.
	The first two digits of the serial number may be used to find out which version of HOSTACCESS is being used, where:
1st 2 digits	HOSTACCESS version number
6n	HOSTACCESS for Windows version 6.n (e.g. 6.0, 6.1 etc.)

PIX.TERMITE.INFO is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.TERMITE.INFO(STATUS.REC)**

This subroutine passes back information about the current HOSTACCESS. The following subroutine parameters need to be passed or will be returned accordingly:

<b>STATUS.REC</b>	Will be returned as a dynamic array as follows:
<1>	1 if HOSTACCESS for Windows running or 0 if DOS version
<2>	1 if current PC is color, 0 if mono
<3>	1 if blinking is enabled on PC, 0 if not
<4>	1 if this PC has a mouse that HOSTACCESS can use, 0 if not

**Note** This routine combines information from previous subroutines to allow new information to be added to later without the need to change host application code. So don 't be surprised if you get more than 4 attributes back in future releases.

PIX.TERMITE.DIRECTORY is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.TERMITE.DIRECTORY(PATH)**

This routine returns the full DOS path including the drive that HOSTACCESS is running from. This is really useful if you need to know a valid disk drive and/or directory that you can write to.

The PIX.PASS.TO routines rely on this to write the PIXELnnn.TMP file and then tell DOS products what drive and directory this file is in.

The following subroutine parameters need to be passed or will be returned accordingly:

**PATH** Is returned for example as C:\host or perhaps A:\ if running from a floppy disk.

PIX.COMMAND.STACK is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.COMMAND.STACK(ON.OFF)**

HOSTACCESS by default logs every command entered at TCL, UNIX shell, ORACLE etc., and then presents a list of the last 'n' commands when you press ALT/R in HOSTACCESS. Commands can be modified and/or sent back to the host again for resubmission.

This is a very powerful command line stacker for all operating systems and obviously gives you a standard stacker even for machines that don't normally support one.

This program tells HOSTACCESS to STOP updating HOSTACCESS's RECALL command stacker (ALT/R). You may want to turn this off around passwords, also perhaps around editors and wordprocessors or in applications that process user input on a single character by character basis.

The following subroutine parameters need to be passed or will be returned accordingly:

**ON.OFF** Should be set to the words 'ON' or 'OFF', nothing happens if neither word is sent.

PIX.SCAN.KEYS is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.SCAN.KEYS(ON.OFF)**

SCAN.KEYS are used when an application requires access to all of the keys on the DOS keyboard which may or may not be available from the particular emulation they are in. By turning scan keys on, HOSTACCESS will return a scan code for all of the non-ASCII keys from the keyboard. This code will be the same, no matter what the emulation is.

If an application has detected that HOSTACCESS is active, it can set the Scancodes ON in order to take control of all 48 function keys, arrow keys, ALT combinations, CTRL combinations etc. If this session has been KEYBOARD configured with REMOTE PAGE KEYS then PAGEUP/DOWN will also generate a SCAN code. If this is set to LOCAL then HOSTACCESS will use PAGEUP/DOWN for control of history pages. Setting SCAN.KEYS to off will reset all keys back to their emulation specific function and/or in the case of function keys back to their user/application programmed sequences. The following subroutine parameters need to be passed or will be returned accordingly:

**ON.OFF** Should be set to the words 'ON' or 'OFF'.

PIX.LOCAL.PRINT.DEVICE is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.LOCAL.PRINT.DEVICE(DEVICE.NAME)**

Subroutine to change the current print device from the printer specification option on the menu **[Configure>Session>Printer>Print device]** in the Print setup... option from the Configure menu in HOSTACCESS for Windows. All LOCAL printing will then go to this print device which can either be a DOS file name including the path (if no path specified then a file will be created in the directory HOSTACCESS is currently running from).

If DEVICE.NAME is a File, then the file should be ERASED first if you do not wish HOSTACCESS to append any existing data in the file.

The following subroutine parameters need to be passed or will be returned accordingly:

**DEVICE.NAME** DEVICE.NAME is the DOS device or filename into which

all print output should be directed for this session. It could be LPT1, LPT2, COM1, COM2 if you have a printer on one of those devices or it could be a DOS file name.

**Note:** Remember to first erase this file, if you do not printing is appended to any data within it.

PIX.SHOW.PC.FONT is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.SHOW.PC.FONT**

This routine allows the user to select from four font tables, each displaying a set of PC characters.

This routine is only intended as a demonstration. To see how it works, select the relevant menu option in the DB command.

PIX.TCL is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.TCL**

This program can be called from any program to give a new TCL environment for executing programs. The calling programs application screen is stored in the slot stack and restored as is before returning to the application.

This program will use SLOT number 16 and refresh the screen as last used by this program.

PIX.CALC is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.CALC(START.VALUE, RETURNED.VALUE)**

This is a pop-up calculator routine, may be called from an application or invoked from TCL via CALCIT verb. This powerful program as much as anything else demonstrates many of the HOSTACCESS facilities. From opening windows, colour, drawing intelligent merging lines and even how to utilize the mouse. This program is not intended to be a replacement for the office calculator but it does prove just what can be done on PICK systems with BASIC HOSTACCESS.

The following subroutine parameters need to be passed or will be returned accordingly:

**START.VALUE**                    Optional value to load into calculator at start, may be null.

**RETURNED.VALUE**            **CURRENT.RESULT** returned upon **EXIT**

PIX.DB is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.DB**

This program provides a PICK/BASIC programmer with a comprehensive set of utilities for editing, copying, formatting, compiling, cataloging routines as well as a gateway back into TCL. It is unlikely that you will really need to call this as a subroutine since we provide a TCL command called [DB - A DATA BASIC Programmer 's Utility](#) to do same thing. All of the facilities are presented through a pop-down menu structure, which may be easily modified to suit the programmer 's host PICK flavor, INFORMATION, REALITY, etc.



PIX.CURSOR is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.CURSOR(TYPE, ON.OFF)**

This program allows you to choose between a block or line cursor or even no cursor at all.

The following subroutine parameters need to be passed or will be returned accordingly:

**TYPE** Should be the words 'BLOCK ' or 'LINE ' or null  
**ON.OFF** Should be the words 'ON ' or 'OFF ' or null. If ON is selected and no type is selected then the currently active cursor shape is turned on.

It is VALID to pass both parameters if you wish.

<a href="#">Determines whether a Windows application is running.</a>	PIX.WIN.RUNNING
<a href="#">Start a Windows Program.</a>	PIX.WIN.RUN
<a href="#">Control the Windows state.</a>	PIX.WIN.CHANGE.STATE
<a href="#">Display a PCX image file.</a>	PIX.DISPLAY.IMAGE
<a href="#">Display a Windows image.</a>	PIX.WIN.DISPLAY.IMAGE
<a href="#">Sends keys in DOS Keyboard stacker format to specified Windows application.</a>	PIX.WIN.SEND.KEYS

PIX.WIN.RUNNING is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.WIN.RUNNING(APP.NAME,RUNNING)**

This subroutine ascertains whether a Windows application is running or not.

**Note** If the application supports being a DDE server, it is recommended that you call [PIX.DDE.INITIATE](#) to detect whether that server application is active or not, because using the server name is more accurate and reliable than depending on APP.NAME.

The following subroutine parameters need to be passed or will be returned accordingly:

**APP.NAME** This must be the name of the Windows application, normally this is the name held in the Window Title as shown exactly on the desktop, e.g.  
WORDPERFECT [DOCUMENT1 - UNMODIFIED]  
**RUNNING** Returned as 1 if the application is running on the desktop, 0 if not.

PIX.WIN.RUN is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.WIN.RUN(WIN.PROGRAM.NAME,STATE,SUCCESSFUL)**

This subroutine allows any Windows program to be started on the desktop.

The following subroutine parameters need to be passed or will be returned accordingly:

**WIN.PROGRAM.NAME**  
This is the drive:\path\winprograme of the Windows program you wish to start e.g. D:\123W\123W.EXE  
If you do not specify a drive and path, the Windows program must be defined in the DOS path so that it can be found.  
You can also specify startup parameters here, for instance  
d:\123W\123W.EXE DEMO.WK3

to start 123 for Windows and open DEMO.WK3 worksheet

- STATE** Is the state in which you want WIN.PROGRAM.NAME to be started. For example:
- 1 - Activates and displays the Window
  - 2 - Activates and minimizes the Window
  - 3 - Activates and maximizes the Window
  - 7 - Displays the window minimized but does not change active window
- SUCCESSFUL** Returned as 1 if WIN.PROGRAM.NAME was started successfully, 0 if not.

PIX.WIN.CHANGE.STATE is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.WIN.CHANGE.STATE(APP.NAME,STATE,SUCCESSFUL)**

This subroutine provides control over the Window state (Minimize, Maximize etc.) of a given application already running on the Windows desktop.

**Note:** It is recommended that you use [DDE](#) if it is available and practical because using the server name is more accurate and reliable than depending on APP.NAME.

The following subroutine parameters need to be passed or will be returned accordingly:

- APP.NAME** This must be the name of the Windows application. Normally this is the name held in the Window Title as shown exactly on the desktop, e.g. WORDPERFECT [DOCUMENT1 - UNMODIFIED]
- If APP.NAME is null, the state change will affect the current HOSTACCESS window.
- STATE** Changes the State of APP.NAME 's window as follows:
- 1 - Activates and displays the window
  - 2 - Activates and minimizes the window
  - 3 - Activates and maximizes the window
  - 7 - Displays the window minimized but does not change active window
- SUCCESSFUL** Returned as 1 if the state change was successful, 0 if not.

PIX.WIN.DISPLAY.IMAGE is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.WIN.DISPLAY.IMAGE(IMAGENAME,TITLE,SCALE,STATE,OPTIONS, STATUS)**

Where:

- IMAGENAME** Is the DOS image file name (including drive and path where necessary) of the image to be displayed. The imagename must be a valid PCX image file.
- TITLE** Is an optional title that will be displayed in the Application Name bar. If omitted, "Image [IMAGENAME]" will be used.
- SCALE** Is an optional integer representing the percentage of the size of the original image to be displayed. The default is 100 (same size). SCALE can be any number greater than 0, such as:
- 25 ¼ size
  - 50 ½ size
  - 75 ¾ size
  - 200 x2 size
  - 400 x4 size
- STATE** Specifies how you want the image displayed, as follows:

- 1 Activates and displays window.
- 2 Activates and minimizes window.
- 3 Activates and maximizes window.
- 7 Displays and minimizes but does not make active.

**OPTIONS**

The following options are available:

- F** Specifies that the image is to fit into the size of the image display window. If specified, this means that if the user changes the image display window size, the image will automatically be scaled to fit as best as possible.
- C** Closes the displayed image window named in **TITLE**.

**STATUS**

0, 1, or 2, as follows:

- 0 IMAGE program started successfully and image displayed.
- 1 IMAGENAME was not found in the specified path.
- 2 IMAGE program was not started.
- 3 No close APP.NAME specified to close image, if "C" option used.

HOSTACCESS enables more than one image to be displayed on the user's desktop at the same time. To achieve this, simply call another subroutine for the next image, changing the scale as required.

There is no limit to the number of images that can be displayed in this manner, simply repeat the subroutine for each image. It is often useful to decrease the size of these images by setting the SCALE parameter to 50 (half size) or less.

Please note that multiple images can only be displayed on the same screen if they *all* have *identical* palettes. Images with different palettes will 'corrupt' each others' screen image (often giving an "infra-red" like display).

PIX.WIN.SEND.KEYS is a BASIC subroutine that can be called from your applications as follows:

**CALL PIX.WIN.SEND.KEYS(APP.NAME,KEYS,SUCCESSFUL)**

This subroutine sends KEYS in the DOS keyboard stacker format to the specified Windows application APP.NAME. This allows almost any Windows application to be driven automatically.

The following subroutine parameters need to be passed or will be returned accordingly:

<b>APP.NAME</b>	This must be the name of the Windows application. Normally this is the name held in the Window Title as shown exactly on the desktop, e.g.  WORDPERFECT [DOCUMENT1 - UNMODIFIED]
<b>KEYS</b>	In the same format as the DOS keyboard e.g.  @F "OC:\wperfdocs\HOSTACCESS.doc" to ALT/F and open a WordPerfect document.
<b>SUCCESSFUL</b>	Returned as 1 if the application was running on the desktop and the keys were sent to it, 0 if not running.

**Note 1** It is recommended that you use [DDE](#) if it is available and practical, because using the server name is more accurate and reliable than depending on APP.NAME.

**Note 2** The DOS shell opened through Microsoft Windows cannot accept keys in the DOS keyboard stacker format. Please refer to the Developer's Guide for more information on passing keystrokes to DOS applications.

[Initiate a DDE channel.](#)

PIX.DDE.INITIATE

[Retrieve data from another Windows application.](#)

PIX.DDE.REQUEST

[Pass data to another Windows application.](#)

PIX.DDE.POKE

[Send commands to the server.](#)

PIX.DDE.EXECUTE.MACRO

[Close DDE link.](#)

PIX.DDE.CLOSE



How DDE Works.



Using DDE with HOSTACCESS.



Initiate a DDE channel.

**Dynamic Data Exchange** (DDE) is used to transfer data between Windows applications.


Two applications that participate in DDE engage in what is known as a DDE **conversation**. The application that initiates the conversation is known as the **client** application, and the application that responds to the client is known as the **server** application.

Any Windows product that supports DDE as a server application must have a **server name**. For example, the server name for Word for Windows is WINWORD and for Quattro Pro it is QPW. To initiate a DDE link with a server application, you would normally use the server name and this would return a **channel number**. Using this channel number you would then send commands (normally in the format of the macro language supported by that server application) and finally close the link with that channel number when all processing is completed.

When communicating with a server you must also always specify a **topic**. Server applications can support many topics depending on which part of that application you want to communicate with. For instance, if you want to request information from Quattro Pro on a specific spreadsheet, the server name would be QPW and the topic name would be the spreadsheet name.

DDE was designed to form a standard way of communicating between Windows applications. However, the fact that each Windows application supports DDE differently (or sometimes not at all) makes it more difficult for the novice to understand it or become involved with it.

If you want to program using DDE, you will have to learn as much, if not more about the server application that you want to talk to, rather than if you were a direct user of the product itself.

Click here  for information on using DDE with HOSTACCESS.

You can use DDE to use HOSTACCESS as a **DDE client** to other Windows applications (servers), sending data and instructions from the host to a Windows application. This gives your own host programs and applications almost total control over any other Windows product.

Using DDE with HOSTACCESS, you only need to specify the server name for any DDE process. HOSTACCESS automatically keeps track of channel numbers internally for you.

All Windows applications support a general topic name **system**. Unless you are setting up more complicated DDE links, this topic should be more than adequate for most developers. (All of the HOSTACCESS PASS.TO's for Windows use the SYSTEM topic).

You can also use DDE to use HOSTACCESS as a DDE server. You can write Windows programs in such applications as Word or Excel, which can send and receive data to and from the host software.

**Note:** You must have a resilient link from the PC to the host. DDE cannot work remotely unless full flow control and error checking are in place.

PIX.DDE.INITIATE is BASIC subroutine that can be called from your application as follows:


**CALL PIX.DDE.INITIATE(SERVER.NAME, TOPIC, STATUS)**

This subroutine opens a DDE channel with the Windows application known by SERVER.NAME for the given TOPIC. See [PIX.DDE.CLOSE](#) for information on closing a DDE link.

The following subroutine parameters need to be passed or will be returned accordingly:

**SERVER.NAME**     i.e. QPW, 123W, WINWORD, EXCEL.  
**TOPIC**             If null then SYSTEM is used as default.  
**STATUS**            Returned as a dynamic array where attribute 1 will be 0 if successful and > 0 otherwise, as follows:  
                         1- Application link not open (no initiate).  
                         2- Timeout.  
                         3- Topic not supported by application.  
                         4- No DDE channels available.  
                         5- Server closed.  
                         6- Server busy.  
                         7- Server NAK.

Click here  for an example.

Click here  for information on DDE.



**CALL PIX.DDE.INITIATE("QPW","SYSTEM",STATUS)**

This **example** would initiate a DDE link between HOSTACCESS as the Client and Quattro Pro for Windows as the server using the SYSTEM topic.

PIX.DDE.REQUEST is a BASIC subroutine that can be called from your application as follows:


**CALL PIX.DDE.REQUEST(SERVER.NAME, TOPIC, ITEM, DATA, STATUS)**

This subroutine allows DATA to be retrieved directly from another Windows application (the server). Most DDE servers have defined elements which the server knows about (called ITEMS) where specific pieces of information reside.

A successful [PIX.DDE.INITIATE](#) must have been made with SERVER.NAME, TOPIC before the DATA can be retrieved.

The following subroutine parameters need to be passed or will be returned accordingly:

<b>SERVER.NAME</b>	i.e. QPW, 123W, WINWORD, EXCEL etc.
<b>TOPIC</b>	If null, SYSTEM is used as default although this would tend to be a document name for REQUEST.
<b>ITEM</b>	Specific to each Server application. An ITEM may be a cell reference such as R1C1 for instance from where you want to receive DATA.
<b>DATA</b>	Returned string of DATA from the given ITEM, - an amount or description for instance. The DATA may contain delimiters from the SERVER and ITEM, - for instance, tabs from some spreadsheet applications for each delimited cell from that spreadsheet
<b>STATUS</b>	Returned as a dynamic array where attribute 1 will be 0 if successful and > 0 otherwise, as follows: 1 - Application link not open (no initiate). 2 - Timeout. 3 - Topic not supported by application. 4 - No DDE channels available. 5 - Server closed. 6 - Server busy. - Server NAK.

Click here  for information on DDE.


PIX.DDE.CLOSE is a BASIC subroutine that can be called from your application as follows:

**CALL PIX.DDE.CLOSE(SERVER.NAME, TOPIC, STATUS)**

PIX.DDE.CLOSE closes a DDE link already established with [PIX.DDE.INITIATE](#). It is recommended that you close any DDE link when DDE conversation is completed.

The following subroutine parameters need to be passed or will be returned accordingly:

<b>SERVER.NAME</b>	i.e. QPW, 123W, WINWORD, EXCEL etc.,
<b>TOPIC</b>	If null, SYSTEM is used as default
<b>STATUS</b>	Returned as a dynamic array where attribute 1 will be 0 if link closed OK

Click here  for information on DDE.

This subroutine allows commands to be sent to the server application to allow it to be driven and updated automatically. Macro information should be in the format as specified and documented by the server application. Normally these are macro strings.

PIX.DDE.EXECUTE.MACRO can be called from your application as follows:

**CALL PIX.DDE.EXECUTE.MACRO(SERVER.NAME, TOPIC, MACRO, DELIMS, STATUS)**


A successful [PIX.DDE.INITIALIZE](#) must have been made with SERVER.NAME, TOPIC before the MACROs can be sent via this routine.

The following subroutine parameters need to be passed or will be returned accordingly:

<b>SERVER.NAME</b>	i.e. QPW, 123W, WINWORD, EXCEL etc.,
<b>TOPIC</b>	If null, SYSTEM is used as default
<b>MACRO</b>	A dynamic array of MACRO command(s) and their arguments that you want to pass through DDE to the server application.
<b>DELIMS</b>	<p>A pair of characters used to wrap around each DDE command specified in MACRO above. If null, defaults to "[ ]".</p> <p>Most Server applications need to be sent the macro command(s) within delimiters, i.e. [command macro1] [command macro2][etc.] It was found that some Windows servers use different characters such as "{ }" for Quattro Pro (QPW).</p> <p>This routine will send each MACRO wrapped in DELIMS as one big DDE string.</p>
<b>STATUS</b>	<p>returned as a dynamic array where attribute 1 will be 0 if successful and &gt; 0 otherwise, as follows:</p> <ol style="list-style-type: none"><li>1- Application link not open (no initiate)</li><li>2- Timeout</li><li>3- Topic not supported by application</li><li>4- No DDE channels available</li><li>5- Server closed</li><li>6- Server busy</li><li>7- Server NAK</li></ol>

To help with debugging, STATUS can be passed as follows:

- 97 Single Mode. Send each MACRO array element as a separate DDE execute. For instance, Quattro Pro has a 128 byte limit.
- 98 Monitor on. Display the DDE string exactly as it will be passed on the screen.
- 99 Single mode and monitor (combination of 97 and 98).

