NIAKWA PROGRAMMING LANGUAGE

MS-DOS SUPPLEMENT

1st Edition - August 1993

Niakwa, Inc.
23600 N. Milwaukee Avenue
Mundelein, IL 60060

PHONE (708) 634-8700   FAX (708) 634-8718   TELEX 3719965 NIAK UB

PREFACE

P.1 The NPL MS-DOS Supplement

The Niakwa Programming Language (NPL) MS-DOS Supplement is designed as an aid to the installation, operation and operating system-specific features of the NPL Interpreter, Compiler, RunTime program and related utilities in the following environments:

Hardware	**Operating System**	**Operating Environment**
IBM-Compatible PC	MS-DOS 2.0 or greater	MS-DOS
IBM-Compatible PC	MS-DOS 2.0 or greater	Novell NetWare 2.10 or greater*
IBM-Compatible PC (80286 or greater)	MS-DOS 3.1 or greater	MS-Windows 3.1 or greater*
IBM-Compatible 80386 PC or greater	MD-DOS 3.1 or greater	Phar Lap 386/DOS-Extender*

* The documentation for operating environment-specific features is found in the appropriate operating environment-specific addenda.

P.1.1 Prerequisite Knowledge

This guide assumes at least a basic knowledge of the IBM Personal Computer, the IBM Disk Operating System Version 3.1 or greater (DOS).

P.1.2 How to Use the NPL Documentation

The NPL documentation has been structured into generic and operating environment dependent manuals to allow developers easy reference to detailed platform-specific information. The NPL Technical Reference Guide (TRG) consists of two manuals, the NPL Programmer’s Guide and the NPL Statements Guide. The TRG set is intended as a generic guide for programmers in the correct use of NPL and its program development and debugging facilities on all supported environments.

The TRG set is designed to be used in conjunction with the NPL operating system-specific supplements, that clearly document the operating system-dependent features of NPL on that specific operating system. Some supplements also contain a series of operating environment-specific addenda. These addenda discuss the NPL operation and options that are either unique to a given operating environment, or function differently from platform to platform.

Additionally, each supplement contains a set of Release Notes, providing information pertaining to an operating system-specific-supplement that was not available at press time.

Refer to the Preface in the NPL Programmer’s Guide for more information on the use of the NPL documentation.

P.1.3 How to Use this Supplement

The NPL MS-DOS Supplement discusses the operation of NPL under MS-DOS based operating environments. The main MS-DOS Supplement is followed by a series operating environment-specific addenda that discuss additional NPL operating considerations under the specific operating environment being covered.

P.1.4 Notational Conventions

This guide uses the following notational conventions:

NOTE: Notes provide information of particular importance.

WARNING--Warnings are special conditions that require extra care by the user.

HINT: Hints provide helpful comments pertaining to the use of particular features.

The following conventions are used in the illustration and definition of NPL:

- Each statement appears on a separate page, with the statement as a page header.
- The general form of each statement is enclosed within a box.
- Uppercase letters ("A" through "Z"), digits ("0" to "9"), and special characters (such as "$", "#", ":") must always appear exactly as presented in the general format.
- All lower-case words indicate information that the user must supply. These words appear in *italic* type.

For example:

```
LEN (alpha-variable)
```

The user must supply the alpha-variable.

- When braces, "{ }", enclose a vertically stacked list, or a horizontal list with each item separated by a comma (","), the user must choose one of the options within braces. Information within braces is shown in *italic* type.

  For example:

```
ALL ({literal-string, alpha-variable, two-hexdigits})
```

  or

```
ALL   ({literal-string  })
       {alpha-variable  }
       {two-hexdigits   }
```

Here, the ALL instruction must be followed by one and only one of the items in the list.

- Brackets, "[ ]", indicate that the enclosed items are optional. When brackets enclose a vertical list or a horizontal list, the user may specify one or none of the items. Information within brackets is shown in *italic* type.

  For example:

```
INPUT [literal-string,] variable [,variable]...
```

  Here, the INPUT instruction may optionally contain a literal-string followed by an optional comma preceding the required "variable". Additional variables may optionally be specified.

  or:

```
CLEAR [V                                ]
      [N                                ]
      [P [line-number1][,[line-number2]]]
```

  Here, either the V, N, or P parameter may be specified, but no parameter is required.

NOTE: Here, line-number parameters may be optionally specified only if the "P" parameter is specified.

- The presence of an ellipsis (...) within any format indicates that the unit immediately preceding the ellipsis can occur one or more times in succession.

  For example:

  ```
  DEFFN'integer[(variable[,variable]...)]
  ```

  Here, any number of "variable" may be specified, but the format ",variable" must be used for the second and subsequent "variables".

- All other punctuation such as commas or parentheses must be included where shown.

P.1.5 Terminology

Throughout this Supplement, the following terms are used:

Niakwa NPL Development Package
This refers to the entire contents of the package, including the NPL Development Package diskettes, the NPL Technical Reference Guide (Programmer's and Statements Guides), and the NPL Supplement. This term may frequently refer to just the software elements of this package.

Niakwa NPL Technical Reference Guide
This refers to the NPL Statements Guide and Programmer's Guide.

NPL Compiler
This refers to the actual compiler program itself, B2C.EXE.

Niakwa NPL Development Package Diskettes
This refers to the physical diskettes delivered as part of the Niakwa NPL Development Package. These diskettes contain the contain the NPL Compiler, Utilities, etc. software used in NPL software development.

Niakwa NPL RunTime Package
This refers to the RunTime Installation Guide and diskettes. This package is used by the Authorized NPL Developer for executing and testing the application object code generated by the compiler. It is also used by end-users for executing application object code purchased from a Authorized NPL Developer.

Niakwa NPL RunTime Program
This refers to the Interpretive (RTI) or Non-Interpretive (RTP) RunTime programs.

Niakwa NPL RunTime Program Diskette(s)
This refers to the physical diskette(s) delivered as part of the Niakwa NPL RunTime Package. The first diskette of this package is also referred to as the Gold Key diskette.

TABLE OF CONTENTS

COMPILER OPERATION

DOS ERROR CODES

"RAW" DEVICE COMPATIBILITY CHART

CO-PROCESSOR SUPPORT

CHAPTER 1

INTRODUCTION

1.1 Overview

The NPL Supplement for MS-DOS on the IBM PC, is intended as an aid in the correct installation and use of the NPL Interpreter, Compiler and RunTime program.

Section 1.2 describes the contents of the NPL Development Package.

Section 1.3 describes the contents of the NPL RunTime Package.

Section 1.4 discusses the main features of the RunTime program which are specific to IBM-compatible PCs operating under MS-DOS.

Section 1.5 provides an overview of the NPL MS-DOS Supplements and its supporting addenda.

1.2 Contents of the NPL Development Package

The NPL Development Package is intended for software vendor use in the development and execution of application software on an IBM-compatible PC. The NPL Development Package consists of the NPL MS-DOS Supplement and six diskettes.

The following is a description of the contents of these diskettes.

Compiler Diskette

ENABLED	The "enable" file necessary for allowing the interpretive RunTime program to function.
B2Cx.BAT	Example batch files showing command-line use of the compiler designed to simplify use.
UPGRTEST.BAT	A batch file that executes the UPGRTEST.OBJ program.
EXSNPL.BS2	A diskimage containing NPL Release IV example programs.
NPLEXAM.BS2	A diskimage containing the NPL Release IV example programs as discussed in Chapter 17 of the NPL Programmer’s Guide.
NPLSYS.BS2	A diskimage containing NPL Release IV library modules used by the example programs as discussed in Chapter 3 of the NPL Statements Guide.
B2C.EXE	The NPL Compiler program.
B2CHELP.HLP	A text file containing help entries for the NPL compiler.
B2CHELP.IDX	A file containing the index used when the NPL compiler help file is accessed.
BOOT.OBJ	The default RunTime boot program in compiled p-code format.

GOLDKEY.OBJ	A boot program that converts a serial number into its corresponding #GOLDKEY value.
REL4EXAM.OBJ	A boot program used to start the Release IV example programs.
UPGRTEST.OBJ	A boot program that can determine what upgrade options are available for an installed RunTime.
UPGRTEST.TXT	A text file that provides the instructions for the UPGRTEST.OBJ program.
BOOT.SRC	The default RunTime boot program in source format.

Utilities Diskette

UTILITY.BS2	A diskimage file containing the NPL Utilities.
UTILHELP.HLP	An indexed help files for use by the NPL Utilities.
UTILHELP.IDX	A file containing index listings used when the help file is accessed.
UTILITY.OBJ	A compiled version of the boot program to start up the NPL Utilities.
UTILITY.SRC	A source code version of the boot program for the Utilities.

Terminal Support Files Diskette

VTFONT.AL5	Font file used for downloading the NPL screen character set for the Altos V terminals.
WYKEYS.OFF	File necessary for correct keyboard operation in application-key mode of Wyse 150 and 160 terminals in Wyse 60 emulation.
WYKEYS.ON	File necessary for correct keyboard operation in application-key mode of Wyse 150 and 160 terminals in Wyse 60 emulation.

FONT.TBL	A character font file which can be used as a reference font file by the NPL Font Table Editor.
FONTIBME.TBL	Modifiable font files used to create downloadable font file for EGA and HGA controller.
FONTIBMH.TBL	Modifiable font files used to create downloadable font file for the IBM EGA board and Hercules MGA board.
FONTxxxx.TBL	Font table files used for editing and creating the above downloadable font files.
VTFONT.VT2	A font file used for downloading the screen character set for the VT200 series terminals.
VTKEYS.VT2	A file necessary for downloading the uppercase key definitions for VT200 series terminals.
WYKEYS.WY3	A file necessary for proper keyboard operation of a Wyse 370 terminal.
WYFONT.WY6	A font file used for downloading the screen character set for the Wyse 60 terminals.
KEYBOARD.xxx	A series of keyboard translation tables used for handling differences between the keyboard and hex codes expected by NPL programs. These files are used on the various NPL-supported terminals that can be used for remote maintenance with the Redirect feature of the RunTime.
SCREEN.xxx	A series of screen translation tables used for displaying the NPL standard character set. These files are used on the various NPL-supported terminals that can be used for remote maintenance with the redirect feature of the RunTime.
W370FONT.xxx	Font files used for downloading the screen character set (80 or 132 columns) for the Wyse 370 terminals.

BESDK Diskette

This diskette contains the required files necessary to allow NPL to interface with external subroutines written in other languages. For a complete description of all files on this diskette, refer to Chapter 11.

Supplementary Files Diskettes

These diskettes contain development files for additional operating environments. Refer to the appropriate addendum in this supplement for a complete description of the diskette contents.

1.3 Contents of the Niakwa NPL Runtime Package

The Niakwa NPL RunTime Package is intended for:

- Vendor use in executing and testing application object code developed in NPL.
- End-user use for execution of application object code purchased from an Authorized NPL Developer.

The NPL RunTime Package is packaged as two unique products. The first product is the Niakwa RunTime Package and is normally shipped for new orders of NPL. The second product is the Niakwa Upgrade RunTime Package. This may be ordered for upgrades to existing customers in cases when the customer already has a Revision 3.xx RunTime installed on the system. The advantage of the Upgrade RunTime is that the old Gold Key does not need to be returned. The contents of each product are different. Refer to the other sections in this chapter for more details on the contents of the Niakwa RunTime and the Niakwa Upgrade RunTime Packages.

The Niakwa RunTime and Niakwa Upgrade RunTime Packages are available on both 5-1/4" and 3-1/2" media. The files for each RunTime Package are contained on two 5-1/4" diskettes or one 3-1/2" diskette.

1.3.1 Niakwa RunTime Package Files

The Niakwa RunTime Package (for new installations) physically consists of the Niakwa RunTime Package Installation Guide and two 5-1/4" diskettes or one 3-1/2" diskette. These diskettes are labeled as follows:

5-1/4"

- Niakwa NPL RunTime Package - GOLD KEY - Disk 1 of 2.
- Niakwa NPL RunTime Package - GOLD KEY - Disk 2 of 2.

3-1/2"

- Niakwa NPL RunTime Package - GOLD KEY

NOTE: For some products, extra RunTime diskettes may be included as supplements to the standard RunTime. Refer to the appropriate operating environment-specific addendum in this supplement for the correct number of diskettes and file contents.

The "Gold Key" is necessary to complete any task requesting or requiring the Gold Key Diskette (i.e., installing the Gold Key security to the hard drive, passing security, or performing a Reset).

Niakwa NPL RunTime Package File Listing

The Niakwa RunTime Package contains the following files:

NIAKINST.BAT	Installs the copy protection on the file server.
NIAKRCLL.BAT	Recalls the copy protection from the file server and allows the Gold Key to be "installed" on another file server or volume.
NIAKMOV1.COM	File used by the "install" procedure.
NIAKMOV2.COM	File used by the "install" procedure.
NIAKMOVE.COM	File used by the "install" procedure.
NIAKSECx.COM	One of six programs used by the copy protection security program (where "x" represents the last character of these files).
RESET.COM	The program that allows the copy protection security install count to be updated remotely.
NIAKSER.DAT	File used by the Runtime security check.

IBMFONT0.EGA	File containing the screen character set for the EGA controller.
RTI.EXE	The Interpretive RunTime program.
RTP.EXE	The Non-interpretive RunTime program.
IBMFONT0.HGA	File containing the screen character set for the HGA controller.
ERRORMSG.HLP	A text file that contains the error messages that are displayed when using the Interpretive RunTime program.
RTIIERR.HLP	A text file that contains the MS-DOS error messages that are displayed optionally when using the Interpretive RunTime program.
ERRORMSG.IDX	File containing the index used when the ERRORMSG.HLP file is accessed.
RTIIERR.IDX	File containing the index used when the RTIIERR.HLP file is accessed.

1.3.2 Niakwa NPL Upgrade RunTime Package

The Niakwa Upgrade RunTime Package physically consists of the Niakwa RunTime Package Upgrade RunTime Installation Guide and two 5-1/4" diskettes or one 3-1/2" diskette. These diskettes are labeled as follows:

5-1/4"

- Niakwa NPL Upgrade RunTime Package - Disk 1 of 2
- Niakwa NPL Upgrade RunTime Package - Disk 2 of 2

3-1/2"

- Niakwa NPL Upgrade RunTime Package

The 5-1/4" diskette labeled Niakwa Upgrade RunTime Package, Disk 1 of 2, or the 3-1/2" diskette labeled Niakwa NPL Upgrade RunTime Package is necessary to complete the upgrade procedure.

Niakwa NPL Upgrade RunTime Package File Listing

The Niakwa Upgrade RunTime Package contains the following files:

UPGRADE.BAT	A file used to start the upgrade procedure.
UPGRADE.BS2	A diskimage used by the upgrade procedure.
NIAKMOVE.COM	File used by the "install" procedure.
NIAKSECx.COM	One of six programs used by the copy protection security program (where "x" represents the last character of these files).
RESET.COM	The program that allows the copy protection security install count to be updated remotely.
IBMFONT0.EGA	File containing the screen character set for the EGA controller.
RTIUPG.EXE	The Interpretive RunTime program.
RTPUPG.EXE	The Non-interpretive RunTime program.
UPGR2.EXE	A file used by the upgrade procedure.
IBMFONT0.HGA	File containing the screen character set for the HGA controller.
ERRORMSG.HLP	A text file that contains the error messages that are displayed when using the Interpretive RunTime program.
RTIIERR.HLP	A text file that contains the MS-DOS error messages that are displayed optionally when using the Interpretive RunTime program.
ERRORMSG.IDX	File containing the index used when the ERRORMSG.HLP file is accessed.

RTIIERR.IDX	File containing the index used when the RTIIERR.HLP file is accessed.
UPGR0.OBJ	A file used by the upgrade procedure.

The NPL Upgrade RunTime Package is ordered separately from Niakwa for each end-user who is upgrading from a Revision 3.xx MS-DOS RunTime Package. Refer to Chapter 2 for further details on the Niakwa NPL Upgrade RunTime Package.

1.4 MS-DOS Specific Features of NPL

The following are MS-DOS specific features of NPL.

1.4.1 Mouse Support

The use of a mouse with NPL is supported if a mouse driver has been loaded prior to executing the RunTime. Mouse support is activated with the /K startup option. Refer to Chapter 4 for more details on the use of a mouse with NPL under MS-DOS.

1.4.2 XMS Memory Support

NPL Release IV supports the optional use of XMS memory. Use of XMS memory can result in up to 200KB of additional memory being available to NPL applications. NPL can access two types of XMS memory:

- HMA (High Memory Area) is a 64KB segment above the 1024KB address. It is made accessible to the RunTime with the /M startup option.
- UMB (Upper Memory Blocks) are segments of memory between the 640KB and 1024KB addresses, and vary in size and quantity depending on hardware configuration. They are made accessible to the RunTime with the /U startup option.

Refer to Chapter 4 for specific details on using the /M and /U startup options with NPL under MS-DOS.

1.4.3 Hardware Support

The MS-DOS RunTime program supports single-user operation on individual IBM-compatible PC's under MS-DOS. IBM-compatible PC's have several hardware characteristics which provide a high level of support for several hardware-dependent features of NPL:

- Wang 2200 320K format "raw" diskettes are supported. In addition, 360K, 720K, 1.2MB, 1.44MB, and 2.88MB "raw" diskettes are supported. Refer to Section 5.2 and Appendix D for details.

- Color and graphics modes are supported for specific controllers and monitors. Refer to Chapter 6 for details.

1.5 MS-DOS Supplement Overview

The MS-DOS Supplement is designed to provide information on operating NPL on MS-DOS based systems. This supplement documents platform-specific features only. General language features are discussed in the NPL Programmer's Guide or the NPL Statements Guide.

Chapter 2 of the Supplement discusses installation and configuration requirements of NPL on MS-DOS systems. It also discusses the NPL Gold Key security and security installation.

Chapter 3 discusses configuration of the NPL system including location of files, and use of auxiliary files.

Chapter 4 discusses RunTime operations, describing various command-line startup options available, and exiting the RunTime.

Chapter 5 discusses devices supported by NPL. These include diskimage files, "raw" diskettes, monitors, printers, tape drives, serial devices, math coprocessors, and mice.

Chapter 6 discusses display characteristics of NPL on various monitor types, including Hercules, CGA, EGA and VGA. It also discusses graphic capabilities and keyboard characteristics.

Chapter 7 discusses multi-user issues for MS-DOS systems and how they can co-exist with networks.

Chapter 8 discusses language features that are specific to MS-DOS, including differences with the system variables $MACHINE and $OPTIONS.

Chapter 9 covers operation of the B2C compiler and conventions used under MS-DOS.

Chapter 10 discusses options for porting programs and data to MS-DOS systems from other platforms.

Chapter 11 discusses the use of mixed-language programming under MS-DOS to create external libraries.

Appendix A contains some common problems and solutions encountered while configuring or executing NPL under MS-DOS.

Appendix B contains common MS-DOS error codes encountered during disk I/O or native operating system commands.

Appendix C discusses support of math co-processors under NPL.

Appendix D diagrams "raw" device compatibility for all current NPL-supported systems.

The Novell NetWare Addendum discusses all specific installation and multi-user operation details relating to NPL running under Novell NetWare.

The MS-Windows Addendum discusses all specific installation and operation details relating to NPL operating in the MS-Windows environment.

The 386/DOS-Extender Addendum discusses all specific installation and configuration requirements for using the 32-bit version of NPL.

CHAPTER 2

INSTALLATION

2.1 Overview

This chapter provides instructions for installing the NPL Development and RunTime package, under MS-DOS.

Section 2.2 discusses the operating system and hardware requirements for use of NPL under MS-DOS.

Section 2.3 discusses the operating system configuration requirements for the CONFIG.SYS and AUTOEXEC.BAT files.

Section 2.4 discusses the installation of the NPL Development Package under MS-DOS.

Section 2.5 discusses the NPL RunTime's Gold Key security.

Section 2.6 discusses the installation of the NPL RunTime package under MS-DOS.

Section 2.7 discusses the installation of all MS-DOS based NPL Upgrade RunTime Packages.

2.2 OS and Hardware Requirements

The MS-DOS implementation of the Niakwa Programming Language is designed to operate on systems meeting the following requirements:

- MS-DOS version 2.0 or greater.
- 640K of memory or greater. The MS-DOS version of NPL only uses conventional memory by default. The RunTime startup options /U and/M can be used to use XMS memory. Refer to Chapter 4 for details on the RunTime startup options.

NOTE: For information on other MS-DOS based operating environments supported by NPL, refer to the addendum at the end of this supplement.

- A hard disk is recommended. In network environments, a hard disk is required only for the server.
- One diskette drive must be present. This can be either a 3-1/2" or 5-1/4" standard or high density drive.
- Standard serial or parallel MS-DOS compatible printers are supported.

NOTE: Some Wang 2200 printers do not work on parallel ports due to hardware-level incompatibilities. Refer to Section 5.5 for details.

2.3 NPL Configuration Requirements

The following modification should be made to the CONFIG.SYS and AUTOEXEC.BAT files on the PC where NPL is installed.

NOTE: The values shown below are minimum requirements.

2.3.1 The CONFIG.SYS File

For efficient operation of the RunTime, it is recommended that the default configuration parameters be modified. The MS-DOS configuration parameters are set up in a file called CONFIG.SYS. This is located in the root directory on the MS-DOS boot disk. Please refer to the MS-DOS manual for details on CONFIG.SYS.

To change an existing CONFIG.SYS file (or to create a new one), use a standard text editing program. The standard MS-DOS text editing program is EDLIN or EDIT (MS-DOS 5.0 or greater) and is documented in the MS-DOS manual.

It is recommended that the CONFIG.SYS file be set up as follows:

```
BUFFERS = 20
FILES = 20
DEVICE =\ANSI.SYS
BREAK = ON
```

NOTE: To make the changes take effect, the system must be reset once the changes are made.

WARNING--Make sure to make a copy of the existing CONFIG.SYS and AUTO-EXEC.BAT files. Improper modification of these files could cause the system to lock when starting the system.

The parameters should be entered exactly as stated, except as noted below. The specific parameters are as discussed below.

Buffers

The BUFFERS statement determines the amount of buffer space allocated in memory.

Modify or add the following to the CONFIG.SYS file:

```
BUFFERS=20
```

NOTE: By increasing the number of allowable BUFFERS, the performance of the RunTime may increase. If large diskimage files are used, a larger buffer value should be set.

If this is already set up at more than 20, do not change it.

Files

The FILES statement determines the maximum number of files which may be open at one time.

Modify or add the following to the CONFIG.SYS file:

```
FILES=20
```

If this parameter is already set at more than 20, do not change it.

ANSI.SYS

This statement is required if using a non-standard monitor. Non-standard monitors require that the ANSI.SYS driver be loaded at boot time. This statement need not be included if a standard PC monitor is being used on the system.

The following should be added to the CONFIG.SYS file if a non-standard monitor or if the Redirect RunTime startup option, /R, is to be used:

```
DEVICE=ANSI.SYS
```

NOTE: If the ANSI.SYS file is not in the root directory, make sure that the full path name of the location of the ANSI.SYS file is specified in the DEVICE statement.

BREAK

The use of the BREAK statement allows the use of Control-C to cancel the RunTime program when it would otherwise be necessary to reboot the system to exit.

To use the BREAK statement, add the following to the CONFIG.SYS file:

```
BREAK=ON
```

STOP *WARNING: Do not use Control-C to exit the RunTime program. Improper use of Control-C may cause loss of data.*

2.3.2 The AUTOEXEC.BAT File

The AUTOEXEC.BAT file in the root director of the PC’s hard drive contains commands that are automatically executed when the system is booted. Niakwa recommends adding or modifying the following AUTOEXEC.BAT file as follows.

NOTE: The system must be booted, after making any changes, for the change to go into effect.

PATH

Add the following line to the AUTOEXEC.BAT file.

```
PATH X:\BASIC2C
```

where X is the drive where the \BASIC2C directory is located. If the NPL files are located in another directory, use that name instead.

NOTE: This discussion assumes the installation of the RunTime to C:\BASIC2C. If another drive or directory is used, use those designations instead.

This statement sets up a X:\BASIC2C directory as an alternate path. This means that the operating system automatically looks for programs in the X:\BASIC2C directory if they cannot be found in the root directory.

The PATH statement may already exist in the AUTOEXEC.BAT file. If so, then modify it by adding the following to the end:

```
;X:\BASIC2C
```

where X is the drive where the \BASIC2C directory is located.

For example, assume that the statement:

```
PATH C:\DOS
```

already exists. Modify this statement to:

```
PATH C:\DOS;C:\BASIC2C
```

Optional Changes

If the NIAKWA_RUNTIME and/or NPL_SECURITY environment variables are being used, it may be useful to add these statements to the AUTOEXEC.BAT file as well. Refer to Section 2.5.5 and 2.5.6 for details on these environment variables.

2.4 Installing the NPL Development Software

This section discusses the installation of the Niakwa NPL Development Package on an MS-DOS based system. The NPL Development Package consists of six diskettes labeled:

- Compiler Diskette
- Utilities Diskette
- Terminal Support Files Diskette
- MS-DOS BESDK Diskette
- MS-Window Supplementary Files Diskette
- 386/DOS-Extender Supplementary Files Diskette

The first four diskettes are necessary for NPL MS-DOS development. The last two (supplementary files diskettes) are specific to their operating environments. Refer to the appropriate operating environment-specific addendum, at the end of this supplement, for more information on these diskettes.

The contents of these diskettes are described in Section 1.2. The first three diskettes should be installed in the\BASIC2C directory or the directory that the NIAKWA_RUNTIME environment variable is set to (refer to Section 2.5.6 for more information about the NIAKWA_RUNTIME variable). The BESDK diskette has a separate installation process, refer to Chapter 11 for details.

NOTE: **Execution of any NPL programs supplied with the Niakwa Development Package requires the Niakwa NPL RunTime Package to be installed on the host system. The installation of the Niakwa NPL RunTime Package is discussed in Section 2.6.**

2.4.1 Installing the Development Software

The following installation instructions assume that A: is the floppy drive and that drive C: is the hard drive. It is also assumed that the files will be copied to the C:\BASIC2C directory. If the drive designations are different on the system where the NPL development software is being installed, be sure to use the proper designations.

To install the Niakwa NPL Development Package's Software follow these steps:

1. If the C:\BASIC2C directory does not exist, it must be created. Enter the following to create the directory:

```
MD C:\BASIC2C
```

2. Insert the diskette labelled "Compiler Diskette" into drive A and enter:

```
COPY A:\BASIC2C\*.*  C:\BASIC2C
```

3. Repeat step 2 for the Utilities Diskette and Terminal Support Files Diskette.

2.4.2 Installing the BESDK

The NPL BESDK diskette has a separate installation procedure. Refer to Chapter 11, in this Supplement for the installation procedures for the BESDK diskette (NPL, formally Basic-2C) External Subroutine Development Kit).

2.5 NPL Runtime Security

The following sections discuss the NPL Gold Key security component of the NPL security and how it is used with the NPL RunTime Package.

2.5.1 Security Overview

The RunTime programs are protected from unauthorized copying by use of a security scheme. The Gold Key diskette issued by Niakwa contains a physically encoded serial number. The RunTime Package contained on the Gold Key and the other associated RunTime diskette(s) (if necessary) contains the same encrypted serial number.

The protection being used is an "install based security" by which the RunTime Program can be copied to a single disk drive. With this type of security the RunTime does not require insertion of the Gold Key diskette. The RunTime Program can be "installed" and "recalled" to allow changing systems.

NOTE: Once the RunTime is installed the Gold Key diskette is used as a backup in case the installed security on the hard drive is lost.

2.5.2 NIAKSER.DAT Files

At start-up time, the RunTime attempts to read the NIAKSER.DAT file and extract the encrypted "Serial Number" from it. To find the NIAKSER.DAT file the RunTime first looks at the directory pointed to by the NIAKWA_RUNTIME variable. It then looks at the current directory, and finally looks at the \BASIC2C directory on the current drive. Refer to Section 2.5.6 for details on the NIAKWA_RUNTIME environment variable.

The RunTime program fails with an appropriate error message whenever NIAK-SER.DAT cannot be read, found, or contains invalid data. Any of the following error messages may be displayed:

- NIAKSER.DAT missing or damaged (followed by an error code).
- RunTime program cancelled.
- Contact the distributor for assistance.

where the numeric code generated can be one of the following:

Error Code	**Description**
Code 0	Cannot find NIAKSER.DAT.
Code 1	Cannot read NIAKSER.DAT.
Codes 2-5	Invalid data in NIAKSER.DAT. File is probably damaged.

In addition, the Runtime generates an error message if the internal revision level in the Runtime does not match the revision level in NIAKSER.DAT.

NOTE: If it is necessary to contact Niakwa Technical Support regarding one of the above errors, please be sure to note the exact error code received.

2.5.3 Gold Key Security TSR Programs

The RunTime security routines require a TSR program to be loaded. This is done automatically by the RunTime. Since the TSR requires availability of a free interrupt, six different TSRs are provided (each using a different interrupt). These are the files NIAKSECA.COM-NIAKSECF.COM on the RunTime diskette. The RunTime automatically determines which interrupt is available on the system and automatically loads the proper TSR. To find these files, the RunTime, first looks in the current directory, then the NIAKWA_RUNTIME directory, and finally the /BASIC directory on the current drive.

NOTE: The RunTime automatically unloads the security TSR after the security check and reuses the memory used by the TSR. Thus, this method has no adverse effect on memory availability.

2.5.4 Security Check

Based on the "SERIAL NUMBER" extracted from NIAKSER.DAT, the RunTime security check then takes place. The security check is comprised of the following steps:

1. If the NPL_SECURITY variable is set to a drive letter (A-Z), then only that drive is used for the security check. NPL does not check the current drive, the NIAKWA_RUNTIME directory, or drive C in this case. If security fails on the indicated drive, the "Mount Gold Key" screen is displayed. If this screen displays, refer to step #5 below.

2. If the NPL_SECURITY variable is not set, then the current drive is checked first for hard disk protection. If hard disk protection is found, execution continues.

3. If the hard disk protection check fails on the current drive, the drive specified in the NIAKWA_RUNTIME environment variable is checked for hard disk protection. If hard disk protection is found, execution continues.

4. If the hard disk protection check fails on the drive specified by the NIAKWA_RUNTIME environment variable, the C: drive is checked for hard disk protection. If hard disk protection is found on the C: drive, execution continues.

5. If the hard disk protection check fails, the "Mount Gold Key" screen displays. The security check can then be performed from any available drive (A-Z) containing the protection. The default drive is A:. The security check is then performed on the specified drive. If protection is found, execution continues. Refer to Section 2.5.9 for details on the diskette-based security check.

6. If the security check fails, the "Mount Gold Key" screen is redisplayed.

2.5.5 NPL_SECURITY Environment Variable

Niakwa has implemented a new environment variable, NPL_SECURITY, that allows the user to specify the drive location of the security files (the files installed with NIAK-INST). The purpose of this variable is to make installations simpler on systems that use compressed drives (MS-DOS 6.0's DBLSPACE, Stacker, etc.). This resolves the potential for losing security due to dynamic reallocation of the compressed drive by the controlling compression program by allowing the security files to be placed on the non-compressed drive.

WARNING--If installed on a compressed drive, manual changes to the size of the compressed drive or the compression ratio can still damage the installed security just as it would on a non-compressed drive. Consequently, the security should be recalled prior to making these manual changes and reinstalled once the changes have taken place.

This environment variable is in addition to the NIAKWA_RUNTIME environment variable which is intended to supply the location of all other NPL supporting files. Refer to Section 2.5.6 for details on the NIAKWA_RUNTIME environment variable.

If the NPL_SECURITY environment variable is not used, the RunTime will refer to the current drive, the NIAKWA_RUNTIME setting, and finally drive C:.

A typical MS-DOS installation that makes use of a compressed drive (C:) where the user wants to locate the security on the uncompressed drive (D:), would have the following environment variables set:

```
SET NPL_SECURITY=D:
SET NIAKWA_RUNTIME=C:\BASIC2C
```

To establish the NIAKWA_SECURITY directory, use the MS-DOS SET command. This is best placed in the AUTOEXEC.BAT file.

For example:

```
SET NPL_SECURITY=D:
```

instructs the RunTime to look at drive D: for the NPL security files.

Refer to Section 2.6 in this Supplement for details on installing the RunTime security when using the NPL_SECURITY environmental variable.

2.5.6 NIAKWA_RUNTIME Environment Variable

The RunTime supports using an environment variable to specify the drive/directory in which the RunTime system files are located. These system files include:

- The NIAKSER.DAT file
- The NIAKSECA-F.COM files
- The ENABLED file
- Font files for EGA and Hercules monitors
- Screen and keyboard translation files
- Error message files
- Patch files
- TERMINAL.TBL

The search path used by the RunTime, is:

1. NIAKWA_RUNTIME drive/directory
2. Current directory on the current drive (except Patch files)
3. \BASIC2C on the current drive

To establish the NIAKWA_RUNTIME directory, use the MS-DOS SET command. This is best placed in the AUTOEXEC.BAT file. For example:

```
SET NIAKWA_RUNTIME=C:\NPL
```

Instructs the RunTime to look for system files in the C:\NPL directory.

```
SET NIAKWA_RUNTIME=D:\BASIC2C
```

instructs the RunTime to look in the \BASIC2C directory on drive D for system files despite the current drive selection. The ability to specify a drive designation as part of the NIAKWA_RUNTIME environment variable is particularly useful on multi-drive installations.

