TI Part #2243724-0001

NaturalLink(TM) Technology Package 2.0
Window Manager Update and Release Information

This document is divided into two sections. The first section contains update information for users of version 1.75 of the NaturalLink Window Manager software. This section briefly describes the enhancements added to version 2.0 and any changes you must make before using this version.

The second section contains release information and information about version 2.0 of Window Manager and its utilities not found in the manual.

This Document: 2243685-0001

SECTION 1

Update Information

1.1 Compatibility Between Versions 2.0 and 1.75

Applications written for version 1.75 of Window Manager are compatible with version 2.0. No source code needs to be changed. However, you MUST recompile your source with the new include files for the language you are using, because the attribute constant values have changed. Once your source is recompiled, just relink your program using the new link stream instructions found in the appendix for the language you are using.

1.2 Multiple Machine Support

Version 2.0 of Window Manager has been tested and verified to run on the following computers.

Texas Instruments	IBM(R)
BUSINESS-PRO(TM) Computer (both modes)	IBM PC
Professional Computer	IBM PC/XT(TM)
Portable Professional Computer	IBM Personal Computer AT(TM)
PRO-LITE(TM) Computer	

Both the utilities and the runtime object code run on all of the mentioned machines except the PRO-LITE without any special modifications. Only the runtime runs on the PRO-LITE. An application developed and linked on the TI machines will run on the IBM machines without re-linking. The NaturalLink code detects which machine it is being run on and automatically makes changes specific to the machine.

1.3 Full MS(TM)-DOS 2.0/3.0 Support

Window Manager now completely supports the MS-DOS 2.0/3.0 operating system. Both the runtime and the utilities accept full DOS 2.0 pathnames. Version 2.0 also permits use of the DOS environment variable feature. You can set the environment variable NLXTOOLS to the directory where all the support files (.PIC, .NM$, and .NS$ files) for the Window Manager utilities reside. Using NLXTOOLS and the DOS PATH variable, you can run the utilities from any drive and directory on your system.

1.4 Compiled BASIC Support

Version 2.0 of Window Manager now provides high-level language support for compiled BASIC. All Window Manager function calls are supported, enabling you to write Window Manager applications in MS-BASIC, compile them using the MS-BASIC compiler, and then link and run them. For more information, see Appendix G, Compiled BASIC Interface, of the Window Manager manual.

1.5 Edit Field Validation Capabilities

A new feature has been added to Window Manager edit windows. Now, after an edit field has been completed, Window Manager calls a validation routine. This validation routine, which the developer must provide, checks the information entered by the user, issues any necessary messages, and returns to Window Manager with the status of the check. Window Manager then continues on the basis of the status. See Chapter 7, Application Validation Routine, for more details.

MS is a trademark of Microsoft Corporation.

1.6 User-Defined Windows and Messages

For special windowing needs not provided for by Window Manager, you can now define and control your own windows. User-defined windows (UDWs) provide a means of displaying data to the user and receiving data from the user in ways that Window Manager does not. Examples are the display of graphics pictures and selection of graphics text, icons, and items from a file. Window Manager will call application program routines to handle displaying, receiving, and deleting information in these windows.

A feature related to user-defined windows is user-defined messages (UDMs). These messages are handled much the same way as user-defined windows. Whenever Window Manager needs to display a user-defined message, an application's routine is called to display the message, handle user input, and delete the message.

See Chapter 8, User-Defined Windows, for an explanation of both features.

1.7 Internal Phrase Editing

There are many places in the NaturalLink runtime code where windows and messages are built at runtime, such as the headings and footers for Help messages. Version 2.0 has added a phrase-editing capability that allows you to modify these internal phrases for your specific applications. These phrase modifications can be link-time or run time bound. See Chapter 9, Internal Phrase Editing, for details.

1.8 New Attributes

New window attributes have been added to version 2.0 of Window Manager. They are listed here and described in Chapter 3, Window Attributes.

* Window border color/intensity
* Reverse video window border
* Window border takes up space
* Multiple line window labels

* Allow cursor to enter window
* Message text color/intensity
* Message border color/intensity

1.9 Application Input Routine

By using the new application input routine, you can now provide users with an alternative to using the keyboard for input to Window Manager applications. This routine is called by Window Manager whenever input is desired. Window Manager polls both this routine and the keyboard for all its input. See Chapter 10, Window Manager Input Devices, for more details.

1.10 New High-Level Language Calls

Version 2.0 of the Window Manager runtime includes several new calls. For a full explanation of these calls, refer to Chapter 6, Window Manager Callable Routines, and the proper appendix for the language you are using.

* DISMSG -- Display Message From Message Manager. This call replaces the 1.75 MSGMGR call. The MSGMGR call is still supported, but the new call supports any variable text separator that you want to use.
* WMWCRE -- Create Window. This call allows you to create new windows at run time. You can add the windows to existing screens or create a new screen to hold the new window.
* WMFLSH -- Flush NaturalLink Memory. This call clears out the memory space used by the NaturalLink runtime.

