TI Part #2249843-0001

NaturalLink(TM) Technology Package 2.0
Toolkit 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 Toolkit 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 the Toolkit not found in the manual.

This Document: 2243731-0001

SECTION 1

Update Information

1.1 Compatibility Between Versions 2.0 and 1.75

Applications written for version 1.75 of the Toolkit are compatible with version 2.0. No source code needs to be changed. However, you MUST recompile your source if any Window Manager attribute constant values are being used. Once your source is recompiled, just relink your program using the new link stream instructions found in Appendix C of the Toolkit manual.

1.2 Multiple Machine Support

Version 2.0 of the Toolkit 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	

BUSINESS-PRO and PRO-LITE are trademarks of Texas Instruments Incorporated.
IBM is a registered trademark and IBM PC/XT and IBM Personal Computer AT are trademarks of International Business Machines Corporation.

Both the Interface Builder utility (IBUILD) and the runtime object code have been tested and verified to run on all of the mentioned machines (except the PRO-LITE, on which only the runtime has been tested) without any special modifications. An application developed and linked on the TI machines will run on the IBM machines without relinking. 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

The Toolkit now completely supports the MS-DOS 2.0/3.0 operating system. Both the runtime and the Interface Builder utility 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 Interface Builder utility reside. Using NLXTOOLS and the DOS PATH variable, you can run IBUILD from any drive and directory on your system.

1.4 Compiled BASIC Support

Version 2.0 of the Toolkit provides high-level language support for compiled BASIC. All Toolkit function calls are supported, enabling you to write Toolkit applications in MS-BASIC, compile them using the MS-BASIC Compiler, and then link and run them. For more information, see Appendix C, High-Level Language Interface, of the Toolkit manual.

1.5 Tailoring the Interface

Two new features make it possible to tailor the interface to meet certain application requirements. They are:

* Interface Customization -- This feature allows the user to edit phrases and/or create synonyms within the interface screen. See Chapter 16, Interface Customization, of the Toolkit manual for more information on this feature.

MS is a trademark of Microsoft Corporation.

* Dynamic Lexical Items -- This feature enables you to make limited modifications to the interface by creating or deleting terminal elements from the lexicon. See Chapter 9, Application Control of User Options, of the Toolkit manual for more information on this feature.

1.6 Interface Builder (IBUILD)

* You must delete the history information file (IBHIST.NM$) for the interfaces before executing IBUILD. Once in IBUILD you must respecify the information for each interface you plan to work on.

* You must regenerate all interface files before using the new runtime to execute them.

* The history information file (IBHIST.NM$) can now be created and maintained on any drive or directory; to do this, set the environment variable NLXHIST to the directory pathname of the directory desired. This feature is particularly important if you are working in a local area network (LAN) environment where IBUILD is being executed from a server that is write-protected. In this case, the history file cannot be created on the server and has to be located on some other drive or directory.

* You can now associate a directory pathname with each interface under development. This pathname specifies the drive and directory where the interface's files (grammar, screen, lexicon, etc.) will be located. You specify the directory pathname when recording the interface's information.

* A new option, Specify Help and Window Coordinates for Sessioner Windows, is now available in the Create and Test an Interface menu. This option enables you to attach help numbers and specify window coordinates for the windows created by the Sessioner during interface execution. These windows are the ones used by the Saved Sentences and Interface Customization features.

* The View a File option in the Create and Test an Interface menu is now invoked by the function key F5.

* The View a File option will have, as a default filename, the last output file that was specified or, if the Generate Representative Sentences option was the last option executed, the <intfile>.NG$ filename.

* The Test the Interface's Translation option, as it executes, now displays in a window each sentence that is being processed.

1.7 Sessioner

The NLXFUN routine can now return the same status codes back to the Sessioner as the NLXEXP routine can. See Chapter 15, The NaturalLink Sessioner, of the Toolkit manual for more details.