Click here  for information on DDE.

```

SERVER.NAME = "WINWORD"
TOPIC = "SYSTEM"
WIN.PROGRAM.NAME = "D:\WINWORD\WINWORD.EXE"
CALL PIX.DDE.INITIATE (SERVER.NAME, TOPIC, STATUS)
IF STATUS<1> # 0 THEN
    STATE = 1 ; * foreground and normal
    CALL PIX.WIN.RUN (WIN.PROGRAM.NAME, STATE, SUCCESSFUL)
    IF SUCCESSFUL THEN
        CALL PIX.DDE.INITIATE (SERVER.NAME, TOPIC, STATUS)
    END
END
END
IF STATUS<1> = 0 THEN
    MACRO = ""
    MACRO<1> = 'FileOpen.Name = "D:\WINWORD\DOCS\LETTER.DOC"'
    MACRO<2> = 'FilePrint'
    CALL PIX.DDE.EXECUTE.MACRO (SERVER.NAME, TOPIC, MACRO, "", ST)
    CALL PIX.DDE.CLOSE (SERVER.NAME, TOPIC, ST)
END

```

This **example** initiates a DDE session with Word for Windows and the SYSTEM topic. If a DDE session cannot be started, Word for Windows is started. If OK, LETTER.DOC is retrieved and printed to the default printer. You could easily carry on with the macro to close the opened document or carry out a mailmerge.

PIX.DDE.POKE is a BASIC subroutine that can be called from your application as follows:


**CALL PIX.DDE.POKE(SERVER.NAME, TOPIC, ITEM, DATA, STATUS)**

This subroutine allows DATA to be passed directly to another Windows application (the server). Most DDE servers have defined elements that the server knows about (called ITEMS) which can accept information from DDE clients.

A successful [PIX.DDE.INITIATE](#) must have been made with SERVER.NAME, TOPIC before the DATA can be sent.

The following subroutine parameters need to be passed or will be returned accordingly:

<b>SERVER.NAME</b>	I.e. QPW, 123W, WINWORD, EXCEL etc.
<b>TOPIC</b>	If null, SYSTEM is used as default, although this would tend to be a document name for POKE.
<b>ITEM</b>	Specific to each server application. An ITEM may be a cell reference such as R1C1 for instance into which you want to send DATA.
<b>DATA</b>	Any string of DATA valid for the given ITEM, - an amount or description for instance. The DATA may contain delimiters that the SERVER and ITEM will recognize - for instance, tabs for some spreadsheet applications to force each delimited field into separate cells.
<b>STATUS</b>	Returned as a dynamic array where attribute 1 will be 0 if successful and > 0 otherwise, as follows: 1 - Application link not open (no initiate). 2- Timeout. 3- Topic not supported by application. 4- No DDE channels available. 5- Server closed. 6- Server busy. 7- Server NAK.

Click here  for information on DDE.

HOSTACCESS allows you to design a sophisticated Windows-style interface for all of your PICK-based applications, using the GUI TOOLKIT. The following topics describes the GUI TOOLKIT, and the PICK BASIC subroutines which you can use to implement a GUI for your host application.

AiF escape sequences are used within these subroutines and are also available to the programmer if required.

Click here [□](#) for a list of available subroutines.

To use the GUI TOOLKIT, you need to call PICK BASIC subroutines from within your application. You can call a subroutine from your application as follows:

**CALL name(parameter1, parameter2, ...)**

Where:

**name** Is the name of the subroutine.

**parameter1,**  
**parameter2, ...** Are the parameters.

The following tables give a list of all of the PICK BASIC subroutines currently available in the GUI TOOLKIT, and describe their functions.

- Screen Sculpting Subroutines.
- Loading controls.
- Managing controls.
- Commands.
- Grids.
- Secondary Windows.
- Control Events.
- Utilities.



These subroutines handle the sculpting features of HOSTACCESS.

Subroutine Name (PIX, ...)

- SCULPTURE.MODE Turns sculpture mode on or off.
- SCULPTURE.BOX Draws a sculpted box.
- SCULPTURE.LINE Draws a sculpted line.
- SET.LINE.COLOURS Changes default sculpture colours.

All the GUI subroutines are prefixed PIX.GUI.

Subroutine Name (PIX.GUI ...)

<u>CONTROL.LOAD</u>	Loads all types of controls.
<u>LOAD.CHECK</u>	Creates a check box control.
<u>LOAD.COLOR</u>	Changes the default control colors.
<u>LOAD.DROPCOMB</u>	Creates a drop-down combo box.
<u>LOAD.DEFGROUP</u>	Creates a base control group.
<u>LOAD.EDIT</u>	Creates a edit box control (single or multiple line).
<u>LOAD.GROUP</u>	Creates a group of specified controls.
<u>LOAD.IMAGE</u>	Draws a static image.
<u>LOAD.PUSHBUTTON</u>	Creates a push-button control with graphics image.
<u>LOAD.LABEL</u>	Creates a text label with a specified font.
<u>LOAD.LIST</u>	Creates a read only List box.
<u>LOAD.RADIO</u>	Creates a radio button control.
<u>LOAD.STRING</u>	Creates a string list, for contents of list/combo boxes.
<u>LOAD.SIMCOMB</u>	Creates a simple combo box.
<u>LOAD.TEXTBUTTON</u>	Creates a push-button control with text.

All the GUI subroutines are prefixed PIX.GUI

Subroutine Name (PIX.GUI ...)

<u>ACC.KEY</u>	Attaches an accelerator key to a control.
<u>ALT.EVENT</u>	Returns an alternate message.
<u>CONTROL.GRAYED</u>	Grays out (disables) a control.
<u>CONTROL.READ.VALUE</u>	Reads the contents of a control.
<u>CONTROL.SET.FOCUS</u>	Set the input focus of a control.
<u>CONTROL.SET.VALUE</u>	Sets the contents of a control
<u>CONTROL.UNGRAYED</u>	Un-grays (enables) a control.
<u>{CONTROL.UNLOAD</u>	Destroys a control.
<u>HIDE.CONTROL</u>	Hides a specific control.
<u>MSG.BOX</u>	Display a message box.
<u>SHOW.CONTROL</u>	Shows a specific control.

All the GUI subroutines are prefixed PIX.GUI.

Subroutine Name (PIX, ...)

<u>LOAD.COMMAND</u>	Create commands for toolbars, toolboxes and menus.
<u>LOAD.MENU</u>	Adds commands to a menu.
<u>LOAD.TOOLBAR</u>	Creates a toolbar on top of session screen.
<u>LOAD.TOOLBOX</u>	Creates a floating toolbox.
<u>MENU.INSTALL</u>	Displays menus on the menu bar.



All the GUI subroutines are prefixed PIX.GUI.

Subroutine Name (PIX.GUI ...)

<a href="#"><u>LOAD.GRID</u></a>	Creating a grid
<a href="#"><u>GRID.CLEAR</u></a>	Clearing a grid.
<a href="#"><u>GRID.COLWIDTH</u></a>	Changing column widths.
<a href="#"><u>GRID.DELETE</u></a>	Deleting rows.
<a href="#"><u>GRID.FOCUS</u></a>	Setting focus to the grid.
<a href="#"><u>GRID.POSITION</u></a>	Obtaining the current grid position.
<a href="#"><u>GRID.SCROLLBARS</u></a>	Setting scrollbars.
<a href="#"><u>LOAD.VBX</u></a>	Loading a VBX control.

All the GUI subroutines are prefixed PIX.GUI.

Subroutine Name (PIX, ...)

<u>SEC.ACTIVE</u>	Gives a secondary window focus
<u>SEC.CLOSE</u>	Close a secondary window
<u>SEC.OPEN</u>	Open a secondary window

All the GUI subroutines are prefixed PIX.GUI.

Subroutine Name (PIX, ...)

CONTROLEVENT Enables reporting of events for a control.

GET.EVENT Collects events from controls.

SET.TIMER Enables timed events

This GUI subroutine is prefixed PIX.GUI.

Subroutine Name (PIX, ...)

SELECTION      Select an item from a list

For clarity, the positioning and drawing of the screen is performed in a grid method. The top left hand corner of any window has the grid co-ordinates of 0,0, and the bottom right can be 23, 79. Each character displayed upon the window takes up one cell, or grid row and column position.

Co-ordinates are given as (**X**, **Y**), where **X** is the number of characters across the screen, and **Y** is the number of vertical characters down the screen.

**Note:** the top-left corner is position 0,0 - not position 1,1.

Many of the PICK BASIC subroutines described in the following sections have a POSITION parameter, with **X** and **Y** co-ordinates. Unless otherwise stated, you can assume that these parameters use the standard (**X,Y**) system as described here.

Click here  to see a diagram of this coordinate system.






You can quickly and simply produce a Windows look and feel to all of your PICK applications, with a minimal change to code, using **sculpting**. This feature allows you to:

- Draw a box of any size
- Use any available colour for the line

Using different colours allows you to effect a raised or sunken look. With the ability to draw boxes within other boxes, you can give a typical GUI effect as seen in other Windows products.


A sculpted image is produced by shading sides of a picture, to mimic the shading produced by light falling on a raised or sunken image. So, when drawing a sculpted box the top and left sides of the box are shaded one colour, and the bottom and right sides of the box are shaded another colour.

For example, to produce an image of a sunken box, you would need to shade the top/left sides a dark colour, and the bottom/right sides a light colour. We recommend that you stick with the Windows colours of BLACK,WHITE for best effect.

Click here  for a list of screen sculpting subroutines.

We have provided a number of subroutines for you to call that displays the sculpture effects. These can be displayed :

- Via a single master subroutine, [PIX.GUI.LOAD.SCREEN](#). All your screen details can thus be held globally and loaded via one subroutine rather than the programmer having to call each subroutine individually for each effect.
- Via individual subroutines, one for each type of screen effect. For example to draw a sculptured box use [PIX.SCULPTURE.BOX](#).

Click here  for a list of screen sculpting subroutines.


To load all types of screen effects into HOSTACCESS 's memory, call the following subroutine:


**CALL PIX.GUI.LOAD.SCREEN(SCREEN.DETAILS,ERROR)**

Where:

**SCREEN.DETAILS** A dynamic array containing screen effects. This subroutine allows you to use GUI screen effects as well as HOSTACCESS 's other screen effects.

**ERROR** Returns 1 if error (for future use)

Click here  for an example.

Click here  for a list of screen sculpting subroutines.

Below are examples of how you can load screen effects using [PIX.GUI.LOAD.SCREEN](#). This is an example and not a demonstrable screen.

```
SCREEN.DETAILS = ""
SCREEN.DETAILS<-1> = "SCOLOR,BLACK,WHITE"
SCREEN.DETAILS<-1> = "SBOX,1,1,78,21,RAISED,BLACK,WHITE"
SCREEN.DETAILS<-1> = "SBOX,10,20,20,1"
SCREEN.DETAILS<-1> = "SLINE,1,11,78,H,WHITE"
SCREEN.DETAILS<-1> = "COLOR,WHITE,BLUE"
SCREEN.DETAILS<-1> = "BOX,5,2,10,10,SINGLE,SHADOW,EXPLODE,WHITE,RED"
SCREEN.DETAILS<-1> = "BOX,40,2,10,10,DOUBLE,,WHITE,RED"
SCREEN.DETAILS<-1> = "LINE,5,5,10,5,SINGLE,MERGE,GREEN"
SCREEN.DETAILS<-1> = "TEXT,0,23,This is Line 23"
SCREEN.DETAILS<-1> = "SPACE,30,23,20"
CALL PIX.GUI.LOAD.SCREEN(SCREEN.DETAILS,"")
```

Click here  for a description of SCREEN.DETAILS.

SCREEN.DETAILS is a dynamic array of screen effects that can be loaded at a given time. They include many of HOSTACCESS's GUI and non GUI effects. Each parameter is separated by a comma. Only the first parameter is common through all of the screen effects, this being the first parameter (TYPE).

TYPE	Description
<b>SLINE</b>	Draws a Sculpted Line. Parameters required are 'SLINE ', X, Y, LENGTH, TYPE, COLOUR. See <a href="#">PIX.SCULPTURE.LINE</a> for details on these parameters.
<b>SBOX</b>	Draws a Sculpted Box. Parameters required are 'SBOX ', X, Y, W, H, APPEARANCE, F/COLOUR, B/COLOUR. See <a href="#">Drawing Sculpted Boxes</a> for details on these parameters.
<b>SCOLOR or SCOLOUR</b>	Sets the colour for use in Sculpted line drawing. Parameters required are 'SCOLOUR ', F/COLOUR, B/COLOUR. See <a href="#">PIX.SCULPTURE.LINE</a> for details on these parameters.
<b>LINE</b>	Draws a character type line. Parameters required are 'LINE ', X1, Y1, X2, Y2, SINGLE or DOUBLE, MERGE, COLOUR. The parameters SINGLE, DOUBLE and MERGE are text literals to explain the line effects. See <a href="#">PIX.DRAW.LINE</a> for details on these parameters.
<b>BOX</b>	Draws a character type box. Parameters required are 'BOX ', X, Y, W, H, LINETYPE, SHADOW, EXPLODE, F/COLOUR, B/COLOUR. The parameters SHADOW and EXPLODE are text literals to explain the BOX effects. See <a href="#">PIX.DRAW.BOX</a> for details on these parameters.
<b>TEXT</b>	Displays text on the screen as you would normally with the CRT @ function in DATABASIC. Parameters required are 'TEXT ', XPOS, YPOS, OUTPUT.
<b>SPACE</b>	Displays spaces on the screen as you would normally with the CRT @ function in DATABASIC. Parameters required are 'SPACE ', XPOS, YPOS, NO.SPACES.
<b>COLOUR or COLOR</b>	Sets HOSTACCESS's foreground and background colours. Parameters required are 'COLOUR ', F/COLOUR, B/COLOUR. See <a href="#">PIX.SET.COLOUR</a> for details on these parameters. (Option D will always be set from this routine)

To turn sculpture mode on or off, call the following subroutine from your application:

**CALL PIX.SCULPTURE.MODE(MODE, CLEAR)**

Where:


**MODE** 'ON ' - Turns Sculpture mode on  
 'OFF ' - Turns Sculpture mode off

**CLEAR** 'C ' - Existing sculpture is cleared.  
 Null - Existing Sculpture is left, if used in conjunction with Mode 'ON '.

To draw a complete sculpted screen very quickly, draw your screen, then set sculpture mode to **on**.

You can also use this routine to just clear sculpted lines and boxes, without switching mode. This stops the screen giving the effect of a wobble.

**Note:** You must have sculpture mode on to display any sculpted lines or boxes.

Click here  for a list of screen sculpting subroutines.

To draw a sculpted box, call the following subroutine from your application:

**CALL PIX.SCULPTURE.BOX(POSITION, APPEARANCE, COLOURS)**

Where:

**POSITION** X, Y, W, H, where X, Y is top left co-ordinate and W, H is the width and height of the box. Home co-ordinate of screen is 0,0.

**APPEARANCE** "SUNKEN" (the default): makes box appear sunken.  
"RAISED": makes box appear raised.  
Uses default Sculpture colours unless COLOURS is specified.

**COLOURS** Colour of top and left sides of box and colour of bottom and right of box. Specified as standard colour literal, e.g. "BLACK, WHITE" . If not used, default colours are last specified colours for sculpture used.

Click here  for a list of screen sculpting subroutines.

Click here  for an example.

```

001 REM *** Example 1
002 REM *** draw a Sculptured box at (10,2), height 5,
003 REM *** width 10, and colour BLACK (top/left) and
004 REM *** colours WHITE (bottom/right) use:
005 REM ***
006 CALL PIX.SET.COLOURS('BLACK,WHITE','D') ; * Set colors of screen.
007 PRINT @(-1) ; * Clear Screen
008 PRINT "Examples of Sculptured Boxes ":
009 CALL PIX.SET.LINE.COLOURS("BLACK,WHITE") ; * Set Sculpture
colours
010 CALL PIX.Sculpture.MODE("ON",'C') ; * Turn Sculpture
on
011 CALL PIX.Sculpture.BOX("20,6,10,5","", "") ; * Draw Sculpted Box
012 CALL PIX.Sculpture.BOX("40,6,10,5","RAISED","") ; * Draw Sculpted Box
013 INPUT XX: ; * Wait
014 CALL PIX.Sculpture.MODE("OFF","") ; * Turn Sculpture
off
015 CALL PIX.SET.COLOURS('GREEN,BLACK','D') ; * Set colours of screen.
016 PRINT @(-1):
017 ***

```

To draw a sculpted line, call the following subroutine from your application:

**CALL PIX.SCULPTURE.LINE(POSITION, LENGTH, TYPE, COLORS)**

Where:

**POSITION** X, Y co-ordinates of start of line.

**LENGTH** Length of line (characters).

**TYPE** "H" - draw a horizontal line (the default)  
"V" - draw a vertical line

**COLOUR** Colour of line. Specified as standard colour literal, e.g. "BLACK".  
By default, the last specified sculpting colour is used.

Click here  for an example.



To draw a sculpted horizontal line at (12,14), 10 characters long with colour white, use:

```
CALL PIX.SCULPTURE.LINE("12,14","10",""," WHITE")
```

To draw a sculpted vertical line at (40,0), 23 characters deep with colour red, use:

```
CALL PIX.SCULPTURE.LINE("40,0", 23, "V","RED")
```

To change the default sculpture colours, call the following subroutine from your application:

**CALL PIX.SET.LINE.COLOURS(COLOURS)**

Where:

**COLOURS** Colour of top and left sides of box and colour of bottom and right of box. Specified as standard literal, e.g. "BLACK, WHITE" .

For a list of colours available click here 

To set the default sculpture colour to (Top/Left) GRAY and (Bottom/Right) WHITE use:

**CALL PIX.SET.LINE.COLOURS("GRAY, WHITE")**

To properly use the GUI TOOLKIT, you should be familiar with the concept of [controls](#). Controls are named Windows objects, contained in HOSTACCESS 's memory. Some controls are actions on other controls (for example, HIDE).

Within HOSTACCESS you can print these controls on to the terminal screen as you would with bold or reverse video. This means that the terminal screen can have Windows controls feeding data to the Host.

- Creating and showing controls.
- Control IDs.
- Loading all types of controls.
- Loading controls Individually.

To create and display Windows controls, simply load them into HOSTACCESS 's memory:

- Via a single master subroutine, PIX.GUI.CONTROL.LOAD. All of the control details can thus be loaded via one subroutine rather than the programmer having to call each subroutine individually for each control.
- Via individual subroutines, one for each type of control. For example, the subroutine PIX.GUI.LOAD.RADIOBUTTON will load a radio button.

Click here [☐](#) for details of loading controls all at once.

See individual subroutine descriptions for details of loading controls individually.

To get any interaction with the Host system you will need to load the [events](#) for each control, and read the values from each control using different subroutines.

When you create controls you reference the control by a name, for ease of use.

The name can be a numeric or alphabetic name, and can be used as a label for reference throughout your host program.

This name is passed as the Control ID parameter in PICK BASIC subroutines.

To load controls into HOSTACCESS 's memory all at once, call the following subroutine:

**CALL PIX.GUI.CONTROL.LOAD(GUI.APP.NAME, CONTROL.DETAILS, CONTROL.VALUE, ERROR)**

Where:

<b>GUI.APP.NAME</b>	This variable is useful to run GUI programs from within GUI programs, i.e. if you "EXECUTE" or "PERFORM" GUI programs from within GUI programs. The variable is prefixed automatically to all control names. If you intend not to do this then you can pass a null. i.e. CALL PIX.GUI.READ.CONTROL.VALUE("",...).
<b><u>CONTROL.DETAILS</u></b>	A dynamic array containing sets of parameters describing each control to be loaded.
<b>CONTROL.VALUE</b>	Not used at present.
<b>ERROR</b>	Set to 1 if Error (for future use)

Each parameter in CONTROL.DETAILS is separated by a comma. The first seven parameters must be in the correct order.

- Parameter 1 = the Control ID (its name)
- Parameter 2 = the control type (for example, RADIO)
- Parameters 3 - 7 depend on the control type
- Parameters 8+ are keywords, and can be in any order.