SECTION 2

Release Information

2.1 High-Level Language Interface

This paragraph contains information about the high-level lanaguage interface in version 2.0 of Window Manager.

* For all languages that support long integers, it is strongly recommended that these integers NOT be used as parameters for any Window Manager call. If long integers are used, the upper 16 bits are ignored and remain unchanged (the same as they were before the call) if used as a return parameter. For example, if 0001 0004 (hexadecimal) were passed, the value used would be 4, and if 5 were returned, the result would be 0001 0005.

* The heap variable in the files li???sh.asm has a maximum memory capacity of 55K bytes (where K equals 1024). The ??? characters in the module name are determined by the language in which your application is written. Please refer to the Memory Considerations section in the appropriate appendix (D, E, F, G) of the Window Manager manual or to Appendix C, High-Level Language Interface, of the Toolkit manual.

* The calls WMIADD, WMIINS, WMIDEL, WMICRE, WMGETV, WMGETS, WMSETV, and WMSETS can be made on windows that have not been added. The screen must be loaded, however.

* When using the WMWCRE call, you cannot create windows in (add windows to) the message screen or any NaturalLink interface screen (screen numbers 20 through 30). You can add windows only to regular Window Manager screens that have been loaded with the WMLOAD call (these screens will have screen numbers of 0 through 19). Passing in a number of a screen that has not been loaded results in the creation of a new screen. An alternative method for creating a new screen is to pass in -1 as the screen number.

* Do not make a WMFLSH call from any application routine (APPDIS, APPRCV, APPVAL, etc.) called by NaturalLink. Doing so could lead to a system crash.

* If you use a window to elicit the value for the LIwmpath string variable (wmpath in FORTRAN) and then want to load the internal phrase file NLXPHRAS.NM$, you must call the WMINIT routine again to load the phrase file. There are two reasons for this: first, the WMINIT routine must be called before any other Window Manager call is made, and second, this routine also loads the phrase file.

* For FORTRAN users, the file WMCINI.OBJ (which must be included in your link stream) is shipped in source form only. You must compile it with the version of FORTRAN you are using before linking your program.

* To create window labels and items with 80 characters of text on a single line, set the attributes LIwlabel (wlabel in FORTRAN) and LIittext (ittext in FORTRAN), respectively, with a WMSETS call.

2.2 Handling of Memory Errors

The following information applies to instances when NaturalLink cannot access enough memory to continue processing.

* If a memory error occurs during a Window Manager call while an application routine that was called by Window Manager (APPDIS, APPRCV, APPVAL, etc.) is being executed, a code of one (1) will be returned to the APP??? routine. Be sure to check return codes on all Window Manager calls so you can tell if a memory error occurs. If a memory error occurs during an APP??? routine, do NOT flush memory with the WMFLSH call. Simply return immediately from the APP??? routine and NaturalLink will automatically flush its memory.

* If NaturalLink runs out of memory while executing a WMICRE, WMIADD, WMIINS, or WMWCRE call, a return code of one (1) is returned immediately to the application. The application must issue a WMFLSH call if memory is to be cleared. For any other Window Manager calls, NaturalLink automatically flushes its memory when a memory error occurs. After making a WMFLSH call, any screens needed must be reloaded.

2.3 Message Attributes

Four message attributes are not mentioned in the manual. These four attributes allow you to set the color/intensity of the message window border, message heading, message text, and message footer. These attributes are used for all message types and will stay the same color until set again.

The names to use when setting these attributes with the WMSETV call follow (FORTRAN users omit the leading LI). These names and their constant values are in the respective include files for the language you are using. Note that if the message heading and message footer color is set to 0, then the heading and footer are displayed in the same intensity as the message text. Also note that when setting these attributes with the WMSETV call you do not need to pass in a valid screen, window, and item number.

LImsglin -- Message heading/label color/intensity
values 0 - 7, default 0/same as message text

LImsgtin -- Message text color/intensity
values 0 - 7, default 7/white

LImsgfin -- Message footer color/intensity
values 0 - 7, default 0/same as message text

LImsgbin -- Message border color/intensity
values 0 - 7, default 4/green

2.4 Message Manager

After Message Manager has displayed and deleted an error, please note, or warning message, it flags the windows covered by the message for repainting. The repainting is done the next time the application makes a receive call. If you want Message Manager to repaint the covered windows immediately, then you must set a flag telling it to do so.

Use the WMSETV call and pass it the field LImsgrep (msgrep for FORTRAN users). (You do not need to pass in a valid screen, window, and item number when setting this flag.) A value of 1 indicates that Message Manager should repaint immediately, and a value of 0 means that Message Manager should just flag the covered windows for repainting. This flag remains set until changed by another WMSETV call.