2.5.7 Security Screen

In the event that the Gold Key "Security Fingerprint" is not installed on the system's hard drive or the hard drive-based security check fails, normal diskette-based security procedures must be performed.

In this event, the following prompt is displayed:

```
PLEASE MOUNT YOUR GOLD KEY DISKETTE
```

NOTE: This prompt appears whether or not the diskette is in the drive:

When this occurs, mount the original Gold Key diskette with its matching "Serial Number" in the diskette drive. If the "Serial Number" is not found on the diskette or if the "Serial Number" is found but it does not match, the security check fails and the message "Please mount your Gold Key diskette" remains on the screen.

NOTE: If the Gold Key is required, it must be the original Gold Key containing the "Security Fingerprint". For example, if a 3.20 RunTime is installed and upgraded to 4.00, the 3.20 Gold Key is required to pass the diskette-based security check (the Upgrade RunTime diskettes contain no security fingerprint that can be used to pass the RunTime security check).

In addition to the prompt, the screen displays the drive number (default is "A") where the Gold Key diskette is expected. The user may change the drive designation if desired and then enter RETURN to proceed or ESC to exit.

When performed, the security check always takes place at the beginning of the RunTime program. Once the security check has been completed, the Gold Key diskette may be removed and returned to a safe place.

2.5.8 What to Do if the Gold Key Security is Lost

If the security check fails on the hard drive, the installed Gold Key security may have been lost. The first step is to check that the RunTime program has been installed. This can be done by trying to reinstall the RunTime program. This installation indicates whether the RunTime still has an install count. If the count is still one, continue with the installation. Refer to Appendix A for more details.

2.5.9 Using the Gold Key Diskette to Pass Security

If the installed Gold Key security is lost, the Gold Key is used as a backup. When the security check fails on the hard drive, the system prompts the user to mount the Gold Key to pass the security check. At this point use the Gold Key to pass security every time the RunTime program is executed.

2.5.10 Field Level Resets

If the Gold Key security becomes lost from the hard drive, Niakwa can provide a remote "reset" of the Gold Key to allow the Niakwa security to be installed again. Resets can only be performed on new Revision 3.20 and 4.00 RunTimes. RunTimes with Gold Key revisions prior to 3.20 cannot be reset.

NOTE: The RunTime can be executed normally, using the Gold Key as a backup until a reset is performed.

Reset Procedure

The following procedure provides a field-level reset.

1. Make sure the RunTime meets the criteria for a field-level replacement of the Gold Key security. This is only possible with new Revision 3.20 or 4.00 MS-DOS installations, not upgrades. Versions can be distinguished by having the user review the Gold Key diskette for the revision number--this must be 3.20 or 4.00--and for the presence of a serial number (only new Revision 3.20 and 4.00 feature a serial number).

2. Have the end-user insert the Gold Key diskette into the floppy drive of the failed system and close the drive door.

3. While at a DOS prompt, enter:

   ```
   A:\BASIC2C\RESET
   ```

4. Have the end-user enter today’s date as displayed and press enter.

5. Have the end-user record the information contained after "Serial Number:" (i.e., 12345-2). This information is all that is necessary for Niakwa to issue a new Gold Key security install.

6. Have the end-user press ESC. This returns the system to the DOS prompt so the user can continue working until the reset code is provided.

7. Contact Niakwa and ask for a field-level reset of the Gold Key. Several questions are asked, including the "Serial Number" information (recorded earlier). Upon approval of a field-level reset, a reset code is issued that allows the reinstallation of the RunTime's Gold Key security.

NOTE: This reset code can be used only once.

8. Contact the end-user. Have the end-user get to the DOS prompt and enter:

```
A:\BASIC2C\RESET
```

9. Have the end-user enter today's date as displayed and press enter.

10. Have the end-user enter the reset code obtained from Niakwa. This must be entered exactly as provided by Niakwa. When the reset code is entered and confirmed, have the end-user press ENTER. The following message is then displayed:

```
Install-Count Reset Successful
```

11. Then have the end-user follow the standard Gold Key security installation procedure as documented in Section 2.6 below for details.

NOTE: The Niakwa RunTime files do not need to be copied again.

2.5.11 Replacement RunTimes

If Gold Key security becomes lost and a reset cannot be performed, a replacement RunTime can be ordered from Niakwa. Contact Niakwa to order a replacement RunTime.

The RunTime can be executed normally, using the Gold Key as a backup until the replacement arrives.

2.6 Installing the NPL Runtime Package

The following sections describe the steps necessary to install the NPL RunTime Package.

2.6.1 CONFIG.SYS

Modifications should be made to the CONFIG.SYS system file for proper operation of the RunTime. Refer to Section 2.3 for details.

2.6.2 AUTOEXEC.BAT

Modifications should be made to the AUTOEXEC.BAT system file for proper operation of the RunTime. Refer to Section 2.3 for details.

2.6.3 NPL_SECURITY Environment Variable

The RunTime supports the use of an environment variable to specify the locations of the NPL security files. Refer to Section 2.5.5 for more details.

2.6.4 NIAKWA_RUNTIME Environment Variable

The RunTime supports the use of an environment variable to specify the location of the NPL files. Refer to Section 2.5.6 for more information.

2.6.5 Creating the NPL Directory

The first step in installing the NPL RunTime Package is to create a directory on the hard drive for the RunTime Package software. For the purposes of this documentation it is assumed that the RunTime package software will be placed in the C:\BASIC2C directory.

NOTE: **If a directory other than the C:\BASIC2C directory is used to install the RunTime, it is necessary to use the NIAKWA_RUNTIME environmental variable. Refer to Section 2.5.6 for more details.**

To create the C:\BASIC2C directory enter:

From the DOS prompt, enter:

```
CD\
MD\BASIC2C
```

Refer to the MS-DOS manuals for details on the use of these commands.

2.6.6 Copying the Software to the Hard Disk

The second step involves copying the RunTime Package software from the \BASIC2C directory on the RunTime diskettes to the \BASIC2C directory on the hard drive.

1. From the MS-DOS prompt, insert the Gold Key diskette and enter:

   ```
   COPY A:\BASIC2C\*.* C:\BASIC2C\*.*
   ```

   All files are now copied to the hard drive.

2. From the MS-DOS prompt, insert the second diskette labeled disk (2 of 2) and enter:

   ```
   COPY A:\BASIC2C\*.* C:\BASIC2C\*.*
   ```

NOTE: The second disk is only present when using 5-1/4" media.

2.6.7 Installing the Gold Key Security

The next step is to install the Gold Key security to the host system's hard drive.

1. If the Niakwa RunTime files were copied to any directory other than C:\BASIC2C, make sure the NIAKWA_RUNTIME environment variable is set as discussed in Section 2.5.6.

2. Select the diskette drive where the Gold Key is mounted by entering:

   ```
   A:
   ```

3. Then enter:

   ```
   CD \BASIC2C
   NIAKINST A: C:
   ```

   The following message appears:

   ```
   Current Available Install Count = 1 - Continue (Y/N)? Y
   ```

   Enter Y.

   Upon successful installation, the following message appears:

   ```
   RunTime Program Successfully Installed on Drive C
   ```

   If any problems occur during the installation, refer to Appendix A for help.

NOTE: The NIAKINST must be executed from the Gold Key diskette.

The Niakwa RunTime is now installed on the hard drive. Remove the Gold Key from the diskette drive and place it in a secure location, along with the other RunTime diskettes, if any.

The NIAKINST program creates, if it does not already exist, a \LOGIN directory off the root directory of the specified hard disk.

NOTE: **If the NPL_SECURITY environment variable is used, the \LOGIN directory is created of the drive specified by the variable. Do not remove the \LOGIN directory.**

The Gold Key diskette is to be used as a backup in case the file server protection is lost. Store this in a safe place! Refer to Section 2.5 for details on security.

WARNING--The security fingerprint files (they are hidden files) are placed in the \LOGIN directory and must not be relocated or have their attributes modified.

2.6.8 Recalling the Gold Key Security

The following conditions may require recalling the Gold Key security.

- The current hard drive must be formatted or replaced.
- The RunTime program must be installed on a different system.
- The RunTime program must be secured from being used.

To recall the Gold Key security on the hard drive, place the original Gold Key diskette in the diskette drive and follow the steps shown below.

1. If the Niakwa RunTime files have been copied to any directory other than C:\BASIC2C, make sure the NIAKWA_RUNTIME environment variable is set as discussed in Section 2.5.6.

2. If the NPL_SECURITY environment variable is being used make sure it is set as discussed in Section 2.5.5.

Select the diskette drive where the Gold Key diskette is mounted by entering:

```
A:
```

3. Enter

```
CD \BASIC2C
NIAKRCLL C: A:
```

NIAKRCLL generates the following message upon successfully recalling the RunTime program:

```
RunTime Program successfully recalled to original diskette.
```

The RunTime program is now recalled from the hard drive and can be installed on another system.

2.7 Installing the Upgrade RunTime Package

A special Upgrade RunTime Package is available. This Upgrade RunTime Package is designed so that when it is installed, it is tied to the Gold Key security and serial number of the existing RunTime that is currently installed on the host system's hard drive. This Upgrade RunTime eliminates the need for the old RunTime to be returned to Niakwa. In addition, use of the Upgrade RunTime provides the same serial number and #GOLDKEY value as the current RunTime in use.

The Upgrade RunTime is similar to the Standard RunTime, except for the following:

- The Upgrade RunTime diskettes do not contain the Gold Key security used to pass the RunTime security check.
- The Upgrade RunTime diskette does not contain a NIAKSER.DAT file. This file is created or updated during the upgrade procedure.

The concept behind the Upgrade RunTime is that the serial number from the currently installed RunTime is extracted by a special upgrade program. This "Serial Number" is then used by the new RunTime to pass the security check.

2.7.1 Requirements for Upgrading

For the upgrade procedure to complete successfully, the following items are required:

- The system being upgraded must have a current 3.xx or greater version of the NPL (formerly Basic-2C) RunTime installed (the Gold Key security from this RunTime must be installed on the hard drive).
- The Upgrade RunTime Package number of users must match the currently installed RunTime's number of users.

- Either RTP.EXE or RTI.EXE must be present and executable (RTP.EXE is used if both RTP and RTI are present).
- All files from the Upgrade RunTime diskettes must be copied to the directory where the existing RunTime files are located (typically C:\BASIC2C).
- Select the \BASIC2C directory or the NIAKWA_RUNTIME environment variable directory if not set to \BASIC2C.
- The currently selected drive and directory must be where the existing RunTime files are located (typically C:\BASIC2C).
- There must be sufficient memory to execute $SHELL from within the current RunTime.
- The Upgrade RunTime Package must match a valid upgrade option, as shown below, or the upgrade procedure will abort.

The following is a complete list of all NPL RunTime "no-return" upgrade options based upon existing RunTime installation type, Gold Key revision, and RunTime revision (can be different from the Gold Key revision if the RunTime was upgraded or updated).
<u>**THESE ARE THE ONLY "NO-RETURN" UPGRADE OPTIONS!**</u>

Upgrade "FROM" Options			**Upgrade "TO" Options**			
Gold Key Revision	**RunTime Revision**	**RunTime Type**	**MS-DOS/ Novell NetWare**	**MS-Windows**	**386/DOS-Extender**	**MS-Windows/ 386/DOS-Extender**
3.20	3.2x	MS-DOS/ Novell NetWare	X	X	X	X
3.20	3.2x	MS-Windows	NA	X	NA	X
2.01 or 3.00	3.xx	MS-DOS/ Novell NetWare	X	NA	NA	NA

where: X Valid
NA Not Available

NOTE: The Gold Key revision number appears on the original Gold Key diskette label.

The RunTime revision appears on the NPL start-up screen when the RunTime is initially executed.

The RunTime Type appears on the original Gold Key diskette label.

This information can be easily obtained from the end-user's site by making use of the UPGRTEST program Niakwa provides on the NPL Development Package Compiler diskette.

2.7.2 How the Upgrade Works

The Upgrade program is used to produce the required NIAKSER.DAT file for the new version of the RunTime being installed. The following steps are performed by the upgrade program:

1. The Upgrade program performs a series of tests to determine if the upgrade should proceed. Various self-explanatory error messages may be generated during this phase. The upgrade procedure is aborted if an error occurs during this phase. Once the error condition is corrected, the Upgrade program can be rerun.

2. All required disk operations are attempted (to ensure access to files, sufficient space on the hard disk, etc.). Again, self-explanatory error messages may be generated during this phase. The upgrade procedure is aborted if an error occurs during this phase. Once the error condition is corrected, the Upgrade program can be rerun.

3. A security check is performed on the upgrade diskette. The procedure is aborted with an error message if the security check fails.

4. Once the above test are done the Upgrade program updates or creates NIAKSER.DAT.

5. The Upgrade Procedure then:

   - Creates a backup copy of NIAKSER.DAT on the upgrade diskette.

   - Deletes RTPOLD.EXE and RTIOLD.EXE if they exist.

- Renames the existing RunTime programs, RTP.EXE and RTI.EXE to RTPOLD.EXE and RTIOLD.EXE.
- Renames the Upgrade RunTime programs, RTPUPG.EXE and RTIUPG.EXE to RTP.EXE and RTI.EXE.
- Exits with a message of successful completion.

NOTE: All error messages produced by the upgrade program are contained in the batch file UPGRADE.BAT. These error messages may be translated for use by non-English users.

There are two possible error conditions which would require extra operator attention:

- If either no diskette or a non-DOS diskette is mounted in the drive specified by the operator, some PCs display the Device Not Ready Screen message. At this point, the operator should select the Kill RunTime option. Then a normal error message is generated and the upgrade procedure is aborted.
- If $SHELL cannot be executed due to insufficient memory or an invalid COMSPEC, a non-recoverable A01-Memory Overflow error is generated. If RTP is being used, this results in the RunTime Error Screen being generated. At this point the operator must select the Kill RunTime Option. Then control returns to the batch file and a normal error message is generated and the procedure is aborted. If RTI is in use, an immediate mode A01-Memory Overflow occurs. At this point, the operator should press HELP and select the Kill RunTime Option. Then control returns to the batch file and a normal error message is generated and the procedure is aborted. However, if a $END statement is entered, the batch procedure does not recognize that an error has occurred and attempts to rename RTI and RTP files as described in step 5 above. If this occurs, the files must be manually renamed back to the original (pre-upgrade procedure) names.

2.7.3 Performing the Upgrade

The following installation instructions assume that "A:" is the floppy drive and "C:\BASIC2C" is the target directory. If the designations are different on the system being used, be sure to use the correct designations instead.

To perform the upgrade, the following steps must be performed:

1. Insert the diskette labeled Niakwa NPL Upgrade RunTime Package (Disk 1 of X) into diskette drive A and enter:

   ```
   COPY A:\BASIC2C\*.* C:\BASIC2C
   ```

   Repeat step 1 for all additional upgrade diskettes, if present.

   This copies the Upgrade RunTime software required to perform the upgrade, to the existing \BASIC2C directory on the hard drive. This must be done before running the upgrade procedure (UPGRADE.BAT).

2. Select the \BASIC2C directory on the hard drive.

   ```
   CD \BASIC2C
   ```

3 Execute the upgrade program (UPGRADE.BAT):

WARNING--Once the upgrade procedure is complete, the Release IV RunTime is permanently tied to the original Gold Key diskette. The original Gold Key must be saved. Both this diskette and the Upgrade RunTime diskette(s) should be stored together in a safe location as an emergency backup.

Do not write-protect the upgrade diskette(s).

Only the original upgrade diskette(s) may be used to perform the upgrade.

Once the upgrade procedure is successfully completed, it cannot be executed again.

```
UPGRADE [DRIVE]
```

where [drive] is the diskette drive in which the Upgrade RunTime Disk 1 of 1 is present.

For example:

```
UPGRADE A:
```

Upon completion of the "upgrade" procedure the following message is displayed:

```
"Upgrade was completed successfully"
```

The Upgrade RunTime Package is now installed. Execution of RTP (or RTI if enabled) causes the new RunTime to be executed.

Refer to Section 2.7.2 for details on the operation of the "upgrade" procedure.

NOTE: If an error occurs, please make a note of the error code before calling Niakwa.

The upgrade procedure can only be run once. Once the upgrade has been successfully completed, it no longer executes. However, should the upgrade procedure be terminated due to some error condition, the execute count is not decremented and the upgrade procedure may be rerun once the error condition is corrected.

Once the upgrade procedure is complete, the NIAKSER.DAT file produced during the Upgrade procedure is automatically backed up to the upgrade diskette. The Upgrade RunTime is now permanently tied to the security fingerprint of the original Gold Key (NPL Revision 3.00 or higher).

2.7.4 Recalling the Upgraded RunTime Package

If any of the following conditions arises, it is necessary to recall the upgraded RunTime from the hard drive:

- The current hard disk must be formatted or replaced.
- The RunTime program must be installed on a different system or drive.
- The RunTime program must be secured from being used.
- The operating system is being updated.
- Installation of a driver-based compression product such as MS-DOS's DoubleSpace or Stac's Stacker.

To recall the Upgrade RunTime, the installed Gold Key security of the original Gold Key that is currently installed on the hard drive must be recalled. There are no special steps to be performed with the actual Upgrade Diskettes. Refer to Section 2.6.8 for details on recalling the Gold Key security from the host system's hard drive.

2.7.5 Reinstalling the Upgrade RunTime Package

The following steps should be performed to reinstall the Upgrade RunTime after it has been recalled for any reason.

1. If the Niakwa Gold Key security files are to be placed on a drive other than the one where the RunTime files are to be located, make sure the NPL_SECURITY environment variable is set as discussed in Section 2.5.5.

2. If the Niakwa RunTime files are to be copied to any directory other than \BASIC2C or the security files are placed on a different drive (refer to step 1), make sure the NIAKWA_RUNTIME environment variable is set as discussed in Section 2.5.6.

3. Copy the original Gold Key to the directory which contains the NPL files.

4. Install the Gold Key security from the original Gold Key to the hard drive. Refer to Section 2.6 for details.

WARNING--Do not install the Gold Key security on a hard drive on which data has been compressed using a driver-based compression product such as MS-DOS's DoubleSpace or Stac's Stacker.

5. Select the \BASIC2C directory or the directory to which the NIAKWA_RUNTIME variable is set.

6. Mount the latest RunTime Upgrade Diskette and copy all files from the RunTime Upgrade diskettes.

NOTE: This now includes the NIAKSER.DAT file created by the Upgrade procedure.

7. Rename RTP.EXE to RTPOLD.EXE.

6. Rename RTI.EXE to RTIOLD.EXE.

7. Rename RTPUPG.EXE to RTP.EXE.

8. Rename RTIUPG.EXE to RTI.EXE.

The upgraded RunTime is now reinstalled and operational.

CHAPTER 3

CONFIGURATION

3.1 Overview

NPL applications consist of one or more diskimage files containing compiled NPL programs and data files, at least one startup "BOOT" program, and possibly other associated files. The application software developer must decide where to place these applications within the MS-DOS file system and how to access them from NPL. The method of setting up NPL applications to operate in an MS-DOS file system on the IBM PC is reviewed in this chapter.

Section 3.2 discusses the MS-DOS file system.

Section 3.3 discusses the location of all Niakwa NPL software.

Section 3.4 discusses the location of application programs and data.

Section 3.5 discusses the various NPL auxiliary files and their functions.

Section 3.6 discusses configuring additional users for operation within the system.

Section 3.7 discusses configuring the user's workstations.

Section 3.8 discusses required access privileges.

Section 3.9 discusses setting up a batch file to invoke the RunTime Package.

Section 3.10 discusses using a menu system to invoke the RunTime Package.

3.2 MS-DOS's File System

MS-DOS uses a hierarchical file system to organize its own system files. Individual directories are used to store the various components of the MS-DOS operating system.

HINT: It is recommended that only the files and programs relative to the MS-DOS operating system reside in the root directory. All other software, including both NPL and the application software, should be stored in subdirectories of the root directory.

By establishing a series of subdirectories within the root directory, multiple application systems can be organized, allowing for ease of operation (including separate backups of individual systems or directories).

3.3 Location of NPL Software

By default, all Niakwa software, including the Development and RunTime Packages, are installed so that they are located in the C:\BASIC2C directory. Other drives and directories may be used, provided that the NIAKWA_RUNTIME environment variable is set as discussed in Chapter 2.

Refer to Chapter 2 for details on copying the Development Package and RunTime Package diskettes.

3.4 Installation of Application Software

The steps described in this section are for installing an NPL application at an end-user site (as opposed to a development system) under MS-DOS on a PC. After installing the NPL RunTime program (refer to Chapter 2 for details), the application software can be installed.

NOTE: **When porting an existing system from a Wang 2200 or another supported NPL environment, refer to Chapter 10 for a discussion of porting software to the IBM PC.**

If the application is already in MS-DOS format, perform the following steps:

1. Create the directories in which the application program and data files are to be installed (refer below).
2. Copy the application files into their appropriate directories.
3. Modify the $DEVICE statements in the "BOOT.OBJ" file to reflect the locations of the application program and data files (refer below).

The applications should now be installed and ready to run.

NOTE: **When porting an application directly from a Wang 2200, the program files must be compiled to operate under NPL. Refer to Chapter 9 for details on compiling.**

Locating Application Programs and Data

As explained in Section 3.2, all application programs and data should be located in separate subdirectories of the drive's root directory.

There are several reasons why application software should be segregated from the NPL software, these include:

- For the purposes of backing up only those data files relevant to an individual application system.
- To avoid the frustration of trying to locate one particular file on a directory which is intermixed with several other application files.

For example, when installing an accounts payable system and an accounts receivable system, the directory structure containing these two application systems might appear as:

\AP	Contains the accounts payable main directory, with two sub-directories named PROGS and DATA.
\AP\PROGS	Contains the AP subdirectory PROGS with all of the AP application programs.
\AP\DATA	Contains the AP subdirectory DATA with all of the AP application data files.
\AR	Contains the accounts receivable main directory, with two subdirectories named PROGS and DATA.
\AR\PROGS	Contains the AR subdirectory PROGS with all of the AP application programs.
\AR\DATA	Contains the AR subdirectory DATA with all of the AP application data files.

3.5 Auxiliary Files

The NPL Development Package contains a series of auxiliary files that NPL uses to address a variety of functions. This section provides a detailed discussion of these files.

Auxiliary files required for the end-user installation may be copied directly from the appropriate NPL Development Package diskette by entering the following MS-DOS commands (this assumes C:\BASIC2C is the directory on the host system):

```
COPY A:\BASIC2C\FILENAME C:\BASIC2C
```

Alternatively, auxiliary files or a modified version of the auxiliary files can be copied from the development system to a diskette and then to the end-user's system by entering the following MS-DOS commands:

On the development system:

```
COPY C:\BASIC2C\FILENAME A:
```

On the end-user system:

```
COPY A:*.* C:\BASIC2C
```

3.5.1 The "ENABLED" File

For the Interpretive RunTime (RTI) to operate, it is necessary to install a special file called "ENABLED". This file is located on the NPL Compiler diskette and must be copied into the \BASIC2C or appropriate NPL files directory of the host system's hard drive.

If the Interpretive RunTime is invoked without the presence of the ENABLED file in the current, NIAKWA_RUNTIME set, or C:\BASIC2C directory, the message:

```
"Interpreter not enabled"
```

appears, and RunTime execution is canceled.

Installing ENABLED in the end user's C:\BASIC2C or appropriate NPL files directory, enables the Interpreter for all applications.

HINT: It is also possible to install the ENABLED file only in the specific application's directory. This allows the Interpreter to be used with some applications while preventing its use with other applications.

NOTE: Use of the Interpreter allows the end-user access to several functions which, if used improperly, could be damaging. These include the RESET and STEP functions of the HELP processor and the HALT key. Refer to Chapter 11 of the NPL Programmer's Guide for more information on the HELP processor. These functions can be suppressed under program control by use of the $OPTIONS system variable. For a detailed discussion of the $OPTIONS system variable, refer to the NPL Statements Guide, $OPTIONS.

3.5.2 Keyboard Files

The IBM PC keyboard is not completely compatible with the NPL character set. These keyboard differences are resolved by the use of a simple look-up table which translates keys received from the keyboard to HEX codes expected by NPL programs. The standard built-in defaults for keyboard remapping are present within the RunTime and should prove adequate for most applications.

Should modifications be required, they can be made by use of the NPL Keyboard Translation Tables Editor. This utility creates a disk file named KEYBOARD.TBL. The RunTime program first searches for the KEYBOARD.TBL file in the currently selected directory when executed. If the file is not located, the RunTime then searches the NIAKWA_RUNTIME set directory, followed by the default C:\BASIC2C directory. If the file is not found, the built-in default values are used for keyboard translation.

The NPL Development Package also provides a series of KEYBOARD.XXX files. These files are used by the RunTime in conjunction with the NPL REDIRECT feature of the HELP facility, which allows for a variety of terminals to be used for remote communication and support with the end-user’s site. These files are located on the NPL Terminal Support diskette.

To modify any of the default keyboard translation tables:

1. Create the modified KEYBOARD.xxx file(s) on the development system.

2. Copy the file(s) to a diskette (refer to Section 3.5 above for details).

3. Copy the file(s) to the end-user’s system as part of the standard application installation procedure (refer to Section 3.5 above for details).

Refer to the NPL Programmer’s Guide, Chapter 13, for details on the use of the NPL Keyboard Translation Tables Editor utility. Refer to Chapter 6 of this Supplement and Chapter 2 of the NPL Programmer’s Guide for details on the Redirect feature. Also, refer to Chapter 6 of this Supplement for details of the IBM PC keyboard characteristics under NPL.

STOP *WARNING--If reinstallation of the Terminals Support Files diskette is necessary (for new versions), any changes made to the above files are overwritten.*

3.5.3 Screen and Font Files

The IBM PC screen is not completely compatible with the NPL character set. Screen character translation is achieved through one of two ways (Standard Mode or Graphics Mode) depending on the video mode on which the RunTime is executed.

When the RunTime is executed in the Standard Mode (non-graphics, no /G), character set compatibility is achieved through the use of a simple lookup table, SCREEN.TBL. This file translates various characters into hex codes which generates an equivalent, or at least similar, display character. This table may be dynamically modified by using the $SCREEN statement or it can be more permanently modified by use of the NPL Screen Translation Tables Editor utility.

When the RunTime is executed in the Graphics Mode (/G) character set compatibility is achieved through the use of a downloadable font file. There are two font files that can be used by the RunTime, IBMFONT0.EGA for Graphics Mode on a EGA video controller or IBMFONT0.HGA for Graphics Mode on a Hercules Graphic controller. Each file contains the Wang 2200 character set. If the font file is not located, it is not loaded and the system reverts to Standard Mode. Should modifications of the IBMFONT0.EGA or IBMFONT0.HGA font files be required, they can be made by use of the NPL Font Table Editor.

Also provided is a series of XXFONT.xxx and SCREEN.xxx files. These files are used by the RunTime in conjunction with the NPL Redirect feature, which provides for a variety of terminals that may be used for remote communication to the end-user’s site.

To modify any of the default font or screen translation tables:

1. Create the modified FONT or SCREEN.xxx file(s) on the development system.

2. Copy the file(s) to a diskette.

3. Copy the file(s) to the end-user’s system as part of the standard application installation procedure.

Refer to the NPL Statements Guide, $SCREEN, for details on $SCREEN. Refer to Chapter 13 of the Programmer’s Guide for details on the Screen Translation Tables Editor.

Refer to Chapter 6 for details of the PC screen characteristics under NPL. For a complete discussion of the NPL Utilities, the Redirect feature or NPL supported terminals, refer to the NPL Programmer’s Guide Chapters 2, 7 and 13, respectively.

STOP

WARNING--If reinstallation of the Terminals Support Files diskette is necessary (for new versions), any changes made to the above files are overwritten.

3.5.4 Printer Control Values

Since printer control protocol standards are virtually non-existent, printer control specifications vary dramatically from printer to printer. The print control feature of the Run-Time program has built-in default control codes (refer to Section 4.6 for actual default values) which should work well with IBM PC Graphics Printer, Epson MX-80, or equivalent.

If other types of printers are in use on the system, use the NPL Edit Printer Control Codes utility to define a PRINTCLR.TBL file of printer control values so that the functions listed on the Printer Control screen operate correctly.

When the RunTime program is executed, it looks for the PRINTCLR.TBL file in the currently selected directory. If the file is not found the RunTime then looks in the NIAKWA_RUNTIME set directory, followed by the default C:\BASIC2C directory. If the file is found, the built-in defaults of the RunTime are used.

Refer to Chapter 13 of the NPL Programmer's Guide for details on the operation of the NPL Edit Printer Control Codes utility.

3.5.5 Error Files

The RunTime Diskette contains four files, labeled ERRORMSG.HLP, ERRORMSG.IDX, RTIIERR.HLP and RTIIERR.IDX, which should be copied to the \BASIC2C or appropriate NPL files directory on the hard drive where the RunTime is installed. The ERRORMSG files are used to display NPL error code messages when an error occurs. The RTIIERR files are used to display the native operating system error code message when an error occurs. The files with the extension of .HLP contain the literal description of each NPL or native operating system error code, while the .IDX files are the indexed help files required to "find" the proper entry in the descriptive files.

The ERRORMSG.HLP and RTIIERR.HLP files can be modified by the NPL developer so that different error descriptions can be displayed. This is particularly useful for distributors of non-English applications. However, if the ERRORMSG.HLP and RTIIERR.HLP files are modified, they must be processed by the NPL Indexed Help File Processor utility (refer to Section 13.16 of the Programmer's Guide for details). Refer to Chapter 11 of the NPL Programmer's Guide, for details on indexed help files.

WARNING--If reinstallation of the RunTime Package is necessary (for new versions or Upgrades), any changes made to the above files are overwritten, since these files are contained on the NPL RunTime Package diskettes.

3.5.6 Additional End-User Security (#GOLDKEY)

The GOLDKEY.OBJ program allows developers to determine the #GOLDKEY number for any Niakwa RunTime based on the Gold Key serial number without having to physically open the RunTime Package.

This program can be found on the NPL Development Package Compiler diskette and, once installed, can be run as any other Niakwa program.

When executed, the GOLDKEY program prompts for the Gold Key serial number as shown below.

```
    Enter Serial Number (1 - 65535) to convert to #GOLDKEY:
```

Once the serial number is entered, the program returns the correct #GOLDKEY code number that is necessary for some application security programs.

3.6 Configuring Additional Users

This feature is not available under DOS.

3.7 Configuring User's Workstations

This function is not necessary under DOS.

3.8 Required Access Privileges

This function is not necessary under DOS.

3.9 Executing the Runtime from a Batch File

Under all versions of MS-DOS, it can be advantageous to invoke the RunTime program from a "batch file". A batch file is an ASCII text file containing a list of commands in the order that they would be entered by the user. This file would typically select the proper directory and then execute the RunTime with the proper "BOOT" program name. This simplifies operations for the end-user since, without a batch file, it would be necessary for every user to remember the directory designations and "BOOT" program name to use.

For example, assume that an NPL application has been set up in directory /AR and that the name of the "BOOT" program is ARBOOT. A typical batch file for executing this application would be:

```
CD /AR
RTP ARBOOT (using the non-interpretive RunTime)
```

(or)

```
RTI ARBOOT (using the interpretive RunTime)
```

NOTE: This example would leave the user's current directory as /AR after exiting the RunTime, until the end of the batch process. This may be undesirable if commands follow the RTP (or RTI) progname command.

For this start-up method to operate correctly, the NIAKWA_RUNTIME environment variable must be set, or the \BASIC2C directory must be established as an alternate search path in the AUTOEXEC.BAT file (this is so that the Command Processor knows where to find the RunTime program). Refer to Sections 2.6.2 and 2.6.3 for details. Refer to the MS-DOS documentation for details on creating batch files.

3.10 Executing the RunTime from a Menu System

Under MS-DOS it is very simple to create batch files or make use of inexpensive third party menu programs which can be used to invoke the RunTime. Predefined menus allow the operator to easily enter the application without having to remember which directory to select, what boot program name to use, etc.

NOTE: Niakwa strongly recommends the use of some form of menu for end-users.

HINT: There are many inexpensive third-party products designed for this use in the MS-DOS environment. In addition, MS-DOS versions 4.0 or higher contain a built-in menu system that can be used. This MS-DOS menu system allows the user to assign passwords and help files, to make it a more "user-friendly" environment. Refer to MS-DOS documentation on using MS-DOS's built-in menu system.