If all 7 primary parameters are not required, you can truncate the string when required but they must be in order specified. Keywords are only accepted in position eight onwards. For example:

**NAME,TYPE,xxx,yyy**

**NAME,TYPE,xxx,yyy,,,Key1,...Keyn**

The parameters passed to PIX.GUI.CONTROL.LOAD are similar to those passed to each individual subroutine, except that this routine will loop through a **dynamic array of controls**, calling the relevant subroutine for each control. The control details passed to this subroutine for each control will be the same as those passed to each individual Control subroutine.

You can load single controls with single subroutines.

Each individual subroutine you call takes the following three parameters:

- GUI.APP.NAME** This variable is useful to run GUI programs from within GUI programs, i.e. if you "EXECUTE" or "PERFORM" GUI programs from within GUI programs. The variable is prefixed automatically to all control names.  
If you intend not to do this then you can pass a null. i.e. CALL PIX.GUI.READ.CONTROL.VALUE("",...).
- CONTROL.DETAILS** When loading controls into HOSTACCESS 's memory, you need to pass a set of parameters as CONTROL.DETAILS. See individual control loading routines for descriptions of these parameters.
- STRING** Set to 'O' if you require output string (AiF) to be formatted and sent back to the calling program, The calling program will need to send this command to HOSTACCESS.

Click here  for an example.



To load a text push-button use the following:

**CALL PIX.GUI.LOAD.TEXTBUTTON(GUI.APP.NAME, CONTROL.DETAILS, STRING)**

To create a push button holding a text label, call the following subroutine from your application:

**PIX.GUI.LOAD.TEXTBUTTON (GUI.APP.NAME, CONTROL.DETAILS, STRING)**

Where:


**CONTROL.DETAILS** = Control ID, TEXTBUTTON, X, Y, W, H, Label, {Key1,...Keyn}

Where:


<b>Control ID</b>	The unique name for the text button.
<b>TEXTBUTTON</b>	Text literal TEXTBUTTON
<b>X</b>	The X co-ordinate of the left of the button
<b>Y</b>	The Y co-ordinate of the top of the button
<b>W</b>	Width of the button (characters)
<b>H</b>	Height of the button (characters)
<b>Label</b>	Textual label associated with that button
<b>Key1...Keyn</b>	Additional parameters if required, in any order - see below.


The following extra keyword parameters are available:

HIDE            SHOW \*        GRAYED        UNGRAYED \*  
TERMITE\*        HELV8            HELV10        SYSTEM

Default settings are denoted by \*. Click here  for details.

Alternatively, you can use the [PIX.GUI.CONTROL.LOAD](#) subroutine.

**Note:** To highlight a specific character in the text label as the Accelerator Key, precede the character with an ampersand (&). For example, 'Pro&CESS'. Click here  for details.

 Program Example

```
061 CALL PIX.GUI.LOAD.TEXTBUTTON ("", "FILE, TEXTBUTTON, 65, 2, 9, 2, &File, HELV10", "")
062 CALL PIX.GUI.LOAD.TEXTBUTTON ("", "CANCEL, TEXTBUTTON, 65, 5, 9, 2, Cancel", "")
063 CALL PIX.GUI.LOAD.TEXTBUTTON ("", "ADD, TEXTBUTTON, 65, 11, 9, 2, Add", "")
```

To create a push button holding an image, call the following subroutines from your application:

**CALL PIX.GUI.LOAD.PUSHBUTTON (GUI.APP.NAME, CONTROL.DETAILS, STRING)**

Where:


**CONTROL.DETAILS** Control ID, PUSHBUTTON, X, Y, W, H, Label, {Key1,...Keyn}

Where:

**Control ID** The unique name for the push button.  
**PUSHBUTTON** Text literal PUSHBUTTON.  
**X** The X co-ordinate of the left of the button  
**Y** The Y co-ordinate of the top of the button  
**W** Width of the button (characters)  
**H** Height of the button (characters)  
**Label** Textual label associated with that button  
**Key1...Keyn** Additional parameters if required, in any order - see below.

**The following extra keyword parameters are available:**


<a href="#">HIDE</a>	<a href="#">SHOW</a> *	<a href="#">GRAYED</a>	<a href="#">UNGRAYED</a> *
<a href="#">PUSH</a> *	<a href="#">FRAME</a>	<a href="#">PUSHEDIN</a>	<a href="#">PUSHEDOUT</a>
<a href="#">SHADOWED</a>	<a href="#">NONE</a>	<a href="#">LABELLED</a>	<a href="#">BITMAP</a> *
<a href="#">ICON</a>	<a href="#">FILEID</a>	<a href="#">LABELPOS</a>	<a href="#">TERMITE</a>
<a href="#">HELV8</a>	<a href="#">HELV10</a>	<a href="#">FILE</a>	<a href="#">ID</a>
<a href="#">SYSTEM</a>	<a href="#">SUBS</a>	<a href="#">FILETYPE</a>	<a href="#">FILE</a>

Default settings are denoted by \*. [Click here](#)  for details.

Alternatively, you can use the [PIX.GUI.CONTROL.LOAD](#) subroutine.

 [Program example.](#)

```
067 CALL PIX.GUI.LOAD.PUSHBUTTON("", "HELP, PUSHBUTTON, 65, 8, 9, 2, \bitmaps\fhhelp.bmp", "")
```

This example is taken from a full program example - [click here](#)  for details.

To display a static image call the following subroutine from your application:

**CALL PIX.GUI.LOAD.IMAGE (GUI.APP.NAME, CONTROL.DETAILS, STRING)**

Where:

**CONTROL.DETAILS** = Control ID, IMAGE, X, Y, W, H, Label, {Key1,...Keyn}

Where:


<b>Control ID</b>	The unique name for the image.
<b>IMAGE</b>	Text literal IMAGE.
<b>X</b>	The X co-ordinate of the left of the image.
<b>Y</b>	The Y co-ordinate of the top of the image.
<b>W</b>	Width of the image (characters)
<b>H</b>	Height of the image (characters)
<b>Label</b>	Textual label associated with that image.
<b>Key1...Keyn</b>	Additional parameters if required, in any order - see below.

**The following extra keyword parameters are available:**


<u>HIDE</u>	<u>SHOW</u> *	<u>CLIPPED</u>	<u>SCALED</u> *
<u>PUSH</u>	<u>FRAME</u>	<u>PUSHEDIN</u>	<u>PUSHEDOUT</u>
<u>SHADOWED</u>	<u>NONE</u> *	LABELLED	<u>BITMAP</u> *
<u>ICON</u>	<u>FILEID</u>	LABELPOS	FILE
ID	SUBS	FILETYPE	TILE

Default settings are denoted by \*. Click here  for details.

Alternatively, you can use the [PIX.GUI.CONTROL.LOAD](#) subroutine.

 Program Example.

```
071     CALL PIX.GUI.LOAD.IMAGE (“”,”LOGO,IMAGE,0,0,15,4,\bitmaps\easy\easyacc.bmp”,””)
```

This example is taken from a full program example - [click here](#)  for details.

To create a check box, call the following subroutine from your application:

**CALL PIX.GUI.LOAD.CHECKBUTTON (GUI.APP.NAME, CONTROL.DETAILS, STRING)**

Where:

**CONTROL.DETAILS = Control ID, CHECK, X, Y, W, H, Label, {Key1,...Keyn}**

Where:


<b>Control ID</b>	The unique name for the check box.
<b>CHECK</b>	Text literal CHECK.
<b>X</b>	The X co-ordinate of the left of the check box.
<b>Y</b>	The Y co-ordinate of the top of the check box.
<b>W</b>	Width of the box (characters)
<b>H</b>	Height of the box (characters)
<b>Label</b>	Textual label associated with that box.
<b>Key1...Keyn</b>	Additional parameters if required, in any order.

The following extra keyword parameters are available:

<a href="#">HIDE</a>	<a href="#">SHOW</a> *	<a href="#">GRAYED</a>	<a href="#">UNGRAYED</a> *
<a href="#">TERMITE</a> *	SYSTEM	<a href="#">HELV8</a>	<a href="#">HELV10</a>
<a href="#">LEFT</a>	<a href="#">RIGHT</a> *	<a href="#">CHECKED</a>	<a href="#">UNCHECKED</a> *


Default settings are denoted by \*. Click here  for details.

Alternatively, you can use the [PIX.GUI.CONTROL.LOAD](#) subroutine.

 [Program Example](#).



```
115 CALL PIX.GUI.LOAD.CHECKBUTTON ("", "CAR,CHECK,47,10,10,1,Car Park,Helv10, LEFT","")
116 CALL PIX.GUI.LOAD.CHECKBUTTON ("", "CAT,CHECK,47,11,10,1,Catalogue,Helv10,LEFT","")
```

This example is taken from a full program example - [click here](#)  for details.

To create a radio button, call the following subroutine from your application:

**CALL PIX.GUI.LOAD.RADIOBUTTON (GUI.APP.NAME, CONTROL.DETAILS, STRING)**

Where:

**CONTROL.DETAILS** Control ID, RADIO, X, Y, W, H, Label, {Key1,...Keyn}

Where:

**Control ID** The unique name for the radio button.  
**RADIO** Text literal RADIO.  
**X** The X co-ordinate of the left of the button.  
**Y** The Y co-ordinate of the top of the button.  
**W** Width of the button (characters)  
**H** Height of the button (characters)  
**Label** Textual label associated with that button.  
**Key1...Keyn** Additional parameters if required, in any order.

The following extra keyword parameters are available:


HIDE            SHOW \*    GRAYED    UNGRAYED \*    RADIO  
GROUP“SEX”  
TERMITE \*    SYSTEM    HELV8    HELV10  
LEFT            RIGHT \*    CHECKED    UNCHECKED \*

Default settings are denoted by \*. Click here  for details.

Alternatively, you can use the PIX.GUI.CONTROL.LOAD subroutine.

Program Example.

```
110 CALL PIX.GUI.LOAD.RADIOBUTTON("", "DOG,RADIO,33,10,10,1,Dog,Helv10,LEFT,CHECK,SEX", "")
111 CALL PIX.GUI.LOAD.RADIOBUTTON("", "BITCH,RADIO,33,11,10,1,Bitch,Helv10,LEFT,SEX", "")
```

This example is taken from a full program example - [click here](#)  for details.

String lists are used in conjunction with list boxes, simple combo boxes and drop-down combo boxes. String lists contain the entries used to populate these boxes. String lists are created and managed quite separately to the list/combo boxes which use them.

You can therefore use a single string list in multiple boxes, and create and destroy boxes without destroying the underlying data.



Creating string lists.



Format of string lists.



Loading a string list.






To create string lists, either download the strings from the host, or read them in from a PC file. The second form is better suited to longer lists. Although there is no inbuilt limit to the number of items in such a list, they are not intended for very large lists, because:

- A list box control cannot contain more than 64k of text, e.g. if the average string length is 90 bytes, a list box will not hold more than approximately 700 items.
- String lists are held in memory at all times.
- The time required to create large lists will be unacceptable to users.
- The time required to populate list/combo boxes with large lists will be unacceptable to users.

The upper 'realistic' limit is probably a few hundred items.

In its simplest form, a string list has an **ID** (a name), and an ordered sequence of strings. The order defines the default order in which the strings are displayed in a list or combo box (although this can be changed when the boxes are actually created).

String lists may also store a second, hidden, string for each item. This string is not displayed to the user, but can be used by the host application as an alternative way for list items to be specified in messages exchanged between HOSTACCESS and the host. See the examples for details.

-  Example 1.
-  Example 2.
-  Example 3.
-  Example 4.
-  Program Example.

To load a string list so it can be used by list type controls, call the following subroutine from your application:

```
CALL PIX.GUI.LOAD.STRING(GUI.APP.NAME, CONTROL.DETAILS, STRING)
```

Where

**CONTROL.DETAILS**      Control ID, STRING, TEXT1,...TEXTn

Where:

**Control ID**            The unique name for the control  
**STRING**                The text literal STRING  
**TEXT1,...TEXTn**        The list of text items to be placed into the String list.

The host application has a screen on which one of the pieces of information the user has to enter is a country name. The host application designer chooses to do this with a simple combo box, which has a list of common countries, but will also let the user type in a country that 's not on the list. All the host application wants to get from the user is the text of the country name.

This is best suited to the simple form of string list, without use of hidden strings. The host downloads the list of countries in a string list, then creates a 'simple combo ' style box. When extracting the selected country, or the name the user entered, HOSTACCESS sends the relevant text to the host.

To create a String list containing the following list "England, America, Canada, Germany, France" use the following.

```
CONTROL.DETAILS = "COUNTRIES, STRING, England, America, Canada, Germany, France"  
CALL PIX.GUI.LOAD.STRING("", CONTROL.DETAILS, "")
```

To amend the above list so that the continents were adjacent to the countries for use in a tabular list box, use the following. (where ">" denotes a tab mark in the data)

```
CONTROL.DETAILS = "COUNTRIES, STRING, England>Europe, America>N.America, Canada>N.America,  
Germany>Europe, France>Europe"
```



The entries in the list contain a 'display' part, and optionally, a 'hidden' part. If present, the hidden part is separated from the display part by a dash or minus sign (so you cannot have dashes in the text). If the hidden part is not given, a default hidden value will be automatically created if it is ever needed- this will be a string representation of the position of the entry in the string list (starting from 1; '1', '2', '3' etc.).

List entries to be added/removed are given directly or indirectly. When given directly, a string parameter specifies the list entry in the format

```
<display-part> <hidden-part>
```

If a string parameter starts with a '@' character, it is treated as an indirect entry. The '@' is stripped off, and the remainder treated as a PC file name. The file contains a list of entries in the above format. It is possible to mix the direct and indirect forms in a single escape sequence.

**Note:** the comma, dash and '@' characters cannot normally be used in display strings, because of their special significance in the above formats.

When adding strings, the second string parameter contains the display text of the existing string list entry before which the new entries are to be inserted. If missing, the new entries are added to the end of the list.

When removing entries, the hidden parts of entries are ignored.

Consider a host application that needs to get the user to select a personnel record from a database. Each record includes the person's name. Each record has a record number. The host application wants to use a dropdownlist style combo box (one in which the user cannot type an entry, but has to select from the list) to get the name. The host application is not really interested in the text of the name, but the record number it relates to.

This is more suited to a string list with hidden strings. The 'display' strings are the names of the people, the hidden strings are the associated record numbers. The host creates such a string list, then associates it with a 'dropdown combo' style box. The host also specifies that it wants to use the hidden strings when exchanging information with HOSTACCESS about the selected list items. HOSTACCESS will then send back the record number of the selected item, which the host application can use directly.

Create a list of people, with hidden strings (record numbers in some database). The bulk of the list is created from a PC file called **people.lst**. To this are added 2 people given directly. The list will be called 'people' and will eventually contain the following entries, in the order given:

<b>Display text</b>	<b>Hidden text</b>	<b>Source</b>
D. Bailey	173	people.lst
M. Woolley	174	people.lst
G. Baker	190	people.lst
F. Carden	191	people.lst
A.Hedgecock	160	direct from host
P.Hall	143	direct from host


Use the following:

```
CONTROL.DETAILS = "PEOPLE, STRING, @people.lst, A.Hedgecock-160,  
P.Hall-143"
```

**people.lst** is a standard DOS file with <CR><LF> characters separating each line of text. The file name could also include the full directory path, for example **c:host\data\people.lst**. By default, the path is your HOSTACCESS directory. In this example, **people.lst** looks like this:

```
D.Bailey-173  
M.Woolley-174  
G.Baker-190  
F.Carden-191
```

```
075 CALL PIX.GUI.LOAD.STRING  
    ("", "GROUP.LIST,STRING,Working,Gundogs,Hounds,Utility,Toys,Terriers", "")
```

This example is taken from a full program example - [click here](#)  for details.

**Note :** The > on line 77 represent the Tab character. Decimal Char(9).

To create a simple combo box, call the following subroutine from your application:

**CALL PIX.GUI.LOAD.SIMCOMB(GUI.APP.NAME, CONTROL.DETAILS, STRING)**

Where

**CONTROL.DETAILS** Control ID, SIMCOMB, X, Y, W, H, Label,  
{Key1,...Keyn}

Where:

**Control ID** The unique name for the combo box.  
**SIMCOMB** Text literal SIMCOMB.  
**X** The X co-ordinate of the left of the box.  
**Y** The Y co-ordinate of the top of the box.  
**W** Width of the box (characters).  
**H** Height of the box (characters).  
**Label** String List associated with the Simple Combo.  
**Key1...Keyn** Additional parameters if required, in any order - see below.

**The following extra keyword parameters are available:**


<a href="#">HIDE</a>	<a href="#">SHOW *</a>	<a href="#">GRAYED</a>	<a href="#">UNGRAYED *</a>
<a href="#">TERMITE *</a>	SYSTEM	<a href="#">HELV8</a>	<a href="#">HELV10</a>
<a href="#">SORT *</a>	<a href="#">NOSORT</a>	<a href="#">HSCROLL</a>	<a href="#">HIDDENMSG</a>
<a href="#">DISPLAYMSG*</a>	<a href="#">HSCROLLBAR</a>	<a href="#">INITSELECT</a>	<a href="#">BORDERON</a>
<a href="#">BORDEROFF *</a>			

Default settings are denoted by \*. Click here  for details.

Alternatively, you can use the [PIX.GUI.CONTROL.LOAD](#) subroutine.

Program Example.

```
098 CALL PIX.GUI.LOAD.SIMCOMB("", "BREED, SIMCOMB, 9, 10, 15, 6, BREEDS.LIST, BORDERON", "")
```

This example is taken from a full program example - [click here](#)  for details.

To create a drop-down combo box, call the following subroutines from your application:

**CALL PIX.GUI.LOAD.DROPCOMB(GUI.APP.NAME, CONTROL.DETAILS, STRING)**

Where:

**CONTROL.DETAILS** Control ID, DROPCOMB, X, Y, W, H, Label,  
{Key1,...Keyn}

Where:


**Control ID** The unique name for the combo box.  
**DROPCOMB** Text literal DROPCOMB.  
**X** The X co-ordinate of the left of the box.  
**Y** The Y co-ordinate of the top of the box.  
**W** Width of the box (characters).  
**H** Height of the box (characters).  
**Label** String List associated with the drop down Combo.  
**Key1...Keyn** Additional parameters if required, in any order - see below.

The following extra keyword parameters are available:


<a href="#">HIDE</a>	<a href="#">SHOW *</a>	<a href="#">GRAYED</a>	<a href="#">UNGRAYED *</a>
<a href="#">TERMITE *</a>	SYSTEM	<a href="#">HELV8</a>	<a href="#">HELV10</a>
<a href="#">SORT *</a>	<a href="#">NOSORT</a>	<a href="#">HSCROLL</a>	<a href="#">HSCROLLBAR</a>
<a href="#">HIDDENMSG</a>	<a href="#">DISPLAYMSG*</a>	<a href="#">BORDERON</a>	<a href="#">BORDEROFF *</a>
<a href="#">READONLY</a>	<a href="#">READWRITE *</a>	<a href="#">INITSELECT</a>	<a href="#">TOPBOX</a>

Default settings are denoted by \*. Click here  for details.

Alternatively, you can use the [PIX.GUI.CONTROL.LOAD](#) subroutine.

 [Program example.](#)

```
094 CALL PIX.GUI.LOAD.DROPCOMB ("", "GROUP, DROPCOMB, 9, 4, 15, 5, GROUP.LIST", "")
```

This example is taken from a full program example - [click here](#)  for details.

To create a list box, call the following subroutines from your application:

**CALL PIX.GUI.LOAD.LIST (GUI.APP.NAME, CONTROL.DETAILS, STRING)**

Where


**CONTROL.DETAILS** Control ID, LIST, X, Y, W, H, Label, {Key1,...Keyn}

Where:

**Control ID** The unique name for the list box.  
**LIST** Text literal LIST.  
**X** The X co-ordinate of the left of the box.  
**Y** The Y co-ordinate of the top of the box.  
**W** Width of the box (characters)  
**H** Height of the box (characters)  
**Label** String List associated with the list box.  
**Key1...Keyn** Additional parameters if required, in any order - see below.