SECTION 2

Release Information

2.1 The Toolkit Manual

Like the Window Manager manual, the Toolkit manual has been written to reflect the keys used on the TI BUSINESS-PRO computer in PC-AT mode.

2.2 Interface Builder

* For interface development (using IBUILD), a system with 512K RAM (K = 1024 bytes) or more is recommended. Trying to develop interfaces on a system with 256K RAM could lead to a system crash and the loss of much time and effort. If you must use a 256K system, note that the following instances can lead to a system crash:

  a. Using the Generate or Modify the Lexicon (Lexicon Builder) utility and

     - Creating more than 10 duplicates at one time. This is regardless of whether the duplicates are of the same lexical item (terminal element) or different items.

     - Specifying all lexical information for every lexical item in the grammar in one session.

  For the examples just mentioned, it is possible to avoid a system crash by exiting the Lexicon Builder utility, saving the changes made to the lexicon, and then exiting to the main IBUILD screen. You should carry out this procedure at least once every 5 to 10 minutes or, in the case of the first example, after creating the tenth duplicate.

  b. Using the Test the Grammar utility and testing the static well-formedness of the grammar.

  c. Executing the Generate the Interface File utility

as the first operation you perform after entering IBUILD.

For these last two situations or others that may arise, a 512K system IS NECESSARY for interface development. There is no way to avoid these problems on a 256K system.

Please note that the preceding item pertains only to Interface Development (IBUILD) and not to Window Manager or its utilities. Also note that the following items pertain to all IBUILD users regardless of the amount of RAM in the system.

* Do NOT set the 'Allow cursor to enter window' attribute to Never (2) for any Command window, Results window, or parse window in NLmenu screens. The Check the Screen Against the Lexicon option generates a warning if this attribute is set to Never. If the attribute is set to Never, it will be forced to Always (0) when the interface file is generated.

* If a memory error occurs during the Lexicon building (Lexicographer) process while you are modifying translation information for experts, the cause may be a bad expert information file (<intfile>.NP$). If this error occurs, delete the expert information file and respecify all experts.

* When you specify help and window coordinates using the Specify Help and Window Coordinates for Sessioner Windows option, the help numbers or window coordinates you specify for the Remove All Changes? Interface Customization window will be used for the Quit (Do you want to save your changes?) window.

* During execution of the interface, the window attributes for the Saved Sentences and Interface Customization windows may differ from the window attributes of these windows viewed through the Specify Help and Window Coordinates for Sessioner Windows option. This difference can arise because this option sets the window attributes according to the window attributes of the first nonpopup list window containing phrases in the NLmenu screen, while the Sessioner uses the window attributes from the first parse window in the NLmenu screen.

* If IBUILD resides in the root directory and will be executed from another directory, the MS-DOS PATH environment variable must be set to include both E: and E:\. This is because of the way the PLINK86(TM) overlay loader works.

* The Lexicon Builder utility does not let you specify phrases with leading blanks.

2.3 Sessioner

* Expert items CANNOT be edited if the Results window is a user-defined window(UDW).

* It is not possible to add synonyms or edit phrases in a user-defined window.

* The Sessioner sets the window attributes for the Saved Sentences and Interface Customization windows according to the window attributes of the first parse window in the NLmenu screen.

* User-defined command/function keys are not restricted to ALT-function key combinations. Any key combination that returns an extended key code can be used, as long as it has not already been assigned a function.

* When NLXEXP sends an invalid return code back to the Sessioner, status code #62 is returned to the application from NLDRIV.

* When NLXFUN sends an invalid return code back to the Sessioner, status code #78 is returned to the application from NLDRIV.

PLINK86 is a trademark of Phoenix Software Associates, Ltd.

2.4 High-Level Language Interface

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

* The heap variable in the files li???sh.asm has a maximum of 55K bytes. The '???' characters in the module name are dependent upon the language that your application is written in. Please refer to the paragraph called Memory Considerations in the appropriate appendix (D, E, F, G) of the Window Manager manual or the Calling and Linking paragraph in Appendix C, High-Level Language Interface, of the Toolkit manual.