CHAPTER 4

RUNTIME OPERATION

4.1 Overview

This chapter covers the general operation of the NPL RunTime under MS-DOS. The chapter discusses various methods of starting and exiting the RunTime program, the available RunTime startup options and memory considerations, and the default printer control values of NPL.

Section 4.2 discusses the two RunTime programs available.

Section 4.3 discusses the general startup form of the RunTime.

Section 4.4 discusses the use of various startup options available in the RunTime.

Section 4.5 discusses the determination of available memory by the RunTime.

Section 4.6 discusses the default printer control values of the RunTime.

Section 4.7 discusses the various methods of exiting the RunTime under MS-DOS.

4.2 RTP Versus RTI

Included in the RunTime Package are two RunTime programs: One program is non-interpretive, (RTP), and the other interpretive (RTI).

The non-interpretive version allows for execution of NPL object code, without any capabilities for "immediate mode" command entry or functions.

The interpretive RunTime program allows for full development capabilities, such as program text editing and debugging. The interpretive version does require more memory. Refer to Section 2.2 for details on memory requirements.

NPL developers may choose whether or not end-user sites have interpretive capabilities. Installation of the "ENABLED" file is necessary to allow the interpretive RunTime Program to execute. Refer to Section 3.6.1 for details on the "ENABLED" file.

NOTE: **Although interpretive capabilities at the end-user site may be useful for support purposes, these capabilities allow an end-user access to the developer's source code, which may not be desirable.**

Use of the Interpreter at end-user sites requires execution of the End-User Support Only License Agreement. Refer to Section 3.5.1 for details on the "ENABLED" file required for operation of the interpretive RunTime program.

4.3 Starting the NPL RunTime

Several methods are available to begin the execution of either RunTime. This section discusses the RunTime's general startup option form.

4.3.1 Starting From the MS-DOS Prompt

The general form of starting the RunTime from an MS-DOS prompt is as follows:

```
     {RTP} [option] [progname]     (non-interpretive version)

     {RTI} [option] [progname]     (interpretive   version  )
```

where:

```
     option    = RunTime startup option or combination of
                 startup options.

     progname =  the filename of the NPL object program to be
                 executed.  If no progname is specified, the
                 RunTime Program assumes the program
                 "BOOT.OBJ" is to be executed. If an extension
                 is not supplied, NPL assumes an extension of
                 ".OBJ" on the progname (the filename is oper-
                 ating system-dependent).
```

4.3.2 Starting from Batch Files

Batch files can be written to directly execute the RunTime. The batch file can perform the various commands necessary to invoke the RunTime for a specific application.

Refer to Section 3.9 for additional information on starting the RunTime from a batch file.

4.3.3 Starting from a Menu

For a more user-friendly approach to starting an NPL application, the RunTime program can be started from a menu system. Many third-party products can be used for this purpose, including some operating systems' menuing facilities.

Refer to Section 3.10 for additional information on starting the RunTime from a menu system.

4.4 Command Line (Start-up) Options

The NPL RunTime environment is set up internally by the RTP and RTI programs upon execution. The following section discusses the available RunTime startup options which may be specified upon execution of the RunTime. These options allow for modification of the default RunTime environment.

4.4.1 /B (Background Partition)

The /B option is not supported for use under MS-DOS.

4.4.2 /D (DET Entries)

The /D RunTime option allows a programmer to specify the number of device equivalence table entries. The number of DET entries may be a range of 16 to 255. If /D is not specified, the default of 16 DET entries is used.

```
     {RTP} [/B /T=xx]

     {RTI} [/B /T=xx]
```

where:

```
      xx =  a defined SuperDOS port number
```

For example:

```
RTI /D=32
```

specifies that 32 DET entries are available.

Special Considerations for /D Option

When using the /D option, be aware of the following operational considerations.

- Each DET entry above 16 results in the use of additional memory. The amount of memory each additional DET entry requires is 64 bytes. This memory is deducted from the available user partition.

NOTE: This memory requirement may increase with future revisions of NPL.

- On versions of MS-DOS prior to 3.3, the maximum number of open files that may be established by the FILES= statement in the CONFIG.SYS file is 20. On MS-DOS 3.3 or higher, FILES= may be set to the value required.

NOTE : Increasing the open files limit under MS-DOS 3.3 or higher uses 45-50 bytes of additional memory per file. Refer to the MS-DOS documentation for details.

Refer to Section 2.4 of the Programmer's Guide, for a complete discussion of the /D option.

4.4.3 /G (Graphics Mode)

The /G option is used to invoke "true" box graphics under MS-DOS. When invoked, the RunTime attempts to use available true box graphics and user-definable fonts on an IBM-compatible PC system.

```
{RTP} [/G[H,E]] [progname]

{RTI} [/G[H,E]] [progname]
```

The /G may be immediately followed by a letter indicating the type of graphics display to use ("E" for EGA, "H" for Hercules). If no letter is used, then the RunTime first determines if an EGA adapter is present, and uses it if available. If no EGA adapter is found, the RunTime looks for a Hercules-compatible display and uses that. If neither adapter is available, or support files such as font download files are not present, the /G option is ignored.

If the /G option is immediately followed by an "E", only an EGA adapter is used (if available). If the /G option is immediately followed by an "H", only a Hercules adapter is used (if available). This allows for the use of Hercules adapters on systems which have both EGA and Hercules displays (either on separate adapters or combination adapters).

For example:

```
RTI /GE [progname]
```

uses the EGA graphics support, while:

```
RTI /GH [progname]
```

uses the Hercules graphics support.

NOTE: The performance of the RunTime is slower when the /G option is used. Refer to Chapter 6 for details.

Refer to Section 2.4 of the NPL Programmer's Guide and Chapter 6 of this Supplement for a complete discussion of the /G option.

4.4.4 /H (Handle Table Size)

The handle table is an internal table used by the RunTime to translate two byte p-code pointers into four-byte memory addresses. Handle table entries are created at program resolution time. An entry is required for:

Every unique variable
Each unique line number
Each DO/ENDDO group
Each internal DEFFN'
Each loop construct
Each PROCEDURE or FUNCTION

If /H is not specified, the RunTime program allocates a small amount of memory for the handle table and expands it as required.

```
{RTP} [Hk]

{RTI} [Hk}
```

where:

```
k =  the number of entries, in kilobytes.
```

For example:

```
RTP /H16
```

allocates a handle table large enough for 16K entries.

The /H option is valid on the MS-DOS version of NPL.

Refer to Section 2.4 of the NPL Programmer's Guide for a complete discussion of the /H option.

4.4.5 /K (Mouse Support)

The /K option is used under MS-DOS to allow for mouse support under NPL.

```
     {RTP} [/K]

     {RTI} [/K]
```

Refer to Section 7.6 of the NPL Programmer's Guide and Section 5.6 for details on mouse support under MS-DOS.

Refer to Section 2.4 of the NPL Programmer's Guide for a complete discussion of the /K option.

4.4.6 /L (Leave Overhead Memory)

The /L option is not supported for use under MS-DOS.

Refer to Section 2.4 of the NPL Programmer's Guide for a complete discussion of the /L option.

4.4.7 /M (XMS Memory)

The /M option allows the RunTime to use the HMA memory, if available.

```
     {RTP} [/M]

     {RTI} [/M]
```

Refer to Section 8.4 of the MS-DOS Supplement for more information on memory management.

Refer to Section 2.4 of the NPL Programmer's Guide for a complete discussion of the /M option.

4.4.8 /P (Pre-Boot)

The "/P" RunTime option allows for a "pre-boot" configuration program. If the /P option is specified on the RunTime command line, NPL loads and executes a "pre-boot" program.

```
     {RTP}[/Pfilename]

     {RTI}[/Pfilename]
where:
     filename = the filename of the pre-boot program to be
                loaded.
```

For example:

```
RTI /PSTARTUP
```

Refer to Section 2.4 of the NPL Programmer's Guide for a complete discussion of the /P option.

4.4.9 /R (Remote Control)

The /R option is used to force the RunTime to use generic native operating system screen access techniques instead of direct video mapping. This feature is primarily intended to be used in conjunction with third party products for remote control maintenance.

```
     {RTP}[/R]

     {RTI [/R]
```

Refer to Section 2.9 of the NPL Programmer's Guide and Section 6.13 for details on using the NPL Redirect feature.

Refer to Section 2.4 of the NPL Programmer's Guide for a complete discussion of the /R option.

4.4.10 /S (Separate Program Segments)

The /S option performs no operation under MS-DOS.

Refer to Section 2.4 of the NPL Programmer's Guide for a complete discussion of the /S option.

4.4.11 /T (Terminal/Port Number)

The /T option is used for specifying a specific #TERM value.

```
     {RTP} [/T=xx]

     {RTI} [/T=xx]
where:
     xx = a numeric-constant which indicates which terminal
          number to use (overrides default terminal number
          determination).
```

Refer to the Section 2.4 of the NPL Programmer's Guide for a complete discussion of the /T option.

4.4.12 /U (UMB Memory)

NPL supports the optional use of XMS memory on standard MS-DOS environments. Use of XMS memory can result in up to 200K additional memory being available to NPL applications.

```
     {RTP}[/U]

     {RTI}[/U]
```

Refer to Section 8.4 for a complete discussion of memory management under MS-DOS.

Refer to Section 2.4 of the NPL Programmer's Guide for a complete discussion of the /U option.

4.4.13 The /X option

The "/X" option is used to specify that an external subroutine library is to be loaded. The filename for an external library must immediately follow the /X option. An extension of .QLB is assumed.

```
     {RTP}[/Xquicklibrary]

     {RTI}[/Xquicklibrary]
```

where:

```
     quicklibrary = the file name of the quick library
                    module.  No spaces should appear
                    between /X and the library name.
```

Refer to Chapter 11 for additional information on the /X option.

Refer to Section 2.4 of the NPL Programmer’s Guide for a complete discussion of the /X option.

4.4.14 The BOOT Program

The RunTime program assumes that the first program to be executed is not in an NPL diskimage, but rather is a native operating system file. This first program can be thought of as a "boot" program. The boot program generally sets up or customizes the Device Equivalence Table and runs a start-up program in a diskimage.

The RunTime looks to the command line for the name of the BOOT file to load. If no boot file is specified, the RunTime looks in the current directory for a file called BOOT.OBJ. If this file is not found, the RunTime starts, but displays an error message indicating that a BOOT file was not found.

Refer to section 4.3 for an example of the general form of starting the RunTime with a BOOT file.

The last statement in the boot program typically is a LOAD RUN < progname > , which loads the first program to execute in a selected diskimage.

Refer to Section 2.4 of the NPL Programmer’s Guide for more information on the BOOT file.

4.5 Available Memory

Upon execution of the MS-DOS version of NPL RunTime, all unused system memory is considered available to the RunTime. SPACEW will always report the maximum amount of memory available to the application. The RunTime then dynamically allocates this memory as needed by the application. The amount of available memory varies from one system to the next depending on the following:

- The revision of MS-DOS being used.
- The number of files, buffers and device drivers initialized in the system’s CONFIG.SYS file.
- The RunTime program is being executed, RTP or RTI.

- The RunTime options invoked at startup.
- The size of the External Subroutine library, if used.

Refer to Section 8.4 for additional details.

4.6 Default Printer Control

Printer control is a function of the RunTime's HELP processor and may be accessed whenever the HELP processor is active. Refer to Chapter 11 of the NPL Programmer's Guide for details on the HELP processor. This section discusses the default RunTime printer control values. For more information on the use of printer control, refer to Section 3.5.

4.6.1 Default Values

The tables below list the default values built into the RunTime program.

The codes in this table are hex codes, except for single ASCII characters preceded by an equal, "= ", sign.

For example:

```
1B=A181B=2 is the equivalent of 1B41181B32
```

This technique is supported for on-line entry of control codes during execution of print control. This avoids the need to look up the hex codes for ASCII control sequences.

IBM-compatible PC

```
Characters per inch option
    10             1412
    N/A
    N/A
    16             140F

Lines per inch options
    3              1B=A181B=2
    4              1B=A121B=2
    6              1B=AOC1B=2
    8T             1B=A091B=2

Line Feed
    OA

Form Feed
    OC
```

4.7 Exiting from the RunTime Program

There are several methods available to either the programmer or the end-user for exiting the RunTime (and, consequently, the application it is executing). In most cases, exiting the RunTime returns the system to the point at which the RunTime was invoked. That is, if the RunTime was started from a MS-DOS prompt, exiting returns the system there. If the RunTime was started from a menu, exiting returns to the same menu. A discussion of the various exiting methods follows.

4.7.1 Exiting Under Program Control

The RunTime may be exited under program control by any one of the following events:

- When working in the interpretive RunTime, execution of the NPL "END" or "STOP" statement. With either of these statements, a ":" is placed on the screen with the cursor immediately following.

- By program logic which falls through the logical end of the program. In this event, the RunTime automatically exits without further action required. This is only true for the non-interpretive RunTime.

- If an application error condition is encountered, the Non-interpretive RunTime is exited.
- By execution of a $END statement. This statement causes the RunTime program to end operation, returning control to MS-DOS. This is the preferred method of exiting under program control.

4.7.2 Exiting using the HELP Key

The end-user may call for cancellation of program execution using the HELP key. Depression of the HELP key causes current program execution to be suspended, the screen is saved and the HELP screen is displayed, with various options. The end-user may, at this point, select the KILL RunTime option to terminate program execution. If the LEAVE HELP option is selected, the screen is restored and program execution resumes.

The HELP key is active only when the application program is polling for keyboard input (i.e., executing a KEYIN, INPUT or LINPUT). The HELP key with the KILL RunTime option can be thought of as a limited 2200 RESET key. For full details on the HELP key, refer to Chapter 11 of the NPL Programmer's Guide.

4.7.3 Exiting using the Interrupt Key

MS-DOS provides its own methods for terminating program execution. The interrupt key (CONTROL-C) termination method is one of these. Upon depression of the interrupt key, the usual NPL HELP display is provided and program execution may be terminated or resumed in the same methods as described for the HELP key.

NOTE: The interrupt key only takes effect when the application program issues a disk I/O request (i.e., DATALOAD, DATASAVE, DATALOAD DC, etc.). If the interrupt key is pressed when the application is waiting for keyboard input, it is treated as a normal keystroke.

NOTE: The interrupt key can be disabled by the DOS command:

```
BREAK OFF
```

CHAPTER 5

DEVICE SUPPORT

5.1 Overview

This chapter discusses the devices supported under the MS-DOS version of NPL. Included in this discussion are any special requirements or implications for the programming of NPL code under MS-DOS.

Section 5.2 discusses supported diskette devices.

Section 5.3 discusses naming conventions for diskimage files.

Section 5.4 discusses supported monitors/controllers

Section 5.5 discusses printer devices.

Section 5.6 discusses mouse support.

Section 5.7 discusses the use of serial devices.

Section 5.8 discusses the support of a tape drive.

Section 5.9 discusses math co-processor support.

Section 5.10 discusses the default device equivalences.

NOTE: The screen and keyboard characteristics of an IBM-compatible PC under MS-DOS are discussed in Chapter 6.

5.2 Diskette Devices

NPL under MS-DOS has the ability of reading and writing "raw" diskettes in a variety of formats. The table in Section 5.2.1 lists all supported formats..

The 320K raw format is compatible with other NPL versions which support these media types and is fully compatible with a Wang 2275 diskette drive on the Wang 2200. This is the default format, unless specified otherwise in the $DEVICE statement. Refer to Section 5.2.2 for details.

The 360K is a 5-1/4" raw format compatible with other NPL versions which support this media type and is partially compatible with the Wang 2200/CS. Refer to Section 5.2.2 for details.

The 1.2MB is a 5-1/4" raw format compatible with other NPL versions which support this media type.

The 720K, 1.44MB and 2.88MB are 3-1/2" raw format diskettes and are compatible with other NPL versions which support these media types. Refer to Section 5.2.2 for details.

NOTE: Refer to Appendix D, for more information on raw device compatibility between current NPL supported operating environments.

5.2.1 Naming Conventions

The following table lists the naming conventions and revision of the RunTime required for the diskette devices supported under the MS-DOS version of NPL.

Media Size	**Type Format**	**Naming Convention**
5-1/4 inch media	320k	A:
	360k	A: 360= Y
	1.2MB	A: 1.2= Y
3-1/2 inch media	720k	A: 720= Y
	1.44MB	A: 1.4= Y
	2.88MB	A: 2.8= Y

5.2.2 Supported Media

The following section discusses the various NPL supported media.

5-1/4" Media

320K Media

The $DEVICE clause "A:" or "B:" defines a raw diskette as 320K format. This format is defined as 320K formatted capacity, 5-1/4" double-sided, double density diskettes containing 40 tracks per side, 16 sectors per track, and 256 bytes per sector.

For example:

```
0010 $DEVICE(/D10)="A:"  :REM 320k raw media on drive A
```

360K Media

The $DEVICE clause "A: 360= Y" defines a raw diskette as 360K format. This format is defined as 360k formatted capacity, 5-1/4" double-sided, double-density diskettes containing 40 tracks per side, 9 sectors per track, and 512 bytes per sector.

For example:

```
0010 $DEVICE(/D10)="A: 360=Y"  :REM 360k raw media on drive A
```

1.2MB Media

The $DEVICE clause "A: 1.2= Y" defines a raw diskette as 1.2MB (1200K) format. This format is defined as 1.2MB formatted capacity, 5-1/4" double-sided, high density diskettes containing 80 tracks per side, 15 sectors per track and 512 bytes per sector.

For example:

```
$DEVICE(/D10)="A: 1.2=Y"  :REM 1.2MB raw media on drive A
```

3-1/2" media

720K Media

The $DEVICE clause "A: 720= Y" defines a raw diskette as 720K format. This format is defined as 720K formatted capacity, 3-1/2" double-sided, double density diskettes containing 80 tracks per side, 9 sectors per track, and 512 bytes per sector.

For example:

```
0010 $DEVICE(/D10)="A: 720=Y"  :REM 720k raw media on drive A
```

1.44MB Media

The $DEVICE clause "A: 1.4= Y" defines a raw diskette as 1.44MB format. This format is defined as 1.44MB formatted capacity, 3-1/2" double-sided, double density diskettes containing 80 tracks per side, 18 sectors per track, and 512 bytes per sector.

For example:

```
0010 $DEVICE(/D10)="A: 1.4=Y"  :REM 1.44MB raw media on drive A
```

2.88 Media

The $DEVICE clause "A: 2.8= Y" defines a raw diskette as 2.88MB format. This format is defined as 2.88MB formatted capacity, 3-1/2" double sided, double density diskettes containing 80 tracks per side, 36 sectors per track and 512 bytes per sector.

5.2.3 Determination of Media Type

The media mounted in the diskette drive must match the type specified by the $DEVICE clause (an "automatic" determination of media types is not supported). Attempting to read or write a diskette which has been designated as the wrong media type results in an NPL error code I93 (format error).

Programs which must access different media types may do so by trying to read sector zero, using each of the formats in turn, with appropriate error coding for recovery.

For example:

```
0010 DIM A$256
   : $DEVICE(/D10)="A:"              :REM try 320K format
   : S=1280                          :REM 320k media size
   : DATALOAD BAT/D10,(0)A$
   : ERROR $DEVICE(/D10)="A: 360=Y"  :REM 320K failed, try 360K
   : S=1440                          :REM 360K media size
   : DATALOAD BAT/D10,(0)A$
   : ERROR $DEVICE(/D10)="A: 1.2=Y"  :REM 360K failed, try 1.2MB
   : S=4800                          :REM 1.2MB media size
   : DATALOAD BAT/D10,(0)A$
   : ERROR $DEVICE(/D10)="A: 720=Y"  :REM 1.2MB failed, try 720K
   : S=2880                          :REM 720K media size
   : DATALOAD BAT/D10,(0)A$
   : ERROR $DEVICE(/D10)="A: 1.4=Y"  :REM 720K failed, try 1.4MB
   : S=5760                          :REM 1.4MB media size
   : DATALOAD BAT/D10,(0)A$
   : ERROR $DEVICE (/D10) = "A: 2.8=Y":REM 1.4MB failed, try 2.8MB
   : S=11520                         :REM 2.8MB media size
   : DATA LOAD BAT/D10, (0)A
   : ERROR PRINT "Cannot read diskette"
   : END
0020 PRINT S
   : REM At this point we can access raw media in /D10. Media
   : REM size is S (256-byte) sectors.
```

HINT: It is recommended that the same NPL address (i.e., /D10) be used to access any one drive (i.e., A:), if access to different types is required. Attempting to define, for example, /D10 as "A:" and /D20 as "A: 1.2= Y" at the same time may result in spurious errors on some systems.

5.2.4 "Raw" Diskettes

Although the 360K, 1.2MB, 720K, 1.44MB and 2.88MB diskette formats use 512-byte sectors, an NPL "sector" continues to refer to 256 bytes of information. All access to the diskettes (such as DATALOAD BA) determines the appropriate 512-byte sector containing the required information. The RunTime then performs a read of the 512-byte sector and then modifies the appropriate 256-byte section and then rewrites the full 512 byte sector. Consequently, the difference between types of diskettes, once they have been initialized (using $FORMAT DISK and SCRATCH DISK statements), is transparent to most applications (provided the $DEVICE specification contains the appropriate media type clause).

When raw diskettes of any class are formatted, Bytes 3 and 4 of sector 0 are set to indicate the media size in 256 byte units (320K = 1280 sectors, 360K = 1440 sectors, 1.2MB = 4800 sectors, 720K= 2880, 1.44MB= 5760, 2.88MB= 11520) which may be used (after subtracting 1 sector) as the END= parameter to a subsequent SCRATCH DISK statement. A value of zero is placed in these bytes by previous MS-DOS versions of the Run-Time.

Performance

Since the minimum amount that may be read from or written to a diskette is one sector, the access to the new types of media may be noticeably slower when writing single sectors, since this entails reading a 512-byte sector, modifying the appropriate part and then rewriting it. Since rewriting incurs an overhead of a complete rotation period, single sector writes can be expected to require an average of 1.5 rotational periods, compared to an average of .5 rotational periods for the 320K media (i.e., three times slower). Multi-sector writes (COPY/MOVE/VERIFY) do not incur this performance penalty, except possibly on the initial and final sectors of a multi-sector access. As a result, these operations (and reads of all types) perform at speeds that are approximately equivalent to the 320K media.

Media Defects

All types of raw diskettes used by NPL must be defect-free. The $FORMAT DISK statement reports an error if media defects are detected during the format procedure.

NOTE: The types of diskettes used for 320K and 360K media are the same, but the 1.2MB (high density) diskettes are a different type. These types of diskettes are not interchangeable in any way. This is also true for 720K, 1.44MB, 2.88MB diskettes.

Drive/Media Compatibility

360K diskette drives are not capable of accessing 1.2MB media. However, the 1.2MB diskette drives are capable of reading 360K diskettes, if the appropriate "360= Y" clause appears in the $DEVICE statement.

NOTE: Due to an inherent problem in 1.2MB drive technology, writing to 360K diskettes with a 1.2MB drive may produce a diskette which cannot be read on a 360K diskette drive.

The 3-1/2" 720K diskette drives are not capable of accessing 1.44MB media. However, 1.44MB diskette drives are capable of reading 720K diskettes, if the appropriate "720= Y" clause appears in the $DEVICE statement. Likewise, 2.88 MB drives can read 720K, and 1.44MB diskettes.

Data Integrity

One advantage of the 320K media format is that, because of the different sector size compared to that used by MS-DOS, NPL and MS-DOS applications were isolated from each other to some extent. An MS-DOS diskette cannot be read from or written to NPL (unless the diskette was formatted for MS-DOS, and access was to a named MS-DOS disk-image file on the diskette), and MS-DOS applications cannot read or write the NPL raw diskettes.

Developers should take precautions to ensure that application programs do not modify diskettes unless they have been validated (i.e., locating an expected file by name).

WARNING--NPL does not attempt to prevent programs from modifying MS-DOS diskettes, which could result in corrupting the data on the diskette. Similarly, it is possible that MS-DOS applications could corrupt the NPL diskettes if direct access to the media is made.

Compatibility

5-1/4" Media

320K

Full compatibility with all other versions of NPL that support 320K "raw" diskette access.

Full compatibility with Wang 2275 and DS 360K drives.

360K

Full compatibility with all other versions of NPL that support 360K "raw" diskette access.

Compatibility with Wang 2275 and DS 360K drives.

1.2MB

Full compatibility with all other versions of NPL that support 1.2MB "raw" diskette access.

3-1/2" media

720K

Full compatibility with all other versions of NPL that support 720K "raw" diskette access.

1.44MB

Full compatibility with all other versions of NPL that support 1.44MB "raw" diskette access.

2.88MB

Full compatibility with all other versions of NPL that support 2.88MB "raw" diskette access.

NOTE: The drive/media compatibility issue described earlier in this chapter applies to 320K or 360K diskettes created on a 1.2 MB drive. Refer to Appendix D for a cross-reference of supported media under other NPL-supported operating environments.

5.3 Diskimage Files

Diskimage files can be defined as any valid MS-DOS filename in any valid MS-DOS directory on any valid MS-DOS drive designation. Any type of physical disk media supported by the MS-DOS operating system on an IBM-compatible PC may be used for the storage of diskimage files. However, there are several special considerations involved in the use of diskimage files on removable media (refer to Section 5.3.6).

All features of diskimage files described in Chapter 7 of the NPL Programmer's Guide, are fully supported on the MS-DOS version of NPL. Refer to Section 5.3.6 and Section 7.3 of the NPL Programmer's Guide for more information about performance issues related to NPL diskimages.

5.3.1 Naming Conventions

$DEVICE statements for diskimage files on an IBM-compatible PC should use the following general form:

```
$DEVICE(/XXX)="[drive:][\pathname]diskimage"
```

where:

XXX	=	Any valid NPL disk device address.
drive	=	Any valid MS-DOS drive designation. If no drive is specified, the current drive is used.
pathname	=	Any valid MS-DOS pathname. If no drive is specified, the currently selected directory pathname for the designated drive is used.
diskimage	=	Full path name of diskimage file.

NOTE: Changing default directories and default drives through the $SHELL statement is permanent and any subsequent references to existing diskimage files may encounter P48 - Illegal Device Specification errors if these files are not found in the current default directory. Therefore, if the $SHELL statement is used to change directories, it is recommended that full pathnames be used.

HINT: It is recommended that the extension .BS2 be used for diskimage filenames in order to clearly distinguish diskimage files from other files stored in the MS-DOS file system.

NOTE: MS-DOS is case-insensitive. Therefore, pathnames and file names for diskimage files are always treated as uppercase, regardless of the case used in the $DEVICE statement.

HINT: It is recommended that only uppercase pathnames and file names be used to provide compatibility with other NPL platforms which are case-sensitive.

5.3.2 Special Considerations for Disk Caches

There are several special considerations relating to the use of diskimage files on systems using disk caches.

Although it is not always apparent, some operating systems may use a write cache as one of many internal mechanisms used to optimize system resources. Alternatively, many third-party software products and disk controllers are now available that can be used to enable or disable read or write caching. However, this can cause problems when dealing with either fixed or removable media. Refer to Section 5.3.4 for special considerations using removable media.

Under MS-DOS, always use raw diskette formats when addressing the diskette drive (i.e., A:, A: 1.2= Y, B: 720= Y, etc.) as opposed to addressing a diskimage directly on diskette (A:PLATTER1.BS2 for example). When addressing a "raw" diskette, NPL bypasses the native BIOS routines that access the FAT Table and maintain the file on diskette. A side benefit of this access method is that write caching is not performed when these routines are bypassed.

Use of disk caching for write operations with a hard disk can cause potential problems. These problems all relate to the fact that there is a time lag between the time when a write operation takes place and the time when the data is written to the disk.

The biggest potential problem is the possibility of events that might prevent the cached data from being written. This could range from a power outage to an operator turning off the PC.

A second problem with write caching is that errors that occur during the physical write to the disk can no longer be detected by the program.

Many disk caches have some form of tunable parameters that allow the user to control the behavior of the cache. At a minimum, a good disk cache should have some type of automatic flush capability so that data is written to disk very frequently. Some caches may be configured as "write through" caches so that writes are written to disk immediately.

Read caching can dramatically improve performance; however, the risks associated with write caching are simply too high for serious use with some business applications. Niakwa recommends turning off write caching or at least configuring it to do frequent flushes to disk.

5.3.3 Implicit $BREAK Implications

$BREAK under MS-DOS performs no operation.

5.3.4 Diskimages on Removable Media

There are several special considerations relating to the use of removable media. These considerations apply to the use of diskettes as well as the use of removable media.

MS-DOS incorporates buffering techniques for disk I/O which could adversely affect operations of NPL programs which use removable media. The problem is that MS-DOS buffers disk I/O and does not automatically clear the buffers when removable media is mounted. Therefore, it is possible that, when accessing the same MS-DOS filename, a read operation may return data from a buffer instead of directly accessing the file. If a new removable media has been mounted, the information in the buffer may not be current and, therefore, may not correspond to the data actually contained on the diskimage file currently mounted. Data may remain in a buffer until the MS-DOS file is logically closed.

NOTE: Write operations typically are not affected by this problem since all write operations are written to disk immediately unless write cache software is in use.

Although the RunTime automatically logically opens a file that has been closed, it cannot automatically perform a logical close. The NPL program using the diskimage file on removable media must logically close the corresponding NPL device whenever a new disk(ette) is mounted. Programs should also close the NPL device when they are finished with the disk(ette) drive so that subsequent programs may successfully access it.

To logically close an NPL device, either a $CLOSE statement or a $DEVICE statement must be executed. For example, assuming that the NPL disk address for the diskimage is D10, and that D10 has been selected as device #1 (SELECT #1/D10) in the internal NPL device table, any of the following statements logically closes NPL device D10 and the corresponding MS-DOS file name:

```
$CLOSE
$DEVICE(/D10)=$DEVICE(/D10)
$DEVICE(#1)=$DEVICE(#1)
```

To summarize, NPL statements which logically close a diskimage file on removable media must be added to programs at the following points:

- Before any access to a newly mounted diskette.
- After the last access to the last diskette in a series (or before exiting the program using the diskettes).

NOTE: **Programmers accessing removable media in Immediate Mode (RTI) should be especially aware of this problem. Before removing a diskette containing a diskimage file which has been modified, the recommended procedure is to press the HELP key, or enter $CLOSE to ensure that no buffered information is retained.**

5.3.5 Using RAM Disks to Increase Performance

A RAM disk is a part of system memory that has been set up to function as a hard drive. Modification to the system's CONFIG.SYS file is necessary to set up a RAM disk. Refer to the MS-DOS manuals for details of setting up a RAM disk.

RAM disks can increase the performance of NPL-based applications because the information stored in the diskimage on the RAM disk is always loaded into memory and immediately available while the PC is on. The diskimage to be used in the RAM disk must be copied to the RAM disk with the use of the MS-DOS COPY command or it must be created by the RunTime once the RAM disk is active. Assuming that the RAM disk is defined as drive E:, the following NPL $DEVICE statement can be used to set PLATTER1.BS2 to NPL device D11:

```
$DEVICE(/D11)="E:PLATTER1.BS2"
```

The above example assumes that PLATTER1.BS2 is already stored on the RAM disk.

WARNING--This disadvantage of a RAM disk is that all data stored there is lost whenever the PC is turned off or when the power is otherwise interrupted. The information stored in a RAM disk should be copied to a hard drive before the system is shut off.

5.4 Supported Monitors/Controllers

NPL supports both monochrome and color graphic controllers and monitors. Chapter 6 discusses the use of these controllers and monitors in detail.

5.4.1 File Naming Conventions

Under the MS-DOS version of NPL, the terminal type for the IBM-compatible PC monitor is determined automatically by the RunTime.

NOTE: **The use of the /G option affects the determination of the monitor type. Refer to Chapter 6 for more information on the /G option.**

The monitor type is stored in byte 3 of $MACHINE. The following are valid monitor types:

Monitor Type	Value of byte 3 of $MACHINE
Monochrome in non-graphic mode	M
CGA/EGA/VGA in non-graphic mode	C
Hercules (HGA) in graphics mode	H
EGA/VGA in graphics mode	E

The downloadable font files used by the RunTime in /G mode are loaded automatically by the RunTime. Refer to Chapter 6 for details.

Serial terminals can be used in conjunction with the Redirect feature of the HELP facility. The terminal type for these serial terminals is set during the execution of the Redirect feature. Chapter 6 of this Supplement and Section 2.9 of the NPL Programmer's Guide describe the operation and compatibility of all serial terminals supported by the Redirect feature.

5.4.2 Using Emulation Products

Many emulation products exist that allow remote control of the RunTime under MS-DOS. These products can be used to allow a developer to check the operation of a RunTime application at a remote location.