You must use this flag if you intend to call Message Manager from within your application validation routine. Otherwise, the screen will not be repainted after a message is deleted.

2.5 Message Builder/Screen Builder

When specifying a message in Message Builder or Screen Builder, you cannot use line feed characters in the message text. Using these characters can cause a crash if you attempt to view the message while running the utility.

2.6 Error Messages

The following three error messages were left out of Appendix H of the manual:

31 - Multiple window error type: ERROR

An error or warning was received from a WMWADD, WMWSEL, WMWREF, WMWREL, WMWDIS, or WMWDEL call that was passing in a -1 for the window number (-1 indicates that the call should be applied to all windows in the given screen). This error indicates that the call had a problem with one of the windows it was dealing with. Either the attributes in the window are set wrong, or the window has not been added, or the operation had already been performed on that window.

32 - NLXPHRAS.NM$ file problem type: ERROR

The NLXPHRAS.NM$ file on your default directory or the WMPATH variable is damaged or invalid. Make sure that your file is of the correct version and undamaged, then restart your program.

104 - Invalid screen for WMWCRE call type: ERROR

An attempt was made to create a window with the WMWCRE call in an interface screen or the message screen. The window was not created. Create the window in an existing Window Manager screen or create a new screen. You can create a new screen by specifying -1 as the value for the screen parameter in the WMWCRE call.

2.7 Other Introductory Information

The Window Manager manual does not explain how the NaturalLink Window Manager is packaged or describe a general development scenario. The following paragraphs clarify both of these points.

2.7.1 Packaging. The Window Manager portion of the NaturalLink Technology Package is packaged on four diskettes as follows:

* Window Manager Utilities Diskette A -- This diskette contains two utilities:
  - Screen Builder -- An interactive utility that permits you to specify a NaturalLink screen's appearance and behavior
  - Message Builder -- An interactive utility that aids in constructing an error/help message file for use with the NaturalLink Message Manager
* Window Manager Utilities Diskette B -- This diskette contains two utilities:
  - Set Function Keys Utility -- An interactive utility that permits you to specify which keys will perform the various Window Manager functions
  - Phrase Editing Utility -- An interactive utility that allows you to change the internal phrases used in the NaturalLink runtime
* Window Manager Runtime Object Diskette -- This diskette provides you with the object for a set of high-level language interface routines and the object code for the Window Manager to link in with your application code.
* Window Manager Demonstration Diskette -- This diskette contains an executable Window Manager demonstration program and the program's source code for the Lattice(R) C, MS-Pascal, and MS-BASIC languages. This demonstration program illustrates the use of several Window Manager features and function calls. A readme file included on the diskette explains the demonstration program.

Lattice is a registered trademark of Lattice, Inc.

2.7.2 Window Manager Application Development. A typical Window Manager development scenario is as follows:

1. Copy all Window Manager diskettes onto the Winchester drive. (A Winchester disk is not a necessity, but for reasons of speed and storage capacity it makes running the utilities easier.) A suggestion is to copy the support files for the utilities (the .PIC, .NM$, and .NS$ files) into one directory and the executable files into the same or another directory. Then set the environment variable NLXTOOLS to the path of where the support files are and put the directory path of the executable files in the MS-DOS command search path. This allows you to run the utilities from any drive or directory on your system.

2. Create the windows needed for your application using the Screen Builder utility (SBUILD). You can draw your windows and the test general functionality of your windows inside the Screen Builder utility without having to write any code. Screen Builder is described in Chapter 4 of the Window Manager manual.

3. Create the Help messages and other messages needed by your application using the Message Builder utility (MBUILD). You can use Message Builder to view each message exactly as it will be displayed in your application. Message Builder is explained in Chapter 5.

4. Once your Help messages are complete you can attach them to the windows and items you have created by using the Screen Builder utility. However, it is recommended that you not perform this step until you are sure your screens are in final form. This is because it is easier to attach all the Help messages for a given screen at the same time.

5. Write your application, which calls Window Manager to display your windows and receive information from the user. Chapter 6 details all the Window Manager calls, and a separate appendix for each language supported details the language-specific syntax of the calls.

6. If you want to use function keys that are different from the default keys, use the Set Function Keys utility (KBUILD) to generate a new file of function key assignments. Chapter 10 lists the default keys and explains how you can change them.

7. If you want to change the default internal phrases used by the Window Manager runtime, either edit and assemble the source file WMSTRDEF.ASM or use the Internal Phrase Editing utility (PBUILD) to create a file of phrases that can be loaded at run time. Chapter 9 explains both methods of changing phrases and provides a list of the internal phrases.

8. Link your application code with the Window Manager runtime object. Appendices D, E, F, and G explain which files apply to the language you are using and the correct link stream order.

NOTE

None of the Window Manager utilities create backups for you, so it is a good idea to keep backup copies of all your screen, message, key, and phrase files. For example, if you specify the name of a file to store a screen file to and a file by that name exists, it will be written over.