* If you are using the Lattice C Compiler(TM) version 2.1 or higher, and it was installed by means of the install batch stream, you must use the appropriate models of the modules C.OBJ and LC.LIB when linking the application program. These modules will replace C?.OBJ and LC?.LIB in the link streams shown in the manuals.

2.5 Handling of Memory Errors

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

* When the Parser runs out of memory, a status is passed to the Sessioner. The Sessioner then displays a message to the user that the sentence is too long and that the user must either back up to an executable state or start over. (Note that if the user does not back up or start over when told to do so, the system will probably crash.) If the memory error occurs while the Sessioner is processing a user-defined expert, the NLXEXP routine will be executed again. If the NLXEXP routine makes any Window Manager calls, NaturalLink may run out of memory while in the NLXEXP routine. Be sure to check return codes on all Window Manager calls so you can tell if a

Lattice C Compiler is a trademark of Lattice Incorporated.

memory error occurs. A code of one (1) indicates that a memory error occurred. If a memory error occurs during the NLXEXP routine, do NOT flush memory with the WMFLSH call. Simply return immediately to the driver (NLDRIV) and NaturalLink will automatically flush its memory. The same is true for the NLXFUN routine and any other application routine called by NaturalLink.

* If at any time any NaturalLink call returns a memory error while in the NLXEXP or NLXFUN routines, do NOT flush memory with the WMFLSH call. Simply return to NLDRIV and NaturalLink will automatically flush its memory.

2.6 Error Messages

* Error code 91 can occur with either the NLXSND or the NLXCRE call.

* Four error codes are missing from Appendix B, NaturalLink Error Codes, of the Toolkit manual. They are as follows:

  103 -- No output file pathname was specified in an NLXEDT call for saving changes made. An output file must be specified, even if the pathname is the same as that of the interface file being edited.

  104 -- 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.

  105 -- An attempt was made to flush NaturalLink's memory after a memory error has occurred. WMFLSH cannot be called from an application routine that was called by NaturalLink. NaturalLink will automatically flush its memory after a memory error has occurred.

  106 -- WMFLSH cannot be called from an application routine that was called by NaturalLink.

2.7 Interface Customization

* Do not use the Dynamic Lexical Items feature with the Interface Customization feature. That is, do not link NLDTLIB.OBJ with NLEDLIB.OBJ and NLSYLIB.OBJ together in the same program. Doing this could lead to a bad interface file or a system crash. This is because the Interface Customization feature enables the user to make changes to the interface and then save them to disk, writing over the existing interface file. See the Note in the User-Defined Window Demonstration Program paragraph later in this document for more information.

* The output filename parameter for the NLXEDT call should NOT be NULL. If it is NULL, an error is generated.

2.8 Dynamic Lexical Items

Do not use the Interface Customization feature with the Dynamic Lexical Items feature. That is, do not link NLDTLIB.OBJ with NLEDLIB.OBJ and NLSYLIB.OBJ together in the same program. Doing this could lead to a bad interface file or a system crash. This is because the Interface Customization feature enables the user to make changes to the interface and then save them to disk, writing over the existing interface file. See the Note in the User-Defined Window Demonstration Program paragraph for more information.

2.9 Linking

If PLINK86 is being used when you are linking your application program, you may receive a warning about duplicate stack 'STACK'. You can ignore the warning; the execution of the application is not affected.

2.10 User-Defined Window Demonstration Program

The program TKDEMO.EXE, on Toolkit Demonstration diskette B, is a Lattice C program that demonstrates how to use the Window Manager and Natural Language interface with user-defined windows. In particular, TKDEMO incorporates user-defined input, display, receive, delete, and Help message routines. The program also

demonstrates the new Dynamic Lexical Items and Interface Customization features.