NOTE: Niakwa does not officially support the use of any emulation product. These products can have an effect on the keyboard and screen translations performed by the Run-Time. Some NPL virtual keys may not operate as expected when using an emulation product. Keyboard remapping using $KEYBOARD or the NPL Utility EDKEY-BOA may be needed, but this can affect the use of the RunTime when the emulation product is not being used. Refer to the NPL Statements Guide for more information on $KEYBOARD and Chapter 13 or the NPL Programmers's Guide for details on the NPL Utilities.

A $OPTIONS bytes are available to provide better results with use of the emulation products. The bytes are described below:

Byte 31 of $OPTIONS

This byte controls certain features for terminal emulators which do not provide 100% support of the terminal being emulated. Refer to the NPL Statements Guide, $OPTIONS, for more information.

5.4.3 NPL Plotter Drivers

Under MS-DOS, use of Niakwa's Plotter Driver software allows NPL programs to perform plot-like functions to the screen (on controllers and monitors which support the equivalent of the CGA, EGA, or HGA controllers).

Refer to the Niakwa Scientific and Communications Drivers Packages for further details on supported hardware and plotting capabilities.

5.5 Printers

On the IBM-compatible PC, print output can be directed to either the parallel port, the serial port, an MS-DOS file, or the standard file handle (refer to Section 5.5.1 for naming conventions).

NPL permits the automatic translation of characters as they are sent to a printer (or spool file) using a simple lookup table. This is performed using the Printer Translation Table option. For a full discussion of this facility refer to Chapters 7 and 13 of the NPL Programmer's Guide.

If no translation is performed, all characters sent by the NPL program are passed directly to the specified print device with no modification. Thus, all printer control sequences are still the responsibility of the programmer.

Section 5.5.2 discusses several special considerations when configuring new printers.

5.5.1 Naming Conventions

The following section discusses NPL naming conventions for printers under MS-DOS.

Parallel Printer

For printing to the parallel printer, the special device name LPTx should be used in the $DEVICE statement.

For example:

```
$DEVICE (/215)="LPTx"
```

directs all print output sent to the NPL print address /215 to the parallel printer where x is the port number (i.e., LPT1, LPT2).

Serial Printer

For printing to the serial printer, the special device name COMx should be used in the $DEVICE statement.

For example:

```
$DEVICE (/216)="COMx"
```

directs all print output sent to the NPL print address /216 to the serial printer, where x is the port number (i.e., COM1, COM2).

MS-DOS File

For directing print output to an MS-DOS file, the device equivalence should be set to the name of the MS-DOS file. Drive designation and path designation may be included.

For example:

```
$DEVICE (/217)="C:\AP\REPORT1.DAT"
```

directs all print output sent to the NPL print address /217 to the file REPORT1.DAT in directory \AP on drive C.

Directing output to an MS-DOS file can be useful in two ways. It can be used for generating ASCII files which can be used as input to native MS-DOS applications. It can also be used to generate an ASCII file that can be printed at a later time.

Use of the optional ERR= Y $DEVICE clause to cause NPL errors to be issued when errors occur while print class output is being directed to an ASCII file is fully supported for printing to MS-DOS ASCII files on the PC. Refer to $DEVICE in the NPL Statements Guide for details on the functionality of ERR= Y.

Printing to Standard File Handles

One of the features of the MS-DOS operating system on an IBM-compatible PC is the ability to reference standard file handles. File handle number four (4) is the standard MS-DOS file handle for the system printer. The actual output device or file associated with any file handle can be modified by a number of MS-DOS functions including the redirect (RDIR) function. By routing print output to file handle number 4, MS-DOS is allowed to direct output to the device or file previously selected. This means that the end-user may direct output to the parallel port, serial port, or an ASCII file independent of NPL and your application programs.

For example:

```
$DEVICE(/215)=">4"
```

directs all print output sent to the NPL print address /215 to the device or file currently established as file handle number 4.

5.5.2 Special Considerations

MS-DOS expects printers to be configured so that they do not automatically perform a line feed upon receipt of a carriage return. Most printers can be set up in this manner by setting a dip switch or modifying the printer’s configuration setup. To be consistent with standard PC type printers and other software native to the PC, the RunTime automatically inserts a line feed following every carriage return sent to a printer.

However, some printers (particularly Wang 2200 printers) do not have the capability of automatically suppressing the generation of a line feed upon receipt of a carriage return. Printers that lack this capability cannot be accessed by standard MS-DOS print functions (double spacing results). However, the RunTime program does provide a mechanism for suppressing the insertion of the linefeed with every carriage return so these printers can be used with NPL applications.

The ALF (Auto Line Feed) Option

The RunTime automatically generates a line feed after a carriage return in all output directed to printer devices. This feature can be suppressed by the operator at execution time using the HELP processor, PRINTER CONTROL selection, setting AUTO-LF OFF.

Alternatively, the Auto Line Feed option may be directly controlled by an NPL program. This is accomplished by the use of a $DEVICE statement for printer type devices. The form of the ALF option may be set to "Y" or "N". A "Y" indicates that the special output options are used, while an "N" indicates that the special output options are not to be used.

For example:

```
$DEVICE(/215)="LPT1 ALF=N"
```

suppresses the automatic line feeds for the selected 215 printer address (the parallel printer). If ALF is not specified, the initial default value is used ("Y"). Refer to Chapter 7 of the NPL Programmer’s Guide for details the use of the ALF $DEVICE clause.

5.6 Mouse Support

The following section details the support for the mouse under the MS-DOS.

To use a mouse device under the MS-DOS version of NPL the following steps are required:

1. A mouse drive must be installed. Refer to the mouse manual for details on installing a mouse driver.

2. A mouse must be connected to the system on system startup.

3. The RunTime must be started with the /K RunTime startup option.

   For example, to start the RunTime for use with a mouse, enter the following:

   ```
                RTI /K MYBOOT
   ```

Byte 29 of $MACHINE indicates whether keyboard mouse events are supported as shown in the following chart.

HEX (00)	Default; mouse not available
HEX (01)	Mouse available

The location of the mouse is reported to $MACHINE bytes 23 and 24. The following table displays the information provided by these two bytes:

Byte	**Description**
Byte 23	Contains the current row position of the mouse pointer when a mouse event occurs (if it is on the screen).
Byte 24	Contains the current columnar position of the mouse pointer when a mouse event occurs (if it is on the screen).

Mouse events are reported to the NPL application through the KEYIN statement by generation of special function key codes as follows:

’F1	Left button pressed
’F2	Left button released
’F3	Left button pressed (within double click time)
’F4	Right button pressed
’F5	Right button released
’F6	Right button pressed (within double click time)
’F7	Dragged North (up)
’F8	Dragged South (down)
’F9	Dragged East (right)
’FA	Dragged West (left)

NOTE: A double-click event is reported if the same key is pressed within approximately 1/2 of a second.

When a key is read by the NPL KEYIN statement, bytes 23 and 24 of $MACHINE contain the Y (row) and X (column) address of the current location of the mouse when the event occurred, if it is "on screen". When the cursor is outside the screen area (or if there is no mouse) these bytes contain high values (HEX(FF)).

NOTE: Several utilities included in UTILITY.BS2 have been modified to support the mouse using the above settings.

To provide correct sequencing of mouse key clicks and keyboard operations, the Run-Time uses the BIOS keyboard request interrupt (INT 16H), and internally buffers both the keyboard and mouse events. The internal buffer can contain up to 255 key strokes.

If the application already takes direct control of the mouse or otherwise hooks the key-board interrupt (i.e., using an external library), it may be incompatible with the /K option.

The default $KEYBOARD map has been modified to return the appropriate special func-tion keys when these codes are returned form the BIOS.

WARNING--If an application has defined a modified KEYBOARD.TBL file, mouse keys may not be returned to the NPL program, although the values in $MACHINE will change. In this event, revise the keyboard table to return the appropriate code for the new complex codes.

If the application prefers to report only selected mouse button presses or releases or wants to have the RunTime automatically replace mouse keys with other key values (i.e., EXEC, CANCEL), the keyboard mapping must be modified. Refer to $KEYBOARD, in the NPL Statements Guide or Chapter 13 of the NPL Programmer's Guide for informa-tion on the EDKEYBOA utility.

The mouse cursor appears as a single character block of inverse video (monochrome) or inverted colors (on a color monitor). In /G mode, under EGA, the cursor appears as an ar-row. The cursor is hidden automatically each time a keystroke is entered and is made vis-ible again any time the mouse moves or a mouse key is pressed or released.

The mouse option is not supported in combination with the /R (remote video) or Hercules /G (graphic) options. It is possible that some very recent mouse drives (compatible with the extended capabilities of Microsoft mouse drivers, version 7.0) will allow support for the Hercules /G option.

Mouse Support in the Help Processor

NPL help screens have recognition of mouse keys as follows:

- The left button click acts as select when positioned on the field.
- The left button double click acts as EXECUTE on the selected action field.
- The right button click acts as the CANCEL key.

Refer to Chapter 11 of the NPL Programmer's Guide for more information on the HELP Processor.

5.7 Serial Devices

The RunTime contains limited ability to read and write raw data from a serial port using the C620 $GIO microcommand. The following section discusses the use of serial devices under MS-DOS for NPL.

5.7.1 Naming Conventions

Serial ports on PCs under MS-DOS can be accessed through the special device designation "COMx" in the $DEVICE statement, where "x" is the physical "COM" port number.

For example:

```
$DEVICE(/211)="COM1"
```

or

```
$DEVICE(/211)="COM1 TMO=Y"
```

directs subsequent I/O operations directed to address 211 to the MS-DOS COM1 port. The optional designation TMO= Y instructs input operations directed to the serial port to return to program control without waiting for a character to be present. Refer to Section 5.7.3 and Chapter 7 of the NPL Programmer's Guide for more details on accessing serial ports.

5.7.2 Sending Output to a Serial Port

NPL PRINT statements may be used to direct output to the serial port. In addition, $GIO output microcommands, both single character and multi-character, may be used.

NOTE: Output to a serial port is handled in the same manner as output to a printer. That is, automatic line feed insertion (unless it is overridden by the ALF option of $DEVICE) and character translation (if specified by the XLA option of $DEVICE) are in effect. Please refer to Section 7.9 of the NPL Programmer's Guide for more details on accessing serial ports.

5.7.3 Input from a Serial Port

NOTE: The following discussion refers to functionality when the TMO= Y option has been specified in the $DEVICE statement, as explained above.

The MS-DOS version of the RunTime contains limited support for accepting input from a serial port. This support is intended to provide a mechanism for using serial input devices where the interface requirements are very simple and straightforward. Applications which require the more sophisticated features of the Wang 2227 board should refer to the Niakwa 2227 emulation driver provided in the Niakwa Scientific and Communication Drivers package.

The $GIO microcommand C620 accesses binary data in the buffer for the port specified and returns as many bytes of data as available, or zero bytes if no data is available.

For example:

```
10 DIM L$10,A$80                  :REM GAO control, buffer
   : $DEVICE(/01C)="COM1 TMO=Y" :REM read from "COM" port 1
20 $GIO/01C(C620,L$)A$           :REM look for data on port
   : L=VAL(STR(L$,9),2)          :REM get # of bytes received
   : IF L=0 THEN $BREAK          :REM check no data available
   : IF L>0 THEN GOSUB xxx       :REM process data
   : GOTO 20
```

In this example, the first L bytes of A$ contain the data returned from the port.

This method of accessing data as input on the serial port is very primitive and has several restrictions that the programmer must consider:

- Communications parameters for the port must be established externally to NPL. The MS-DOS "MODE" command may be used to define the baud rate, number of data bits, stop bits, and parity for the serial port. These parameters may be defined in the AUTOEXEC.BAT file or may be altered dynamically at any time from the DOS prompt. Consult the MS-DOS manual for more information regarding the "MODE" command.

- If the buffer overflows, incoming characters are lost. There is no flow control for incoming data.

- There is no error detection.

- There is no method for checking signals on the line (i.e., device not attached or not ready).

- On PC's the standard MS-DOS COM driver is interrupt-driven and, in general, is suitable for reception of multi-byte data. However, more reliable results are usually obtained at lower baud rates, or if data is slowed down by the sending devices to ensure that the port is "read" at least once in each character time. Communication parameters are set by the MS-DOS "MODE" command.

- None of the $GIO microcommands supported by the Niakwa 2227 emulation driver are supported, using this technique. Program modifications are required to support input from the serial device using this method.

- This technique is supported under MS-DOS only for compatibility with other hardware versions of NPL. For many applications, use of the 2227 emulation driver remains the recommended choice for a synchronous communications on the IBM PC.

5.7.4 NPL Communications Drivers

Under MS-DOS, use of Niakwa's 2227 communication driver software allows NPL programs to perform serial communications through the serial ports equivalent to those of the Wang 2227B Buffered Asynchronous Communications Controller.

Refer to the Niakwa Scientific and Communications Drives Package for further details.

5.8 Tape Drive Support

NPL does not directly support access to a tape drive. However, all NPL diskimage files (i.e., PLATTER1.BS2) may be backed up using standard MS-DOS backup utilities or third-party tape drives. The $SHELL command can be used to access both methods to perform a backup during program operation.

5.9 Math Co-Processor Support

Use of the 80x87 math co-processor support is enabled by setting of byte 16 of the $OPTIONS system variable. Acceptable values for the $OPTIONS byte are:

HEX(00)	Do not use math co-processor even if available.
HEX(01)	Use math co-processor for transcendental function if available.

Other values are reserved and should not be assigned to this byte.

Applications which now use the math co-processor should set byte 16 of $OPTIONS to HEX(01) or they may experience a decrease in performance, since the RunTime default is not to use the co-processor.

For example:

```
0010 DIM X$64
  : X$=$OPTIONS
  : STR(X$,16,1)=BIN(1)        :REM use co-processor
  : $OPTIONS=X$
```

NOTE: The $MACHINE system variable indicates whether a math co-processor is available. Byte 10 of $MACHINE contains the following values:

HEX(00)	No co-processor available
HEX(01)	80X87-class co-processor available

Developers making use of the math co-processor should be aware that there may be differences in precision of results and range of functional domain to some functions. In particular, the 80x87-class co-processors are generally accurate with 48- bit precision and have an exponent of 2 in the range +/- 16383. This does not normally present a problem, except where results or arguments approach overflow or underflow values.

5.10 Default Device Equivalences

When the NPL RunTime program is started under MS-DOS, the Device Equivalence Table contains the following default entries:

2200 Address	MS-DOS Equivalent
/B10 or /D10	A:
/310 or /D11	PLATTER1.BS2 in the current directory
/D12	PLATTER2.BS2 in the current directory
/215	/dev/prn
/204	/dev/prn

The "current directory" is the directory selected at the time the RunTime program was invoked. No entry is required of the keyboard or CRT screen. These are defaulted (/X01 & /X05, where X= 0 or 2) by the RunTime.

CHAPTER 6

SUPPORTED DISPLAY CHARACTERISTICS

6.1 Overview

The MS-DOS version of NPL supports the use of color and graphics on specific color/graphics monitors and controllers. Many NPL screen handling features are affected by the type of controller/monitor used. These include standard features such as character set support, box graphics support, and attribute support. Refer to Chapter 7 of the Programmer's Guide for details on standard NPL screen handling features.

This chapter discusses the use of enhanced color and graphics support and how the NPL screen handling features are affected by the various controllers supported by NPL under MS-DOS. For each supported controller, the use of different monitor types is also discussed.

NOTE: NPL color support is specific to the Color Graphics (CGA) controller and the Enhanced Graphics (EGA) controller. NPL graphics support is specific only to the Enhanced Graphics (EGA) controller or the Hercules Monochrome controller. VGA controllers are supported in either text mode or in EGA graphics mode.

Section 6.2 discusses operating system considerations.

Section 6.3 summarizes display devices.

Section 6.4 discusses the standard monochrome controller.

Section 6.5 discusses the use of a Hercules controller with a monochrome monitor.

Section 6.6 discusses the color graphics controller (CGA) with a color monitor.

Section 6.7 discusses the use of an Enhanced Graphics Adapter (EGA) with an Enhanced color monitor.

Section 6.8 summarizes the use of a Video Graphics Array (VGA).

Section 6.9 discusses the support of color by NPL.

Section 6.10 discusses the support of graphics.

Section 6.11 discusses plotting capabilities of the PC under MS-DOS.

Section 6.12 discusses the keyboard characteristics of the PC under MS-DOS.

Section 6.13 discusses the use of serial terminals in conjunction with the Redirect feature of the NPL HELP facility.

6.2 Operating System Considerations

The following sections discuss the characteristics of the operating system which affect the system display. Among the topics discussed are how terminal type is determined by the RunTime and where the terminal files are to be found. Also described are the HALT key function, local printer support and terminal configuration requirements.

6.2.1 Determination of Terminal Type

Under the MS-DOS version of NPL, the terminal type for the PC monitor is determined automatically by the RunTime.

Use of the /G, RunTime start-up option affects the determination of the monitor type. Refer to Section 4.4.3 for more information on the /G option. The monitor type is stored in byte 3 of $MACHINE. The following are valid monitor types:

Monitor Type	**Value of Byte 3 of $MACHINE**
Monochrome (MGA) in non-graphic mode	M
CGA/EGA/VGA in non-graphic mode	C
Hercules (HGA) in graphics mode	H
EGA/VGA in graphics mode	E

6.2.2 Location of Terminal Files

On startup, the RunTime searches for the keyboard and screen translation table files (SCREEN.xxx and KEYBOARD.xxx) based on the following order of search:

- Current directory on the current drive.
- NIAKWA_RUNTIME drive/directory.
- The \BASIC2C directory on the current drive.

Refer to Chapter 13 of the Programmer's Guide for details on the use of the NPL Utilities to modify the screen and keyboard files.

6.2.3 Downloading Font Files

Font files are only necessary under the /G RunTime start-up option. The necessary files are loaded automatically to the RunTime when the /G option is used. Refer to Section 6.10 for details on the /G option.

6.2.4 The HALT Key

Under the MS-DOS release of NPL, the HALT key is (CTRL-ALT). The HALT key is active at all times on the supported monitors, unless disabled using $OPTIONS byte 13. The HALT key cannot be modified by the user or by the application.

NOTE: The HALT key is not available when using the Redirect feature.

6.2.5 Local Printer Support

This function is not necessary under MS-DOS and is not supported.

6.2.6 Terminal Configuration Requirements

This function is not necessary under MS-DOS and is not supported.

6.3 Display Device Summary

The following table presents several of the display devices supported by NPL and their various options.

IBM Monitor (Console) Features Supported by MS-DOS

Operating System	Controller	Monitor	Mode	Box Type	Attributes				Color	Columns	Fonts	Alternate Character Set
					Blink	Bright	Reverse	Underline				
MS-DOS	MGA CGA	Mono	Character	Character	Yes	Yes	Yes(8)	Yes(8)	No	Only 80	No	No
	CGA, EGA VGA(2)	Color	Character	Character	Yes	Yes	Yes	Yes (3)	Yes	Only 80	No	No
	HGA, EGA, VGA(2)	Mono	Graphics (1)	Both	Yes (6,7)	Yes (6,7)	Yes(8)	Yes	No	Only 80	Yes	Yes
	EGA, VGA	Color	Graphics (1)	Both	Yes (4,5)	Yes(5)	Yes	Yes	Yes	Only 80	Yes	Yes

Features by Operating System Notes:

(1) /G option used. If /G is not used, the display behaves as described in "Character" mode.

(2) True VGA mode is supported only under the MS-Windows version of NPL.

(3) Underline is not supported on CGA. A color combination may be displayed instead. The default is bright white on blue background, but some compatibles may display the background in shades of gray.

(4) Appears red on black background. Color choices may be modified.

(5) Bright/Blink used jointly appears bright white on red background. Color choices may be modified.

(6) Bright/Blink used jointly appears blinking (same as blink only).

(7) All combinations of reverse, bright or blink appear as reverse video on HGA.

(8) Reverse/Underline used jointly appears only as underline on Monochrome.

Refer to Chapter 7 of the Programmer's Guide for more information on video attributes.

6.4 Monochrome Controllers (MGA)

The following section discusses how the RunTime operates under MS-DOS with the standard Monochrome Controller.

6.4.1 Graphics Availability

The /G (graphics) RunTime startup option is not supported on the standard monochrome controller.

6.4.2 Screen Character Set

Downloadable Fonts

The standard monochrome controller does not support the use of downloadable fonts.

Alternate Character Set (Pixel Graphics)

On the monochrome controller, use of the Alternate Character Set, which displays the characters from HEX(C0) to HEX(FF) in a "pixel" graphic mode, is supported. However, not all characters are fully equivalent.

Word Processing Graphic Characters (HEX(80-9F)):

The most commonly used WP graphics characters are available. However, there are some exceptions which have no equivalent in the IBM font. These include:

HEX(87)	Accent umlaut/diaeresis
HEX(88)	Accent acute
HEX(90)	Centered Dot
HEX(91)	Hollow Diamond
HEX(9B)	Hollow Square

Pixel Graphics (HEX(C0-FF)):

Exact equivalences are available only for blank, full character, left half, and right half graphics. These four characters are set up in the default screen translation table. However, no attempt has been made to assign default translation values for pixel graphics for which there is no direct equivalent. If use of such characters is required, use the screen translation editor or $SCREEN statement to set up the equivalences which would work best in the application. Refer to Chapter 13 of the Programmer's Guide or details on the screen translation table editor.

Refer to Chapter 7 of the Programmer's Guide for details on pixel graphics.

Screen Translation

Screen character translation is performed using an internal lookup table. The RunTime program contains standard default values for this table which should suffice for most applications. However, some modification of this table may be required to support specialized functions such as non-English characters. This table can be modified by the NPL statement $SCREEN or by the NPL utility EDSCREEN which creates a file (SCREEN.TBL) with the default screen translation values to be used. The default table can be displayed by executing the EDSCREEN utility. Refer to the NPL Statements Guide for details on $SCREEN and refer to Chapter 13 of the NPL Programmer's Guide for details on the NPL Utilities, (EDSCREEN).

6.4.3 Box Graphics Support

The standard Monochrome Graphics controller does not support "true" box graphics under NPL. "Character" box graphics are supported. To use "character" box graphics it is necessary to modify byte 1, of the NPL $BOXTABLE system variable. The default value of byte 1 is HEX(00) and indicates that "character" boxes are disabled. A value of HEX(01) enables "character" boxes.

Refer to the NPL Statements Guide for details on the use of the $BOXTABLE system variable.

6.4.4 Attributes Support

Under MS-DOS, NPL screen attributes on the monochrome controller operate as documented in Chapter 7 of the Programmer's Guide with the following exceptions:

- Inverse video does not work in conjunction with underline. Underline only is produced when these attributes are combined.

6.4.5 Cursor Appearance

The monochrome controller does not have the ability to display a steady cursor. A "steady" cursor appears as a thicker, blinking cursor.

6.5 Hercules Graphics Controllers (HGA)

The following section discusses how the RunTime operates under MS-DOS with the Hercules Graphics Controllers.

6.5.1 Graphics Availability

The /G (graphic) RunTime start-up option is supported on the Hercules controllers. The remainder of this section assumes that the /G (graphic) option is in use on the Hercules controllers.

NOTE: **If a system is using both Hercules and EGA displays (either on separate adapters or "combination" adapters), the /GH option should be specified if Hercules mode is desired and the /GE option should be specified if EGA mode is desired. If the /G option is not in use on the Hercules controllers, the RunTime treats them as standard monochrome controllers. Refer to Section 6.4 for details on supported monochrome controllers.**

6.5.2 Screen Character Set

Downloadable Fonts

The character set for the Hercules controllers is supported through the use of a downloadable font. The downloadable font file IBMFONT0.HGA is loaded into memory at the time of execution of the RunTime with the /G option. This file must be present or the /G option is ignored. Refer to Section 6.10 for more information on downloadable fonts.

The NPL utility program EDFONT can be used to modify the downloadable fonts. Refer to Chapter 13 of the NPL Programmer's Guide for details on the Font Editor utility program.

Alternate Character Set (Pixel Graphics)

Use of the alternate character set is fully supported on the Hercules monochrome Graphics controllers.

Screen Translation

If the graphic option (/G) is used at the time NPL is executed, the values in $SCREEN are reset at startup to indicate that no character translation should take place. Developers whose applications modify $SCREEN, under the assumption that the MS-DOS character set is in use, should consider the effects of downloadable fonts.

6.5.3 Box Graphics Support

The Hercules Monochrome Graphics controller supports "true" box graphics under NPL. To use "true" box graphics, byte 1 of the NPL $BOXTABLE system variable must be set to its default, HEX(00) setting. Setting byte 1 of $BOXTABLE to HEX(01) enables "character" boxes.

Refer to NPL Statements Guide for details on the use of the $BOXTABLE system variable.

6.5.4 Attributes Support

The Hercules Monochrome Graphics Controllers support the use of underline as a standalone attribute. Any other combinations of NPL attributes appear as inverse video. Modifying byte 24 of the $OPTIONS system variable allows NPL to ignore the attributes. Refer to Section 6.10 for more details on video attributes in graphics mode.

6.5.5 Cursor Appearance

Refer to Section 6.10 for more information on the cursor appearance under graphics mode.

6.6 Color Graphics Controllers (CGA)

The following section discusses how the RunTime operates under MS-DOS with the Color Graphics Controller.

6.6.1 Graphic Availability

The /G (graphic) option is not supported on the CGA controller.

6.6.2 Screen Character Set

Downloadable Fonts

The standard CGA controller does not support the use of downloadable fonts.

Alternate Character Set (Pixel Graphics)

Under MS-DOS, use of the Alternate Character Set, which displays the characters from HEX(C0) to HEX(FF) in a "pixel" graphic mode, is supported. However, not all characters are fully equivalent. Refer to Section 6.10 of this manual and Chapter 7 of the Programmer's Guide for more information.

Screen Translation

Screen character translation is performed using an internal lookup table. The RunTime program contains standard default values for this table which should suffice for most applications. However, some modification of this table may be required to support specialized functions such as non-English characters. This table can be modified by an NPL statement (refer to $SCREEN, in the Statements Guide) or by the NPL utility EDSCREEN (refer to Chapter 13 of the NPL Programmer's Guide) which creates a file (SCREEN.TBL) with the default screen translation values to be used. The default table can be displayed by executing the EDSCREEN utility.

6.6.3 Box Graphics Support

The Color Graphics Adapter does not support "true" box graphics under NPL. "Character" box graphics are supported. To use "character" box graphics, it is necessary to modify byte 1, of the NPL $BOXTABLE system variable. The default value of byte 1 is HEX(00) and indicates that "character" boxes are disabled. A value of HEX(01) enables "character" boxes.

Refer to the NPL Statements Guide for details on the use of the $BOXTABLE system variable.

6.6.4 Attributes Support

NPL screen attributes operate as documented in Chapter 7 of the Programmer's Guide with the following exceptions:

- Underlines are not supported on the Color Graphics Controller. A color combination (background and foreground) may be displayed instead. The default color combination is bright white on a blue background. This default may be modified by the NPL application program by use of the $OPTIONS system variable. Refer to Sections 6.9 and 6.10 of this Supplement and the NPL Statements Guide, $OPTIONS, for more information on $OPTIONS bytes concerning color and graphics.

Byte 1 of the $OPTIONS system variable contains the default value used by the Run-Time program to replace the underline attribute on color/graphics PC monitors. This byte should contain a HEX value which relates to the following table:

	HEX(x0)	Background color (refer below for values of x)
+	HEX(0x)	Foreground color (refer below for values of x)
+	HEX(08)	Bright (optional)
+	HEX(80)	Blink (optional)

NOTE: Addition is binary.

Values for "x" are:

0	Black
1	Blue
2	Green
3	Cyan
4	Red
5	Magenta
6	Brown
7	White

The default value in this byte is HEX(1F) which is comprised of:

```
     HEX(10)         Blue background
+    HEX(07)         White foreground
+    HEX(08)         Bright
===========
     HEX(1F)
```

Alternatively, if the NPL color options are in effect (byte 22 set to any value other than HEX(00)), changing the foreground color of underline text can be accomplished by modifying the low order nibble of byte 23 to the appropriate color value desired, and issuing a power-on reset sequence. Refer to Chapter 8 of this Supplement and Chapter 7 of the Programmer's Guide for details on $OPTIONS bytes 22 and 23.

6.6.5 Cursor Appearance

The CGA controller does not have the ability to display a steady cursor. A "steady" cursor appears as a thicker, blinking cursor.

6.6.6 Color Support

Programmable specification of foreground, background, underline and perimeter colors is supported. Refer to Section 6.9 for details.

6.6.7 Noise Suppression

Old versions of the Color/Graphics adapter produces some transient screen interference ("noise") when the screen is updated using direct video access. This interference is most noticeable while a large amount of updating to the screen is being performed, and while the screen scrolls.

Special logic has been incorporated into the RunTime program to suppress this noise where this is required. However, since this logic is specific to the Color Graphics Adapter only, the default is for the RunTime Package not to do noise suppression.

Byte 3 of the $OPTION system variable is used to enable or disable the noise suppression logic. A value of HEX(00) (the default) means no noise suppression is performed. A value of HEX(01) means noise suppression is selected. Programs which wish to conditionally select noise suppression should use the $OPTIONS statement in conjunction with the $MACHINE statement. Refer to the NPL Statements Guide, $MACHINE and $OPTIONS, for details on these statements.

6.7 Enhanced Color Graphics Controller (EGA)

The following section discusses how the RunTime operates under MS-DOS with the Enhanced Color Graphics Controller.

6.7.1 Graphics Availability (/G option)

The /G RunTime start-up option is supported on the EGA controller.

NOTE: If a system is using both EGA and Hercules displays (either on separate adapters or "combination" adapters), the /GE option should be specified if EGA mode is desired and the /GH option should be specified if Hercules mode is desired. If the /G option is not being used on an EGA controller the RunTime treats it as a CGA controller.

Refer to Section 6.6 for details regarding the CGA controller. Refer to Sections 6.4 and 6.5 for details on supported monochrome displays and adapters.

6.7.2 Screen Character Set

Downloadable Fonts

The character set for the EGA controller is supported through the use of a downloadable font. The downloadable font file IBMFONT0.EGA is loaded into memory at the time of execution of the RunTime with the /G option. This file must be present or the /G option is ignored.

The NPL utility program EDFONT can be used to modify the downloadable fonts. Refer to Chapter 13 of the NPL Programmer's Guide for details on the Font Editor utility program.

Alternate Character Set (Pixel Graphics)

Use of the alternate character set is fully supported on the EGA controller.

Screen Translation

If the graphic option (/G) is used at the time NPL is executed, the values in $SCREEN are reset at startup time to indicate that no character translation should take place. Developers whose applications modify $SCREEN, under the assumption that the MS-DOS character is in use, should consider the effects of downloadable fonts.

6.7.3 Box Graphics Support

The EGA controller supports "true" box graphics under NPL. To use "true" box graphics, byte 1 of the NPL $BOXTABLE system variable must be set to its default, HEX(00) setting. Setting byte 1 of $BOXTABLE to HEX(01) enables "character" boxes.

Refer to the NPL Statements Guide for details on the use of the $BOXTABLE system variable.

6.7.4 Attributes Support

Under graphics mode the EGA controller does not support the Blink attribute of NPL on a color display. It displays this attribute as a combination of colors. The RunTime does give the ability to change these colors using the NPL variable $OPTIONS. Refer to Section 6.10 for more details on attributes in graphics mode.

6.7.5 Cursor Appearance

Refer to Section 6.10 for more information on the cursor appearance under graphics mode.

6.7.6 Color Support

On EGA color monitors, programmable specification of foreground, background, underline and perimeter colors is supported. Older EGA adapters (with only 64K of display memory) have a limited number of colors to choose from. Refer to Section 6.9 of this Supplement and Chapter 7 of the NPL Programmer's Guide for details.

6.7.7 Use of a Monochrome Monitor on a EGA Controller

There are a variety of monochrome monitors that have the ability to work with an EGA controller. When this type of configuration is used, the RunTime ignores any color specified. In addition, the "blink" attribute functions normally. However, "blink" and "bright" appear as "blink" only. Refer to Section 6.10 for further details on attributes.

6.8 Video Graphics Array Controller (VGA)

The VGA controller is supported by NPL as an EGA controller in graphics mode and as a CGA controller in text mode. Refer to Section 6.7 and 6.6 respectively for more information on these controllers.

6.9 Color Support under NPL

Under the MS-DOS version of NPL, applications can be enhanced by the use of color on various types of monitors and controllers supported by NPL. Color support can be used by an application using $OPTIONS or colors can be selected dynamically by using NPL screen control sequences. The programming considerations to implement color support within NPL applications are discussed in detail in Section 7.4.7 of the NPL Programmer's Guide.

6.9.1 Supported Controllers and Monitors

NPL supports the following controllers and monitors that are intended for use within a color environment. A palette of 16 colors is available.

- A CGA monitor using a Color Graphics Adapter.
- An EGA color monitor using an Enhanced Graphics Adapter (EGA) in graphics mode (/G) or non-graphics mode (no /G). In graphics mode, and EGA adapter must have more than 64K of memory or only four colors are available.
- A VGA color monitor using a Video Graphic Array (VGA) adapter in graphics mode (/G) or non-graphics mode (no /G).