The following extra keyword parameters are available:

<a href="#">HIDE</a>	<a href="#">SHOW *</a>	<a href="#">GRAYED</a>	<a href="#">UNGRAYED *</a>
<a href="#">TERMITE *</a>	SYSTEM	<a href="#">HELV8</a>	<a href="#">HELV10</a>
<a href="#">SORT *</a>	<a href="#">NOSORT</a>	<a href="#">HSCROLL</a>	<a href="#">HSCROLLBAR</a>
<a href="#">HIDDENMSG</a>	<a href="#">DISPLAYMSG*</a>	<a href="#">INITSELECT</a>	SIZEBOX
<a href="#">TOPBOX</a>	<a href="#">BORDERON</a>	<a href="#">BORDEROFF *</a>	<a href="#">TABBED</a>
<a href="#">MORE</a>			

Default settings are denoted by \*. Click here  for details.


Alternatively, you can use the [PIX.GUI.CONTROL.LOAD](#) subroutine.



Program Example.



```
102     CALL PIX.GUI.LOAD.LIST ("", "CLASS, LIST, 33, 4, 25, 6, CLASS.LIST, TABBED", "")
```

This example is taken from a full program example - [click here](#)  for details.

To create an edit box, call the following subroutines from your application:

**CALL PIX.GUI.LOAD.EDIT (GUI.APP.NAME, CONTROL.DETAILS, STRING)**

Where:

**CONTROL.DETAILS** Control ID, EDIT, X, Y, W, H, Label, {Key1,...Keyn}

Where:

**Control ID** The unique name for the edit box.  
**EDIT** Text literal EDIT.  
**X** The X co-ordinate of the left of the edit box.  
**Y** The Y co-ordinate of the top of the edit box.  
**W** Width of the box (characters)  
**H** Height of the box (characters)  
**Label** Edit text to be passed to the Edit Control.  
**Key1...Keyn** Additional parameters if required, in any order - see below.

The following extra keyword parameters are available:


<a href="#">HIDE</a>	<a href="#">SHOW*</a>	<a href="#">GRAYED</a>	<a href="#">UNGRAYED *</a>
<a href="#">TERMITE *</a>	<a href="#">HELV8</a>	<a href="#">HELV10</a>	<a href="#">PASSWORD</a>
<a href="#">READONLY</a>	<a href="#">READWRITE *</a>	<a href="#">NOSELECT</a>	<a href="#">SELECT *</a>
<a href="#">HSCROLL</a>	<a href="#">VSCROLL</a>	<a href="#">BORDERON</a>	<a href="#">BORDEROFF *</a>
<a href="#">HSCROLLBAR</a>	<a href="#">VSCROLLBAR</a>	<a href="#">UPPER</a>	<a href="#">LOWER</a>
<a href="#">NOCASECONVERT*</a>	<a href="#">LINELEN</a>	<a href="#">SCULPTURE</a>	<a href="#">INTRANGE</a>
<a href="#">LONGDATE</a>	<a href="#">SHORTDATE</a>	<a href="#">ABBRDATE</a>	SYSTEM

Default settings are denoted by \*. Click here  for details.

Alternatively, you can use the [PIX.GUI.CONTROL.LOAD](#) subroutine.

Program Example.

```
085 CALL PIX.GUI.LOAD.EDIT ("", "NAME,EDIT,9,2,26,1,Lezand, HSCROLL,SCULPTURE", "")
086 CALL PIX.GUI.LOAD.EDIT ("", "D.O.B,EDIT,45,2,10,1,21 Apr 92,SCULPTURE", "")
087 CALL PIX.GUI.LOAD.EDIT
("", "ADDRESS,EDIT,33,13,25,3,,HSCROLL,VSCROLL,VSCROLLBAR,SCULPTURE", "")
```

This example is taken from a full program example - [click here](#)  for details.

To create a static text label, call the following subroutine from your application:

**CALL PIX.GUI.LOAD.LABEL(GUI.APP.NAME, CONTROL.DETAILS, STRING)**

Where:


**CONTROL.DETAILS** Control ID, LABEL, X, Y, W, H, Label, {Key1,...Keyn}

Where:


**Control ID** The unique name for the text label.  
**LABEL** Text literal LABEL.  
**X** The X co-ordinate of the left of the label.  
**Y** The Y co-ordinate of the top of the label.  
**W** Width of the label (characters)  
**H** Height of the label (characters)  
**Label** Text to be printed as Label.  
**Key1...Keyn** Additional parameters if required, in any order - see below.

The following extra keyword parameters are available:

<a href="#">HIDE</a>	<a href="#">SHOW</a> *	<a href="#">GRAYED</a>	<a href="#">UNGRAYED</a> *
<a href="#">TERMITE</a> *	SYSTEM	<a href="#">HELV8</a>	<a href="#">HELV10</a>
<a href="#">BORDERON</a>	<a href="#">BORDEROFF</a> *	FONTNAME	<a href="#">RIGHT</a>
CENTRE	<a href="#">LEFT</a> *		

Default settings are denoted by \*. Click here  for details.

Alternatively, you can use the [PIX.GUI.CONTROL.LOAD](#) subroutine.

 [Program Example](#).

```
052 CALL PIX.GUI.LOAD.LABEL("" , "LAB1,LABEL,1,2,6,1,Name : ,Helv10,Right", "")
053 CALL PIX.GUI.LOAD.LABEL("" , "LAB2,LABEL,1,4,6,1,Group : ,Helv10,Right", "")
054 CALL PIX.GUI.LOAD.LABEL("" , "LAB3,LABEL,1,10,6,1,Breed : ,Helv10,Right", "")
055 CALL PIX.GUI.LOAD.LABEL("" , "LAB4,LABEL,37,2,6,1,D.O.B : ,Helv10,Right", "")
056 CALL PIX.GUI.LOAD.LABEL("" , "LAB5,LABEL,25,4,6,1,Class : ,Helv10,Right", "")
057 CALL PIX.GUI.LOAD.LABEL("" , "LAB6,LABEL,25,13,6,1,Address : ,Helv10,Right", "")
```

This example is taken from a full program example - [click here](#)  for details.

Commands are controls associated with menus, toolboxes or toolbars. Commands can be associated with text (for use in menus), or with button images (for use in toolbars and toolboxes).

Once a command has been created, you make it visible to the user by adding it to a 'command container' object, such as a toolbar, and then make the toolbar visible. For example, you can load ten command controls into HOSTACCESS 's memory and then associate some of them to menus, some to the Toolbar and some to a floating toolbox.

A single command can also be loaded into all three of the container objects at the same time. For example, a Command control called FILE can be loaded into a Menu, Toolbar and Toolbox at the same time.



Creating a command.



Command program example.



To create a command as a control, call the following subroutine from your application:

**CALL PIX.GUI.LOAD.COMMAND (GUI.APP.NAME, CONTROL.DETAILS, STRING)**

Where

**CONTROL.DETAILS** Control ID, COMMAND, MENUTEXT,  
STATUSTEXT, BUTTONSPEC, {Key1,...Keyn}

Where:

**Control ID** The Unique name for the Control  
**COMMAND** The text literal COMMAND or CHOICE  
**MENUTEXT** Text to be displayed in Menu item e.g. Save As.. in the Session menu. If the Command is to be used in a toolbar or toolbox option then this entry can be null.  
**STATUSTEXT** Single line help text to be displayed on the Status line if selected.  
**BUTTONSPEC** Specification of button to be displayed in toolbar or toolbox. Click here  for information on Text push-buttons and here  for image push-buttons. If Command is to be used in a menu option then this entry can be null.  
**Key1...Keyn** Are the additional parameters if required in any order (see below).

The following extra keyword parameters are available:

<a href="#">HIDE</a>	<a href="#">SHOW</a> *	<a href="#">CLIPPED</a>	<a href="#">SCALED</a> *
<a href="#">PUSH</a>	<a href="#">FRAME</a>	<a href="#">PUSHEDIN</a>	<a href="#">PUSHEDOUT</a>
<a href="#">SHADOWED</a>	<a href="#">NONE</a> *	<a href="#">LABELED</a>	<a href="#">BITMAP</a> *
<a href="#">ICON</a>	<a href="#">FILEID</a>	LABLEPOS	<a href="#">TERMITE</a>
<a href="#">HELV8</a>	<a href="#">HELV10</a>	SUBS	TEXTBUTTON
FILETYPE	<a href="#">FILEID</a>		TILESIZE
TILE	MENUTEXT		

Default settings are denoted by \*. Click here  for details.

Alternatively, you can use the [PIX.GUI.CONTROL.LOAD](#) subroutine.

```
028 CALL PIX.GUI.LOAD.COMMAND("", "COMM1,CHOICE,Command1,This is Command 1, \BITMAPS\ICON_BMP\HALT.BMP", "")
029 CALL PIX.GUI.LOAD.COMMAND("", "COMM2,CHOICE,Command2,This is Command 2, \BITMAPS\ICON_BMP\TYPE.BMP", "")
030 CALL PIX.GUI.LOAD.COMMAND("", "COMM3,CHOICE,Command3,This is Command 3, \BITMAPS\ICON_BMP\TOOLS.BMP", "")
031 CALL PIX.GUI.LOAD.COMMAND("", "COMM4,CHOICE,Command4,This is Command 4, \BITMAPS\ICON_BMP\SHUTDOOR.BMP", "")
032 CALL PIX.GUI.LOAD.COMMAND("", "COMM5,CHOICE,Command5,This is Command 5, \BITMAPS\ICON_BMP\SHREEK.BMP", "")
033 CALL PIX.GUI.LOAD.COMMAND("", "COMM6,CHOICE,Command6,This is Command 6, \BITMAPS\ICON_BMP\QUESTION.BMP", "")
034 CALL PIX.GUI.LOAD.COMMAND("", "COMM7,CHOICE,Command7,This is Command 7, \BITMAPS\ICON_BMP\POSTBOX.BMP", "")
035 CALL PIX.GUI.LOAD.COMMAND("", "COMM8,CHOICE,Command8,This is Command 8, \BITMAPS\ICON_BMP\PHONE.BMP", "")
036 CALL PIX.GUI.LOAD.COMMAND("", "COMM9,CHOICE,Command9,This is Command 9, \BITMAPS\ICON_BMP\PAPER.BMP", "")
037 CALL PIX.GUI.LOAD.COMMAND("", "COMM10,CHOICE,Command10,This is Command 10, \BITMAPS\ICON_BMP\
    OPENDOOR.BMP", "")
```

This example is taken from a full program example - [click here](#)  for details.



The toolbar is also referred to as a speed bar and is placed above the session screen. HOSTACCESS has a standard toolbar that will be replaced with the newly specified toolbar.

To create a toolbar containing associated commands, call the following subroutine from your application:

**CALL PIX.GUI.LOAD.TOOLBAR (GUI.APP.NAME, CONTROL.DETAILS, STRING)**


Where

**CONTROL.DETAILS**    “Control ID, TOOLBAR, ADD,COMMAND1,...COMMANDn”

Where:

<b>Control ID</b>	Is the Unique name for the Control
<b>TOOLBAR</b>	Is the text literal TOOLBAR
<b>ADD</b>	Is the literal ‘ADD ’; to append commands to the current toolbar. By default, a new toolbar is created (the current toolbar is over-written)
<b>COMMAND1... COMMANDn</b>	This is a list of command Control ID ’s that wish to be used in the toolbar. Each Command control ID will be separated by commas or Value marks. The icons /bitmaps will be loaded from left to right in order specified. If a new line is required to give multiple toolbars then the text literal “NEWLINE” as a Command Control Id will be needed.


Alternatively, you can use the [PIX.GUI.CONTROL.LOAD](#) subroutine.

This example is taken from a full program example - click here  for details.



Program Example.

```
042      CALL PIX.GUI.LOAD.TOOLBAR ("", "BAR1, TOOLBAR, ADD, COMM7, COMM8, COMM9, COMM10", "")
```

This example is taken from a full program example - [click here](#)  for details.

The floating toolbox is a set of icons/bitmaps that are placed in a group on the session screen. This toolbox can be moved around the screen and even minimized if required by the user.

To create a toolbox containing the associated commands, use the following subroutine from your application:

**CALL PIX.GUI.LOAD.TOOLBOX(GUI.APP.NAME, CONTROL.DETAILS, STRING)**

Where


**CONTROL.DETAILS**      “Control ID, TOOLBOX, X,Y, Title, C1,...Cn,  
                                  {Key1,...Keyn}”

Where:

<b>Control ID</b>	The Unique name for the Control
<b>TOOLBOX</b>	The text literal TOOLBOX
<b>X</b>	The X co-ordinate of the toolbox.
<b>Y</b>	The Y co-ordinate of the toolbox.
<b>Title</b>	The title text to display on the toolbox.
<b>C1... Cn</b>	A list of command Control ID 's that wish to be used in the toolbox. Each Command control ID will be separated by commas or Value marks. The icons /bitmaps will be loaded from left to right in order specified. If a new line is required to give a block effect then the text literal “NEWLINE” as a Command Control Id will be needed.
<b>Key1..Keyn</b>	Are the additional parameters if required in any order (see below)


The following extra keyword parameters are available:

**MODELESS**      **MIN**      **ICON**      **BORDEROFF**


Default settings are denoted by \*. Click here  for details.

Alternatively, you can use the [PIX.GUI.CONTROL.LOAD](#) subroutine.

This example is taken from a full program example - click here  for details.

 [Program Example.](#)

```
041 CALL PIX.GUI.LOAD.TOOLBOX("", "FLOAT1, TOOLBOX, 60, 2, FLOATING TOOLBAR, COMM1, COMM2,  
COMM3, COMM4, COMM5, COMM6", "")
```

This example is taken from a full program example - [click here](#)  for details.

You can alter HOSTACCESS 's pull down menu that appears at the top of the session screen, adding extra menu items. You cannot remove HOSTACCESS 's menu items, as they are the control options for the HOSTACCESS session. The only existing HOSTACCESS menu option that can be amended is the **Help** menu option where items can be appended to the end. New menus are placed to the left of the **Help** menu.

To show menus, you need to first create the menu holding pre-defined commands, and then display the menu. This process is described in the following sections.

Click here  for details of creating commands.

To create a menu, holding pre-defined commands, call the following subroutine from your application:

**CALL PIX.GUI.LOAD.MENU(GUI.APP.NAME, CONTROL.DETAILS, STRING)**

Where:

**CONTROL.DETAILS**      “Control ID, MENU, MENUTEXT,  
COMMAND1,...COMMANDn”

Where:


<b>Control ID</b>	The Unique name for the Control
<b>MENU</b>	The text literal MENU
<b>MENUTEXT</b>	Title to display on Menu status line as a description of the menu.
<b>COMMAND1...</b> <b>COMMANDn</b>	A list of command control ID 's that you wish to use in a menu set. Each Command control ID will be separated by commas or Value marks. The command controls will be loaded from top to bottom in order specified. If a new line is required to give a separator line to divide groups of menu items then the text literal “NEWLINE” as a Command control Id will be needed.

Alternatively, you can use the [PIX.GUI.CONTROL.LOAD](#) subroutine.



Menu Program Example.

```
043 CALL  
PIX.GUI.LOAD.MENU("", "MENU1, MENU, EXAMPLES, COMM1, COMM2, COMM3, COMM4, COMM5, COMM6, COMM7, COMM8, COMM9, C  
COMM10", "")
```

This example is taken from a full program example - [click here](#)  for details.

To display a created menu on the Windows Menu bar, with associated commands, call the following subroutine from your application:

**CALL PIX.GUI.MENU.INSTALL(GUI.APP.NAME, CONTROL.DETAILS, STRING)**

Where

**CONTROL.DETAILS** "Control ID, MENUINSTALL, COMMAND1,...COMMANDn"

Where:

**Control ID** The Unique name for the Control

**MENUINSTALL** The text literal MENUINSTALL

**COMMAND1...** A list of Menu Control ID s that will be loaded on the Menu bar.  
**COMMANDn** Each Command control ID will be separated by commas or Value marks. The Menu controls will be loaded from left to right in order specified after the last HOSTACCESS menu.


Alternatively, you can use the [PIX.GUI.CONTROL.LOAD](#) subroutine.



Program Example.



```
044 CALL PIX.GUI.MENU.INSTALL("", "MENU, MENUINSTALL, MENU1", "")
```








This example is taken from a full program example - [click here](#)  for details.

The following topics explain the grid features available with HOSTACCESS 6.2 only.

Grids are a way of representing a two dimensional table of information on the screen. This is a very useful way of displaying related information, across columns, row by row. Additionally, if showing values, this gives the appearance of a spreadsheet 'look & feel' to your application.

**Note:** This version of grids in HOSTACCESS is not backwards compatible with any previous product versions. The grid used is a Visual Basic custom VBX control which specifically works with HOSTACCESS.

When creating a grid, it is necessary to define some of the key elements so that the grid can be displayed. If other features are then desired, either at a user request, or coded options, then the appropriate calls can be made to 'turn on/off' the features. See the topics listed below for more detail:

-  Creating a grid.
-  Setting scrollbars.
-  Obtaining the current grid position..
-  Setting focus to the grid
-  Changing column widths.
-  Deleting rows.
-  Clearing the grid.

To load a grid, the BASIC routine PIX.GUI.LOAD.GRID can be called from your host application in the following way:

**CALL PIX.GUI.LOAD.GRID(GUI.APP.NAME, CONTROL.DATA, OUTPUT.STRING)**

Where:

**GUI.APP.NAME** Is an optional application label used to uniquely identify all or some controls. This can prove to be useful when wishing to hide/unload a group of controls.

**CONTROL.DATA** Is the grid parameters, and has the following syntax.

**Control id,GRID,startcolumn,startrow,width,height,maxcolumns,maxrows,keywords**

Where:

**control id** Is unique name for the grid.

**GRID** Is a literal string.

**start column** Is the starting column position of the grid.

**start row** Is the starting row position of the grid.

**width** Is the visible width of the grid on the screen.

**height** Is the visible height of the grid on the screen.

**maxcolumns** Is the maximum number of columns in the grid.

**maxrows** Is the maximum number of rows in the grid.

**keywords** All keywords are the same as the existing documentation.  
Note: GRAYED or GREYED are both acceptable.

**output.string** If set to literal 'O' the command string that would be used to create the grid is returned and not actioned. If null, this is returned as null.

Click here  for an example.

This is an example PICK program which will create a simple grid and set the column headings and widths. In this example, the data has been manually typed in. An alternative would have been to call the PIX.GUI.GRID.SET.VALUE routine to set the entire grid data.

```
001 *
002 * Example program to show various Grid program calls
003 *
004     PROMPT ""
005     PRINT @(-1):@(25,0):"Monthly Balances Display"
006 *
007 * Display the grid
008 *
009     CALL PIX.GUI.LOAD.GRID("", "FIGURES, GRID, 5, 2, 37, 14, 4, 12, NOROW", "")
010 *
011 * Now change the column headings and widths
012 *
013     GRID.PARAM = ""
014     GRID.PARAM<1> = "Month, Opening, Closing, Balance"
015     GRID.PARAM<4> = "5, 8, 8, 8"
016 *
017     CALL PIX.GUI.GRID.SET.PARAM("", "FIGURES", GRID.PARAM)
018 *
019 END
```

To clear a grid, the BASIC routine PIX.GUI.GRID.CLEAR can be called from your host application in the following way:

**CALL PIX.GUI.GRID.CLEAR(GUI.APP.NAME, GRID.NAME)**

Where:

- GUI.APP.NAME** Is an optional application label used to uniquely identify all or some controls. This can prove to be useful when wishing to hide/unload a group of controls.
- GRID.NAME** Is the name of the grid to clear.

To change the width of a column, the basic routine PIX.GUI.GRID.COLWIDTH can be called from your host application in the following way.

**CALL PIX.GUI.GRID.COLWIDTH(GUI.APP.NAME, GRID.NAME, COLUMN.NUMBER, NEW.WIDTH)**

Where:

- GUI.APP.NAME** Is an optional application label used to uniquely identify all or some controls. This can prove to be useful when wishing to hide/unload a group of controls.
- GRID.NAME** Is the name of the grid to change.
- COLUMN.NUMBER** Is the column number whose width is to be changed.
- NEW.WIDTH** Is the new width of the column.

To delete a specific or selected row of a grid, the BASIC routine PIX.GUI.GRID.DELETE can be called from your host application in the following way.

**CALL PIX.GUI.GRID.DELETE(GUI.APP.NAME, GRID.NAME, ROW.NUMBER)**

Where:

- GUI.APP.NAME** Is an optional application label used to uniquely identify all or some controls. This can prove to be useful when wishing to hide/unload a group of controls.
- GRID.NAME** Is the name of the grid to change.
- ROW.NUMBER** Is the row number that you wish to delete. If the row number is not specified or is set to zero, the selected grid row will be deleted.