The B diskette contains all the files necessary to run the program; type "TKDEMO" to execute it. The Toolkit Demonstration diskette A contains the source files to the program:

TKDEMO.C is the driver routine.
UDWCODE.C contains the user-defined window routines.
DBCODE.C contains the database routines.
TKDEMO.SCR is the driver's screen.
TRAVEL.* are the files needed to build the interface.
DMDEFINE.H is the include file for the database.

TKDEMO uses the HALO(TM) graphics runtime library to give examples of using graphics and mouse input with the new Toolkit features. However, Texas Instruments does not endorse or require that product; you can use any other suitable package with your applications. Since TKDEMO uses the TI version of the HALO runtime, it will not work on any IBM computer or on the TI BUSINESS-PRO in PC-AT mode.

HALO supports three mice: Mouse Systems(TM), Summagraphics(R), and Microsoft. The program is currently set for the Mouse Systems mouse. A mouse is not required, however, because keyboard input is also accepted for all features.

When you execute TKDEMO, the first window that appears offers you a choice of two options:

* TRAVEL INFORMATION ONLY

* ADD STATISTICAL INFORMATION

Choosing the second option (ADD STATISTICAL INFORMATION) puts into effect the Dynamic Lexical Items feature, adding to the first parse window three new items that are not included with the first option:

(Show) "the population of"

(Show) "the median income of"

(Show) "the average tornadoes in"

HALO is a trademark of Media Cybernetics, Inc.
Mouse Systems is a trademark of Mouse Systems Corporation.
Summagraphics is a registered trademark of Summagraphics Corporation.

After you choose an option the interface screen appears. The window on the right is a user-defined parse window, using graphics instead of text. Choosing "the cities with" from the first parse window takes you to the user-defined parse window, where an arrow appears instead of a cursor. Either the mouse or the keyboard will move the arrow, and selection is similar to that in other icon-based systems. When a sentence is executed, a window demonstrating graphics and text appears with the answer to the query.

Selecting "the National Parks in" brings up another user-defined window, this one a pop-up. You can demonstrate user-defined Help messages by selecting "the points of interest in" from the first window and then pressing the Help key for any item in the list.

TKDEMO also demonstrates the Interface Customization feature, which allows the user to change the wording of items in the interface or add synonyms for them. Press the F3 key to bring up this feature's menu.

For further information on user-defined windows, the Interface Customization feature, and the Dynamic Lexical Items feature, consult the Window Manager and Toolkit manuals.

IMPORTANT NOTE

The Dynamic Lexical Items feature and the Interface Customization feature should not be linked in to the same program at the same time. However, in order to demonstrate how these features are linked into a program and invoked, the TKDEMO program includes both features. The demo program will function correctly as long as you DO NOT SAVE CHANGES when leaving the Interface Customization feature. You can use the feature as much as you like to get the feel of it, but on quitting the feature, answer "no" when prompted for whether or not to save the changes you have made. If you do save the changes, the next time you run the demo program you will get an error stating that NLXCRE could not create dynamic lexical items because of a bad DTS (dynamic terminal symbol) code. You can fix this problem by restoring the file TRAVEL.INT from your master NaturalLink Toolkit Demo diskette B. The following sequence shows how and why the error occurs and why the two features should not be included in the same program:

1. The program loads the interface file.

2. The program calls NLXCRE to replace the DTS entries

in the lexicon with new lexical entries. This means that the DTS entries are no longer in the copy of the lexicon in memory.

3. The program calls NLDRIV to bring up the interface, including the new lexical entries.

4. The user presses the Interface Customization key to invoke the Interface Customization feature.

5. The user makes some modifications to the interface and specifies that the changes be saved.

6. The interface, AS IT EXISTS IN MEMORY (that is, minus the DTS entries necessary to the NLXCRE call), is written to disk.

7. The user plays with the demo for a while and then exits the program.

The next time the program is invoked and it calls NLXCRE to add the new lexical entries, the dynamic terminal symbols are no longer in the lexicon because the interface was saved after they had been removed.