Refer to Section 6.10 for details on graphics mode support under NPL.

NOTE: Various combinations of the above controllers and monitors can be used (including a monochrome display); however, color is only displayed on color monitors. Some monochrome monitors shades of gray, or whatever color the screen phosphor produces, instead of true color. Refer to Section 6.3 for a summary of these options.

6.9.2 Video Attributes

Refer to the Attributes Support section in each of the controller sections of this chapter for information on supported video attributes. Also, refer to Chapter 7 of the NPL Programmer's Guide.

6.10 Graphics Support under NPL

The following section discusses NPL graphics support under MS-DOS.

6.10.1 Invoking

Graphics mode must be specifically requested at the time of execution of the RunTime by use of the /G (graphics) start-up option. To start the RunTime program in graphics mode, enter:

```
    RTI /G[H|E] [progname]
```

When invoking NPL with the graphics mode /G (graphics) option, the RunTime program supports the use of "true" box graphics characters. The RunTime creates these characters on the screen by using information from a downloaded font file. When the RunTime program is executed, it determines the type of graphics board being used (EGA or Hercules) and automatically downloads the appropriate font file, IBMFONT0.EGA for EGA graphics boards or IBMFONT0.HGA for Hercules Monochrome Graphics boards. These font files are only used when the /G mode is used. The /G may be immediately followed by a letter indicating the type of graphics display to use ("E" for EGA, "H" for HERCULES). If the /G option is immediately followed by an "E", only an EGA adapter is used. If the /G option is immediately followed by an "H", only a Hercules adapter is used. This allows for the use of Hercules adapters on systems which have both EGA and Hercules displays (either on separate adapters or combination adapters).

Effects on $MACHINE under Graphics Mode

When graphics mode has been invoked $MACHINE displays the following values:

Byte Number	Description	Allowable Values

3	Monitor Type	E= EGA
		H= Hercules Monochrome Graphics

4	Graphics Enabled	G= "true" box graphics are available

When graphics mode has not been invoked, $MACHINE displays the following values:

Byte Number	Description	Allowable Values

3	Monitor Type	C= Color
		M= Monochrome

4	Graphics Enabled	Blank, HEX(20)="true" box graphics are not available

6.10.2 Supported Controllers

The NPL RunTime Package supports the following graphic monitor and controller combinations in graphics mode.

- Enhanced Graphics Adapter (EGA) with a monochrome monitor.
- Enhanced Graphics Adapter (EGA) with an enhanced color monitor.
- Hercules Graphics Adapter attached to a Monochrome monitor.

To enable the RunTime to load the FONTIBM0.EGA and the FONTIBM0.HGA font files, the files must be in one of the following locations.

- Current directory on the current drive.
- NIAKWA_RUNTIME drive/directory.
- The \BASIC2C directory on the current drive.

If the appropriate file is not found in one of the above locations the /G option is ignored.

These files must also be in the above locations whenever a $SHELL is executed.

These font files contain the full NPL character set and provide the support of pixel graphics.

NOTE: All systems that support VGA graphics include support of EGA graphics as a subset. Consequently, on these systems the graphics mode (/G option) of the RunTime program is always supported.

The PS/2 Model 30 does not support full VGA or EGA graphics, thus the NPL /G option is not supported on the Model 30.

Support of the Hercules Graphics Adapter includes the Hercules Plus Graphic Adapter.

6.10.3 Performance

The use of graphics mode produces somewhat slower screen access than use of the default (text) mode. This difference is most apparent with programs which do large amounts of screen display and/or scrolling.

6.10.4 Cursor Appearance

Under the text mode the cursor is controlled by the computer hardware being used. Most graphic monitors do not support display of a cursor in graphics mode. The RunTime controls the cursor in graphics mode. The RunTime displays a cursor which is a reverse video copy of an entire character under the cursor, producing a block cursor. For replacement of a "blinking" cursor, the RunTime displays a cursor which is a reverse video copy of the bottom few lines of the character displayed. This produces a cursor similar to an underline character. This cursor is fairly visible. However, on displays which use a number of different video modes (especially inverse), the cursor may be less visible than in text mode.

6.10.5 Video Attributes

Depending on the type of graphics board used, "bright" and "blink" video attributes may not be supported on a dot-by-dot basis when in graphics mode. Characters using these video attributes in text mode may not be as easily distinguished from "normal" characters when using the graphics option. The following describes the results displayed when using "bright" or "blink" attributes on supported monitors.

Enhanced Graphics Adapter (EGA) with color:

Bright	Appears bright white on black background.
Blink	Appears red on black background.
Bright/Blink	Appears bright white on red background.

Refer below for more information on using color replacements for attributes.

Enhanced Graphics Adapter (EGA) with monochrome:

Bright	Appears bright.
Blink	Appears blinking.
Bright/Blink	Appears blinking (same as blink only).

Hercules graphics controllers with monochrome:

Bright	Appears inverse video.
Blink	Appears inverse video.
Bright/Blink	Appears inverse video.

In cases where bright or blink attributes are not exactly reproduced (Hercules or EGA color), $OPTIONS byte 24 may be used to indicate that these attributes should appear the same as normal characters. The values that can be used for $OPTIONS byte 24 are:

HEX(00)	Distinguishes blink/bright characters from normal characters using color or inverse video (default).
HEX(01)	Ignores bright attribute, but distinguishes blink from normal characters using color or inverse video.
HEX(02)	Ignores blink attribute, but distinguishes bright from normal characters using color or inverse video.
HEX(03)	Ignores blink and bright attributes.

Color Replacement for Video Attributes

When using the graphics mode with a EGA/COLOR monitor, the RunTime program allows use of replacement colors for video attributes. The number of colors available with an EGA display depends on the amount of memory installed on the EGA card.

NOTE: An EGA card with 64K of memory can only display four colors at one time in the graphics mode used by the RunTime program. These four colors are black, blue, green and white. If more than 64K of memory is available on the card (or VGA is used), all 16 colors are available. On an EGA card with 256K, color replacement is only supported for the blink and bright/blink attribute combinations.

The following is a list of the 16 colors available and their associated HEX codes.

* 0 - BLACK	8 - GRAY
* 1 - BLUE	9 - LIGHT BLUE
2 - GREEN	A - LIGHT GREEN
3 - CYAN	B - LIGHT CYAN
* 4 - RED	C - PINK
5 - MAGENTA	D - LAVENDER
6 - BROWN	E - YELLOW
* 7 - WHITE	F - BRIGHT WHITE

NOTE: The colors marked with an asterisk (*) are the default colors used by the RunTime when color is invoked.

The NPL variable $MACHINE byte 12 can be viewed to determine the number of colors available on the monitor card in use. The following values are possible:

HEX(02) 2 colors - A monochrome display is in use (in graphic or non-graphic modes).

HEX(04) 4 colors - A base memory (64K only) EGA card is in use.

HEX(10) 16 colors - A full memory EGA card is in use,(or a CGA is being used in a non-graphic mode).

When using on an EGA color monitor with less than 64K of display memory, only four colors are available on the screen at one time (colors 0, 1, 4, and 7). Bytes 29 and 30 of $OPTIONS allow definition of alternative colors for each of the four default codes 0, 1, 4, and 7. These bytes should contain:

Byte 29 HEX(xy) replaces color x for black, y for blue (default HEX(0F) replaces blue with bright white).

Byte 30 HEX(xy) replaces color x for red, y for white (default HEX(47) leaves red and white unchanged).

Where x and y can be any one of the sixteen color codes in the above list.

The replacement color defined then takes on the value of the color being replaced. For example, byte 29 is used to define replacement colors for color 0 (black) and color 1 (blue). The default value of byte 29 is HEX(0F). This value is replacing color 1 (blue) with color F (bright white). When color 1 is used after the replacement, this produces bright white and not blue.

The background and foreground colors are determined by the values of $OPTIONS bytes 25, 26, 27, and 28. These values are as follows:

NOTE: Each byte is defined as HEX(bf), where b is the background color, and f is the foreground color, and determines the colors to use for any one combination of bright and blink video attributes. The colors are affected by the selection of replacement colors as explained above.

Byte 25	Used for no bright and no blink. Only supported on EGA controllers with 64K.
	Default, HEX(07) = white on black.
Byte 26	Used for bright and no blink. Only supported on EGA controllers with 64K.
	Default, HEX(01) = bright white on black.
Byte 27	Used for no bright and blink. Supported on all EGA controllers.
	Default, HEX(04) = red on black.
Byte 28	Used for bright and blink. Supported on all EGA controllers.
	Default, HEX(14) = red on bright white

To be compatible with all color EGA monitors, the colors selected in bytes 25 through 28 should only use the color values 0, 1, 4, and 7, with appropriate replacement colors in $OPTIONS bytes 29 and 30 as described previously.

NOTE: Replacement colors affect the entire screen, including characters already displayed. A terminal reset sequence (HEX(020D0C030E) must be printed to the screen before color replacement is performed.

6.10.6 Other Issues

The following section discusses the other issues to be considered when using graphics mode.

Memory Requirements

Use of graphics mode requires a small amount of extra memory for the use of the downloadable font file. When running the /G (graphics) option, the RunTime program requires about 4K more of memory than required in text mode.

Subshells

When programs leave the RunTime environment to execute a DOS program using the $SHELL or invoke (!) command, the RunTime switches from graphics mode to text mode (this is because many DOS applications do not properly handle a screen that is in graphics mode). When this switch takes place, the information on the screen is translated to the closest representation supported by the text mode. This means that true box graphics vanish, and characters which do not have the same appearance in ASCII and NPL character set are transformed to a different appearance. When the $SHELL or invoke (!) command completes, the RunTime switches back to graphics mode and the screen is translated back. There may be some unexpected side effects when the change between modes occurs. In particular:

- Enhanced Graphics (EGA) Color--Blink and bright/blink colors being displayed under graphics mode become blink and bright/blink under text mode. Underline modes change to colors that are appropriate to a CGA display. The aspect ratio of the color display may change.

- Enhanced Graphics (EGA) Monochrome--Characters with the underline and inverse video attributes under graphics mode become underline only under text mode. Characters which are displayed using a 8h by 14v font under graphics mode changes to a 9h by 14v font under text mode. This makes text characters a bit "skinnier" than graphics characters and puts a vertical font line between characters, which are usually a blank space, except for character box values in the range of HEX(C0-DF), for which the right-most vertical font column is duplicated.

- Hercules Monochrome--Bright and blink characters under graphics mode that are displayed by reverse video become bright/blink characters under text mode. Underline and inverse video modes under graphics mode become underline only under text mode. The screen may "bounce" for a short period (less than one second) after changing modes. Some Hercules-compatible boards use a different font (i.e., sans-serif) from the monochrome display when in text mode, so the appearance of even normal ASCII characters may change in this case.

Effects on $SCREEN

If the graphics option (/G) is used at the time NPL is executed, the values in $SCREEN are reset at start-up time to indicate that no character translation should take place. Programs which change the value of $SCREEN under the assumption that the IBM character set is in use need to consider the effects of downloaded fonts.

Use of $BOXTABLE

Character boxes produced using the $BOXTABLE command supersede the "true" boxes if the first byte of $BOXTABLE is not set to HEX(00). The character boxes, if selected, use the box characters HEX(A0-AF) of the downloaded fonts, which are normally single line boxes.

6.10.7 Example Use

The following is a short program designed to present an example of the concepts discussed in this section.

```
0100 DIM X$64
   : M$=$MACHINE
   : IF STR(M$,3,1)<>"E" THEN 110 :REM Options only effect EGA
   : X$=$OPTIONS
   : REM If you like black characters on a white screen...
   : STR(X$,29,1)=HEX(7F)         :REM using palette, for black, display white
   : REM for blue, display bright white
   : STR(X$,30,1)=HEX(40)         :REM leave red as red
                                  :REM for white display black
   : REM now set background and foreground colors for bright and blink
   : STR(X$,28,1)=HEX(14) :REM blink/bright = red on bright white
   : REM (color 1 is bright white due to palette)
   : $OPTIONS=X$                  :REM set new options
   : PRINT HEX(020D0C030E):REM reset to screen puts changes into effect
0110 IF STR(M$,4,1)"G" THEN 120   :REM true boxes available?
   : X$=$BOXTABLE                 :REM Yes!
   : $BOXTABLE = BIN(0)&STR(X$,2) :REM be sure to use them...
0120 REM .....
```

6.11 Plotting Capabilities

Under MS-DOS, use of Niakwa’s Plotter Driver software allows NPL programs to perform plot-like functions to the screen, on monitors which support the equivalent of the CGA display, EGA display or Hercules adapter.

Refer to the Niakwa Scientific and Communications Drivers Package for further details on supported hardware and plotting capabilities.

6.12 Keyboard Characteristics

The standard PC keyboard is substantially different from those used on other implementations of NPL and on the Wang 2200 systems (models 2236DE and 2236DW). The possibility of keyboard differences is resolved through the use of a keyboard translation table, which allows for keyboard remapping. The standard built-in defaults for remapping should be adequate for most applications. However, developers may need to be aware of these differences when converting software designed to run with other implementations of NPL, or with Wang 2200 keyboards.

NOTE: **Keyboard equivalences detailed in this section are the built-in defaults. These values may be modified by the EDKEYBOA utility. Refer to Chapter 13 of the NPL Programmer's Guide or to the NPL Statements Guide, $KEYBOARD, for details.**

The rest of this section explains the logic used in terminal key sequences, the keys which are not equivalent under the MS-DOS implementation of NPL, keyboard characteristics, and the effects of the CAPS LOCK and NUM LOCK keys.

6.12.1 Keys which are not Equivalent

The following keys are not equivalent under NPL.

Underscore Key

If a special function HEX(A0) is defined in the Keyboard Translation Table, the system allows entry of underlined characters as follows (during INPUT or LINPUT):

- The text which is to be underlined is keyed in.
- The arrow keys are used to reposition to the start of the text.
- The defined SF key is pressed once for each character requiring underlining. After each depression, a single character is underlined and the cursor advances one place.

For programs using a KEYIN operation, the key defined as special function HEX(A0) is reported as special function HEX(A0), with no other side effects.

NOTE: By default, the underline key is defined as normal HEX(5F). This is the value necessary for use in the NPL identifier names.

CLEAR

There is no built-in default CLEAR key or equivalent on the keyboards supported on the MS-DOS based computers.

CONTINUE

There is no CONTINUE key or equivalent on the keyboards supported on the PC.

6.12.2 Keyboard Characteristics

The default values for the keyboard used with the MS-DOS version of NPL are built-in to the RunTime programs. If modifications are made to these defaults, they are saved to disk in the file called KEYBOARD.TBL.

Default equivalences for commonly used keys are:

HALT	CTRL-ALT (not modifiable)
EXECUTE	HOME
CANCEL	END
HELP	ESC

The following editing keys are available in edit mode:

Keypad arrows	(NORTH, SOUTH, EAST, and WEST).
Right arrow	Recalls the previous command entry. If a line number is entered before pressing the right arrow key, that specific program line is recalled.
CTRL-F1	Inserts a linefeed within a line of text.
SHIFT-F10	Deletes characters from the current cursor position to the end of line.
INSERT	Inserts a blank space within a line of text or toggles between Insert and Overstrike mode. This behavior depends on the value of byte 44 of $OPTIONS.

DELETE	Deletes a character at the current cursor position.
PG UP	Causes the screen display to shift to the previous screen (if more than 23 lines of text exist) or moves the cursor to the first line of text.
PG DN	Causes the screen display to shift to the next screen (if more than 23 lines of text exist) or moves the cursor to the last line of text.
CTRL-P	Recalls the previous command from the "multi-command" keyboard buffer. Refer to Chapter 5 of the Programmer's Guide for more information.
CTRL-N	Recalls the next command from the "multi-command" keyboard buffer. Refer to Chapter 5 of the Programmer's Guide for more information.
ALT-0	Insert soft carriage return.
ALT- -	Delete soft carriage return.
CTRL-R	Recalls program line or LIN and inserts it at cursor location.

In Edit Mode, SF keys are assigned the following functions:

F5	Moves the cursor to the end of the line being edited.
F6	Moves the cursor down one line.
F7	Moves the cursor up one line.
F8	Moves the cursor to beginning of line being edited.
F9	Erases all text from current position to end of line.
F10	Deletes one character at the current cursor position.
ALT F1	Inserts one space at the current cursor position.
ALT F2	Moves cursor 5 spaces to the right.
ALT F3	Moves cursor one space to the right.
ALT F4	Moves cursor one space to the left.
ALT F5	Moves cursor 5 spaces to the left.
ALT F6	Recalls the current line.

6.12.3 Default Equivalence Table

When determining a key sequence in the default Keyboard Equivalences Table, the following terminology is used. When a "-" (dash) is used in a key sequence, it indicates that the keys should be pressed simultaneously. When a "," (comma) is used in a key sequence, it indicates that the keys should be pressed in sequential order. A "/" is used to separate different key sequences that are assigned to one NPL virtual key. For example, CTRL-x indicates press and hold the CTRL key while pressing x; ESC,x indicates press and release the ESC key, then press x, and; SHIFT-PREV / CTRL-P indicates that both the SHIFT-PREV and the CTRL-P perform the same function.

IBM-Compatible PC Default Keyboard Equivalences Table

NPL Code Key	**NPL Virtual Key**	**MS-DOS Key**
08	BACKSPACE	BACKSPACE
0D	RETURN	ENTER
5F	UNDERSCORE	UNDERLINE
81	CLEAR	?
82	EXECUTE	7/HOME
84	CONTINUE (LOAD)	?
A1	LOAD	CTRL 7/CTRL HOME ALT-1 *
E5	SHIFT ERASE	CTRL-W
FF’A0’xx	UNDERSCORE (DEAD KEY)	?
’00 ... ’09	SF ’0 ... ’9	F1 ... F10
’0A ... ’0F	SF ’10 ...’15	ALT-F1 ... F6
’10 ... ’19	SHIFT SF ’0...’9	SHIFT-F1 ... F10
’1A ... ’1F	SHIFT SF’10...’15	CTRL-F1 ... F6
’42	PREV-SCREEN	9/PG UP
’43	NEXT-SCREEN	3/PG DN
’45	SOUTH	2/SOUTH
’46	NORTH	8/NORTH
’48	ERASE	CTRL-F10
’49	DELETE	./DEL
’4A	INSERT	0/INS
’4C	EAST	6/EAST
’4D	WEST	4/WEST
’4F	RECALL	CTRL-R

IBM-Compatible PC Default Keyboard Equivalences Table

NPL Code Key	NPL Virtual Key	MS-DOS Key
’50	SHIFT-CANCEL	CTRL-1/END CTRL-F7 CTRL-K
’52	SHIFT-PREV-SCREEN	CTRL-9/PG UP ALT-9 * CTRL-P
’53	SHIFT-NEXT-SCREEN	CTRL-3/PG DN ALT-3 * CTRL-N
’55	SHIFT-SOUTH	ALT-2 *
’56	SHIFT-NORTH	ALT-8 *
’59	SHIFT-DEL (LINE DEL)	ALT-- *
’5A	SHIFT-INS (LINE INS)	ALT-0 *
’5C	SHIFT-EAST (EAST-5)	CTRL-6/EAST ALT-6 *
’5D	SHIFT-WEST (WEST-5)	CTRL-4/WEST ALT-4 *
’5F	D TAB	ALT-F10 CTRL-T
’7C	GL	ALT-F9 CTRL-G
’7D	SHIFT-GL	CTRL-F9 CTRL-Z
’7E	TAB	TAB
’7F	SHIFT-TAB	SHIFT-TAB
’E1	HELP	ESC ALT-F8
’F0	EDIT	1-END ALT F7

NOTE: A question mark (?) indicates that no key is assigned to the NPL code.

Entries followed by a "*" indicates that the key refers to the standard numeric key, NOT the keypad, even if the NUM LOCK is on.

Several of the NPL Virtual Keys have two or three assigned keys on the PC. Here, all keys have been mapped and are active.

HINT: The keyboard should NOT be placed in the "NUM LOCK" mode (so that arrow keys are available without shifting). This is the power-up default.

6.12.4 Keyboard Lock Status

The standard MS-DOS keyboard has two LOCK keys which affect the current "mode" of the keyboard, CAPS LOCK and NUM LOCK.

When CAPS LOCK is not selected, the alphabetic keys produce lowercase letters if unshifted, and uppercase when shifted. When CAPS LOCK is selected, this is reversed: the alphabetic keys (A-Z) return uppercase letters when unshifted, and lowercase letters when shifted.

When NUM LOCK is not selected, the keypad keys produce the cursor motion and functions if unshifted, and numbers (0-9) when shifted. When NUM LOCK is selected, this is reversed, the keypad returns numbers when unshifted, and cursor motion and functions when shifted.

The MS-DOS version of the RunTime program has the ability to maintain the current status of both LOCK keys on the 25th line of the display (which may not be accessed by application programs). The display of this status information is optional. The 2nd byte of the $OPTION system variable is used to enable or disable the LOCK status displays. A value of HEX(00) means no LOCK status is displayed. A value of HEX(01) (the default) means LOCK status is displayed.

Lock status is only available where direct video access is used. In particular, it is not available when using the remote (/R) option.

6.13 Terminals Support with REDIRECT

The following serial terminals are supported by the redirect feature and can only be used with this feature.

Terminal	**Keyboard File**	**Screen**	**Font File**	**Keys File**
Altos III	KEYBOARD.AL3	SCREEN.AL3	NA	NA
Altos V	KEYBOARD.AL5	SCREEN.AL5	VTFONT.AL5	NA
Bull HDS 1	KEYBOARD.WY5	SCREEN.WY5	NA	NA
Bull HDS 3	KEYBOARD.VT2	SCREEN.VT2	VTFONT.VT2	VTKEYS.VT2
DEC VT100	KEYBOARD.VT1	SCREEN.VT1	NA	NA
DEC VT200 Series	KEYBOARD.VT2	SCREEN.VT2	VTFONT.VT2	VTKEYS.VT2
IBM 3151	KEYBOARD.I31	SCREEN.I31	NA	NA
NCR 4970	KEYBOARD.VT2	SCREEN.VT2	VTFONT.VT2	VTKEYS.VT2
Spectrix SPX 701	KEYBOARD.SPX	SCREEN.SPX	NA	NA
Wang 2110A	KEYBOARD.W21	SCREEN.W21	NA	NA
Wang 2x36 DE/DW	KEYBOARD.W22	SCREEN.W22	NA	NA
Wyse 50	KEYBOARD.WY5	SCREEN.WY5	NA	NA
Wyse 60, 150, 160	KEYBOARD.WY6	SCREEN.WY6	WYFONT.WY6	NA
Wyse 370	KEYBOARD.WY3	SCREEN.WY3	WYFONT.80 WYFONT.132	WYKEYS.WY3

For details on the topics covered in this chapter as they pertain to the specific supported serial terminals, refer to Appendix D of the NPL Programmer’s Guide.

CHAPTER 7

MULTI-USER CAPABILITIES

7.1 Overview

Multi-user capabilities are not supported on PCs under MS-DOS.

Section 7.2 discusses device sharing and "hogging".

Section 7.3 discusses global partitions.

Section 7.4 discusses terminal identification.

Section 7.5 discusses intertask communications.

Section 7.6 discusses the user count.

Section 7.7 discusses the single user MS-DOS RunTime on a Novell NetWare network.

If used on the single-user PC under MS-DOS, NPL multi-user statements behave as described in Section 7.2 through 7.6.

7.2 Device Sharing and "Hogging"

When using $OPEN, no operation is performed.

Using $CLOSE with no device specified causes a logical close of all open files or devices. Otherwise, no operation is performed.

7.3 Global Partitions

Global partitions are not supported by NPL. However, global variables are supported. Applications that depend upon the global partitions for record locking must be modified to use a disk-based record-locking system. MS-DOS is a single-user operating system, thus there is no need for record locking. Refer to the appropriate operating environment addenda and Section 4.6 of the NPL Programmer's Guide for more information.

7.4 Terminal Identification

This feature operates as follows under MS-DOS.

#TERM always returns a value of 1.

#PART always returns a value of 1.

7.5 Intertask Communications

The following section discusses intertask communication options for NPL under MS-DOS.

7.5.1 DATE and TIME

The DATE and TIME functions operate normally. Refer to the NPL Statements Guide, DATE, TIME for details on these functions.

7.5.2 System Messages

There is no capability for system messages under MS-DOS. $MSG can be set and examined.

7.6 User Count

Multi-user operation is not supported under MS-DOS. The user count is always 1.

7.7 Use on Networks

A single-user NPL RunTime operating on a PC can be attached to a Novell NetWare network. The RunTime does not attempt to enforce the network user limit as long as security is passed from the local drive (i.e., typically C:). This is accomplished by setting the NPL_SECURITY environment variable to the local drive.

HINT: This is very useful for laptop systems that are typically operated remotely and, occasionally, are attached to an in-house network.

For example, to set the NPL_SECURITY environment variable to drive C:, enter the following:

```
NPL_SECURITY = C:\
```

Refer to Section 2.5.4 for details on the NPL_SECURITY environment variable.

When security is passed in this way, the $MACHINE bytes (6 and 25-26) indicating "users at login" are not available and contain a value of 0. Refer to the NPL Statements Guide for details on the $MACHINE environment variable.

When used in this way, multi-user capabilities are present. Refer to Chapter 5 in the Novell NetWare Addendum.

CHAPTER 8

PLATFORM-SPECIFIC LANGUAGE FEATURES

8.1 Overview

There are a series of NPL statements which are specific to the type of hardware/operating environment in which they are executed. This chapter discusses the language features specific to the NPL under MS-DOS.

Section 8.2 discusses the environment-specific statements.

Section 8.3 discusses background partition support.

Section 8.4 discusses memory management.

8.2 MS-DOS Specific Statements

This section discusses the NPL language features that are specific to the MS-DOS operating system. Refer to the appropriate operating environment-specific addendum for other MS-DOS based operating environments.

8.2.1 $MACHINE

The $MACHINE system variable contains information about the hardware environment in which the RunTime is executing. In the MS-DOS version, the following platform specific values are set:

Byte 1	RunTime Version : "I" for MS-DOS.
Byte 2	Hardware Manufacturer Code - HEX(00) for MS-DOS.
Byte 3	Monitor Type - Refer to Chapter 6 for available types .
Byte 4	Graphics Enabled ("G" = graphics enabled; " " = no graphics available).
Byte 5	Hardware Model Code. HEX(FF)= IBM PC. Codes may vary by manufacture.
Byte 7	RunTime type in use - "I" = Interpretive version; "P" = Non-interpretive version.
Byte 8	Display width in binary. Always HEX(50) on MS-DOS 80-column screens.
Byte 10	Math co-processor present. A HEX(00) indicates that no co-processor is present. A HEX(01) indicates that a co-processor is present and may be used for some math operations by setting byte 16 of $OPTIONS to a value of HEX(01).
Byte 12	Number of colors available. Refer to Section 6.8 for details on color support in NPL under MS-DOS.
Byte 21	Contains the maximum number of entries (in binary) allocated to the handle table (in K) during a RunTime session.

Byte 22	Indicates the XMS usage as specified in the /M and /U startup options. Contains several bit flags (0= no; 1= yes) as follows: HEX(01) /M option specified (HMA use permitted). HEX(02) /U option specified (UMB use permitted). Refer to Section 8.4 for details. The remaining flags are all 0 unless one of the above is set:	
	04	An XMS driver is installed
	08	RTI/RTP has allocated the HMA area
	10	RTI/RTP has used a UMB for storing code
	20	RTI has used a second UMB for storing interpreter-only code
	40	UMB's have been allocated to extend the user partition
	80	Unused; always 0
Byte 23	Contains the current row position of the mouse pointer when a mouse event occurs (if it is on screen).	
Byte 24	Contains the current column position of the mouse pointer when a mouse event occurs (if it is on screen).	
Byte 29	Indicates whether keyboard mouse events are supported.	
	HEX(00)	Default; no mouse available.
	HEX(01)	Mouse available. Under DOS 3.0 and greater. HEX(01) only occurs if the NPL /K startup option is specified and a mouse driver is installed and detected.

NOTE: $MACHINE is a 64-byte variable. The above bytes are specific to the MS-DOS Run-Time. For a complete description of all bytes within the $MACHINE system variable, refer to the Statements Guide, $MACHINE.

8.2.2 $OPTIONS

The NPL Runtime allows for the inspection or modification of the $OPTIONS system variable, which consists of a variety of options. Bytes which have specific meanings under the MS-DOS version of NPL include:

Byte 1:	Replacement attribute for the underline character to be used on color monitors. The default value is HEX(1F), bright white on blue background.

Byte 2:	Switch to determine whether or not keyboard status information (CAPS LOCK and NUM LOCK indicators) is to be displayed on line 25 of the monitor (IBM version only). A value of HEX(00) disables the status display. A value of HEX(01) enables the status display.
Byte 3:	Switch for "noise" suppression option on color monitors. A value of HEX(00) (the default) indicates that no "noise" suppression is to take place. A value of HEX(01) indicates that noise suppression is to take place.
Byte 7:	Keyboard translation complex key lead value. When this value is received by the keyboard handlers from the native operating system, a complex key sequence is assumed to follow.
Byte 9:	Keyboard translation complex key trailing value. After receiving a complex key sequence from the native operating system, if this byte is not HEX(00), the keyboard handlers wait for a trailing code to finish the sequence.
Byte 10:	Alternate keyboard translation complex key lead value. When this alternate value is received by the keyboard handlers from the native operating system, a complex key sequence is assumed to follow. A value of HEX(00) means no alternate code is allowed.
Byte 11:	Alternate keyboard translation complex key trailing value. After receiving an alternate complex key sequence from the native operating system, if this byte is not HEX(00), the keyboard handlers wait for a trailing code to finish the sequence.
Byte 16:	Switch for using math co-processor. A value of HEX(00), the default, indicates that the math co-processor should not be utilized. A value of HEX(01) indicates that the math co-processor should be utilized.
Byte 20:	"Raw" output in remote mode. The default value of HEX(00) indicates that "raw" output mode should not be used. A value of HEX(01) indicates that "raw" output mode should be used. "Raw" output mode is applicable only on PC monitors under MS-DOS when the non-interpretive RunTime is operating in "remote" mode. Remote mode is used when the "R" startup option is specified.
Byte 21:	Controls display of "bright" attribute on terminals where "normal" is actually "dim". Refer to Chapter 6 for information on specific controller/monitors where this feature is supported. A value of HEX(00) indicates that "bright" is displayed as "bright" and "normal" is displayed as "dim". A value of HEX(01) indicates that "bright" is displayed as "dim" and "normal" is displayed as "bright".

Byte 22:	Provides power-on default background/foreground color selection for supported color terminals. Refer to Chapter 6 for details. HEX(00) is the default.
Byte 23:	Provides power-on default perimeter/underline color selection for supported color terminals. Refer to Chapter 6 for details. HEX(00) is the default.
Byte 24:	Controls attribute display on PC monitors in graphics (/G) mode. Refer to Chapter 6 for details.
Byte 25:	Color replacement value for no bright, no blink in EGA graphics mode. HEX(01) indicates white on black. Refer to Chapter 6 for details.
Byte 26:	Color replacement value for bright, no blink in EGA graphics mode. HEX(01) indicates bright white on black. Refer to Chapter 6 for details.
Byte 27:	Color replacement value for no bright. Refer to Chapter 6 for details. HEX(04) is the default.
Byte 28:	Color replacement value for bright. Refer to Chapter 6 for details. HEX(14) is the default.
Byte 29:	Allows for remapping of available colors for video attribute replacement in graphics mode on terminals with less than 64K and only four colors. The format is HEX(xy), where x replaces color for black and y replaces blue. HEX(0F) replaces blue with bright white. Refer to Chapter 6 for details.
Byte 30:	Allows for remapping of available colors for video attribute replacement in graphics mode on terminals with less than 64K and only four colors. The format is HEX(xy) where x replaces red and y replaces white. HEX(47) is the default
Byte 39:	Used to specify that no implicit $OPEN is to be performed on DATA LOAD BA and DATA LOAD BM when the platter had not been explicitly hogged by any $OPEN. HEX(00) is the default
Byte 46	Allows customization of the HELP processor’s use of keys HEX(01) - 0 Return = TAB/EAST HEX(01) - 1 Return = EXEC

NOTE: $OPTIONS is a 64-byte variable and must be treated as such or unpredictable results may occur. Refer to the Statements Guide, $OPTIONS, for details on the exact syntax and use of this statement, as well as the contents of the remaining bytes of the variable.

8.2.3 $PSTAT

Refer to the NPL Statements Guide, $PSTAT, for details on the exact syntax and use of this statement.

8.2.4 $SHELL

The changing of default directories and default drives through the Invoke process is permanent. Commands such as:

```
!cd \ARPROGS
```

or

```
10 $SHELL "cd \ARPROGS"
```

changes the default directory to \ARPROGS as indicated, and the current directory remains set to \ARPROGS when control is passed back to the RunTime program. Similarly:

```
10 $SHELL "E:"
```

changes the default drive designation.