To set focus to a specific column and row of the grid, the BASIC routine PIX.GUI.GRID.FOCUS can be called from your host application in the following way.

**CALL PIX.GUI.GRID.FOCUS(GUI.APP.NAME, GRID.NAME, COLUMN.NUMBER, ROW.NUMBER)**

Where:

- GUI.APP.NAME** Is an optional application label used to uniquely identify all or some controls. This can prove to be useful when wishing to hide/unload a group of controls.
- GRID.NAME** Is the name of the grid to change.
- COLUMN.NUMBER** Is the column number that you wish to set focus onto.
- ROW.NUMBER** Is the row number that you wish to set focus onto.

To obtain your current grid position column and row of a grid, the BASIC routine PIX.GUI.GRID.POSITION can be called from your

host application in the following way.

**CALL PIX.GUI.GRID.POSITION(GUI.APP.NAME, GRID.NAME, COLUMN.NUMBER, ROW.NUMBER)**

Where:

**GUI.APP.NAME** Is an optional application label used to uniquely identify all or some controls. This can prove to be useful when wishing to hide/unload a group of controls.

**GRID.NAME** Is the name of the grid to change.

**COLUMN.NUMBER** Is the column number returned, that the grid is active on.

**ROW.NUMBER** Is the row number returned, the grid is active on.

To display/remove scrollbars for a grid, the BASIC routine PIX.GUI.GRID.SCROLLBARS can be called from your host application in the following way.

**CALL PIX.GUI.GRID.SCROLLBARS(GUI.APP.NAME, GRID.NAME, horizontal, vertical)**

Where:

**GUI.APP.NAME** Is an optional application label used to uniquely identify all or some controls. This can prove to be useful when wishing to hide/unload a group of controls.

**GRID.NAME** Is the name of the grid to change.

**HORIZONTAL** If set to a value other than zero, will display a horizontal scrollbar. If zero, no scrollbar is displayed.

**VERTICAL** If set to a value other than zero, will display a vertical scrollbar. If zero, no scrollbar is displayed.



Example.

```
018 *  
019 * Display both scrollbars  
020 *  
021     CALL PIX.GUI.GRID.SCROLLBARS("", "FIGURES", 1, 1)
```

To load a custom VBX control, the BASIC routine PIX.GUI.LOAD.VBX can be called from your host application in the following way. Custom VBX controls allow the ability to integrate Visual Basic with host software and utilise their features.

**CALL PIX.GUI.LOAD.VBX(GUI.APP.NAME, CONTROL.DATA, OUTPUT.STRING)**

Where:

**GUI.APP.NAME** Is an optional application label used to uniquely identify all or some controls. This can prove to be useful when wishing to hide/unload a group of controls.

**CONTROL.DATA** Is the grid parameters, and has the following syntax.

**control id,GRID,startcolumn,startrow,width,height,maxcolumns,maxrows,keywords**

Where:

**control id** Is unique name for the grid.

**start column** Is the starting column position of the grid.

**start row** Is the starting row position of the grid.

**width** Is the visible width of the grid on the screen.

**height** Is the visible height of the grid on the screen.

**caption** Is the caption to display in the initial window.

**filename** Is the filename of the VBX custom control that you wish to load.

**VBXname** Is the name of the VBX custom.

**keywords** The following keywords can also be specified :

HIDE, SHOW\*, GREYED, UNGREYED\*,  
SCULPTURE,TERMITE\*, SYSTEM, HELV8, HELV10

\*denotes default settings.

Note: GRAYED or GREYED are both acceptable.

**output.string** If set to literal 'O' the command string that would be used to load the VBX is returned and not actioned. If null, this is returned as null.

The following topics describe PICK subroutines which perform general control management functions on a named control.



Showing controls.



Hiding controls.



Changing control colours.



Setting input focus to a control.



Using control groups.



Returning an alternative message.



Setting the accelerator character.



Defining a base control group.



Reading the contents of a control.



Setting the contents of a control.



Graying a control.




Un-graying a control.



Unloading a control.



Moving a control.

Click here  for a description of controls.



To show a specific control or group of controls, call the following subroutine from your application:

**CALL PIX.GUI.SHOW.CONTROL (GUI.APP.NAME, CONTROL.DETAILS, STRING)**

Where:

**CONTROL.DETAILS**      Control ID,SHOW

Where:

**Control ID**              The name of the control or control group you wish to show.

**SHOW**                      The text literal SHOW

Alternatively, you can use the [PIX.GUI.CONTROL.LOAD](#) subroutine.

For example, to show the group of controls called SOFTWARE use the following:

```
CONTROL.DETAILS<1> = 'SOFTWARE, SHOW'  
CALL PIX.GUI.SHOW.CONTROL (GUI.APP.NAME, CONTROL.DETAILS, STRING)
```

To hide a specific control or group of controls call the following subroutine from your application:

```
CALL PIX.GUI.CONTROL (GUI.APP.NAME, CONTROL.DETAILS, STRING)
```

Where

**CONTROL.DETAILS**      Control ID,HIDE

Where:

**Control ID**            The name of the control or control group you wish to hide

**HIDE**                    The text literal HIDE.

Alternatively, you can use the [PIX.GUI.CONTROL.LOAD](#) subroutine.

For example, to hide the group of controls called HARDWARE use the following:

```
CONTROL.DETAILS<1> = 'HARDWARE,HIDE'  
CALL PIX.GUI.CONTROL.SHOW (GUI.APP.NAME, CONTROL.DETAILS, STRING)
```

To change the foreground and background colours for a control, call the following subroutine from your application:

**CALL PIX.GUI.LOAD.COLOUR (GUI.APP.NAME, CONTROL.DETAILS, STRING)**

Where:

**CONTROL.DETAILS** Control ID, COLOUR, FOREGROUND, BACKGROUND, GRAYED COLOUR

Where:

**Control ID** The control ID of the control to be affected.  
To affect all subsequent controls, use the literal ALLCONTROLS.

**COLOUR** The text literal COLOR or COLOUR.

**FOREGROUND** The foreground colour of the control - see below for details.

**BACKGROUND** The background colour of the control - see below for details.

You can use the following literal keywords to define your colours:

BLACK	BLUE	GREEN	CYAN
RED	MAGENTA	BROWN	GRAY
LIGHTBLACK	LIGHTBLUE	LIGHTGREEN	LIGHTCYAN
LIGHTRED	LIGHTMAGENTA	LIGHTBROWN	LIGHTGRAY
CLEAR	WHITE	YELLOW	

Alternatively, you can use the [PIX.GUI.CONTROL.LOAD](#) subroutine.

Click here  for an example.

To set the colour of the control called TITLE to RED on WHITE use the following.

```
CONTROL.DETAILS<1> = 'TITLE,COLOUR,RED,WHITE '
```

```
CALL PIX.GUI.LOAD.COLOUR (GUI.APP.NAME, CONTROL.DETAILS, STRING)
```

To set all proceeding controls to colour BLACK on WHITE use the following:

```
CONTROL.DETAILS<1> = 'ALLCONTROLS,COLOUR,BLACK,WHITE '
```

```
CALL PIX.GUI.LOAD.COLOUR (GUI.APP.NAME, CONTROL.DETAILS, STRING)
```

To set the input focus of a control call the following subroutine from your application:

**CALL PIX.GUI.CONTROL.SET.FOCUS (GUI.APP.NAME, Control ID, ERROR)**

Where:

**GUI.APP.NAME** Used to identify a given set of controls. Can be null.

**Control ID** Control ID of the Control that is to gain Focus.

**ERROR** Set to 1 is Error (for future use)

Groups are very useful types of controls. They can be used as the name implies to group a number of associated controls. This comes into its element when the programmer wishes to hide, show or gray out a sub set of all of the controls. You can define many groups within your list of controls and you refer to them by one Control ID rather than many.

For example if your application has a number of buttons referring to actions on an update screen you can put all these action buttons under one control group. You can then gray-out the group when there is no record on the screen. i.e. on startup and un-gray the group when a record is present i.e. Amend mode. This saves programming time when developing applications and also reduces communications traffic to and from HOSTACCESS.

To define a group of controls, call use the following subroutine from your application:

```
CALL PIX.GUI.LOAD.GROUP(GUI.APP.NAME, CONTROL.DETAILS, STRING)
```

Where:

**CONTROL.DETAILS** Control ID, GROUP, CONTROL1,...CONTROLn

Where:

**Control ID** The Unique name for the Control

**GROUP** The text literal GROUP

**CONTROL1,...  
CONTROLn** A list of Control ID that are contained within this group.

Alternatively, you can use the [PIX.GUI.CONTROL.LOAD](#) subroutine.

To assign the following Controls to the group ACTION use the following:

Controls to be assigned are: FILE, LOOKUP, DELETE

**CONTROL.DETAILS<1> = 'ACTION,GROUP,FILE,LOOKUP,DELETE '**


As well as returning a true event to the Host you can also assign a string against a given event for a control. This is useful for returning keystrokes against an action on a control. For instance you may wish to send the ESCape key when the Cancel PUSHBUTTON is pressed. At present you can not use the scancode keys to send special keys (CR ES etc.) so the text you require to send must have the correct decimal characters in the text. (i.e. CHAR(27) for ESCape and CHAR(13) for Carriage Return).

To tell a control or group of controls to return a different string to the host when a specific event occurs, call the following subroutine from your application:

**CALL PIX.GUI.ALT.EVENT (GUI.APP.NAME, Control ID, EVENT, TEXT,"")**

Where:

<b>GUI.APP.NAME</b>	Used to identify a given set of controls. Can be null.
<b>Control ID</b>	Control ID of the Control that the event is to be associated with.
<b><u>EVENT</u></b>	Event that alternative string is to be associated with.
<b>TEXT</b>	Text string to be returned to the Host instead of the Event.

Click here  for an example.

Alternatively, you can use the [PIX.GUI.CONTROL.LOAD](#) subroutine.



To load the alternative events of “/FI<CR>“ to the control FILE which is a TEXTBUTTON set the control details as follows.

```
CALL PIX.GUI.ALT.EVENT (“”, “FILE”, “CLICKED”, “/FI”: CHAR(13),””
```

or, in [PIX.GUI.CONTROL.LOAD](#) set

```
CONTROL.DETAILS<1> = “FILE, ALTEVENT, CLICKED,/FI”: CHAR(13)
```

```
CALL PIX.GUI.CONTROL.LOAD (GUI.APP.NAME, CONTROL.DETAILS,””)
```

This subroutine enables accelerator keys to be attached to controls. Accelerator keys are accessed by pressing the Alternate key and the letter you have assigned to be the accelerator key. This makes the screen access quicker to use as the user does not need to use the mouse or tab through to the selected field.

To load a Control with an accelerator key use the following subroutines.

**PIX.GUI.CONTROL.EVENT** is a basic subroutine that can be called from your application:

**CALL PIX.GUI.ACC.KEY (GUI.APP.NAME, Control ID, KEY,ERROR)**

Where:

<b>GUI.APP.NAME</b>	Used to identify a given set of controls. Can be null.
<b>Control ID</b>	Name of the control that the Accelerator key is to be assigned against
<b>KEY</b>	Key letter to be used in association with the ALT key.
<b>ERROR</b>	Returns 1 if error (for future use).

Alternatively, you can use the [PIX.GUI.CONTROL.LOAD](#) subroutine.

To associate the ALT+F keystroke to the FILE control use the Following.

```
CALL PIX.GUI.ACC.KEY (“”, ‘FILE ’, “F”)
```

When used with [PIX.GUI.CONTROL.LOAD](#), this is automatically done for you - you specify the caption with a leading ‘& ’ before the desired key. For example,

```
CONTROL.DETAILS<1> = ‘HELPBUTT, TEXTBUTTON, 10,19,9,2, &Help, HELV8 ’
```

```
CALL PIX.GUI.CONTROL.LOAD(GUI.APP.NAME,CONTROL.DETAILS,“”)
```

To specify a base control group, to which all subsequent controls will belong, call the following subroutine from your application:

**CALL PIX.GUI.LOAD.DEFGROUP(GUI.APP.NAME, CONTROL.DETAILS, STRING)**

Where:

**CONTROL.DETAILS**      Control ID, DEFGROUP

Where:

**Control ID**            The unique name for the control

**DEFGROUP**            The text literal DEFGROUP or DEFGRP

Normally you would put one of these controls at the beginning of all the control definitions. This is so you can globally access all of your controls, for example, so you can SHOW or HIDE them.

Click here  for an example.

To set all proceeding controls to the STAFF.DEMO group use the following:

```
CALL PIX.GUI.LOAD.DEFGROUP (GUI.APP.NAME, "STAFF.DEMO, DEFGROUP", "")
```

or, in [PIX.GUI.CONTROL.LOAD](#) set

```
CONTROL.DETAILS<1> = 'STAFF.DEMO,DEFGRP '
```

```
CALL PIX.GUI.CONTROL.LOAD(GUI.APP.NAME,CONTROL.DETAILS, "")
```

To read the contents of a control call the following subroutine from your application:

**CALL PIX.GUI.CONTROL.READ.VALUE (GUI.APP.NAME, CONTROL/TYPE, VALUE, ERROR)**

Where:

<b>GUI.APP.NAME</b>	Used to identify a given set of controls. Can be null.
<b>CONTROL/TYPE</b>	This is set to the control ID and the type of Control with a comma separating the 2 fields. e.g. "NAME, EDIT"
<b>VALUE</b>	This is the returning value of the contents of the control.
<b>ERROR</b>	Set to 1 if error has occurred (for future use).

To load the contents of a control, call the following subroutine from your application:

**CALL PIX.GUI.CONTROL.SET.VALUE (GUI.APP.NAME, CONTROL/TYPE, VALUE, ERROR)**

Where:

<b>GUI.APP.NAME</b>	Used to identify a given set of controls. Can be null.
<b>CONTROL/TYPE</b>	This is set to the control ID and the type of Control with a comma separating the 2 fields. e.g. "NAME, EDIT"
<b>VALUE</b>	This is the value of the contents of the control that wish to be loaded.
<b>ERROR</b>	Set to 1 if Error has occurred (for future use).

This subroutine will disable or 'gray out' a control. If the user is not allowed access to a control it can be disabled by calling this routine. Disabled controls will remain visible on the screen, but will not send any events to the Host.

To disable a control or a group of controls call the following subroutine from your application:

**CALL PIX.GUI.CONTROL.GRAYED (GUI.APP.NAME, Control ID, ERROR)**

This subroutine will enable a control that has previously been disabled. If the user is not allowed access to a control it can be disabled by calling PIX.GUI.CONTROL.GRAYED and enabled by calling this routine.

To enable a control or a group of Controls call the following subroutine from your application:

**CALL PIX.GUI.CONTROL.UNGRAYED (GUI.APP.NAME, Control ID, ERROR)**

This routine is used to remove controls that are no longer needed. Removing the controls will remove them from the screen and from memory. All controls should be unloaded when finished with or you may experience memory problems later on. You can unload single controls, groups of controls and even the DEFGROUP. (All controls)

To unload a control or a group of Controls call the following subroutine from your application:

**CALL PIX.GUI.CONTROL.UNLOAD (GUI.APP.NAME, Control ID, ERROR)**

Where:


**GUI.APP.NAME** Used to identify a given set of controls. Can be null.  
**Control ID** Name of the control or group of controls that you wish to disable.  
**ERROR** Set to 1 if Error (for future use).

To change the size and position of a control, call the following subroutine from your application:

**CALL PIX.GUI.REPOSITION (GUI.APP.NAME, X, Y, W, H, Control ID)**

where

**GUI.APP.NAME** Used to identify a given set of controls. Can be null.  
**X** The new X co-ordinate of the control.  
**Y** The new Y co-ordinate of the control.  
**W** The new width of the control.  
**H** The new height of the control.  
**Control ID** Control ID of the control to be repositioned.

Click here  for an example.



To move the control LOGO to a new position at 10,10 with width 20 and depth 3 use :

**CALL PIX.GUI.REPOSITION (GUI.APP.NAME, 10,10,20,3,“LOGO”)**

A control can be loaded into HOSTACCESS 's memory, but will have no interaction with the Host system unless told to do so by the means of **events**.

- Enabling Event Reporting.
- Event Keywords Defined.
- Enabling Timed Events.
- Capturing Events.

Before HOSTACCESS can inform you that the user has done anything to a control you need to associate an event against a control. For example you could load three text buttons into HOSTACCESS 's memory and the user can click the mouse on each button. However, until you load the events associated to these controls, no communication will be sent to the host telling it that these buttons have been clicked. To analyze the events returned from the Host use [PIX.GUI.GET.EVENT](#)

To enable event reporting for a control or group of controls call the following subroutine from your application:

**CALL PIX.GUI.CONTROL.EVENT (GUI.APP.NAME, CONTROL.DETAILS, OPTIONS)**

where Control Details is a dynamic array of events to be globally loaded.

```
CONTROL.DETAILS = Control ID, EVENT, ON/OFF, EVENTS1,...EVENTSn
```

Where:

<b>Control ID</b>	The unique name for the Control
<b>EVENT</b>	The text literal EVENT.
<b>ON/OFF</b>	'ON ' - Control is to return events. 'OFF ' - Control is not to return events.
<b>EVENTS... EVENTSn</b>	Events that are associated with the control.


**Note:** **OPTIONS** are not used at present and will need to be set to null.

Alternatively, you can use the [PIX.GUI.CONTROL.LOAD](#) subroutine.

Events can be set with the following keywords.

<b>ENTER</b>	ENTER pressed
<b>ESCAPE</b>	ESCape pressed
<b>CLICKED</b>	Button clicked
<b>BUTTONSTATE</b>	Check box or Radio button check state change.
<b>EDITCHANGE</b>	Contents of edit box, or the contents of the edit box part of a simple & drop down combo boxes, have been changed.
<b>LISTSELECT</b>	List box selection changed (List, simple & drop down combo)
<b>LISTDBCLICKED</b>	List box double clicked. (List, simple & drop down combo)
<b>FOCUS</b>	Gained focus - focus has changed from one control to another. See note below.
<b>TAB</b>	Control has been tabbed from.
<b>CLICKEDON</b>	Same as Clicked but Control that loses focus will be returned in Params (see <a href="#">PIX.GUI.GET.EVENT</a> )
<b>SECFOCUS</b>	Secondary Window active: the user is trying to change focus from a secondary window.
<b>SECCLOSE</b>	Secondary Window close: the user is trying to close a secondary window.
<b>LISTSCROLL</b>	The user has scrolled off the end of an Incremental List box.
<b>GRIDCELLCHANGE</b>	Grid cell change: a cell has changed in value.
<b>GRIDROWCHANGE</b>	Grid row change: a cell on a row has been changed.

**Note:** If a control has the FOCUS event assigned to it, no other events can be assigned against this control. This stops controls from sending double events. For example, If you tried to assign FOCUS and CLICKED against a button, you would get an event to say the button has gained focus, and an event to say the button was clicked.

Click here  for an example.

To enable the events against the Control FILE so the Host is informed if the user presses the Enter key if the control has focus or the user clicks the mouse over the control, use the following.

**CONTROL.DETAILS<1> = "FILE,EVENTS,ON,ENTER,CLICKED"**

**CALL PIX.GUI.CONTROL.EVENT(GUI.APP.NAME,CONTROL.DETAILS,"")**

You may wish HOSTACCESS to report back to your application ever so often. This is very useful if wish you application to time out after a given time or run a task after every n seconds.

To enable timed event reporting call the following subroutine from your application:

**CALL PIX.GUI.SET.TIMER(SECONDS,TYPE)Where:**

**SECONDS** Number of seconds before HOSTACCESS returns a timed event

**TYPE** Set to **ONCE** if you require HOSTACCESS to respond only once (after n seconds).

Set to **MANY** if you require HOSTACCESS to respond more than once (ever n seconds).

To make your application return the event TI every 60 seconds call the subroutine as follows :

**CALL PIX.GUI.SET.TIMER (60,"MANY")**

This subroutine is normally the central routine which collects the events from the controls. In the case of a fully GUI style program this routine replaces your INPUT statement. Typically this is used in a loop with CASE statements to action the returned event.