NOTE: To illustrate, assume that the following device equivalence has been set up:

```
$DEVICE(/D40)="PLATTER1.BS2"
```

Device address D40 is set up as diskimage file PLATTER1.BS2 on the current drive and directory, since no other drive designation or directory path is indicated. If the Invoke command is used to change directories without changing back to the original directory when the Invoke process is completed, further attempts to reference PLATTER1.BS2 refer to the PLATTER1.BS2 file in the new drive\directory. If no such file exists, this results in an error P48 - Illegal Device Specification.

HINT: Due to the problems associated with changing directories or drives through the Invoke process, it is recommended that this procedure not be used under program control. Instead, a better place to perform these commands is in a batch file.

The return code variable, if used, always contains a value of HEX(00) for each of the first two bytes. Refer to the NPL Statements Guide for more information on $SHELL.

8.3 Background Partition Support

Background partitions are not supported under MS-DOS. $RELEASE TERMINAL performs no operation in this environment. For information regarding background partitions, refer to the NPL Statements Guide, $RELEASE TERMINAL.

8.4 Memory Management

This section discusses Release IV memory allocation as it pertains to the MS-DOS implementation of NPL. For additional information concerning RunTime memory usage, refer to Chapter 3 of the NPL Programmer's Guide and Section 4.5 of this Supplement.

8.4.1 MS-DOS Considerations

All NPL program code and defined variables reside within a section of memory defined as the "user partition". Under Release IV this "partition" includes all modules currently on memory.

Available memory in the RunTime partition can be determined with the use of the SPACEF and SPACEW functions.

Under the MS-DOS version of NPL, the maximum size of the user partition is reported by the SPACEW function. The amount of memory currently available within the user partition is returned by the SPACEF function.

The amount of available memory under MS-DOS can be increased with the use of XMS memory. Refer to Section 8.4.2 for information on XMS memory usage.

8.4.2 XMS Memory Usage

The use of XMS memory can result in up to 200K additional memory being available to NPL applications.

XMS memory is memory that resides between 640K and 1MB. The RunTime under MS-DOS can use two types of XMS memory:

HMA memory (High Memory Area)

This is a 64K area that resides just above the 1MB boundary. HMA memory is available of both 80286 and above-based machines that have at least 1MB of physical memory.

UMB memory (Upper Memory Block)

UMBs can exist on 80386 and above machines that have at least 1MB of physical memory. UMBs reside between 640K and 1MB. Depending on the hardware and software configuration, the number and size of UMBs available may vary widely.

Memory Management Software Required

Use of the XMS memory requires that a memory manager be installed on the PC where XMS memory is to be used. NPL supports the use of the following memory management products (others may work but they have not been tested by Niakwa):

HIMEM.SYS from Microsoft may be used for HMA memory support on 80286 and above machines. HIMEM.SYS is available with Microsoft Windows 3.1 and MS-DOS 5.0 and above. HIMEM.SYS is installed as a device driver which uses an entry in CONFIG.SYS. No special options must be specified when installing HIMEM.SYS. Refer to Microsoft’s documentation for further details.

QEMM.SYS by Quarterdeck may be used for HMA and UMB support on 80386 and above machines. QEMM.SYS is installed as a device driver using an entry in CONFIG.SYS. Use of the RAM option is recommended to maximize the amount of memory available. Refer to Quarterdeck’s documentation for further details on installing and configuring QEMM.SYS.

The Quarterdeck VIDRAM program (included with QEMM) may be used to gain additional memory on systems where EGA or VGA controllers are in use, but where use of graphics capabilities is not required. Graphics capabilities are used by the RunTime when:

- The /G startup option is specified.
- The PLOT driver is used.
- In addition, applications that use external subroutines may directly access graphic capabilities.

NOTE: Use of the VIDRAM extends the effective size of conventional memory past 640K. No option (/M) is required to use this memory.

Refer to Quarterdeck's documentation for further information on installing and configuring VIDRAM.

Version 5.1 and above of 386MAX by Qualitas may be used as an alternative to QEMM.SYS. 386MAX provides both HMA and UMB support on 80386 and 80486 machines. Refer to Qualitas' documentation for information on installing and configuring 386MAX.

Startup Options for Enabling XMS Memory Utilization

By default, the RunTime does not use XMS memory. The following RunTime startup options are used to access XMS memory.

The /M option enables the use of the HMA.

The /U option enables the use of UMBs.

For example:

```
RTP /M /U
```

enables use of both HMA and UMBs.

NOTE: Enabling XMS memory does not guarantee that it is available and can be used; refer to the Limitations section below for details.

Memory Utilization

XMS memory is used by the RunTIme as follows:

Block	RTI	RTP
1	12K in HMA or UMB	58K in HMA or UMB
2	53K in UMB	54K in UMB
3	53K in UMB	
4	62K in UMB	

HINT: Since RTI's use of HMA memory is not efficient, we recommend that HMA memory be used to load DOS into high base memory for use by RTI or other programs.

The total amount of memory available to the application is returned in the SPACEW and SPACEF system variables as in previous revisions.

XMS memory that is allocated to the user partition is first allocated to the NPL program code. Then, any remaining XMS memory is allocated to variable space. Any program code or variables that cannot be stored in XMS memory are allocated to base memory until base memory is filled.

Rarely, an application may encounter an unexpected A01 - Memory Overflow error when attempting to dimension a large variable. This error can only occur when all of the following conditions are met:

1. There is sufficient XMS memory to store the variables beyond the program text.
2. All base memory has been allocated.
3. The remaining XMS memory is fragmented such that the total amount of XMS memory (as reported by SPACEF) exceeds the desired size of the variable, but no single fragment is large enough to store the variable (variables must be stored in contiguous memory).

This is an unusual combination of circumstances that typically may only be encountered by an application that uses SPACEF to determine the size of variable or that has a fairly small code size and that dimension many large variables and few small variables.

HMA memory is always used first if enabled and available.

The utilization of XMS memory is stored in Byte 22 of the $MACHINE system variable. Refer to the NPL Statements Guide and Section 8.2.1 for more information on $MACHINE.

Limitations

The following limitations apply:

- XMS memory may be used by other products. The RunTime conforms to the XMS version of the 4.0 specification and checks for the availability of XMS memory before attempting to allocate it. If any portion of XMS memory is allocated by another program, the RunTime simply does not use that portion.

NOTE: In some cases, it may be advantageous to allow another program to use XMS memory instead of RunTime. For example, MS-DOS 5.0 and 6.0 can be loaded into HMA memory. Refer to the MS-DOS manuals for details on loading DOS high. By allowing DOS to use the HMA memory, more memory is actually available for non-NPL applications in base memory.

- The hardware and BIOS in use must be suitable for using XMS memory. Check manufacturer's specifications. Typically the XMS memory manager detects and reports incompatibilities.

- Some (usually older) software products that use XMS memory may not conform to XMS specifications. If using such a product, unpredictable results may occur if XMS memory is enabled by the RunTime. If this situation occurs, do not enable XMS memory use by the RunTime until the problem with the other product has been resolved. Many older products may have recent updates that conform to XMS specifications.

- Troubleshooting problems as described in point C above, or configuring systems to provide the maximum XMS memory can be quite complex. Developers are cautioned not to make specific claims or promises to users based on this feature unless they either have complete control over the user's hardware and software configuration or are prepared to gain the expertise required.

CHAPTER 9

COMPILER OPERATION

9.1 Overview

This chapter provides an overview of the general operation of the NPL Compiler (B2C) on an IBM-compatible PC running under MS-DOS. For a complete discussion of the NPL Compiler, refer to Chapter 14 of the NPL Programmer's Guide.

Section 9.2 discusses how to invoke the NPL Compiler.

Section 9.3 discusses file-naming conventions under the MS-DOS operating system.

Section 9.4 discusses the print devices available under the NPL Compiler.

Section 9.5 discusses the use of batch files to operate the NPL Compiler.

9.2 Invoking the Compiler

The NPL Compiler may be invoked in one of two ways. The first and most common form of invoking the compiler is using the command line method.

For example:

```
    B2C /SRCLOC C:\basic2c\2200disk.bs2 PROG1 PROG2
```

the designation of "B2C" at beginning of the MS-DOS command line, refers to the name of the NPL compiler and causes MS-DOS to invoke it. Upon execution, the compiler inspects the balance of the command line to determine the desired compiler options and the list of input programs to compile. All compiler options are discussed fully in Chapter 14 of the NPL Programmer's Guide.

The second method of invoking the compiler is to simply enter "B2C" at the MS-DOS command line with no specified programs to compile. This implicitly invokes the use of the user-friendly compiler display. This display provides an easy-to-use, alternate method of specifying compiler parameters. Refer to Section 14.19 of the Programmer's Guide for a detailed explanation of the user-friendly compiler display.

9.3 File Naming Conventions

File naming conventions vary from one operating system to the next. This section discusses the MS-DOS conventions and their implications to NPL.

9.3.1 Supported Characters

The file-naming conventions of MS-DOS are not entirely compatible with the names of files within NPL diskimages. Specifically, within diskimages, filenames may be any eight characters, with no restrictions. MS-DOS also allows eight-character filenames, but the permitted character set is restricted. Only the letters A-Z, digits 0-9, and a few additional graphic characters are allowed. Exactly which characters are allowed, depends on the release of the operating system.

Consequently, problems may arise when the compiler attempts to take the name of a program within a diskimage (example: "GL G/L > ") and create a .SRC file with "the same name" under the MS-DOS operating system, since "/" and "> " have special meanings in MS-DOS file names. This problem has no general solution, but, if a system does not use too "strange" a subset of ASCII on filenames in the diskimage, it can be circumvented with the TRANSLATE option.

The TRANSLATE Option

The TRANSLATE option is used to inform the compiler of the assumed character equivalences between MS-DOS filenames and filenames within a diskimage file. The TRANSLATE verb is followed by a series of groups of four hex digits. Each group specifies a from/to pair of hexcodes, the first two hexdigits representing the MS-DOS character, the second two representing the equivalent character to be used for filenames within diskimage files.

For example:

```
/TRANSLATE 5F20
```

informs the compiler that underline characters (HEX(5F)) in MS-DOS filenames are equivalent to space characters (HEX(20)) in names of files within NPL diskimage files. Consequently, whenever the compiler needs to place names of files from within NPL diskimage files directly into an MS-DOS directory, it replaces the embedded blanks in the filenames from within the diskimage file with underlines. Conversely, when the compiler needs to place an MS-DOS filename into a diskimage it replaces underlines with spaces.

For legibility, groups of codes may be separated by a decimal point (".").

For example:

```
/TRANSLATE 5F20.7B3C.7D3E.=%=/
```

Defines the following equivalences (where D.I. stands for Diskimages):

MS-DOS (HEX)	5F	7B	7D	25
MS-DOS (display)	-	{	}	%
Filenames in Diskimage (display)		<	>	/
Filenames in Diskimage (HEX)	20	3C	3E	2F

NOTE: Trailing spaces in a filename are not considered to be part of the name. For obvious reasons, use of filenames which use only upper and lower case A-Z, 0-9 and space is preferred in the MS-DOS environment and is strongly encouraged.

For a complete discussion of the NPL Compiler, please refer to Chapter 14 of the NPL Programmer's Guide.

9.3.2 Case Sensitivity

Since the NPL Compiler may compile from ASCII text files, the fact that MS-DOS is case insensitive must be considered when compiling from ASCII text files to NPL diskimage files.

If program file names in diskimage files contain lowercase letters, the LCASE option must be used to generate the correct file name in the diskimage file when compiling from ASCII files to a diskimage. Refer to Chapter 14 of the NPL Programmer's Guide for a discussion of the LCASE option.

9.3.3 Wildcard Usage

In addition to single program names, the compiler accepts "wildcard" names. The presence of a wildcard name in the input program list causes the compiler to compile all programs in the specified directory or diskimage (as specified with SRCLOC) that "match" the wildcard pattern. A wildcard name is any program name which contains at least one of the "?" or "*" wildcard characters.

- A "?" in a wildcard matches any single character in the same position of the input program name.
- A "*" in a wildcard matches any characters or no characters to the end of the program name.

For example:

```
"AR?UPD"
```

matches "AR1UPD", "ARXUPD", "AR@UPD", but not "AR1UPD2".

```
"AR*"
```

matches all file names beginning with "AR".

In the following example:

```
B2C /SRCLOC A: /OBJLOC platter1.bs2 *
```

compiles all programs on the 320K format diskette in drive A, placing compiled output in diskimage file PLATTER1.BS2 in the current directory.

```
B2C /SRCLOC A: /OBJLOC platter1.bs2 AR??001
```

compiles all programs with the first two characters "AR" and characters 5-7 as "001" from the 320K format diskette in drive A to diskimage file PLATTER1.BS2.

When wildcard-scanning a directory, the compiler automatically adds the .SRC extension. When wildcard-scanning a diskimage, the scanner ignores data files even if they match the wildcard pattern.

9.3.4 Pathnames

The NPL Compiler (B2C) assumes that all input and output files reside in the current directory. This is modifiable by specifying an alternate pathname in the compiler command line.

For example:

```
B2C /SRCLOC C:\progs\platter1.bs2 /OBJLOC C:\progs\platter2.bs2 *
```

locates the "progs" directory on the C drive and compiles all the programs found in PLATTER1.BS2, into PLATTER2.BS2 in the "PROGS" directory.

Along with specified directories, the compiler also respects all MS-DOS drive designators (i.e., A: B: C:).

For example:

```
B2C /SRCLOC A: /OBJLOC C:\progs\platter1.bs2 *
```

compiles all programs from the "raw" 320K diskette in drive A and places all compiled output in the diskimage file PLATTER1.BS2 in the "PROGS" directory on the C drive.

NOTE: Under MS-DOS, the compiler is only capable of addressing "raw" 320K diskettes from any diskette device on the system, unlike the RTI and RTP programs which have the capability of addressing "raw" 360K, 720K, 1.2MB, 1.44MB, and 2.88MB diskettes.

Output for any of the compiler operations can be discarded by specification of the null device as the output parameter. Under MS-DOS, the null device must be specified as /dev/nul.

For example:

```
B2C /SRCLOC \progs\platter1.bs2 /OBJLOC /dev/nul /LSTLOC \source
/LSTFORMAT 2200 *
```

creates files in 2200 atomized format, directly from a diskimage file containing already compiled programs (\progs\platter1.bs2). The /dev/nul designator for the /OBJLOC option is used to suppress the generation of a second set of compiled programs. Refer to the discussion on the LSTLOC option in Section 14.9 of the NPL Programmer's Guide.

9.4 Print Devices

Hardcopy listings may be produced by specifying LSTLOC as the name of a native operating system print-device. The following device designations are valid print-device names for the operating system.

/LSTLOC lptx	Direct listing to parallel printer. The "x" indicates the number of the parallel printer port to use.
/LSTLOC comx	Direct listing to a serial printer. The "x" indicates the number of the com port to use.
/LSTLOC /dev/nul	Suppresses listing.
/LSTLOC \pathname	Store source listing for each file in the specified directory. Filenames are derived from the input program name, with an extension of .LST or .SRC, depending on the /LSTFORMAT designation.

/LSTLOC \pathname\filename	Concatenates all program listings into the specified filename (with a .SRC file extension) in the specified directory.

9.5 Batch Files

Batch files are functions which allow the user to set up and execute files containing MS-DOS commands and parameters. Batch files can be used to invoke the NPL Compiler.

For example, the following compiler command line example:

```
B2C /SRCLOC platter1.bs2 /OBJLOC platter2.bs2 /LSTLOC . /LSTFORMAT
.src AR*
```

can be set up as a batch file called COMPAR.BAT (the .BAT extension is required), and then be executed from the command processor by simply entering the name of the batch file:

```
COMPAR
```

In this case, the batch file would perform in exactly the same way as the example command line from which it was taken.

It is possible to generalize batch files so that some options may be entered during execution of the batch file. This is handy for specifying the list of input program names to compile. This can be done by a simple modification to the example above. The modification would be to replace the literal program name designation with a replaceable parameter value ("%N") where N is a number from 1 to 9:

```
B2C /SRCLOC platter1.bs2 /OBJLOC platter2.bs2 /LSTLOC . /LSTFORMAT
.src %1 %2 %3 %4 %5 %6 %7 %8 %9
```

This batch file would now be able to accept up to nine source program names.

For example:

```
COMPAR AR* GL101 OE*
```

compiles the following sets of programs:

- All programs with the prefix "AR"
- Program GL101

- All programs with the prefix "OE"

The same compiler options are used for all programs.

9.5.1 Example Batch Files

Niakwa has provided some batch files for your convenience (some developers may want to set up their own). These examples may be invoked from the MS-DOS command processor by name. These batch files were designed to meet the most frequent requirements of the NPL developer.

Following are the four batch files Niakwa has supplied:

B2CA.BAT

```
B2C /DISPLAY on /SRCLOC A: /OBJLOC platter1.bs2 /TRANSLATE 5F20.252F
%1 %2 %3 %4 %5 %6 %7 %8 %9
```

The above batch file will compile input programs from a raw diskette in 320K format, creating p-code files in UNSCRAMBLED format which will be placed in the diskimage file PLATTER1.BS2 in the current directory, displaying any compiler warning messages on the screen. Note that the translate option has no effect since conversion of filenames between MS-DOS and filenames within a NPL diskimage is not performed. However, if the options are changed during the compiler display, conversion of filenames between filenames within NPL diskimages and MS-DOS filenames would be performed for the character pairs " " (NPL) and "_" (MS-DOS), and "/" (NPL) and "%" (MS-DOS).

Up to nine filename wildcards may be specified when executing this batch file. These filename wildcards, if specified, will appear on the Source Programs line of the generated display.

B2CB.BAT

```
B2C /DISPLAY on /SRCLOC A: /OBJLOC /dev/nul /LSTLOC . /LSTFORMAT .src
/WARNINGS off /TRANSLATE 5F20.252F %1 %2 %3 %4 %5 %6 %7 %8 %9
```

The above batch file will compile input programs from a raw diskette in 320K format, creating ASCII text files in the current directory, which may be edited and resubmitted to the compiler. Note that conversion of filenames between filenames within NPL diskimages and MS-DOS filenames will be performed for the character pairs " " (NPL) and "_" (MS-DOS), and "/" (NPL) and "%" (MS-DOS) when creating the text files.

Up to nine filename wildcards may be specified when executing this batch file. These filename wildcards, if specified, will appear on the Source Programs line of the generated display.

B2CC.BAT

```
B2C /DISPLAY on /REM$ on /OBJLOC platter1.bs2 /TRANSLATE 5F20.252F %1
%2 %3 %4 %5 %6 %7 %8 %9
```

The above batch file will compile input programs in ASCII format from the current directory, creating p-code files in UNSCRAMBLED format which will be placed in the diskimage file PLATTER1.BS2 also in the current directory, compiling any statements which start with a "REM $ PC", and displaying any compiler warning messages on the screen. Note that the translate option is used when determining the name of the filename in the PLATTER1.BS2 diskimage, based on the .SRC files used as input.

Up to nine filename wildcards may be specified when executing this batch file. These filename wildcards, if specified, will appear on the Source Programs line of the generated display.

B2CD.BAT

```
B2C /DISPLAY on /LSTLOC A: /OBJLOC /dev/nul /LSTFORMAT 2200 /WARNINGS
off /TRANSLATE 5F20.252F %1 %2 %3 %4 %5 %6 %7 %8 %9
```

The above batch file will compile input programs in ASCII format from the current directory, creating output programs in 2200 atomized format directly on a 320K diskette, which may then be ported over to the 2200. Note that if the MS-DOS filenames contain "_" or "/" characters, the names of the files within the diskimage will automatically replace any "%" characters with the "/" character, and replace the"_" with a " ".

Up to nine filename wildcards may be specified when executing this batch file. These filename wildcards, if specified, will appear on the Source Programs line of the generated display.

9.5.2 Example Compiles

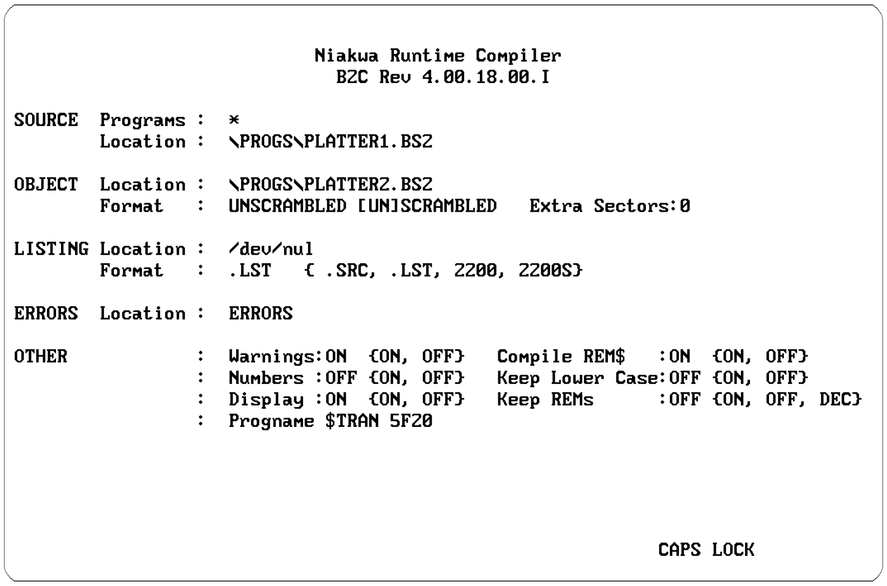
Chapter 14 of the NPL Programmer's Guide describes each of the options available for use when compiling programs. The remainder of this chapter provides several examples of commonly used compilations, indicating both the particular compiler command line and the compiler display. A brief discussion of what each compile is accomplishing, as well as an explanation of each compiler option, is also included.

From a Compiled Diskimage To A Compiled Diskimage

Command line:

```
B2C /DISPLAY on /SRCLOC \progs\platter1.bs2 /OBJLOC \progs\plat-
ter2.bs2 /ERRLOC errors /REM$ on *
```

Display



```
                             Niakwa Runtime Compiler
                               B2C Rev 4.00.18.00.I

SOURCE  Programs :  *
        Location :  \PROGS\PLATTER1.BS2

OBJECT  Location :  \PROGS\PLATTER2.BS2
        Format   :  UNSCRAMBLED [UN]SCRAMBLED   Extra Sectors:0

LISTING Location :  /dev/nul
        Format   :  .LST   { .SRC, .LST, 2200, 2200S}

ERRORS  Location :  ERRORS

OTHER            :  Warnings:ON  {ON, OFF}   Compile REM$    :ON  {ON, OFF}
                 :  Numbers :OFF {ON, OFF}   Keep Lower Case:OFF {ON, OFF}
                 :  Display :ON  {ON, OFF}   Keep REMs      :OFF {ON, OFF, DEC}
                 :  Progname $TRAN 5F20



                                                             CAPS LOCK
```

Discussion:

The above example indicates the procedure which would be used to create compiled programs in a diskimage file, directly from a diskimage file containing already compiled programs. With the KEEPREMS option set to OFF, REMs and spacing information are suppressed from the object programs. This would be useful for generating compressed program code which would then occupy less space on the end-user's system. Unless warnings appear indicating that a statement requires a later revision of the RunTime, the code is considered compatible with all prior revisions of the RunTime. The diskimage file could then be copied onto diskette(s) using a compiler utility, and ported over to another machine operating under the same revision of the RunTime or an earlier release.

Explanation of Options:

SOURCE Programs:	the "*" indicates that all programs in the specified SRCLOC are to be compiled.
Location:	indicates that the input programs are located in a diskimage file named PLATTER1.BS2, in the \PROGS directory.
OBJECT Location:	indicates a diskimage file, PLATTER2.BS2, in the \PROGS directory, which will receive the compiled programs.
Format:	the default UNSCRAMBLED indicates that program encryption will not take place.
Extra Sectors:	the default 0 indicates that extra free sectors will not be allocated to the programs.
LISTING Location:	/dev/nul indicates that no list output will be generated.
Format:	the .LST default is irrelevant since no list output will be generated.
ERRORS Location:	ERRORS indicates the name of the file in the current directory which will receive all warnings and errors during the compilation.
OTHER Warnings:	ON indicates that all warnings encountered during the compile will be displayed on the screen.
Numbers:	the default OFF indicates that additional object code to keep track of multiple program statements is not generated.
Display:	ON indicates that the compiler display screen will be shown for your review.
Compile REM$:	ON indicates that any statements beginning with REM $ PC will be retained. Note also that the REM $ PC designation will be removed, due to KEEPREMS OFF.

Keep Lower Case: the default OFF is irrelevant since it only pertains to filename translations between programs stored as stand-alone MS-DOS files and programs stored in NPL diskimage files.

Keep REMs: the default OFF indicates that any program statement beginning with a REM will be removed during the compile, the original format of certain literals (either HEX or quote string) may not be retained, and "special" program spacing will be removed.

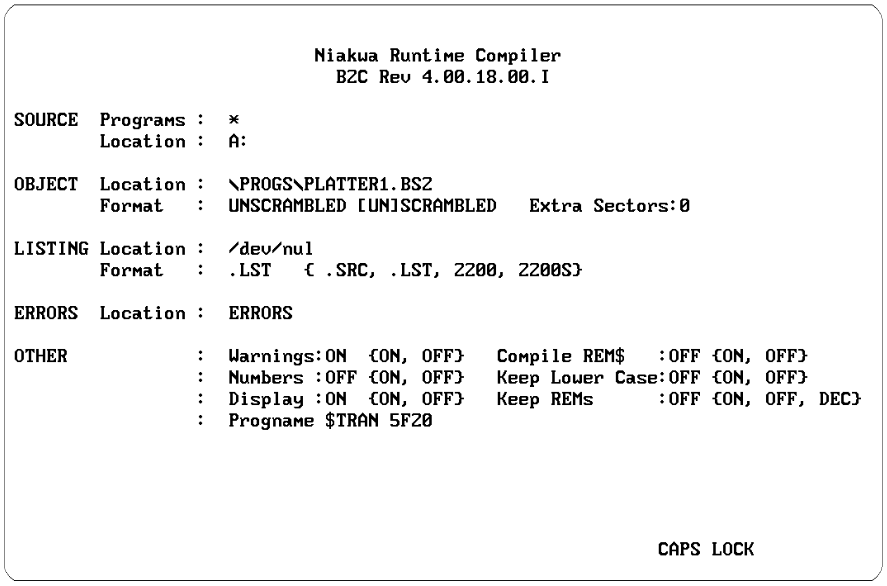
Progname $TRAN: the default 5F20 is irrelevant since it only pertains to filename translations between programs stored as stand-alone MS-DOS files and programs stored in NPL diskimage files.

From a 2200 To a Compiled Diskimage

Command line:

```
B2C /DISPLAY on /SRCLOC A: /OBJLOC \progs\platter1.bs2 /ERRLOC errors *
```

Display:



```
                              Niakwa Runtime Compiler
                                B2C Rev 4.00.18.00.I

SOURCE  Programs :  *
        Location :  A:

OBJECT  Location :  \PROGS\PLATTER1.BS2
        Format   :  UNSCRAMBLED [UN]SCRAMBLED   Extra Sectors:0

LISTING Location :  /dev/nul
        Format   :  .LST   { .SRC, .LST, 2200, 2200S}

ERRORS  Location :  ERRORS

OTHER            :  Warnings:ON  {ON, OFF}   Compile REM$    :OFF {ON, OFF}
                 :  Numbers :OFF {ON, OFF}   Keep Lower Case:OFF {ON, OFF}
                 :  Display :ON  {ON, OFF}   Keep REMs      :OFF {ON, OFF, DEC}
                 :  Progname $TRAN 5F20



                                                      CAPS LOCK
```

Discussion:

The above example indicates the compilation procedure which would be used for the initial porting of programs from a Wang 2200 to a diskimage file on the PC.

Explanation of Options:

SOURCE Programs:	the '*' indicates that all programs in the specified SRCLOC are to be compiled.
Location:	A: indicates that the input programs are located on a 320K format diskette in drive A.
OBJECT Location:	indicates a diskimage file, PLATTER1.BS2, in the \PROGS directory, which will receive the compiled programs.
Format:	the default UNSCRAMBLED indicates that program encryption will not take place.
Extra Sectors:	the default 0 indicates that extra free sectors will not be allocated to the programs.
LISTING Location:	/dev/nul indicates that no list output will be generated.
Format:	the .LST default is irrelevant since no list output will be generated.
ERRORS Location:	ERRORS indicates the name of the file in the current directory which will receive all warnings and errors during the compilation.
OTHER Warnings:	ON indicates that all warnings encountered during the compile will be displayed on the screen.
Numbers:	the default OFF indicates that additional object code to keep track of multiple program statements is not generated.
Display:	ON indicates that the compiler display screen will be shown for your review.

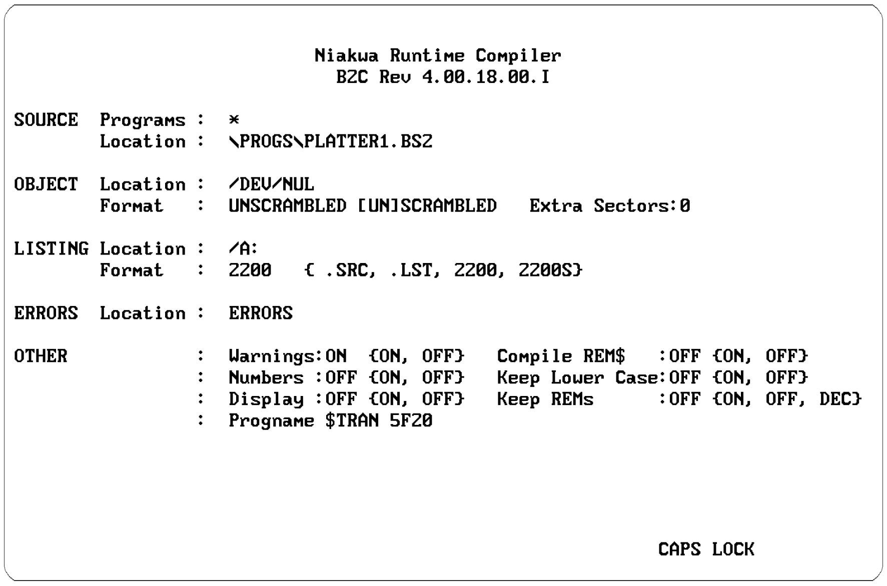
Compile REM$:	the default OFF indicates that any program statement beginning with REM $ PC will be ignored by the compile.
Keep Lower Case:	the default OFF is irrelevant since it only pertains to filename translations between programs stored as stand-alone MS-DOS files and programs stored in NPL diskimage files.
Keep REMs:	the default OFF indicates that any program statement beginning with a REM will be removed during the compile, the original format of certain literals (either HEX or quote string) may not be retained, and 'special' program spacing will be removed.
Progname $TRAN:	the default 5F20 is irrelevant since it only pertains to filename translations between programs stored as stand-alone MS-DOS files and programs stored in NPL diskimage files.

From a Compiled Diskimage To 2200 Atomized Code

Command line:

```
B2C /DISPLAY on /SRCLOC \progs\platter1.bs2 /OBJLOC /dev/nul /LSTLOC
A: /LSTFORMAT 2200 /ERRLOC errors *
```

Display:



```
                               Niakwa Runtime Compiler
                                 B2C Rev 4.00.18.00.I

SOURCE  Programs :  *
        Location :  \PROGS\PLATTER1.BS2

OBJECT  Location :  /DEV/NUL
        Format   :  UNSCRAMBLED [UN]SCRAMBLED   Extra Sectors:0

LISTING Location :  /A:
        Format   :  2200   { .SRC, .LST, 2200, 2200S}

ERRORS  Location :  ERRORS

OTHER            :  Warnings:ON  {ON, OFF}   Compile REM$   :OFF {ON, OFF}
                 :  Numbers :OFF {ON, OFF}   Keep Lower Case:OFF {ON, OFF}
                 :  Display :OFF {ON, OFF}   Keep REMs      :OFF {ON, OFF, DEC}
                 :  Progname $TRAN 5F20




                                                           CAPS LOCK
```

Discussion:

The above example indicates the procedure which would be used to create programs in 2200 atomized format, directly from a diskimage file containing already compiled programs. These programs could be created directly on a diskette in 320K format, as the example indicates, if they will fit on one diskette. If not, then the LSTLOC should be a diskimage file on the hard disk, which could then be copied onto multiple diskettes using the compiler utilities, and ported directly over to the 2200.

Explanation of Options:

SOURCE Programs:	the '*' indicates that all programs in the specified SRCLOC are to be compiled.
Location:	indicates that the input programs are located in a diskimage file named PLATTER1.BS2, in the \PROGS directory.
OBJECT Location:	/dev/nul indicates that the compiler will suppress generating a second set of compiled programs.