If you are experiencing spurious data being returned from this routine please contact Pixel Innovations for assistance.

To capture the events from HOSTACCESS's controls call the following subroutine from your application:

**CALL PIX.GUI.GET.EVENT (GUI.APP.NAME, Control ID, EVENT, PARAMS)**

where

<b>GUI.APP.NAME</b>	Used to identify a given set of controls. Can be null.
<b>Control ID</b>	Control ID of the Control that the event is to be associated with. If the EVENT that is reported is the TIMED event then this will be set to T1.
<b>EVENT</b>	<a href="#">Event</a> that is returned.
<b>PARAMS</b>	If the event is CLICKEDON or FOCUS, this is the Control ID of Control that loses Focus.  If Mouse is turned on, returns "MS" as the Event and the x & y co-ordinates of the mouse are in PARAMS. Use the <a href="#">PIX.MOUSE.ON</a> subroutine to turn the mouse on.  If the event is LISTDBCLICKED, this is the contents of the selected item from a list box.

**Note:** If the event is FOCUS, no other events should be specified against the controls. This is to stop double events from being returned.

Click here  for an example.

Below is an example of how you can load multiple events, capture the response and perform actions on the responses:

```
CD=""
*
* Controls
  CD<-1>="OK,TEXTBUTTON,1,1,10,2,&Ok"
  CD<-1>="CANCEL,TEXTBUTTON,13,1,10,2,&Cancel"
  CD<-1>="DATAFIELD,EDIT,3,5,10,1"
*
* Events
  CD<-1>="OK,EVENTS,ON,CLICKED,ENTER"
  CD<-1>="CANCEL,EVENTS,ON,CLICKED"
  CD<-1>="DATAFIELD,EVENTS,ON,FOCUS"
*
* Load the controls and events
  CALL PIX.GUI.CONTROL.LOAD("",CD,"","")
*
* Setup the timer to go at 60 second intervals
  CALL PIX.GUI.SET.TIMER(10,"MANY")
*
* Enable mouse events
  CALL PIX.MOUSE.ON("LEFT")
*
  DONE=0
  LOOP UNTIL DONE DO
*
* Wait until an event occurs
  CALL PIX.GUI.GET.EVENT("",CONTROLID,EVENT,PARAMS)
  BEGIN CASE
*
  CASE CONTROLID="OK"
* User did something with the "OK" button
    BEGIN CASE
      CASE EVENT="CLICKED" ;* Was it a CLICKED event
        PRINT "OK,CLICKED"
      CASE EVENT="ENTER" ;* Was it an ENTER event
        PRINT "OK,ENTER"
    END CASE
```



```

*      CASE CONTROLID="CANCEL"
          DONE=1 ;* Cancel was clicked so Exit the program
          PRINT "CANCEL CLICKED"
*
          CASE CONTROLID="DATAFIELD"
* My edit control has gained focus
          BECAUSE=FIELD(PARAMS,"",1)
* This is where it came from
          COMEFROM=FIELD(PARAMS,"",2)
          PRINT "DATAFIELD FROM ":COMEFROM:" BECAUSE OF ":BECAUSE
*
          CASE CONTROLID="TI"          ;* Timer event has occurred
          GOSUB 100                    ;* Process something
*
          CASE CONTROLID="MS"
* user has clicked with left mouse button
          xpos=FIELD(PARAMS,"",1) ;* Collect coordinates
          ypos=field(PARAMS,"",2)
          PRINT "MOUSE CLICKED AT ":xpos:",":ypos
*
          CASE 1
            END CASE
          REPEAT
            CALL PIX.GUI.CONTROL.UNLOAD("",CD,"")
            CALL PIX.MOUSE.OFF          ;* Turn off mouse events
            CALL PIX.GUI.SET.TIMER(0,"") ;* Turn off timer events
          STOP
100*
          PRINT "Timer event"
          RETURN

```

Secondary windows are one of the most powerful facilities available to the programmer within the GUI TOOLKIT. They can give your application the ability to display screens of data off the primary screen. These windows can be closed before returning to the primary screen or left open for the user to refer back to (Modal / Modeless).

These windows can be configured to be any size for example 10x10, 80x24 or 132x24 to display your data.

- Using Secondary Windows.
- Designing Secondary Windows.
- Opening a Secondary Window.
- Closing a Secondary Window.
- Giving a Secondary Window Focus.

Once open the user can reposition, re-size and even minimize the window to review the underlying screen.

The simplest example is from an order entry screen. The application would normally need to repaint the screen if the user wished to inquire on the customer details. Once the inquiry was made then the screen would need to re-paint the original screen. With Secondary windows the application can open a Modeless window to display the customer details and come back to the primary window leaving the details on the screen. The user can then refer to this window, move it around the desktop, re-size the window and even minimize the window. The user can then close it when required or the application can close it on exit from the order screen.

These windows can be addressed as you would the primary screen, you can display text as normal but you can also display all the Windows controls available to you within the developers toolkit.

When designing applications using secondary windows the programmer must remember that the host session is most likely not Multi tasking. Therefore if a window is left open, then its main purpose will be for inquiry and not for update.

You can design your application software to allow hot keys to update screens from any point to any secondary window. However, your application will need to know what windows are open, which record is currently being updated and even which field they are currently on.

The created window can be modal or modeless. The window settings (for example, Colours) are set to the startup settings.

To open a secondary window call the following subroutine from your application:

**CALL PIX.GUI.SEC.OPEN (GUI.APP.NAME, CONTROL.DETAILS, STRING)**

Where:

**CONTROL.DETAILS = Control ID, WINDOW, X, Y, W, H, LABEL, {Key1,...Keyn}**

Where:

<b>Control ID</b>	The unique name for the Control.
<b>WINDOW</b>	The text literal WINDOW.
<b>X</b>	The X co-ordinate of the window.
<b>Y</b>	The Y co-ordinate of the window.
<b>W</b>	The width of the window.
<b>H</b>	The height of the window.
<b>LABEL</b>	The title to be displayed in the secondary window 's title bar
<b>Key1...Keyn</b>	Additional parameters if required in any order (see below)

Additional keyword parameters available are:

MIN            MAX            MODAL            BORDERNONE  
BORDERNORM    BORDERTHICK    BORDERTHIN    COLOR  
NOTITLE

Alternatively, you can use the PIX.GUI.CONTROL.LOAD subroutine.

If you use this routine, it must be the first line of controls. Only one window can be opened from this routine.

If you open a **modal** window, the user can not have access to the previous or primary window until this window has been closed. If you open a **modeless** window, the user can gain focus to any other opened window. The host program must control each window in turn. So, to run multiple activities in each window, your host program must be able to control this.

To close a secondary window call the following subroutine from your application:

**CALL PIX.GUI.SEC.CLOSE (GUI.APP.NAME, Control ID, STRING)**

Alternatively, you can use the [PIX.GUI.CONTROL.LOAD](#) subroutine.

When opening a secondary Window it immediately gains focus. If your application has more than one Window open and you wish to set focus to another Window or the Base window then call the following subroutine from your application :

**CALL PIX.GUI.SEC.ACTIVE(GUI.APP.NAME,Control ID, STRING)**

**Note :** To gain focus to the Base/ Primary Window set the Control ID = “\_BASE”



This routine will display a Windows style Message box with the ability to enter options on exit. Its main use is for messages that require confirmation. The message box will be created as a modal style secondary window. (Cannot return to primary window until window cleared.) The response to the message box will be returned to the calling program.

To display a Message box call the following subroutine from your application:

**CALL PIX.GUI.MSG.BOX (TITTLE, MESSAGE, TYPE, ERROR)**

Where:

**TITLE** Message to appear in Windows Header bar.

**MESSAGE** Text message to appear in Message box

**TYPE** Button Style. The buttons set as default (i.e. those that respond when the ENTER key is pressed) are shown in bold. The following combinations are available:

OK , **CANCEL**

**OK** , CANCEL

YES , NO , **CANCEL**

**YES** , NO , CANCEL

YES , **NO** , CANCEL

**OK**

**YES** , NO

YES , **NO**

**ERROR** 1 = NO

2 = YES or OK

3 = CANCEL, escape pressed, or message box closed



Example.

```
CALL PIX.GUI.MSG.BOX("Error Message", "An Error Has occurred in the Application",  
"OK, CANCEL", ERROR)
```

If you design any useful utilities that you don't mind making accessible to others, please send them to Pixel and we can include them in future releases of the Host programs and documentation.

For example, here is a useful dialog box type selection screen that can be included into your applications if a selection is required.

This subroutine allows the user to select an item from a list. It is built up from an edit box , a list box and some user installable text push-buttons. The box is placed in a secondary window. It will destroy itself when finished and return the selected item and a name of the text push-button which the user selected, (if one was selected).

To use this utility, call the following subroutine from your application:

**CALL PIX.GUI.SELECTION(BUTTONS, TITLES,DEFAULT,LIST,SELECTION, CONTROL.ID)**

Where:

<b>BUTTONS</b>	A comma separated list of button names i.e. OK,CANCEL
<b>TITLES</b>	Comma separated text for window title, edit box text and list box text
<b>DEFAULT</b>	This contains one of the contents of the USER.LST. When the screen is displayed this item will be highlighted.
<b>LIST</b>	A list of items that the user may select from.
<b>SELECTION</b>	Is returned, the text of the item that the user choose
<b>CONTROL.ID</b>	Is returned, the name of the installed button the user clicked on or LISTDBCLICKED if the user double clicked the item. The user may also close the window by using Alt-F4. If this happens, the control.id will be ALTF4.

Below is an example of how to use this subroutine :

```
001 EQU TAB TO CHAR(9)
002 LIST="PETER, PAUL, JOHN, DAVID, HAZEL, JOHN MIKE, PAUL, PETER, SIMON, STEVE, UTAH, BILLY, ANDREW, STEVE"
003 TITLES="Choose a person, Se&lection, pick &from"
004 BUTTONS="&OK, Cancel, Help,, Delete,, Exit"
005 CONTROL.ID=""
006 CALL PIX.GUI.SELECTION(BUTTONS, TITLES, "PETER", LIST, SELECTION, CONTROL.ID)
007 PRINT "On return :":SELECTION, CONTROL.ID
eoi 007
```

The following table describes, in alphabetic order, all of the keywords you can use as optional parameters in PICK BASIC subroutines. Defaults are shown with an asterisk (\*) character.

<u>ABBRDATE</u>	HELV10	MODELESS	<u>RADIO GROUP</u>
BITMAP	HIDDENMSG	MORE	READONLY
<u>BORDER</u>	HIDE	NAMECOL	READWRITE *
BORDERNONE	HSCROLL	NAMEROW	RIGHT
<u>BORDERNORM</u>	<u>HSCROLLBAR</u>	NOCASECONVERT*	SCALED*
BORDERON	ICON	NOCOL	<u>SCULPTURE</u>
BORDEROFF	INITSELECT	NODELETE	SELECT*
BORDERTHICK	INTRANGE	NOROW	SHADOWED
BORDERTHIN	LABELED	NONE*	<u>SHORTDATE</u>
CENTER	LABLEPOS	NOSELECT	SHOW*
CHECKED	LEFT	NOSORT	SORT
CLIPPED	LINELN	NOTITLE	TABBED
COLOR	LINESOFF	NUMCOL	TERMITE*
DISPLAYMSG	LONGDATE	NUMROW	TOPBOX
FILEID	LOWER	PASSWORD	UNCHECKED
FRAME	MAX	<u>PUSH</u>	UNGRAYED*
GRAYED	MIN	PUSHEDIN	<u>UPPER</u>
HELV8	MODAL	<u>PUSHEDOUT</u>	VSCROLL
			VSCROLLBAR

**ABBRDATE**

Validation for a date entered in abbreviated format, for a validated edit box. For example, DD/MM/.

**BITMAP**

Followed by a .bmp file name, associated with LABELED (if specified.)



**BORDER**

Border type: follow with the type of border required. i.e. BORDER, followed by NONE, NORM, THIN, or THICK

**BORDERNONE**

No border for a secondary window

**BORDERNORM**

Create a normal border for a secondary window.

**BORDERON**

Control is surrounded by a box, extending 4 pixels above and below the normal box. You cannot have two bordered controls on consecutive lines

**BORDEROFF**

Control is not surrounded by a box

## **BORDERTHICK**

Create a thick border for a secondary window.

**BORDERTHIN**

Create a thin border for a secondary window.

**CENTER**

Text is centre-justified for labels within its boundary.



**CHECKED**

Control is checked when displayed. (RADIO, CHECK controls) Only one radio button within its group will be checked, even if all are set with checked set. (the last to be loaded)

**CLIPPED**

Clips bitmap to image co-ordinates (does not re-scale)

**COLOR**

Colour of a secondary window.

**DISPLAYMSG**

Display the normal text in the string list. This is returned by PIX.GUI.READ.VALUE.

**FILEID**

The following keywords will be the file ID (e.g. HOST.EXE) and the next keyword will be the resource number (e.g. 103)

**FRAME**

No border

**GRAYED**

Control is inactive when created

**HELV8**

Control uses the Helvetica 8 font to display the text



**HELV10**

Control uses the Helvetica 10 font to display the text

**HIDDENMSG**

Display the hidden text in the string list. This is returned by PIX.GUI.READ.VALUE

**HIDE**

Control is hidden when created

**HSCROLL**

Control has ability to scroll horizontally

**HSCROLLBAR**

Control has horizontal scroll bar attached

**ICON**

If ICON specified then the following keywords will be the Icon: .ico file, associated with LABELED if specified

**INITSELECT**

If specified, the following keyword will be the item in the string list to be highlighted when control gains focus. e.g. If a string called LETTER contained items 'A' to 'G' and 'C' was selected to be highlighted, when displayed the item 'C' would be highlighted. i.e. 'INITSELECT, C'

**INTRANGE**

When specified, gives an integer number range for validated edit boxes. Range is defined by the following two keywords lownum and highnum



**LABELED**

If labeled with a bitmap then text (label) appears with the bitmap

**TABLEPOS**

If TABLEPOS specified then the following keywords will be the (X,Y) position of the label. e.g. TABLEPOS, 10,10

**LEFT**

Text associated with Radio and Check controls is placed to the left of the control or Text will be Left justified for Labels within its boundary

**LINELEN**

If LINELEN specified then the following keyword will be the maximum number of characters allowed to be entered. This is the total limit, not just the limit on a single line. (Multi-line edit box user take note) Default is 64K characters.  
For example, LINELEN, 30 will set the maximum line length to 30

**LINESOFF**

Does not display Grid lines in Grid Controls

**LONGDATE**

validated date entered in long format, for a validated edit box. For example, 12 December 1994

## **LOWER**

Convert returning text to lower case

**MAX**

Creates a maximize box for a secondary window.



**MIN**

Creates a minimize box for a secondary window.

**MODAL**

The control is modal.

## **MODELESS**

The control is modeless

**MORE**

Sets how many elements the list box will hold, for incremental list boxes. Follow this by the number of elements. You can use this feature to create list boxes with room for many entries, and create corresponding string lists with only a few items. This allows you to update the list box incrementally as the user scrolls downwards

**NAMECOL**

Allows Columns to have named titles in Grid Controls

**NAMEROW**

Allows Rows to have named titles in Grid Controls

**NOCASECONVERT\***

Text returned from control is not case-converted

**NOCOL**

Specifies no titles for Columns in Grid Controls



**NODELETE**

Makes Grid control READONLY

**NOROW**

Specifies no titles for Rows in Grid Controls

**NONE**

No shadow

**NOSELECT**

Contents not selected when control receives focus

**NOSORT**

String list is not sorted when loaded into a list/combo box

**NOTITLE**

Set no title for a secondary window

**NUMCOL**

Gives Column titles as numbers in Grid controls

**NUMROW**

Gives Rows titles as numbers in Grid controls



**PASSWORD**

Text is encrypted to display password character. Default set to #

**PUSH**

Displays image with a button border. When image pushed by user, the border colors are swapped

**PUSHEDIN**

Image sunken.

**PUSHEDOUT**

Image raised.

**RADIO GROUP**

This is the Control ID of the Group for a set of Radio buttons. You will not need to specify this group using the GROUP control type if placed as a keyword

**READONLY**

Control is read-only, No changes can be made to the data

**READWRITE**

Control is Read/Write. Changes can be made to the data

**RIGHT**

Text associated with radio and check controls is placed to the right of the control (or text is right-justified for labels within its boundary).



**SCALED**

Scales bitmap to image co-ordinates

## **SCULPTURE**

Control is surrounded with sculpted line drawing.

**SELECT**

Contents selected when control receives focus

**SHADOWED**

Give image a shadow

**SHORTDATE**

Validation for a date entered in short format, for a validated edit box. For example, DD/MM/YY.

**SHOW**

Control is shown when created.

**SORT**

String list is alphabetically sorted when loaded into a list/combo box

**TABBED**

Allows you to present your list box in tabular columns. The String list loaded needs each column of data separated by a TAB. e.g. 'STRING, Andy<TAB>18, Francis<TAB>13 ' where <TAB> is char(9)



**TERMITE**

Control uses the TERMITE display font to display the text

**TOPBOX**

If specified, the following keyword will be the item in the string list to be placed at the top of the box. e.g. If a string called LETTERS contained items 'A' to 'Z' and 'F' was selected to be at the top of the box, when displayed the user could scroll back to 'A'. If not specified then 'A' would be at the top of the box. i.e. 'TOPBOX, F'.

**UNCHECKED**

Control is not checked when displayed. (RADIO, CHECK Controls) One Radio button within it 's group will have the Checked state even if all are set with NO Check set. (the last to be loaded).

**UNGRAYED**

Control is active when created.

**UPPER**

Convert returning text to upper case.

**VSCROLL**

Control has ability to scroll vertically.

**VSCROLLBAR**

Control has vertical scroll bar attached.

The following topics are example programs.

The first program (BOB) uses PIX.GUI.CONTROL.LOAD.

The second program (RBOB) calls the individual loading subroutines for each type of control.

The third program example (ORDER) is a fully-working program with events returned. All these programs are available for use in the Host programs. Please look at and amend these if needed.

- The PIX.B.O.B Program.
- PIX.GUI.FORM.F for PIX.B.O.B.
- The PIX.R.B.O.B Program.
- ORDER Program Example.
- ORDER.CONTROLS for ORDER Example.
- ORDER.SCREEN for ORDER Example.
- ORDER.EVENTS for ORDER Example.



```
001 REM *** Example Program to Use most of the Controls available.
002 REM *** This program calls PIX.GUI.CONTROL.LOAD
003 REM *** See eg PIX.R.B.O.B for same prog using subroutine method.
004 REM ***
005 REM *** Author : Andy Hedgecock 23 Sept 1994
006 REM ***
007 REM ***
008 REM *** Set DEFGROUP for all Controls that follow.
009 REM *** Note HOSTACCESS does not put Commands,Toolbars,Toolboxes &
010 REM *** Menus into Group. We hope to change this later.
011 REM ***
012 CALL PIX.SET.COLOR('BLACK,WHITE','D') ; * Set colors of screen.
013 PRINT @(-1):
014 REM ***
015 REM *** Open file that contains array of all the Controls.
016 REM ***
017 OPEN 'PIX.GUI.FORM.F' TO DBASE ELSE STOP
018 REM ***
019 REM *** Read in the Controls.
020 REM ***
021 READ REC FROM DBASE,'B.O.B' ELSE STOP
022 REM ***
023 REM *** Turn Sculpture ON
024 REM ***
025 CALL PIX.SCULPTURE.MODE('ON','C')
026 REM ***
027 REM *** Load the Controls
028 REM ***
029 CALL PIX.GUI.CONTROL.LOAD("",REC,"","")
030 PROMPT ""
031 INPUT XX:
032 REM ***
033 REM *** Turn Sculpture OFF and Unload all the Controls
034 REM ***
035 CALL PIX.GUI.CONTROL.UNLOAD("",REC,"")
036 CALL PIX.SCULPTURE.MODE('OFF',"")
037 END
EOI
```

```

005 REM *** Set DEFGROUP for all Controls that follow.
006 REM ***
007 MAINGROUP,DEFGROUP,DOG.CONTROLS
008 REM ***
009 REM *** Set Control Colors
010 REM
011 ALLCONTROLS,COLOR,BLACK,LIGHTGRAY
012 REM ***
013 REM *** Load Each Command Control
014 REM ***
015 COMM1,COMMAND,Command1,This is Command 1,\BITMAPS\ICON_BMP\HALT.BMP
016 COMM2,COMMAND,Command2,This is Command 2,\BITMAPS\ICON_BMP\TYPE.BMP
017 COMM3,COMMAND,Command3,This is Command 3,\BITMAPS\ICON_BMP\TOOLS.BMP
018 COMM4,COMMAND,Command4,This is Command 4,\BITMAPS\ICON_BMP\SHUTDOOR.BMP
019 COMM5,COMMAND,Command5,This is Command 5,\BITMAPS\ICON_BMP\SHREEK.BMP
020 COMM6,COMMAND,Command6,This is Command 6,\BITMAPS\ICON_BMP\QUESTION.BMP
021 COMM7,COMMAND,Command7,This is Command 7,\BITMAPS\ICON_BMP\POSTBOX.BMP
022 COMM8,COMMAND,Command8,This is Command 8,\BITMAPS\ICON_BMP\PHONE.BMP
023 COMM9,COMMAND,Command9,This is Command 9,\BITMAPS\ICON_BMP\PAPER.BMP
024 COMM10,COMMAND,Command10,This is Command 10,\BITMAPS\ICON_BMP\OPENDOOR.BMP
025 REM ***
026 REM *** Put Commands into a TOOLBOX, TOOLBAR & MENU
027 REM ***
028 FLOAT1,TOOLBOX,60,2,FLOATING TOOLBAR,COMM1,COMM2,COMM3,COMM4,COMM5,COMM6
029 BAR1,TOOLBAR,TITLE,COMM7,COMM8,COMM9,COMM10
030 MENU1,MENU,EXAMPLES,COMM1,COMM2,COMM3,COMM4,COMM5,COMM6,COMM7,COMM8,COMM9,COMM10
031 MENU,MENUINSTALL,MENU1
032 REM ***
033 REM *** Load Static Labels on Screen.
034 REM *** Note : The examples below use the Fixed font Helv10. For normal screens
035 REM ***           these look good but when the screen is reduced to a small size the
036 REM ***           font remains fixed and screen are corrupted.( BEWARE )
037 REM ***
038 REM ***
039 LAB1,LABEL,1,2,6,1,Name : ,Helv10,Right
040 LAB2,LABEL,1,4,6,1,Group : ,Helv10,Right
041 LAB3,LABEL,1,10,6,1,Breed : ,Helv10,Right
042 LAB4,LABEL,37,2,6,1,D.O.B : ,Helv10,Right
043 LAB5,LABEL,25,4,6,1,Class : ,Helv10,Right
044 REM ***
045 REM *** Load Textbutton Controls.
046 REM ***
047 FILE,TEXTBUTTON,65,2,9,2,&File,HELV10
048 CANCEL,TEXTBUTTON,65,5,9,2,Cancel
049 ADD,TEXTBUTTON,65,11,9,2,Add
050 REM ***
051 REM *** Load Pushbutton Controls.
052 REM ***
053 HELP,PUSHBUTTON,65,8,9,2,\host\bitmaps\flhelp.bmp
054 REM ***

```

```
055 REM *** Load Image Controls.
056 REM ***
057 REM LOGO,IMAGE,0,0,15,4,\bitmaps\easy\easyacc.bmp
058 REM ***
059 REM *** Load String Lists for use within Lists, Combos ect.
060 REM ***
061 GROUP.LIST,STRING,Working,Gundogs,Hounds,Utility,Toys,Terriers
062 BREEDS.LIST,STRING,Alaskan Malamutes,Bearded Collies,Belgian Shepherd Dogs,Boxers,Collies
(Rough),Collies (Smooth),Dobermann,G.S.D,Rottweiler,Swedish Valhunds
063 CLASS.LIST,STRING,1.Minor Puppy,2.Puppy,3.Novice,4.Junior,5.Graduate,6.Post
Graduate,7.Limit,8.Open,9.Veteran
064 REM ***
065 REM *** Set Control Colors
066 REM ***
067 ALLCONTROLS,COLOR,BLACK,WHITE
068 REM ***
069 REM *** Load Edit Controls.
070 REM ***
071 NAME,EDIT,9,2,26,1,Lezand,HSCROLL,SCULPTURE
072 D.O.B,EDIT,45,2,10,1,21 Apr 92,SCULPTURE
073 ADDRESS,EDIT,33,13,25,3,,HSCROLL,VSCROLL,VSCROLLBAR,SCULPTURE
074 REM
075 REM *** Load Dropdown Combo containing the list "GROUP.LIST"
076 REM
077 GROUP,DROPCOMB,9,4,15,5,GROUP.LIST
078 REM
079 REM *** Load Simple Combo containing the list "BREEDS.LIST"
080 REM ***
081 BREED,SIMCOMB,9,10,15,6,BREEDS.LIST,BORDERON
082 REM
083 REM *** Load a Tabbed LIST control containing the list "CLASS.LIST"
084 REM ***
085 CLASS,LIST,33,4,25,6,CLASS.LIST,TABBED
086 REM ***
087 REM *** Set Control Colors
088 REM ***
089 ALLCONTROLS,COLOR,BLACK,LIGHTGRAY
090 REM ***
091 REM *** Load Radio Button Controls
092 REM ***
093 DOG,RADIO,33,10,10,1,Dog,Helv10,LEFT,CHECK,SEX
094 BITCH,RADIO,33,11,10,1,Bitch,Helv10,LEFT,SEX
095 REM ***
096 REM *** Load Check Button Controls
097 REM ***
098 CAR,CHECK,47,10,10,1,Car Park,Helv10,LEFT
099 CAT,CHECK,47,11,10,1,Catalogue,Helv10,LEFT
100 REM ***
101 REM *** Set Control Colors
102 REM ***
103 ALLCONTROLS,COLOR,BLACK,WHITE
104 REM ***
```

```

105 REM *** Load a Grid Control with Nammed cols and numbered rows.
106 REM ***
107 GRID1,GRID,4,18,72,4,10,6,NAMECOL,NUMROW
108 REM ***
109 REM *** Set Col headings and Widths of Cols.
110 GRID1,GRIDPARAM,Name,D.O.B,Breed,Class, Sex,Amount;;;26,10,15,5,3,6
111 REM ***
112 REM *** Load Data into Grid
113 REM ***
114 GRID1,GRIDSETVALUE,0,0,Lezand,Lezand,Lezand;21 Apr 92,21 Apr 92,21 Apr 92;
    Dobermann,Dobermann,Dobermann;5,6,7;B,B,B;10.00,2.00,2.00
eoi:

```

```

001 REM *** Example Program to Use most of the Controls available.
002 REM *** This program calls each Subroutine rather than using PIX.GUI.CONTROL.LOAD
003 REM *** Refer to previous example to see the same program using PIX.GUI.CONTROL.LOAD
004 REM ***
005 REM *** Author : Andy Hedgecock 23 Sept 1994
006 REM ***
007 REM ***
008 REM *** Set DEFGROUP for all Controls that follow.
009 REM *** Note: At present HOSTACCESS does not put Commands, Toolbars, Toolboxes & Menus
010 REM ***      into the Group. This we hope to change in a later release.
011 REM ***
012 CALL PIX.GUI.LOAD.DEFGROUP("", "DOG.CONTROLS", "")
013 REM ***
014 CALL PIX.SET.COLOR('BLACK,WHITE','D')          ;* Set colors of screen.
015 PRINT @(-1):                                  ;* Clear Screen
016 REM ***
017 REM *** Set Sculpture Colors
018 REM ***
019 CALL PIX.SET.LINE.COLORS("BLACK,WHITE")
020 CALL PIX.SCULPTURE.MODE("ON", "")             ;* Turn Sculpture ON
021 REM
022 REM *** Set Control Colors
023 REM
024 CALL PIX.GUI.LOAD.COLOR("", "ALLCONTROLS,COLOR,BLACK,LIGHTGRAY", "")
025 REM ***
026 REM *** Load Each Command Control
027 REM ***
028 CALL PIX.GUI.LOAD.COMMAND("", "COMM1,COMMAND,Command1,This is Command 1, \BITMAPS\ICON_BMP\
    HALT.BMP", "")
029 CALL PIX.GUI.LOAD.COMMAND("", "COMM2,COMMAND,Command2,This is Command 2, \BITMAPS\ICON_BMP\
    TYPE.BMP", "")
030 CALL PIX.GUI.LOAD.COMMAND("", "COMM3,COMMAND,Command3,This is Command 3, \BITMAPS\ICON_BMP\
    TOOLS.BMP", "")

```

```

031 CALL PIX.GUI.LOAD.COMMAND("", "COMM4,COMMAND,Command4,This is Command 4, \BITMAPS\ICON_BMP\
SHUTDOOR.BMP", "")
032 CALL PIX.GUI.LOAD.COMMAND("", "COMM5,COMMAND,Command5,This is Command 5, \BITMAPS\ICON_BMP\
SHREEK.BMP", "")
033 CALL PIX.GUI.LOAD.COMMAND("", "COMM6,COMMAND,Command6,This is Command 6, \BITMAPS\ICON_BMP\
QUESTION.BMP", "")
034 CALL PIX.GUI.LOAD.COMMAND("", "COMM7,COMMAND,Command7,This is Command 7, \BITMAPS\ICON_BMP\
POSTBOX.BMP", "")
035 CALL PIX.GUI.LOAD.COMMAND("", "COMM8,COMMAND,Command8,This is Command 8, \BITMAPS\ICON_BMP\
PHONE.BMP", "")
036 CALL PIX.GUI.LOAD.COMMAND("", "COMM9,COMMAND,Command9,This is Command 9, \BITMAPS\ICON_BMP\
PAPER.BMP", "")
037 CALL PIX.GUI.LOAD.COMMAND("", "COMM10,COMMAND,Command10,This is Command 10, \BITMAPS\ICON_BMP\
OPENDOOR.BMP", "")
038 REM ***
039 REM *** Load Commands into a TOOLBOX, TOOLBAR & MENU
040 REM ***
041 CALL PIX.GUI.LOAD.TOOLBOX("", "FLOAT1,TOOLBOX,60,2,FLOATING TOOLBAR,COMM1,COMM2,
COMM3,COMM4,COMMS5,COMM6", "")
042 CALL PIX.GUI.LOAD.TOOLBAR("", "BAR1,TOOLBAR,TITLE,COMM7,COMM8,COMM9,COMM10", "")
043 CALL PIX.GUI.LOAD.MENU("", "MENU1,MENU,EXAMPLES,COMM1,COMM2,COMM3,COMM4,COMM5,
COMM6,COMM7,COMMS8,COMM9,COMM10", "")
044 CALL PIX.GUI.MENU.INSTALL("", "MENU,MENUINSTALL,MENU1", "")
045 REM ***
046 REM *** Load Static Labels on Screen.
047 REM *** Note : The examples below use the Fixed font Helv10. For normal screens
048 REM ***           these look good but when the screen is reduced to a small size the
049 REM ***           font remains fixed and screen are corrupted.( BEWARE )
050 REM ***
051 REM ***
052 CALL PIX.GUI.LOAD.LABEL("", "LAB1,LABEL,1,2,6,1,Name : ,Helv10,Right", "")
053 CALL PIX.GUI.LOAD.LABEL("", "LAB2,LABEL,1,4,6,1,Group : ,Helv10,Right", "")
054 CALL PIX.GUI.LOAD.LABEL("", "LAB3,LABEL,1,10,6,1,Breed : ,Helv10,Right", "")
055 CALL PIX.GUI.LOAD.LABEL("", "LAB4,LABEL,37,2,6,1,D.O.B : ,Helv10,Right", "")
056 CALL PIX.GUI.LOAD.LABEL("", "LAB5,LABEL,25,4,6,1,Class : ,Helv10,Right", "")
057 CALL PIX.GUI.LOAD.LABEL("", "LAB6,LABEL,25,13,6,1,Address : ,Helv10,Right", "")
058 REM ***
059 REM *** Load Textbutton Controls.
060 REM ***
061 CALL PIX.GUI.LOAD.TEXTBUTTON("", "FILE,TEXTBUTTON,65,2,9,2,&File,HELV10", "")
062 CALL PIX.GUI.LOAD.TEXTBUTTON("", "CANCEL,TEXTBUTTON,65,5,9,2,Cancel", "")
063 CALL PIX.GUI.LOAD.TEXTBUTTON("", "ADD,TEXTBUTTON,65,11,9,2,Add", "")
064 REM ***
065 REM *** Load Pushbutton Controls.
066 REM ***
067 CALL PIX.GUI.LOAD.PUSHBUTTON("", "HELP,PUSHBUTTON,65,8,9,2,\bitmaps\flhelp.bmp", "")
068 REM ***
069 REM *** Load Image Controls.

```

```
070 REM ***
071 REM LOGO, IMAGE, 0, 0, 15, 4, \HOST\bitmaps\easy\easyacc.bmp
072 REM ***
073 REM *** Load String Lists for use within Lists, Combos ect.
074 REM ***
075 CALL PIX.GUI.LOAD.STRING("", "GROUP.LIST, STRING, Working, Gundogs, Hounds, Utility, Toys, Terriers", "")
076 CALL PIX.GUI.LOAD.STRING("", "BREEDS.LIST, STRING, Alaskan Malamutes, Bearded Collies, Belgian Shepherd
Dogs, Boxers, Collies (Rough), Collies (Smooth), Dobermann, G.S.D, Rottweiler, Swedish Valhunds", "")
077 CALL PIX.GUI.LOAD.STRING("", "CLASS.LIST, STRING, 1.Minor
Puppy, 2.Puppy, 3.Novice, 4.Junior, 5.Graduate, 6.Post Graduate, 7.Limit, 8.Open, 9.Veteran", "")
078 REM ***
079 REM *** Set Control Colors
080 REM ***
081 CALL PIX.GUI.LOAD.COLOR("", "ALLCONTROLS, COLOR, BLACK, WHITE", "")
082 REM ***
083 REM *** Load Edit Controls.
084 REM ***
085 CALL PIX.GUI.LOAD.EDIT("", "NAME, EDIT, 9, 2, 26, 1, Lezand, HSCROLL, SCULPTURE", "")
086 CALL PIX.GUI.LOAD.EDIT("", "D.O.B, EDIT, 45, 2, 10, 1, 21 Apr 92, SCULPTURE", "")
087 CALL PIX.GUI.LOAD.EDIT("", "ADDRESS, EDIT, 33, 13, 25, 3, , HSCROLL, VSCROLL, VSCROLLBAR, SCULPTURE", "")
091 REM ***
092 REM *** Load Dropdown Combo containing the list "GROUP.LIST"
093 REM ***
094 CALL PIX.GUI.LOAD.DROPCOMB("", "GROUP, DROPCOMB, 9, 4, 15, 5, GROUP.LIST", "")
095 REM ***
096 REM *** Load Simple Combo containing the list "BREEDS.LIST"
097 REM ***
098 CALL PIX.GUI.LOAD.SIMCOMB("", "BREED, SIMCOMB, 9, 10, 15, 6, BREEDS.LIST, BORDERON", "")
099 REM ***
100 REM *** Load a Tabbed LIST control containing the list "CLASS.LIST"
101 REM ***
102 CALL PIX.GUI.LOAD.LIST("", "CLASS, LIST, 33, 4, 25, 6, CLASS.LIST, TABBED", "")
103 REM ***
104 REM *** Set Control Colors
105 REM ***
106 CALL PIX.GUI.LOAD.COLOR("", "ALLCONTROLS, COLOR, BLACK, LIGHTGRAY", "")
107 REM ***
108 REM *** Load Radio Button Controls
109 REM ***
110 CALL PIX.GUI.LOAD.RADIOBUTTON("", "DOG, RADIO, 33, 10, 10, 1, Dog, Helv10, LEFT, CHECK, SEX", "")
111 CALL PIX.GUI.LOAD.RADIOBUTTON("", "BITCH, RADIO, 33, 11, 10, 1, Bitch, Helv10, LEFT, SEX", "")
112 REM ***
113 REM *** Load Check Button Controls
114 REM ***
```

```

115 CALL PIX.GUI.LOAD.CHECKBUTTON("", "CAR,CHECK,47,10,10,1,Car Park,Helv10,LEFT","")
116 CALL PIX.GUI.LOAD.CHECKBUTTON("", "CAT,CHECK,47,11,10,1,Catalogue,Helv10,LEFT","")
117 REM ***
118 REM *** Set Control Colors
119 REM ***
120 CALL PIX.GUI.LOAD.COLOR("", "ALLCONTROLS,COLOR,BLACK,WHITE","")
121 REM ***
122 REM *** Load a Grid Control with Nammed cols and numbered rows.
123 REM ***
124 CALL PIX.GUI.LOAD.GRID("", "GRID1,GRID,4,18,72,4,10,6,NAMECOL,NUMROW","")
125 REM ***
126 REM *** Set Col headings and Widths of Cols.
127 PARAMS = ""
128 PARAMS<1> = "Name,D.O.B,Breed,Class,Sex,Amount"
129 PARAMS<4> = "26,10,15,5,3,6"
130 CALL PIX.GUI.GRID.SET.PARAM("", "GRID1", PARAMS)
131 REM ***
132 REM *** Load Data into Grid
133 REM ***
134 DATA = ""
135 DATA<1> = "Lezand,Lezand,Lezand"
136 DATA<2> = "21 Apr 92,21 Apr 92,21 Apr 92"
137 DATA<3> = "Dobermann,Dobermann,Dobermann"
138 DATA<4> = "5,6,7"
139 DATA<5> = "B,B,B"
140 DATA<6> = "10.00,2.00,2.00"
141 CALL PIX.GUI.GRID.SET.VALUE("", "GRID1", 0, 0, DATA)
142 REM ***
143 REM *** Wait for a keystroke
144 REM ***
145 PROMPT ""
146 INPUT XX:
147 REM ***
148 REM *** Unload all controls & clear Sculpture.
149 REM *** Note : Command, Toolbox, Toolbar & Menus can not be unloaded via group
150 REM ***       This we hope to change in a later release.
151 REM ***
152 CALL PIX.GUI.CONTROL.UNLOAD("", "COMM1","")
153 CALL PIX.GUI.CONTROL.UNLOAD("", "COMM2","")
154 CALL PIX.GUI.CONTROL.UNLOAD("", "COMM3","")
155 CALL PIX.GUI.CONTROL.UNLOAD("", "COMM4","")
156 CALL PIX.GUI.CONTROL.UNLOAD("", "COMM5","")
157 CALL PIX.GUI.CONTROL.UNLOAD("", "COMM6","")
158 CALL PIX.GUI.CONTROL.UNLOAD("", "COMM7","")

```

```
159 CALL PIX.GUI.CONTROL.UNLOAD("", "COMM8", "")
160 CALL PIX.GUI.CONTROL.UNLOAD("", "COMM9", "")
161 CALL PIX.GUI.CONTROL.UNLOAD("", "COMM10", "")
162 CALL PIX.GUI.CONTROL.UNLOAD("", "FLOAT1", "")
163 CALL PIX.GUI.CONTROL.UNLOAD("", "BAR1", "")
164 CALL PIX.GUI.CONTROL.UNLOAD("", "MENU1", "")
165 CALL PIX.GUI.CONTROL.UNLOAD("", "DOG.CONTROLS", "")
166 CALL PIX.SCULPTURE.MODE("OFF", "")
167 REM ***
168 REM *** End of Program
169 REM ***
170 END
eoi
```



```

001 PROMPT ""
002 GUI.APP.NAME = "ORDER"
003 GOSUB 1000;* and get control and screen formats
004 EQU TRUE TO 1, FALSE TO 0, OTHERWISE TO 1
005 CALL PIX.PUSH.SLOT ; CALL PIX.CURSOR("", "OFF")
006 CALL PIX.SCULPTURE.MODE("ON", "") ; CALL PIX.SET.COLOUR("BLACK,GREY", "D")
007 PRINT @(-1):
008 CALL PIX.GUI.LOAD.SCREEN(SCREEN, "")
009 CALL PIX.GUI.CONTROL.LOAD(GUI.APP.NAME, CONTROLS, "", "")
001 CALL PIX.GUI.CONTROL.EVENT(GUI.APP.NAME, CONTROL.EVENTS, "")
011 *
012 FIELD.VALUES = "";* This transaction
013 FINISHED=FALSE
014 CALL PIX.GUI.CONTROL.SET.FOCUS(GUI.APP.NAME, "USERS", "")
015 *
016 LOOP UNTIL FINISHED DO
017 *
018 CALL PIX.GUI.GET.EVENT(GUI.APP.NAME, CONTROL.ID, EVENT.TYPE, PARAMS)
019 CALL PIX.GUI.CONTROL.UNLOAD(GUI.APP.NAME, "MOUSE", "")
020 *
021 BEGIN CASE
022 CASE CONTROL.ID="SAVE"
023 FINISHED=1 ; GOSUB 100
024 CASE CONTROL.ID="CANCEL" OR PARAMS="CANCEL"
025 FINISHED=1
026 CASE INDEX("VISA,AMEX,DINERS,MCARD", CONTROL.ID, 1)
027 PRINT @(54,7):CONTROL.ID "L#8": ; FIELD.VALUES<12>=CONTROL.ID
028 CASE CONTROL.ID = "EXCEL"
029 CALL PIX.GUI.CONTROL.READ.VALUE(GUI.APP.NAME, "USERS, EDIT", VALUE, "")
030 IF NUM(VALUE) AND VALUE # "" THEN
031 PRINT @(0,0):
032 CALL EXCEL(VALUE)
033 END
034 CASE CONTROL.ID = "ROOT"
035 IF EVENT.TYPE = "Q" THEN
036 FINISHED=1
037 END
038 CASE OTHERWISE
039 GOSUB 100;* validate fields if necessary
040 END CASE
041 *
042 REPEAT
043 *