Format:	the default UNSCRAMBLED is irrelevant since an OBJLOC of /dev/nul is specified.
Extra Sectors:	the default 0 is irrelevant since an OBJLOC of /dev/nul is specified.
LISTING Location:	A: indicates that the compiler will generate output programs directly on a diskette in 320K format.
Format:	the 2200 option indicates that the programs generated on the specified LSTLOC will be in 2200 atomized format. Use of the 2200S option would additionally remove syntactically insignificant spaces from all programs, thus creating "compressed" program code.
ERRORS Location:	ERRORS indicates the name of the file in the current directory which will receive all warnings and errors during the compilation.
OTHER Warnings:	ON indicates that all warnings encountered during the compile will be displayed on the screen. Note that warnings are not sent to the LSTLOC if the LSTFORMAT option is either 2200 or 2200S.
Numbers:	the default OFF is irrelevant since an OBJLOC of /dev/nul is specified.
Display:	ON indicates that the compiler display screen will be shown for your review.
Compile REM$:	the default OFF is irrelevant since an OBJLOC of /dev/nul is specified.
Keep Lower Case:	the default OFF is irrelevant since it only pertains to filename translations between programs stored as stand-alone MS-DOS files and programs stored in NPL diskimage files.
Keep REMs:	the default OFF is irrelevant since an OBJLOC of /dev/nul is specified.

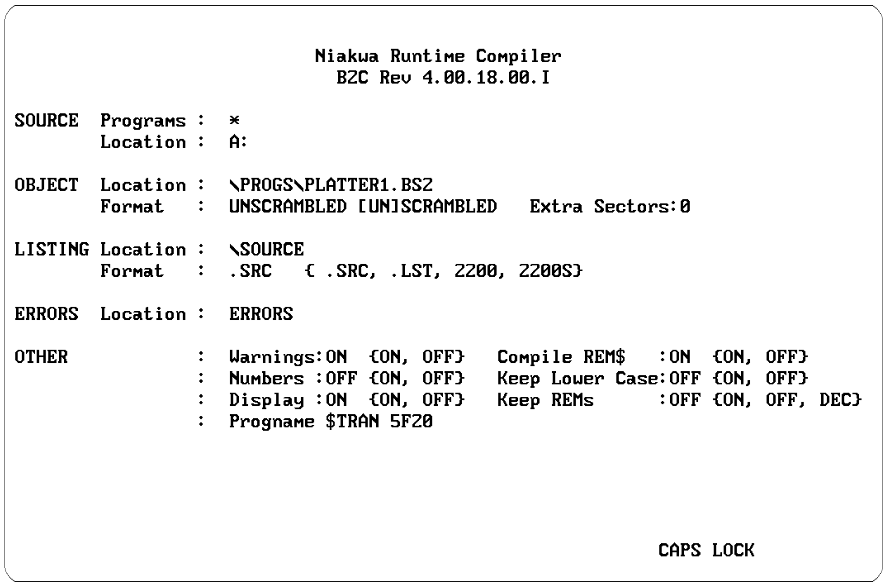
Progname $TRAN: the default 5F20 is irrelevant since it only pertains to file-name translations between programs stored as stand-alone MS-DOS files and programs stored in NPL diskimage files.

From a 2200 To a Compiled Diskimage and ASCII Text Files

Command line:

```
B2C /DISPLAY on /SRCLOC A: /OBJLOC \progs\platter1.bs2 /LSTLOC \source
/LSTFORMAT .src /REM$ on /ERRLOC errors *
```

Display:



```
                          Niakwa Runtime Compiler
                           B2C Rev 4.00.18.00.I

SOURCE  Programs :  *
        Location :  A:

OBJECT  Location :  \PROGS\PLATTER1.BS2
        Format   :  UNSCRAMBLED [UN]SCRAMBLED   Extra Sectors:0

LISTING Location :  \SOURCE
        Format   :  .SRC   { .SRC, .LST, 2200, 2200S}

ERRORS  Location :  ERRORS

OTHER            :  Warnings:ON  {ON, OFF}   Compile REM$   :ON  {ON, OFF}
                 :  Numbers :OFF {ON, OFF}   Keep Lower Case:OFF {ON, OFF}
                 :  Display :ON  {ON, OFF}   Keep REMs      :OFF {ON, OFF, DEC}
                 :  Progname $TRAN 5F20



                                                        CAPS LOCK
```

Discussion:

The above example indicates the compilation procedure which would be used for the initial porting of programs from a Wang 2200 to the PC, creating both a diskimage file containing the compiled programs, and "source" code program listings as ASCII text files in an MS-DOS directory.

Explanation of Options:

SOURCE Programs:	the '*' indicates that all programs in the specified SRCLOC are to be compiled.
Location:	A: indicates that the input programs are located on a 320K format diskette in drive A.
OBJECT Location:	indicates a diskimage file, PLATTER1.BS2, in the \PROGS directory which will receive the compiled programs.
Format:	the default UNSCRAMBLED indicates that program encryption will not take place.
Extra Sectors:	the default 0 indicates that extra free sectors will not be allocated to the programs.
LISTING Location:	indicates that source program listings will be generated in the \SOURCE directory.
Format:	.SRC indicates that the programs generated in the specified LSTLOC will be in ASCII text format.
ERRORS Location:	ERRORS indicates the name of the file in the current directory which will receive all warnings and errors during the compilation.
OTHER Warnings:	ON indicates that all warnings encountered during the compile will be displayed on the screen and in the .SRC files.
Numbers:	the default OFF indicates that additional object code to keep track of multiple program statements is not generated.

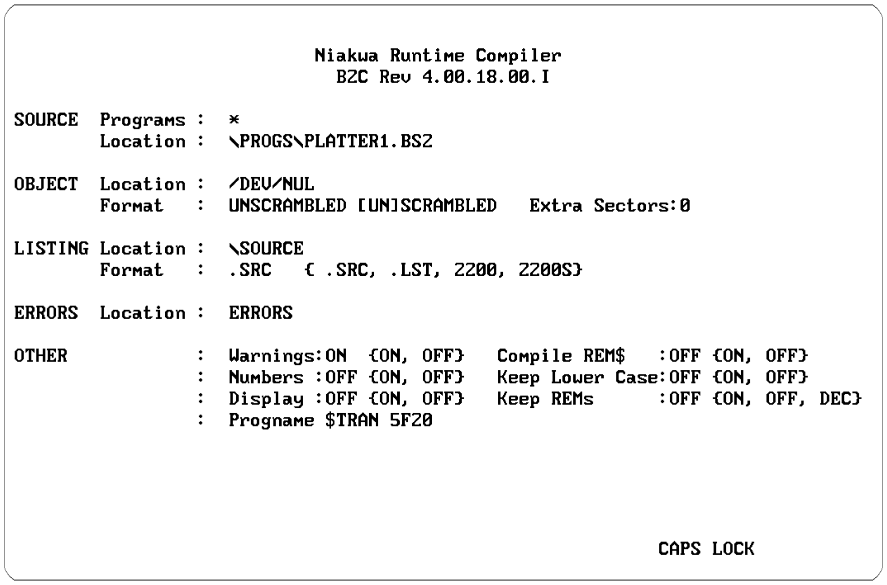
Display:	ON indicates that the compiler display screen will be shown for your review.
Compile REM$:	the ON option indicates that any program statement beginning with REM $ PC will have the program code contained in the statement compiled.
Keep Lower Case:	the default OFF is irrelevant because even though ASCII .SRC files are being created, MS-DOS is case insensitive. Therefore, no distinction can be made in the MS-DOS files between upper and lower case.
Keep REMs:	the default OFF, in conjunction with the REM$ option set to ON, instructs the compiler to compile statements beginning with REM $ PC, and then remove the REM $ PC designation from the program statement. Also, the original format of certain literals (either HEX or quote string) may not be retained, and "special" program spacing will be removed.
Prognam e $TRAN:	the default 5F20 indicates that spaces in input program names will be converted to underline characters when creating ASCII text files in the specified LSTLOC.

From a Compiled Diskimage To ASCII Text Files

Command line:

```
B2C /DISPLAY on /SRCLOC \progs\platter1.bs2 /OBJLOC /dev/nul /LSTLOC
\source /LSTFORMAT .src /ERRLOC errors *
```

Display:



```
                             Niakwa Runtime Compiler
                               B2C Rev 4.00.18.00.I

SOURCE  Programs :  *
        Location :  \PROGS\PLATTER1.BS2

OBJECT  Location :  /DEV/NUL
        Format   :  UNSCRAMBLED [UN]SCRAMBLED   Extra Sectors:0

LISTING Location :  \SOURCE
        Format   :  .SRC   { .SRC, .LST, 2200, 2200S}

ERRORS  Location :  ERRORS

OTHER            :  Warnings:ON  {ON, OFF}   Compile REM$    :OFF {ON, OFF}
                 :  Numbers :OFF {ON, OFF}   Keep Lower Case:OFF {ON, OFF}
                 :  Display :OFF {ON, OFF}   Keep REMs      :OFF {ON, OFF, DEC}
                 :  Progname $TRAN 5F20




                                                              CAPS LOCK
```

Discussion:

The above example indicates the procedure which would be used to create "source" code program listings in ASCII text format in an MS-DOS directory, directly from a diskimage file containing already compiled programs.

Explanation of Options:

SOURCE Programs:	the ’*’ indicates that all programs in the specified SRCLOC are to be compiled.
Location:	indicates that the input programs are located in a diskimage file named PLATTER1.BS2, in the \PROGS directory.
OBJECT Location:	/dev/nul indicates that the compiler will suppress generating a second set of compiled programs.
Format:	the default UNSCRAMBLED is irrelevant since an OBJLOC of /dev/nul is specified.

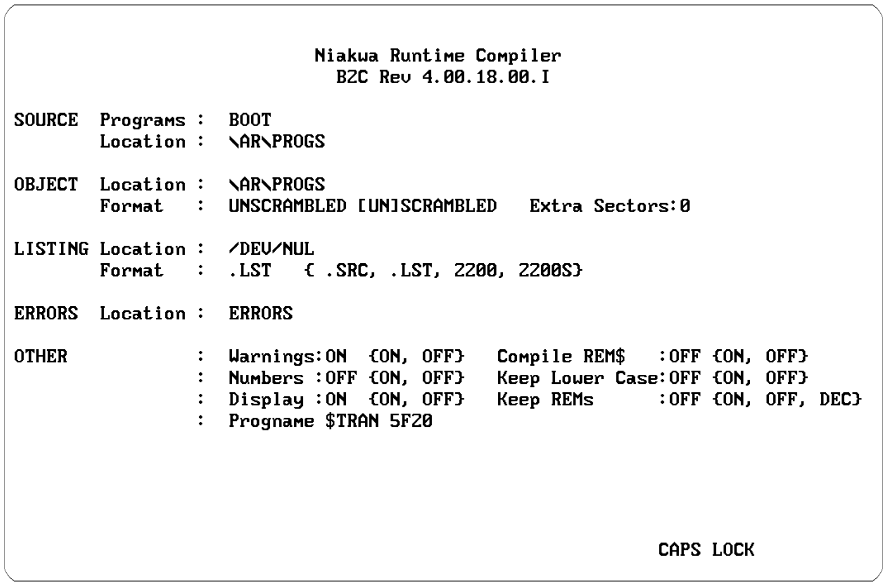
Extra Sectors:	the default 0 is irrelevant since an OBJLOC of /dev/nul is specified.
LISTING Location:	indicates that source program listings will be generated in the \SOURCE directory.
Format:	.SRC indicates that the programs generated in the specified LSTLOC will be in ASCII text format.
ERRORS Location:	ERRORS indicates the name of the file in the current directory which will receive all warnings and errors during the compilation.
OTHER Warnings:	ON indicates that all warnings encountered during the compile will be displayed on the screen and in the .SRC files.
Numbers:	the default OFF is irrelevant since an OBJLOC of /dev/nul is specified.
Display:	ON indicates that the compiler display screen will be shown for your review.
Compile REM$:	the default OFF is irrelevant since an OBJLOC of /dev/nul is specified.
Keep Lower Case:	the default OFF is irrelevant because even though ASCII .SRC files are being created, MS-DOS is case insensitive. Therefore, no distinction can be made in the MS-DOS files between upper and lower case.
Keep REMs:	the default OFF is irrelevant since an OBJLOC of /dev/nul is specified.
Progname $TRAN:	the default 5F20 indicates that spaces in input program names will be converted to underline characters when creating ASCII text files in the specified LSTLOC.

From ASCII Text Files To Compiled Format in a Directory

Command line:

```
B2C /DISPLAY on /SRCLOC \ar\progs /OBJLOC \ar\progs /ERRLOC errors BOOT
```

Display:



```
                          Niakwa Runtime Compiler
                            B2C Rev 4.00.18.00.I

SOURCE  Programs :  BOOT
        Location :  \AR\PROGS

OBJECT  Location :  \AR\PROGS
        Format   :  UNSCRAMBLED [UN]SCRAMBLED   Extra Sectors:0

LISTING Location :  /DEV/NUL
        Format   :  .LST   { .SRC, .LST, 2200, 2200S}

ERRORS  Location :  ERRORS

OTHER            :  Warnings:ON  {ON, OFF}   Compile REM$    :OFF {ON, OFF}
                 :  Numbers :OFF {ON, OFF}   Keep Lower Case:OFF {ON, OFF}
                 :  Display :ON  {ON, OFF}   Keep REMs      :OFF {ON, OFF, DEC}
                 :  Progname $TRAN 5F20




                                                        CAPS LOCK
```

Discussion:

The above example indicates the procedure which would be used to take a single "boot" program stored in ASCII text format, and compile it into a stand-alone .OBJ object file stored in an MS-DOS directory. For example, a simple boot program may appear as:

```
10 $DEVICE(/D11)="\AR\PROGS\PLATTER1.BS2"
20 $DEVICE(/D12)="\AR\DATA\PLATTER1.BS2"
30 SELECT DISK D11
40 LOAD RUN"ARSTART"
```

Explanation of Options:

SOURCE Programs:	indicates that a single program named BOOT is to be compiled.

Location:	indicates that the input program is an ASCII text file with an extension of .SRC, and is located in the \AR\PROGS directory.
OBJECT Location:	indicates that the compiler will create an object file with an extension of .OBJ in the \AR\PROGS directory.
Format:	the default UNSCRAMBLED indicates that program encryption will not take place.
Extra Sectors:	the default 0 is irrelevant since a stand-alone .OBJ object file is being created.
LISTING Location:	/dev/nul indicates that no list output will be generated.
Format:	the .LST default is irrelevant since no list output will be generated.
ERRORS Location:	ERRORS indicates the name of the file in the current directory which will receive all warnings and errors during the compilation.
OTHER Warnings:	ON indicates that all warnings encountered during the compile will be displayed on the screen.
Numbers:	the default OFF indicates that additional object code to keep track of multiple program statements is not generated.
Display:	ON indicates that the compiler display screen will be shown for your review.
Compile REM$:	the default OFF indicates that any program statement beginning with REM $ PC will be ignored by the compile.
Keep Lower Case:	the default OFF is irrelevant since it only pertains to filename translations between programs stored as stand-alone MS-DOS files and programs stored in NPL diskimage files.

Keep REMs:	the default OFF indicates that any program statement beginning with a REM will be removed during the compile, the original format of certain literals (either HEX or quote string) may not be retained, and "special" program spacing will be removed.
Progname $TRAN:	the default 5F20 is irrelevant since it only pertains to filename translations between programs stored as stand-alone MS-DOS files and programs stored in NPL diskimage files.

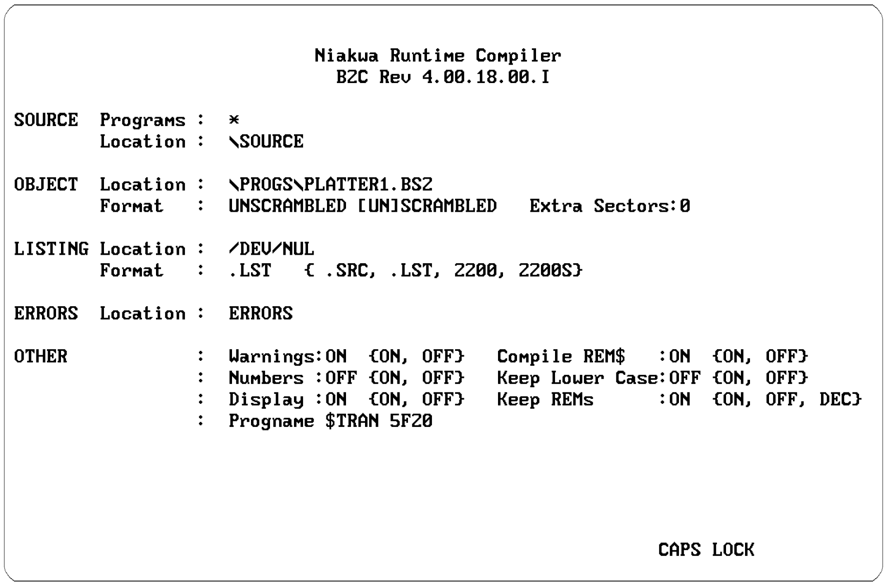
From ASCII Text Files To A Compiled Diskimage

Command line:

```
B2C /DISPLAY on /SRCLOC \source /OBJLOC \progs\platter1.bs2 /ERRLOC er-
rors /REM$ on /KEEPREMS on *
```

Display:

Discussion:



```
                              Niakwa Runtime Compiler
                                B2C Rev 4.00.18.00.I

SOURCE  Programs :  *
        Location :  \SOURCE

OBJECT  Location :  \PROGS\PLATTER1.BS2
        Format   :  UNSCRAMBLED [UN]SCRAMBLED   Extra Sectors:0

LISTING Location :  /DEV/NUL
        Format   :  .LST   { .SRC, .LST, 2200, 2200S}

ERRORS  Location :  ERRORS

OTHER            :  Warnings:ON  {ON, OFF}   Compile REM$   :ON  {ON, OFF}
                 :  Numbers :OFF {ON, OFF}   Keep Lower Case:OFF {ON, OFF}
                 :  Display :ON  {ON, OFF}   Keep REMs      :ON  {ON, OFF, DEC}
                 :  Progname $TRAN 5F20



                                                           CAPS LOCK
```

The above example indicates the procedure which would be used to take input "source" programs stored in ASCII text format, and compile them into a diskimage file.

Explanation of Options:

SOURCE Programs:	the "*" indicates that all programs in the specified SRCLOC are to be compiled.
Location:	indicates that the input programs are ASCII text files with an extension of .SRC, and are located in the \SOURCE directory.
OBJECT Location:	indicates a diskimage file, PLATTER1.BS2, in the \PROGS directory, which will receive the compiled programs.
Format:	the default UNSCRAMBLED indicates that program encryption will not take place.
Extra Sectors:	the default 0 indicates that extra free sectors will not be allocated to the programs.
LISTING Location:	/dev/nul indicates that no list output will be generated.
Format:	the .LST default is irrelevant since no list output will be generated.
ERRORS Location:	ERRORS indicates the name of the file in the current directory which will receive all warnings and errors during the compilation.
OTHER Warnings:	ON indicates that all warnings encountered during the compile will be displayed on the screen.
Numbers:	the default OFF indicates that additional object code to keep track of multiple program statements is not generated.
Display:	ON indicates that the compiler display screen will be shown for your review.

Compile REM$:	the ON option indicates that any program statement beginning with REM $ PC will have the program code contained in the statement compiled.
Keep Lower Case:	the default OFF is irrelevant in this case because of the use of the program name wildcard. Note that if individual programs were specified and it was desirable to specify the case of the program name for the files generated in the diskimage, the LCASE option should be set to ON.
Keep REMs:	the ON option indicates that REM statements and spacing information is retained in the object program. Also, the original format of certain literals (either hex or quote string) will be retained.
Progname $TRAN:	the default value of 5F20 will cause underline characters in stand-alone MS-DOS .SRC files to be converted to a space within the object diskimage.

CHAPTER 10

PORTING PROGRAMS AND DATA

10.1 Overview

This chapter provides an overview of porting NPL application and data files to or from a PC under MS-DOS.

Section 10.2 discusses the NPL specific options available for porting.

Section 10.3 discusses options for porting to/from specific environments outside the Run-Time.

NOTE: The discussions in this chapter are intended to provide general information on file transfer and porting options. For a complete description of the options mentioned in this chapter, refer to Chapter 15 of the NPL Programmer's Guide.

10.2 Porting Options

Two options can be used for transfer of NPL applications and data on the PC. These options are serial communication and diskette transfer. These options are discussed in the following sections.

NOTE: The discussion below presents several third-party products. None of the products are officially endorsed by Niakwa. Before using any third-party product, be aware that possible compatibility problems may exist.

10.2.1 Serial Communications

Direct file transfer from or to the PC is possible using serial communication. For serial transfer of NPL programs and data to/from a MS-DOS system, any program that supports binary file transfer can be used. Many of these products also support terminal emulation. Be aware that the features used by NPL (downloadable fonts, local printer support, etc.), may not function properly on terminal emulation products. This can cause problems for NPL users.

NOTE: Some communication programs may use keys that conflict with NPL key mapping. These conflicts can be avoided using the $KEYBOARD feature.

Any file transfer product used must be capable of transferring eight-bit binary data.

10.2.2 "Raw" Diskette Transfer

The ability of NPL to access "raw" diskettes provides a means of transferring NPL diskimages from one machine to another, where the native operating system's file formats are incompatible. A "raw" device is a physical disk which does not contain a native file system.

The MS-DOS version of NPL accepts 5-1/4", 320K, 360K, 1.2MB and 3-1/2", 720K, 1.44MB, 2.88MB "raw" formatted diskettes. Most NPL-supported systems work with one or more of the above formats. Refer to Appendix D for a list of all supported media on the PC. Provided that diskette compatibility exists between two supported platforms, the following steps are required to port data between the PC and the other supported platform.

1. Select a compatible "raw" format for the source and destinations.
2. Use either the NPL Backup and Recovery utilities or custom transfer utilities to transfer programs and data between systems.

NOTE: Although the "raw" formatted diskettes being used are identical from one system to the next, the naming conventions of these diskettes may change from one system's native operating system to the next.

For example:

Under MS-DOS	**$DEVICE(/D10)="A: 360= Y"**	**Instructs NPL to treat A: as a "raw" 360k device.**
Under SuperDOS	**$DEVICE(/D10)="1: 360= Y"**	**Instructs NPL to treat 1: as a "raw" 360k device**

Refer to Chapter 5 of the appropriate NPL operating system-specific Supplement for supported diskette naming conventions.

Refer to Chapter 15 of the NPL Programmer's Guide for more information on porting.

10.3 Specific Environments

10.3.1 From a Wang 2200

There are many serial communication products that can used for porting from a Wang 2200. The most common product is PC2200 by Computer Concepts. This program emulates Wang 2x36 terminals and supports true box graphics in the character mode. The file transfer utility copies files from the Wang 2200 and automatically places them in a specified NPL diskimage on the PC. PC2200 is capable of creating NPL diskimages on the fly.

For more information on PC2200 contact:

Computer Concepts
8375 Melrose Dr.
Lenexa, KS 66214
(913) 541-0900

10.3.2 From a UNIX-Based System

One product Niakwa has used for the transfer of files to PC's from UNIX systems is Reflection 2. This product can transfer diskimage files and boot program files in the format required by NPL so no conversion is necessary after transfer. To properly transfer diskimage and boot program files with Reflection 2, be sure to specify the following parameters on the file transfer screen:

Method	B (for binary)
Host file name	Filename/F (generate fixed length records)
Host record size	256

NOTE: The Reflection 2 product also provides a terminal emulation feature which allows a PC to be used as a VT220 or VT100 terminal. This has worked well with one exception. NPL uses downloadable fonts to generate the full NPL character set on the VT220. This does not work with a terminal emulation product such as Reflection 2. Therefore, the full NPL character set is not available in this environment. For the most part, this will affect only those applications which use graphics type characters in the range HEX(80) and above.

Reflection 2 requires the installation of the UNIX development system for proper operation.

For more information on Reflection 2, contact:

Walker, Richer & Quinn, Inc.
2815 Eastlake Ave. E.
Seattle, WA 98102
(800) 872-2829

Niakwa programs and data can also be ported to MS-DOS based systems from UNIX-based systems by using the UNIX DOS copy utilities. Most UNIX flavors can read and write MS-DOS format 5-1/4" and 3-1/2" diskettes using the UNIX dos copy commands (specific to each flavor of UNIX). The use of these DOS copy commands is fully documented in the UNIX Reference Manual.

10.3.3 From a SuperDOS-Based System

The simplest way to transfer data between MS-DOS and SuperDOS is to use the PCFILE utility which is part of the SuperDOS operating system package. This program runs on SuperDOS and has a menu interface. However, this program can only run in SuperDOS real mode. While SuperDOS 6.0 runs only in protected mode, it has a DOS file transfer utility called PCXFER. For further details, refer to the SuperDOS Utilities Guide.

Alternatively, PC2200 by Computer Concepts transfers NPL data and programs between DOS and SuperDOS. It can receive as well as transmit between SuperDOS and an MS-DOS based system. This program emulates a Wang 2236/DW terminal. There are several considerations in using such a terminal with a SuperDOS system. Refer to the NPL SuperDOS Supplement regarding Wang 2X36 terminal support under SuperDOS.

10.3.4 From Another MS-DOS-Based System

There are several methods for transferring data between MS-DOS systems using floppy diskettes. Among those are the DOS COPY and BACKUP/RESTORE commands. In addition, there are many third-party MS-DOS based backup and transfer products offered as alternatives (i.e., Fastback, LapLink, etc.). These third-party products are readily available and may offer such features as data compression and batch processing.

Serial communication between MS-DOS systems is also possible using the many third-party MS-DOS products on the market.

CHAPTER 11

MIXED LANGUAGE PROGRAMMING

11.1 Overview

The NPL External Subroutine Development Kit (BESDK), formerly Basic-2C, provides an interface to external subroutines written in other programming languages. However, there are both benefits and penalties which may occur as a result of using mixed languages programming under NPL. The benefits include a potential increase in execution speed for selected processor-intensive functions, and the capability to access resources and features of a specific environment. The penalties include increased memory requirements, limited portability to other NPL environments and a potentially less friendly environment for testing and error diagnosis.

This chapter concerns itself with the operating system and language-specific features of the NPL External Subroutine Development Kit (BESDK) for MS-DOS. For a complete discussion on the general operations of mixed language programming, refer to Chapter 16 of the NPL Programmer's Guide.

Section 11.2 discusses the contents of the NPL External Subroutine Development Kit.

Section 11.3 discusses the installation of the NPL External Subroutine Development Kit.

Section 11.4 discusses MS-DOS support.

Section 11.5 discusses support of Microsoft C under MS-DOS.

Section 11.6 discusses support of Microsoft Assembler under MS-DOS.

Section 11.7 discusses support of Microsoft Pascal under MS-DOS.

Section 11.8 discusses flow control for External Libraries.

Section 11.9 discusses error messages when loading Quick Libraries.

11.2 Contents of the BESDK

The BESDK package is contained on a single diskette labeled NPL Development Package, BESDK Files Diskette. The BESDK includes a number of directories, each of which illustrates an example of linking an external subroutine in a particular environment using a particular language. The function performed by the subroutine is the same in each case, and is analogous to the Microsoft C example followed in the text in Chapter 16 of the NPL Programmer's Guide.

The examples are provided mainly to allow a simple test of whatever versions of compilers, assemblers, linkers, etc., being used with source files which have been pre-tested, and to help clarify any points which may be unclear in this documentation.

HINT: Try to produce the stand-alone example and quick library before creating a new project, to ensure that the various utilities work together as they should.

The contents of the BESDK Files Diskette are listed below:

In the root directory:

\

README.DOC	This file (if present) contains additions and corrections to the BESDK documentation and installation procedure. Read this document before installing the BESDK.
INSTALLI.BAT	This file is a batch file used to transfer files from a floppy to the PC's hard drive.
\INCLUDE	This directory contains files which are common to all implementations or to all implementations of a specific language.
MYBOOT.SRC	An NPL source file which performs a simple test of the example external subroutine.
MYBOOT.OBJ	Compiled version of the MYBOOT.SRC boot program to test the example external subroutines and FUNCTIONs. MYBOOT contains only configuration commands and loads MYSTART from the MYMODULE.BS2 diskimage.
MYSTART.SRC	Source version of the NPL program used to test the example subroutines and FUNCTIONs.
MYMODULE.SRC	Source version of the NPL library module used to specify the interface to the example subroutines and FUNCTIONs and containing a sample CALLBACK function.
MYMODULE.BS2	Compiled version of the MYMODULE.SRC and the MYSTART.SRC programs in a diskimage.
RTPALL.H	Standard include file for C programs, containing structure and type definitions needed to write C code for the BESDK. The Release IV version of this file is expanded and contains a number of macros and types that did not exist in Release III.

MAKEFILE	Gives instructions on how to make the MYBOOT.OBJ and MYMODULE.BS2 files.
RTPALL.PPI	For programmers that have BESDK libraries in Metaware Pascal. Release IV features are not supported under this language.
RTPALL.PI	For programmers that have BESDK libraries in Miscrosoft Pascal. Release IV features are not supported under this language.
RTPALL.INC	For programmers that have BESDK libraries in Microsoft Assembler. Release IV features are not supported under this language.
\INCLUDE\DOS	This directory is specific to the MS-DOS implementation of BESDK.
RPTPARM.OBJ	Compiled version of RTPPARM.C using Microsoft C.
RTPPARM.C	C subroutines to provide the rtpfn_getparminfo() function used to check function declarations.
QLBHEAD.OBJ	Compiled version of QLBHEAD.C using Microsoft C.
RTPDEFFN.H	Different versions of the C compiler can require variant declarations for functions to ensure that they use the exact calling conventions required by NPL. These variances are isolated in this file.
MAKEFILE	Gives instructions on how to make the QLBHEAD.OBJ, RTPPARM.OBJ, and the CNULLCHK.OBJ files, which are the files required by MS-DOS BESDK to make quick libraries work with NPL.
QLBHEAD.ASM	The source required to allow NPL to properly load and interface a .QLB format.
CNULLCHK.C	Handles possible segmentation problems when generating a .QLB.

CNULLCHK	Compiled version of CNULLCHK.C using Microsoft C.
\DOSCEXAM	Contains example files for DOS implementations using Microsoft C.
\DOSMEXAM	Contains example files for DOS implementations using MASM.
\DOSPEXAM	Contains example files for DOS implementations using Pascal.

The above three directories include the following files:

MYMAIN.x	Source file for example mainline.
MYRTP.x	Source file for example RTP test subroutine.
MYRTPEXT.x	Source file for example RTPEXT subroutine.
MYSUB.x	Source file for example DEFFN' subroutine.

where x=

C	for C programs
ASM	for MASM programs
PAS	for Microsoft Pascal programs

MAKEMAIN.BAT — Batch file to compile and link example mainline. To execute, enter:

```
makemain
```

MAKEQLB.BAT — Batch file to compile and link example quick library. To execute, enter:

```
makeqlb
```

MAKEFILE — Script for Microsoft "nmake" utility to produce both mainline and quick library. Assumes Microsoft "nmake" 6.00 or later. To execute, enter:

```
NMAKE/F MAKEFILE
```

The /DOSCEXAM directory also contains the following files, specifically for Release IV callback features:

MYCALLBK.C	Source code to illustrate the use of a C function (mykeyin) that performs a callback to the NPL function ’CallBack-Keyin.
MYCALLBK.H	Include file containing the parameter block specifications required by mycallbk.c.
MYPROC.C	Source code to illustrate the implementation of the example external PROCEDURE in C.
MYPROC.H	Include file containing the parameter block specification required by myproc.c.

NOTE: To change any include files or the makefile itself, delete all previously made .OBJ files in the directory before running make again.

Both the .BAT files and the MAKE script assume that:

- Compiler executables (such as cl, masm, pl, link, b2c, etc.) which may be required can be accessed (PATH is set to allow these to be found).
- The current directory is the same as that containing the source files.
- The example INCLUDE directory can be accessed as "..\INCLUDE".
- All required system libraries are in a directory specified by the LIB environment variable.
- Environment variables which may supply default parameters (i.e., CL, LINK, MASM) are set to null values.

11.3 Installation of the BESDK

The installation of the BESDK Files Diskette is performed by the use of a simple batch file INSTALLI.BAT. To install, enter:

```
INSTALLI "source" "target"
```

where each of "source" and "target" is a drive and optional directory specification. If no directory specification is given, the root directory is assumed.

For example:

```
INSTALLI A: C:\EXTERN
```

installs the files to the directory \EXTERN on drive C. The \EXTERN directory is created if it does not already exist. All required subdirectories are created.

11.4 MS-DOS Support

The following section discusses extended call support under MS-DOS.

11.4.1 Environments

External libraries are supported for MS-DOS in the native environment. It is possible to compile, link and run programs and quick libraries under MS-DOS.