```

```
044 CALL PIX.GUI.CONTROL.UNLOAD(GUI.APP.NAME,"ORDER","")
045 CALL PIX.SCULPTURE.MODE("OFF","C")
046 CALL PIX.POP.SLOT ; CALL PIX.CURSOR("LINE","ON")
047 *
048 IF CONTROL.ID = "SAVE" THEN
049 RECORD = FIELD.VALUES      ;* more processing here
050 OPEN "ORDERS" TO ORDERS ELSE STOP
051 WRITE RECORD ON ORDERS,DATE():TIME()
052 END
053 *
054 STOP
055 *
056 1000* Build Control and Screen variables
057 *
058 OPEN "PIX.GUI.FORM.F" TO DBASE.FILE ELSE STOP
059 *
060 READ CONTROLS FROM DBASE.FILE,"ORDER.CONTROLS" ELSE STOP
061 READ SCREEN FROM DBASE.FILE,"ORDER.SCREEN" ELSE STOP
062 READ CONTROL.EVENTS FROM DBASE.FILE,"ORDER.EVENTS" ELSE STOP
063 RETURN                      ;* from 1000
064 *
065 100* Validate
066 *
067 *
068 * Call ORDER.VALIDATE to validate 1 or many (if finished)
069 CALL ORDER.VALIDATE(GUI.APP.NAME,CONTROL.ID,FINISHED,FIELD.VALUES)
070 *
071 RETURN                      ;* from 100
072 *
073 END
```

ORDER.CONTROLS  
ORDER, DEFGRP  
ALLCONTROLS, COLOUR, BLACK, LIGHTGREY  
CANCEL, TEXTBUTTON, 60, 21, 9, 2, &Cancel  
SAVE, TEXTBUTTON, 40, 21, 9, 2, &Save  
F1HELP, TEXTBUTTON, 50, 21, 9, 2, &Help  
SIZE3, RADIO, 10, 4, 9, 1, 3 1/2, RIGHT, CHECKED, SIZE  
SIZE5, RADIO, 20, 4, 9, 1, 5 1/4, RIGHT, SIZE  
ASYNC, RADIO, 10, 5, 9, 1, ASYNC, RIGHT, TYPE  
NET, RADIO, 20, 5, 9, 1, NET, RIGHT, CHECKED, TYPE  
LOCAL, RADIO, 10, 7, 9, 1, Local, RIGHT, CHECKED, INSTALL  
SERVER, RADIO, 20, 7, 9, 1, Server, RIGHT, INSTALL  
WIN, RADIO, 10, 8, 9, 1, Windows, RIGHT, CHECKED, WINDOWS  
DOS, RADIO, 20, 8, 9, 1, DOS, RIGHT, WINDOWS  
VISA, PUSHBUTTON, 45, 4, 11, 3, BITMAPS\ORDER\VISA.BMP  
AMEX, PUSHBUTTON, 57, 4, 11, 3, BITMAPS\ORDER\AMEX256.BMP  
MCARD, PUSHBUTTON, 45, 8, 11, 3, BITMAPS\ORDER\MASTERC.D.BMP  
DINERS, PUSHBUTTON, 57, 8, 11, 3, BITMAPS\ORDER\DINERS.BMP  
EXCEL, PUSHBUTTON, 21, 20, 8, 3, BITMAPS\ORDER\MSEXCEL.BMP  
USERS, EDIT, 20, 10, 5, 1  
COST, EDIT, 20, 12, 10, 1, 0.00  
CARDNO, EDIT, 20, 14, 15, 1, 4929 933  
CARDNAME, EDIT, 20, 16, 15, 1  
CARDEXPIRY, EDIT, 20, 18, 8, 1  
ADSTR, STRING, Spectrum, Pick World, News & Review, Dec User,  
IBM today, Financial Times, PC World, Windows Magazine, Other User  
ADVERT, DROPDOWN, 50, 12, 18, 6, ADSTR  
ADVERT, COLOUR, BLACK, LIGHTWHITE  
CARDADDR, EDIT, 40, 15, 30, 5, , VSCROLL, HSCROLL, VSCROLLBAR, MULTILINE  
TITLE, LABEL, 13, 0, 50, 2, HOSTACCESS Telesales Order-II Processing,  
CENTER, Times New Roman  
TITLE, COLOUR, LIGHTRED, LIGHTGREY  
LAB1, LABEL, 8, 10, 10, 1, # of Users, Times New Roman Bold Italic  
LAB2, LABEL, 8, 12, 10, 1, Total Cost, Times New Roman Bold Italic  
LAB3, LABEL, 6, 14, 11, 1, Card Number, Times New Roman Bold Italic  
LAB4, LABEL, 5, 16, 12, 1, Name on Card, Times New Roman Bold Italic  
LAB5, LABEL, 5, 18, 11, 1, Expiry Date, Times New Roman Bold Italic  
LAB6, LABEL, 40, 12, 9, 1, Sales Ref, Times New Roman Bold Italic  
LAB7, LABEL, 40, 14, 20, 1, Delivery/Card Address, Times New Roman  
Bold Italic

ORDER.SCREEN  
COLOUR, BLACK, GREY  
SCOLOUR, BLACK, WHITE  
SBOX, 5, 3, 70, 20  
SLINE, 37, 3, 20, V, GREY  
SBOX, 45, 4, 23, 7  
SBOX, 10, 4, 20, 5  
SBOX, 13, 0, 50, 2, RAISED  
SBOX, 40, 15, 30, 5  
SBOX, 20, 10, 5, 1  
SBOX, 20, 12, 10, 1  
SBOX, 20, 14, 15, 1  
SBOX, 20, 16, 15, 1  
SBOX, 20, 18, 8, 1

ORDER.EVENTS

CANCEL, ON, CLICKED

SAVE, ON, CLICKED

AMEX, ON, CLICKED

DINERS, ON, CLICKED

VISA, ON, CLICKED

MCARD, ON, CLICKED

USERS, ON, ENTER, TAB

EXCEL, ON, CLICKED

CARDEXPIRY, ON, ENTER, TAB

After you have installed the HOSTACCESS Host Programs on the host, several control and program files are created. This section details each file, its usage and where necessary any record structures.

All HOSTACCESS 's files can be found in the Master Dictionary or VOC of the account in which you have installed the HOSTACCESS Host Programs. These files have a naming convention to allow you to identify a HOSTACCESS specific file easily. This convention is as follows:

**PIX.filename.F**

Where **PIX.** is the prefix and **.F** is the suffix. The HOSTACCESS specific files are summarized below :

<b>File Name</b>	<b>Description</b>
<a href="#"><u>PIX.CONTROL.F</u></a>	HOSTACCESS control file
<a href="#"><u>PIX.EASY.ACCESS.F</u></a>	Used by the EASY.ACCESS application
<a href="#"><u>PIX.DB.F</u></a>	Used by the DB application
<a href="#"><u>PIX.PROGS.F</u></a>	Main HOSTACCESS programs file
<a href="#"><u>PIX.TEXT.F</u></a>	Text used by each program
<a href="#"><u>PIX.TRANSFER.F</u></a>	Used by HOSTACCESS 's File Transfer
<a href="#"><u>PIX.OUTPUT.nn.F</u></a>	Used by the GET application where 'nn' is the port number for any port that has used GET.
PIX.DEMO.F	This is a demonstration file only. It is created by the PIX.MAKE.DEMO.FILE program and contains dummy data. If you review the TERMITE.DEMO facility, you will notice that the EASY.ACCESS demonstration uses this file.
PIX.GUI.FORM.F	Used by the GUI demonstration programs to load the controls, screens and events.
PIX.ORDERS.F	Used by the ORDER demonstration program.
<a href="#"><u>PIX.CONTACT.F</u></a>	Used by the CONTACT and CONTACT.GUI programs.
PIX.STAFF.F and PIX.STAFF.DEPT.F	Used by the STAFF.DEMO demonstration program.

You will also find 'Q' pointers to the same file names within any Account which you have updated with HOSTACCESS Update Account procedure. These 'Q' pointers normally point back to the main HOSTACCESS host programs installation account. [PIX.OUTPUT.nn.F](#) is always created locally within each account and should not be "Q" pointed to. The HOSTACCESS control files are more fully detailed in the following sections.

This file contains many control records used by HOSTACCESS for a multitude of purposes. These records are described in their functional groups. Control records that may be amended to suit your operating environment are also detailed here. Please make sure that you keep copies of the original records before editing them.

- Control Records for DB and EASY.ACCESS.
- Control Records for the PASS.TO Routines.
- Control Records for DOS.PICK and PICK.DOS Transfers
- Control Records for Host Programs installation.
- Control Record for Host Pick programs.

The following records are used by the HOSTACCESS applications DB and EASY.ACCESS respectively:

DB.TEXT.

EASY.ACCESS.TEXT.

DB.COMMANDS.

EASY.ACCESS.COMMANDS.

EASY.ACCESS.COMMANDS.HELP.

The `.TEXT` records contain attributes of multi-values of text presented to the user as menu options where each attribute is a separate menu. This text may be customized to meet user requirements. The `.COMMAND` records are also attribute/multi-valued and match the corresponding attributes/multi-values in the `.TEXT` records. The `.HELP` records follow the same attribute/multi-value associations as the `.COMMAND` records and are used to provide on-line help within the EASY.ACCESS application. (note: this help is usually only available on those PICK systems, such as Prime INFORMATION, that support an on-line HELP facility.)

You must ensure that you maintain the attribute/multi-value relationships when changing and moving options.

The basic rule is that you can delete, change and move most options within the same attribute (menu) but you cannot move options from one attribute to another.

In most cases, you are unlikely to change the `.COMMAND` records too much, other than deleting unrequired options or changing verbs. Any other control COMMAND strings should NOT be changed in here since they are used by the programs to determine what option you have selected.

You will also find more copies of each of the above records with the extensions `.UNIVERSE` and `.PRIME`. These are the records that are used when you are running on those systems. They have been provided to take into account additional commands and verbs on those systems.



PASS.TO.SUPERCALC	PASS.TO.SYMPHONY
PASS.TO.LOTUS	PASS.TO.LOTUS.WINDOWS
PASS.TO.WORDPERFECT	PASS.TO.WORDPEFECT.WINDOWS
PASS.TO.QUATTRO	PASS.TO.QUATTRO.WINDOWS
PASS.TO.DOS	PASS.TO.WORD
PASS.TO.EXCEL	

#### Attributes 1 and 2

The first two attributes of these records are used by EASY.ACCESS when selecting to transfer data to a DOS or Windows package from the 'Pass to' option. Attribute 1 is the text of the options available when calling the package. Attribute 2 is a list of the actual command line option corresponding to the text. E.g. PASS.TO.LOTUS record attribute 1 contains the text option 'Stacked' Graph. Attribute 2 contains a corresponding 'S' which is passed as an option when PASS.TO.LOTUS is called from EASY.ACCESS. The 'text' may be customized to suit user requirements.

**Attribute 3** is more important. By default, it contains a single DOS or Windows command that will be used when calling the corresponding DOS or Windows package. E.g. PASS.TO.LOTUS attribute 3 has '123' as the DOS command to call LOTUS. If all of the PC's running HOSTACCESS can run a DOS command 123 from any directory then this attribute need never be changed.

However, if some or all of your PCs call LOTUS differently, this attribute can be used to identify which DOS command to use on a port by port basis. This is done by storing in the multi-value position (port number + 2), the relevant DOS command to call LOTUS on that port's PC. For example :

If attribute 3 contains :

```
123]]MY123]]CD\LOTUS;123
```

Port 1 will run MY123 to call LOTUS. Port 3 will change to directory LOTUS and run 123. All ports with a null multi-value (0, 2, 4 and above) will call the default DOS command 123 held in the first multi-value.

**Note:** for network/LAN users, we recommend you set your PCs to all run the same DOS or Windows command to call the relevant DOS package, as the port number may change each time you log in.

#### Attribute 4

This is used for PASS.TO.LOTUS and PASS.TO.WORDPERFECT. If you use multiple versions of these products, then specify the version number (3 or 4 for LOTUS, or 5 or 6 for WORDPERFECT) in this attribute.

**nn.FT.LOG**

This record contains the details of the last file transfer for port 'nn'. This is mainly supported for backwards compatibility with HOSTACCESS 2.0.

With HOSTACCESS 3.0, the file transfer programs can be called as subroutines and the details of the transfer are returned in a separate variable.


**PROGS.TO.COMPILE**

This record contains a list of programs that would not compile during the Host Programs Installation process. Pixel would like to hear from anyone who finds that this record is NOT empty. We have tested these programs on a massive range of PICK machines and we want to ensure that we support all of our shipped programs on all platforms and PICK flavors.


## ENVIRONMENT

This record is the main HOSTACCESS control record and has the following structure:

<b>Attribute</b>	<b>Function</b>	
001	Machine Type.	(do not change)
002	HOSTACCESS account	(do not change)
003	Reformat Type	(do not change)
004	Item Size Limit.	
005	System support for CHAR(255) (0 or 1).	

Click here  for attribute descriptions,

Attributes 1,2 and 3 are used by HOSTACCESS programs and should not be modified. If the machine type in attribute 1 does not match that for your system, you should re-install the HOSTACCESS host programs for your machine type into a clean account.

Attribute 4 is used mainly by HOSTACCESS 's file transfer upload programs. If the size of the DOS file being uploaded exceeds the Item Size Limit held here, it is split into records of that size accordingly. Click here  for information on the format of these split records. This value is normally set to 30000 bytes by default. Systems that are known to support unlimited item size will have this value set to 5 megabytes. This value affects all users on the system.

Attribute 5, is used by the file transfer upload programs. If set to 1 (true), this tells HOSTACCESS that this system can print and store the character value 255. Most generic PICK systems cannot support this character and therefore if it is found during any file transfer upload it will be converted to a 'tilde '. Without character 255 support, many DOS graphics files cannot be printed via the DOS.VS virtual spooler routine. If you need to store files that contain a character 255 then you should look at both the 'X ' and 'T ' options on DOS.PICK and PICK.DOS file transfer routines.

If you are interested in storing DOS files that either contain a character 255 and/or your DOS files are too big for your PICK host then you should see the 'T ' option associated with [PIX.DOS.PICK](#) and [PIX.PICK.DOS](#).

**Note:** If you wish to change attribute 5, ensure that your PICK system supports char(255).

**PIX.MENU.NAMES**

**PIX.SELECTION.BOX.NAMES**

These records are used internally by the host "PIX." selection box and menu subroutines. These contain a multi-value list of a names corresponding to a menu or selection box number to use for that name. These should not be changed.

This file is used by the EASY.ACCESS application and the PIX.EASY.ACCESS subroutine to store saved sentences. Normally this file is local to the main HOSTACCESS account and any other accounts will point to it via a Q Pointer. However, if you prefer to store saved sentences on an account by account basis, the Q Pointer can be deleted and a new file of the same name created in the relevant account.

This file is used by the DB application and the PIX.DB subroutine to store the current DB environment for each port.



This file is the main HOSTACCESS programs and subroutines file. Most programs in this file are subroutines and SHOULD NOT be run either directly from TCL or via the RUN command. Each of these programs is CATALOGed and should be called directly as a PICK/BASIC subroutine CALL with the appropriate parameters.

For each program and subroutine that HOSTACCESS uses, there is normally a corresponding record in this file. Most of the important user text has been removed from each program and stored it in these records. This allows the text for the programs to be customized to suit the user requirements. All of the text in this file may be modified or translated into other languages, as required. Since the DOS part of HOSTACCESS also has it 's text stored in a separate DOS file and can be modified to suit other languages, HOSTACCESS really is an international product.

This file is used by the File Transfer commands [DOS.PICK](#) and [PICK.DOS](#) and by the subroutines [PIX.CALL.DOS.PICK](#), [PIX.DOS.PICK](#), [PIX.CALL.PICK.DOS](#) and [PIX.PICK.DOS](#):

1. To store DOS files that have been uploaded but exceed the 'Item Size Limit' and therefore cannot be stored in one PICK record. This limit is held in the ENVIRONMENT record in the [PIX.CONTROL.E](#) file. Each large DOS file will be split into smaller records with each record id in the format 'FT.SPLIT,portno.n' where portno is the port number that initiated the transfer. The 'n' will start at 1 and will be incremented each time the data is SPLIT. The application developer or user must extract the data after it has been stored here. This is only a temporary storage area since it will be overwritten the next time a large DOS file is uploaded by this port.
2. When the 'T' option is used to upload DOS records using [PIX.DOS.PICK](#), those records are stored in HOSTACCESS's file transfer 'TRANSMIT' format in this file. This overcomes any item size limitation or character 255 restriction for storing DOS records on PICK. It also speeds up transfers, as HOSTACCESS does not need to unpack/repack the protocol packets. Using the 'T' option on [PIX.PICK.DOS](#) to send the record back results in the DOS file being created exactly as it was when uploaded. This may include any DOS file from .exe to spreadsheets.

Click here  for an example.

An example format of a transmit record would be as follows:

```
itemname.1  
itemname.2  
itemname.3
```

and

```
itemname.HDR
```

where **itemname** is the name of the record passed to PIX.DOS.PICK at upload time. And where itemname.n denotes a split, if the item is too big for the PICK system to store as one record. For each itemname uploaded there will be an associated .HDR record with 4 attributes as follows:

```
001 Total bytes of the actual DOS file  
002 Number of times itemname has been split  
003 Time and date stamp when created  
004 Port number who created the record
```

The actual transmit format data records are virtually in non user readable form. That is, they are the format of the protocol blocks that HOSTACCESS 's file transfer uses. In this way, the blocks can be sent back very quickly without the overhead of rebuilding them. These records should not be modified in any way, but can be deleted when and if they are no longer required.

You do not need to know how the format of these records are made up - when you specify to transfer 'itemname ' back to DOS using [PIX.PICK.DOS](#) and the 'T ' option, HOSTACCESS brings together all of the split records and create the single correct DOS record for you.

If the DOS record uploaded using the 'T ' option is smaller than the maximum item size limit, then there will only be 2 records created ; 'itemname ' (no extension) and 'itemname.HDR '.

This file is used by the [GET](#) command and the [PIX.GET](#) subroutine. There could be one file for each port number 'nn' that has ever run GET. This file may be deleted if it is no longer required by that port. Most users will have no use for the detailed format of this file. Any developers requiring program access to this file should contact their HOSTACCESS dealer or Pixel Innovations directly for details of the file format.

The format of the output file is self-explanatory, except for the header record. This record is structured as follows:

<b>Header</b>	<b>Meaning</b>
<1>	Column Width
<2>	No. of output dictionary
<3>	No. of items listed
<4>	Split into <b>N</b> chunks
<5>	No. of display lines.
<6>	No. of header lines
<7>	Justification of each field
<8>	Header of each field
<9>	1 = Date for each field 0 = No date.