The examples assume Microsoft LINK version 5.01.21 or later. Earlier versions may also work but have not been tested.

11.4.2 The DGROUP Area

In the standard Microsoft languages convention, the DS and SS segment registers point to an area containing a maximum of 64K which is referred to as the DGROUP area; in some in-line code routines, DS may temporarily point to other segments. Near pointers (16-bit) always refer to data in this area. The DGROUP area is divided into three main sections.

Initialized (near) Data

This is normally where small data items are placed. In large model C, each static and external variable which does not exceed the size of the data threshold (the default data threshold size is 256 bytes, but this may be changed with the /Gt option) is placed here. The initial values of all variables in this area (at load time) are defined by the program.

Stack

This is where subroutine parameters, subroutine stack frames (auto class variables) and subroutine return addresses are placed. The stack grows downward from the top of its section. Because subroutines may be recursive, and the allocation for a subroutine stack frame is only required while the subroutine is active, determining the exact stack requirement for a program is problematical. Where maximum stack requirements cannot be precisely determined, the usual approach is to make a generous estimate and then check that the stack pointer has sufficient space at the entry to satisfy each subroutine's requirements. Running out of stack space is usually fatal to an application, because the stack area cannot be "grown" at run time. If this problem is not detected, the data area is overwritten, causing unpredictable and often fatal errors. C programs, by default, use a 2K stack space.

(Near) Heap

The heap area is located above the stack. Memory in this area is usually used for dynamic allocation of memory to routines which either have a temporary requirement for data, or have a requirement for size which cannot be determined until run time.

11.4.3 How DGROUP is Initialized by the Program Loader

Only the initialized data area and stack are included as part of an executable file and are allocated by the program loader. Normally, the stack does not contain initial values. The heap never contains initial values, and may not have any bytes available, depending on memory available when DOS loads the image.

The C start-up routines normally determine the available memory for the near heap, and initialize control information to allow the heap area to be allocated dynamically to small model programs (and other models which specifically require "near" memory). Applications which have a minimum requirement for near heap space may have this information placed in the .EXE file. If this is done, the DOS program loader ensures that there is sufficient additional memory for the near heap before allowing the program to run.

Similarly, it is possible to place information into an .EXE file to indicate that the program requires a larger stack area. If this is done, the DOS program loader ensures that there is sufficient memory for the expanded stack before allowing the program to run.

11.4.4 Quick Library--Stack and Heap Allocation

When loaded as a quick library, the non-DGROUP part of the image is loaded into the high part of system memory. The DGROUP area is loaded in low memory immediately following NPL with no extra space reserved for heap allocations. To the application it looks as if DOS has loaded the program into a free memory area in which it "just fit". Additional stack or heap space may be reserved by modifying the "STACK" of the quick library using the "EXEHDR" utility:

For example:

```
exehdr mylib.qlb /STACK:2048 /MIN:8192
```

Adjusts the file so that 2048 bytes (0800H paragraphs, or 2K) are reserved for stack and 8192 bytes (2000H paragraphs, or 8K) are reserved for the (near) heap in the task space above the image. Increasing the allocation for the external library reduces the space available to the NPL partition. Examples assume the Microsoft EXEHDR version 2.01 or later. Earlier versions may also work but are not tested.

The entry point of the module must be labeled "__astart" (two underlines) and must be a PUBLIC symbol. Linking with the standard C startup and support library meets this condition.

11.5 Microsoft C under MS-DOS

As indicated in the example, writing external subroutines in Microsoft C for MS-DOS is relatively straightforward. Examples assume Microsoft C version 7.00 or later. Earlier versions may also work but are not tested.

11.5.1 General

Use the large model (/AL) option on all "cl" compile commands or ensure that this is part of the CL environment variable.

Make sure the include files provided with the BESDK are available either in a directory specified by the INCLUDE environment variable or by a "/Idirectory" option to the "cl" command.

11.5.2 Mainline

The starting label of user code is called main ("_main" to linker).

NOTE: Substantial startup code from the C library is executed before reaching the "main" routine. The standard entry point of an image linked using Microsoft C is __astart, a library routine.

The RTP subroutine should be referenced as an external with the C naming conventions (linker label is "_RTP").

When linking the quick library, suppress null-pointer checks generated by the C/Pascal startup. This is done by defining a do-nothing function called "_nullcheck". If desired, place this function in a separate module, so that the stand-alone test program can use the null pointer checking facility.

11.5.3 Calling Conventions for BESDK Subroutines

Test RTP Subroutines

Declare all GOSUB' routines using "pascal" calling conventions. The "rtpdeffn.h" include file for MS-DOS defines "rtpdeffn_ext" as equivalent to this designation.

RTPEXT Subroutine

The RTPEXT subroutine should be defined as a procedure with the C naming conventions (linker label is "_RTPEXT"). When called, the address of the rtpdef structure (defined in include file rtpdef.h) is the only parameter. The first field of this structure is a rtpreq structure (defined in include file rtpreq.h).

GOSUB' Subroutines

Use the "pascal" designation on declarations of all subroutines which is called using the GOSUB' interface. The "rtpdeffn.h" include file for MS-DOS defines "rtpdeffn_ext" as equivalent to this designation.

Formats of strings in NPL do not have a zero-terminator and are not of variable length. If strings are to be used by C library routines, make copies which have trailing spaces removed and a zero terminator added.

11.5.4 Linkage of Test Program

If the "cl" command is provided with the names of all source files (or .obj files produced on previous compiles) it automatically invokes the linker with appropriate options (unless the /c option is specified) to produce the stand-alone test program. It is not necessary to specify which libraries are to be used by the compiler, provided the appropriately named libraries can be found using the LIB environment variable.

The files required for production of the stand-alone should include:

- The mainline
- The RTP test subroutine. If callbacks to NPL are made by the test subroutines, it is usually not practical to link a stand-alone test module which includes the code that makes these callbacks.
- The RTPEXT subroutine (optional, but recommended)
- The GOSUB' subroutines
- Any FUNCTION or PROCEDURE subroutines. If these are used, the rtpparm.obj module (located in include\dos) must also be included.

Implied in the linkage are:

- The C support library (LLIBCE.LIB) (name may vary)
- The C startup module (CRT0.OBJ) (may be in support library)

11.5.5 Linkage of Quick Library

If the "cl" command is provided with the names of all source files (or .obj files produced on previous compiles) plus the options "/link /quicklib" (these must be the LAST options on the "cl" command line), it automatically invokes the linker with appropriate options (unless the /c option is specified) to produce the quick library. It is not necessary to specify which libraries are to be used by the compiler, provided the appropriately named libraries can be found using the LIB environment variable. Specify the name of the result file with a ".QLB" extension, e.g., using "/Femylib.qlb".

The files required for production of the stand-alone should include:

- The mainline
- Quick library header (qlbhead.obj)
- The RTPEXT subroutine
- The GOSUB' subroutines
- Any FUNCTION or PROCEDURE subroutines. If these are used, the rtpparm.obj module (located in the include\dos) must also be included.
- Null pointer check suppression (cnullchk.obj)

Implied in the linkage are:

- The C support library (LLIBCE.LIB) (name may vary)
- The C startup module (CRT0.OBJ) (may be in support library)

11.6 Microsoft Assembler under MS-DOS

External subroutines and the mainline can be written entirely in Macro assembler if required. Macro assembler has the advantage that support code dragged in from libraries is usually minimal and, so, the resulting library is often more compact than if written in a high-level language. However, the code is generally much more difficult to write and less portable when complete. Examples assume Microsoft MASM version 5.10 or later. Earlier versions may also work but are not tested.

External libraries written in Macro Assembler do not support the FUNCTION or PROCEDURE interfaces or callbacks to NPL.

11.6.1 General

Make sure the include files provided with the BESDK are available either in a directory specified by the INCLUDE environment variable or by a "/Idirectory" option to the "masm" command.

If a module refers to an external routine which is located in a library, linkage is simplified by adding an INCLUDELIB statement to the module which refers to the required library.

11.6.2 Mainline

The entry point of the module must be labeled "__astart" (two underlines) and must be a PUBLIC symbol. The module containing this label should be assembled with the "/Mx" or "/Ml" to ensure that the lowercase label is not translated by the assember to uppercase. The entry point is designated by placing the entry point after the END statement of the module.

For example:

```
END __astart
```

The RTP subroutine should be referenced as a far external with the name "_RTP".

Use Microsoft conventions for segment names. This is most easily done using the simplified segmentation directives introduced on version 5.0 of MASM. If explicit segment names are used, be sure that:

- All code segments have combine class "CODE"
- All near-data segments (and standard stack) have combine class "DATA", and are part of the group named DGROUP.

The module must not depend on any specific segment ordering. All quick libraries are linked in /DOSSEG order, i.e., all "CODE" first, DGROUP last, others in between. DGROUP segments of class "BSS" and "STACK" are always moved to the end of DGROUP. External symbols "_edata" and "_end" are defined by the linker to be offset DGROUP:BSS and offset DGROUP:STACK, respectively. When loaded as a quick library, the non-DGROUP part of the image is loaded into high memory (in a different DOS allocation area from DGROUP), so no attempts should be made to reference DGROUP using other segment bases.

At the entry point, SS:SP is set to the stack area of DGROUP.

NOTE: SS is not equal to DGROUP.

The startup routines should normalize the stack (make SS= DGROUP, SP adjusted accordingly) as part of initialization.

Do not make any assumptions about additional memory being available in the allocation area containing DGROUP. When the RunTime loads the quick library, DGROUP is given the minimum size possible, including reserved stack space. No (near) heap space is available unless the EXEMOD utility has been used to change the "MIN_BSS" field (refer above). The paragraph address of the end of the DOS allocation area containing DGROUP is available at offset 2 in the Program Segment Header (ES: and DS: point to the PSH at __astart label). If additional memory is required, use the DOS call to get it, and free it, if it is no longer required.

11.6.3 Calling Conventions for BESDK Subroutines

Test RTP Subroutines

Declare RTP subroutine as public with name "_RTP".

Call GOSUB' subroutines using pascal calling conventions (push arguments in order used in GOSUB' statements, assume arguments popped by subroutine).

RTPEXT Subroutine

The RTPEXT subroutine should be defined as a far procedure with the name "_RTPEXT". When called, the address of the RTPDEF structure (defined in include file RTPDEF.INC) is on the stack as a far pointer. The first field of this structure is a RTPREQ structure (defined in include file RTPREQ.INC).

GOSUB' Subroutines

Each subroutine should be defined as a far procedure. When called, the parameters are on the stack below the return address. The first parameter of the GOSUB' is pushed first, last parameter pushed last.

A string parameter is passed as:

```
PUSH SEG <string>
PUSH              OFFSET <string>
PUSH              SIZE <string>
```

NOTE: Pushing immediate values is not supported, except on 286 processors. To operate the quick library on PC class machines, push constant values using an intermediate register.)

A numeric parameter is passed as:

```
PUSH               SEG <rtpnum structure>
PUSH               OFFSET <rtpnum structure>
```

The rtpnum structure is defined in the include file RTPNUM.INC.

To conform to pascal calling conventions, use the form of the "RET" instruction that automatically pops parameters from the stack (4 bytes per numeric parameter + 6 bytes per string parameter).

11.6.4 Linkage of Test Program

Programs written in MASM must be specifically linked to produce an executable file. All input files to LINK must be the result of previously run assemblies or libraries of files.

The files required for production of the stand-alone should include:

- The mainline
- The RTP test subroutine
- The RTPEXT subroutine (optional, but recommended)
- The GOSUB’ subroutines
- Any libraries referenced by the above modules.

11.6.5 Linkage of Quick Library

Programs written in MASM must be specifically linked to produce an executable file. All input files to LINK must be the result of previously run assemblies or libraries of files. A special option ("/QUICKLIB") to the LINK program results in the production of a quick library instead of an executable file.

The files required for production of the quicklib should include:

- The mainline (e.g., mymain.obj)
- Quick library header (qlbhead.obj)
- The RTPEXT subroutine (e.g., myrtpext.obj)
- The GOSUB' subroutines (e.g., mysub.obj)
- Any libraries referenced by the above modules. (e.g., libs.lib)

For example:

```
LINK /QUICKLIB mymain qlbhead myrtpext mysub,mylib,,libs;
```

produces "mylib.qlb".

11.7 Microsoft Pascal under MS-DOS

External subroutines can be written in MS-Pascal if required. It is usually necessary to use extensions to standard Pascal to interface with NPL since the parameters passed do not, in general, correspond to standard Pascal types or pointers. Example files assume MS-Pascal 4.00 or later. Earlier versions may also work but are not tested.

External libraries written in Macro Assembler do not support the FUNCTION or PROCEDURE interfaces or callbacks to NPL.

11.7.1 General

The large model is the only model supported by MS-Pascal so no special designation of pointers or addresses is required.

Make sure the include files provided with the BESDK are available either in a directory specified by the INCLUDE environment variable or by a "/Idirectory" option to the "pl" command.

11.7.2 Mainline

Declare RTP as an external function returning type WORD result with the [C] attribute. It is not necessary to pass parameters to RTP in this case. The value returned by RTP can be passed back to DOS by assigning the external WORD variable DOSEQQ to the value returned by RTP.

When linking the quick library, suppress null-pointer checks generated by the C/Pascal startup. This is done by defining a do-nothing function called "_nullcheck". If desired, place this function in a separate module, so that the stand-alone test program can use the null pointer checking facility.

11.7.3 Calling Conventions for BESDK Subroutines

Test RTP Subroutines

Declare RTP as a function named "RTP" with [C] attribute returning integer.

Declare GOSUB' subroutines with argument types "x:rtpstr_pointer, len:word" for a string, and "x:rtpnum_pointer" for a numeric.

Call GOSUB' subroutines with arguments in format "ADS x, len" for a string, and "ADS x" for a numeric.

RTPEXT Subroutine

Declare RTPEXT as a function named "RTPEXT" with [C] attribute returning an integer.

Assign the address of the GOSUB' procedure to rtpdef_pointer using the ADS operator.

Assign the address of the LIST' description string using the ADS operator. Strings used for this purpose should not be declared as local variables to RTPEXT, since this places them in a volatile area (the stack).

GOSUB' Subroutines

Subroutines called by the GOSUB' interface should be declared as functions returning type integer results. Since they are usually in a source file separate from the RTPEXT subroutine and require linking, MS-Pascal requires that the function declaration have the "[public]" attribute.

The data type of NPL strings is somewhat problematical for Pascal, since fixed strings of different lengths are usually incompatible types in standard Pascal, and extensions to support variable-length strings require a different format and parameter passing convention from that used by NPL. The approach used by the example Pascal subroutines is to declare a "rtpstr" type which is the largest array of characters, with indices starting at 1. This means, of course, that subscript checking on these strings is not enforced within the Pascal subroutines, and caution must be taken to ensure that string bounds are not exceeded.

11.7.4 Linkage of Test Program

If the "pl" command is provided with the names of all source files (or .objfiles produced on previous compiles) it automatically invokes the linker with appropriate options (unless the /c option is specified) to produce the stand-alone test program. It is not necessary to specify which libraries are to be used by the compiler, provided the appropriately named libraries can be found using the LIB environment variable.

The files required for production of the stand-alone should include:

- The mainline
- The RTP test subroutine
- The RTPEXT subroutine (optional, but recommended)
- The GOSUB' subroutines

Implied in the linkage are:

- The Pascal support library (LIBPASE.LIB) (name may vary)
- The Pascal/C startup module (CRT0.OBJ) (may be in support library)

11.7.5 Linkage of Quick Library

If the "pl" command is provided with the names of all source files (or .obj files produced on previous compiles) plus the options "/link /quicklib" (these must be the LAST options on the "pl" command line), it automatically invokes the linker with appropriate options (unless the /c option is specified) to produce the quick library. It is not necessary to specify which libraries are to be used by the compiler, provided the appropriately named libraries can be found using the LIB environment variable. Specify the name of the result file with a ".QLB" extension, e.g., using "/Femylib.qlb".

NOTE: The "pl" program with Pascal 4.00 has an apparent bug which causes the .qlb extension to be ignored, producing a .exe extension instead.

The files required for production of the quicklib should include:

- The mainline
- Quick library header (qlbhead.obj)
- The RTPEXT subroutine
- The GOSUB’ subroutines
- Null pointer check suppression (cnullchk.obj)

Implied in the linkage are:

- The Pascal support library (LIBPASE.LIB) (name may vary)
- The Pascal/C startup module (CRT0.OBJ) (may be in support library)

11.8 Flow Control for External Libraries

The flow of control (in chronological order) for RunTime using quick libraries is as follows:

1. NPL runs, and does some initial configuration work, including processing options and loading the quick library specified after the /X option. After loading the quick library, locations in the memory image of the quick library are patched to allow it to call the RTP subroutine.

2. NPL jumps into the entry point of the quick library, which is usually the library start-up routine for the language in use.

3. The C start-up routines in the external library execute and eventually get into the external library mainline (e.g., main()), which calls RTP() as a subroutine.

4. The RTP subroutine completes NPL start-up, including loading the bootstrap program.

5. NPL scans the external library for numbered DEFFN'S with named aliases, using the LIST' calls starting at function number 0. An internal table of identifiers and equivalent numbered externals is built.

6. NPL execution proceeds. At some point, a GOSUB' (e.g., GOSUB'100) is executed, and no local GOSUB' subroutine is found. If the GOSUB' is to a named DEFFN', and the identifier is found in the table created in step 5, the equivalent number is used to query RTPEXT.

7. NPL calls RTPEXT to find out whether an external '100 subroutine exists and, if so, where it is and what parameter types it needs.

8. RTPEXT supplies the requested information (e.g., GOSUB'100 exists, has three parameters with types string, string, and numeric, which is located at mysub()) and returns (to NPL).

9. If the RTPEXT indicated that the subroutine does not exist, or if the number and type of parameters don't match, an NPL error is generated on the GOSUB' statement. Otherwise, NPL evaluates parameters and calls the external subroutine (mysub) with an address provided by RTPEXT.

10. The external subroutine (mysub) executes and returns to NPL. NPL execution proceeds until it returns to step 5 (another GOSUB') or the RunTime ends ($END, Killed from Help, etc). In the second case proceed to the next step.

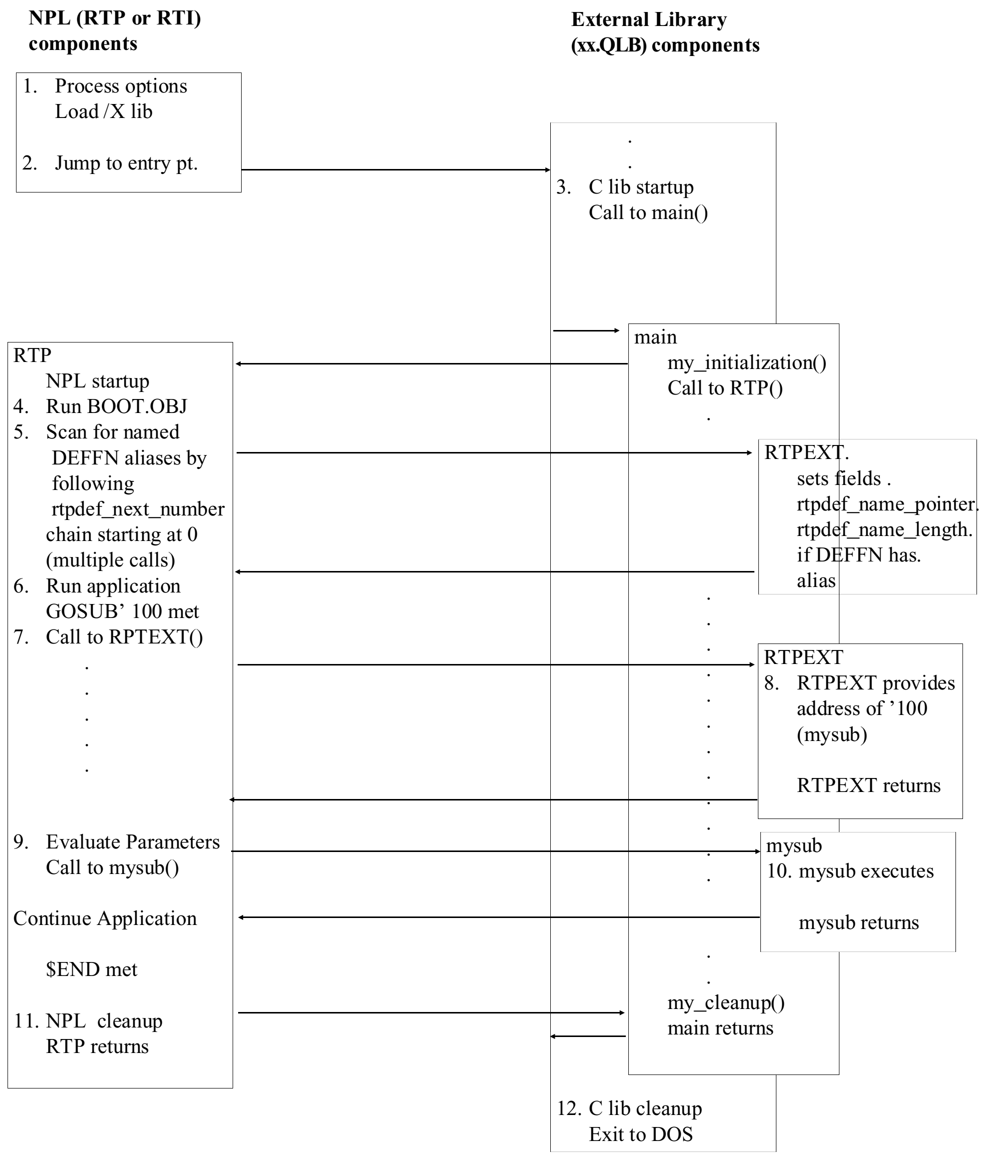
11. NPL does its clean-up, then the RTP subroutine returns to the caller (in the external library mainline).

12. The external library mainline does C library shutdown, and eventually exits back to DOS.

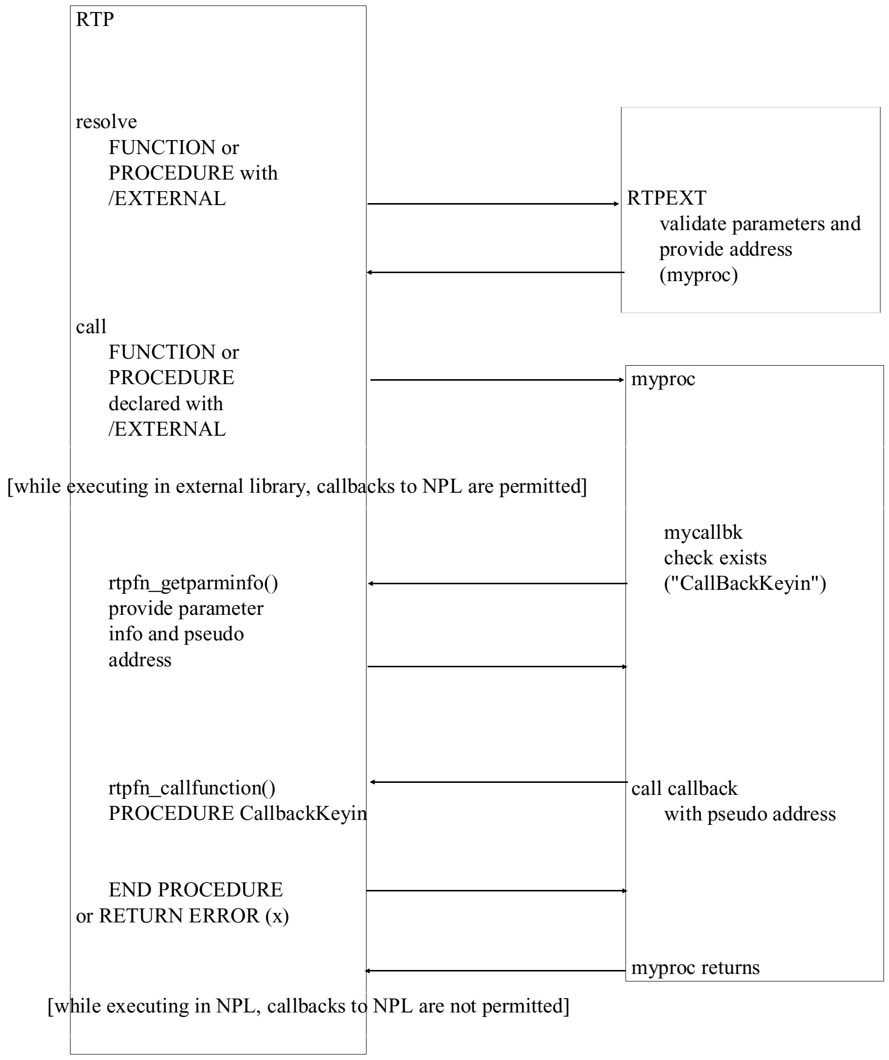
RTPEXT can be called by LIST’ to determine information about DEFFN’ subroutines, without actually calling the subroutines.

NOTE: The subroutines should not depend on the above flow control order to work (e.g., a subroutine should not expect RTPEXT to always be called immediately before it). The above outline is provided merely as a guide to understanding how the external mainline, RTP, RTPEXT and external subroutines interact.

The following diagram shows how execution proceeds using the various software components, with the above steps labeled:



The following diagram shows the execution picture for the FUNCTION/PROCEDURE (interface):



11.9 Errors when Loading Quick Libraries

A fairly large number of errors may occur when NPL is loading a quick library specified by the /X option, if the file is not in a format appropriate to a quick library. These error messages are all diagnosed with the message:

```
Failure loading /X external library (code xx).
```

The xx value in the message may assist in determining exactly what it is about the quick library that NPL does not like. The following list provides some explanation of the errors which are most likely to occur:

Code	Meaning
01	Cannot locate the specified filename at all.
02, 05, 06, 09, 10, 12, 13, 25, 26, 29, 30, 31, 40, 42, 43, 44, 45	Seek or read failed on the file. Usually indicates the .QLB file has been damaged.
03	The file does not contain the correct "magic number" placed by Microsoft LINK into all .EXE and .QLB files.
07	The file does not contain the correct "magic number" placed by Microsoft LINK into all .QLB files.
11	The CODE area does not start at offset 0. This can happen if the mainline contains an ORG directive, or when trying to use some Microsoft support libraries for small/compact models.
15	The size of the DGROUP area exceeds 64K. This should be reported as an error by LINK.
16	The DGROUP area starts at an invalid offset.
20	Insufficient memory in task area for LDT to be allocated (Protected SuperDOS). Insufficient memory for DGROUP area, or value set using /min option of EXEMOD is too big (DOS).
21	Insufficient memory in task area for stack/heap to be allocated (SuperDOS).
22	Insufficient memory in task area for DGROUP to be allocated (SuperDOS).
28	Insufficient memory for CODE area to be allocated.
41	Insufficient memory to load .QLB symbol table.

Code	**Meaning**
46	Too many non-DGROUP segments--use /packcode option to LINK (Protected SuperDOS).
47	Too many DGROUP segments--use /packcode option to LINK (Protected SuperDOS).
48	Cannot find [_]RTPPTR symbol in quick library data symbols.
49	Cannot find [_]RTPEXT symbol in quick library code symbols.

APPENDIX A

COMMON PROBLEMS

A.1 Overview

Many of the problems discussed below relate to the manner in which files and devices are accessed under the MS-DOS operating system.

Section A.2 discusses problems and errors generated by the RunTime.

Section A.3 discusses the general problems and errors generated by the Niakw NPL Upgrade RunTime program.

Section A.4 discusses problems and errors generated by the Compiler.

A.2 Runtime Problems

This section describes specific problems which may be encountered in using the Run-Time program, along with possible solutions.

Problem 1: Program RTP (or RTI) not found.

This problem can result from two possible causes:

The RTP.EXE (or RTI.EXE) program is not in the NPL RunTime files directory. Install the RunTime Package.

The NPL RunTime files directory has not been properly set up as an alternate path or the NIAKWA_RUNTIME environment variable has not be set properly. Modify the AUTO-EXEC.BAT file as described in Chapter 2. It is necessary for the user to reboot the system after this change has been made for it to take effect.

NOTE: **This problem may also apply to any batch files used to invoke the RunTime. The same solutions apply.**

Problem 2: A RunTime error P48-Illegal Device Specification is generated.

This error means that the RunTime was unable to open the specified diskimage file. This could be due to a number of factors:

The RunTime cannot locate the specified diskimage file. Be sure that the $DEVICE equivalence is specified correctly. If the $DEVICE statement does not include a directory specification (i.e., is simply a file name--PLATTER1.BS2, for example), be sure that the current directory is established properly using a cd statement prior to execution of the RunTime.

If the device being accessed is a "raw" format diskette, check that the $DEVICE statement refers to the proper MS-DOS device:

```
$DEVICE(/D10)="A:"  or
$DEVICE(/D10)="A: 360=Y"
$DEVICE(/D10)="A: 1.2=Y"
$DEVICE(/D10)="A: 720=Y"
$DEVICE(/D10)="A: 1.4=Y"
$DEVICE(/D10)="A: 2.8=Y"
```

Problem 3: The RunTime generates an immediate D82-File Not in Catalog error stating that the boot program cannot be found.

This error means that the RunTime program could not find the boot program specified. This could be caused by the following:

The current directory is not defined. Check batch files used to invoke the RunTime.

The boot program exists, but is specified incorrectly. Either modify your batch file or rename the boot program.

Problem 4: "True" box graphics do not print.

Make sure that a graphics card is installed in the PC. Refer to Section 2.2 for configuration requirements or refer to the NPL Statements Guide, $BOXTABLE, for details on box graphics. Character boxes may be used instead.

Problem 5: The message "DEVICE /DEV/PRN not ready, please check before proceeding" appears.

The printer is not ready to receive output. Either it is turned off, disconnected, not selected, out of paper, or otherwise malfunctioning.

Correct the problem with the printer and press HOME to proceed.

If the problem with the printer cannot be corrected, it is necessary to execute the "Kill RunTime" option. This exits the RunTime entirely.

NOTE: The "printer not ready" message is generated for each character being sent to the printer. Pressing HOME when the printer is still not ready causes the character currently being sent to the printer to be lost. Furthermore, repeatedly pressing HOME when the printer is not ready may eventually cause the system to hang.

Problem 6: The message "DRIVE (X) not ready, please check before proceeding", where (X) is a DOS drive designation, appears.

This message suggests that the program cannot access the drive specified. Typically, this occurs when the program is attempting to read or write from the diskette drive and either no diskette is in the drive, a diskette is in the drive but is not inserted properly, or a diskette of a format type other than the one expected by the program is in the drive.

Insert the proper diskette into the diskette drive and press HOME. Should this not work, then contact Niakwa guidance.

Problem 7: The RunTime ERROR screen appears:

The RunTime has encountered an error condition in the application program being run.

Refer to Appendix B of the NPL Programmer's Guide..

Problem 8: The message "Interpreter not enabled." appears:

The interpretive RunTime Package (RTIxxx) has not been enabled on the configuration.

Use the non-interpretive version of the RunTime Package (RTPxxx) or install the ENABLED file in the directory where the NPL RunTime files are located. Refer to Chapter 3 for details.

Problem 9: The message "'FROM' or 'TO' parameters missing or invalid." appears when installing the Gold Key security.

During the installation of the Gold Key security under the NIAKINST program, an improper drive was specified.

Retype the NIAKINST command making sure that the correct diskette drive is specified where the Gold Key is mounted and the correct destination file server.

Problem 10: The message: "Protection Error = 7030, Hex - Sence = 00 00 00 00 Current Available Install Count = 0 RunTime program has been previously installed on another drive. Please 'recall' from the other system first before installing here." appears when trying to install the Gold Key security.

This may result from trying to install the RunTime program when it has been previously installed on another system.

Recall the protection from the other system.

Problem 11: The message "Unauthorized copy" appears when trying to install the RunTime program.

NIAKINST was not run from the Gold Key diskette drive.

NIAKINST must be run from the Gold Key diskette drive.

Problem 12: The message "'FROM' or 'TO' parameters missing or invalid. Recall failed." appears when recalling the Gold Key.

Please check that the original RunTime program diskette was properly mounted and the drives were specified correctly.

During the recall of the Gold Key security under the NIAKRCLL program, an improper drive was specified.

Reenter the NIAKRCLL command making sure that the correct diskette drive is specified (where the Gold Key is mounted) and the correct destination hard drive.

Problem 13: The message "Protection on Disk A: is not installed on Disk X: Recall failed." appears when recalling the Gold Key security.

Please check that the original RunTime program diskette was properly mounted and the drives were specified correctly.

NIAKRCLL has been run when the Gold Key diskette is not installed on the file server.

The drive specifier used is not the drive on which the RunTime program was installed.

The Gold Key that is mounted is not the original diskette that was used for the installation.

Use the original Gold Key diskette that was used for the installation.

Problem 14: The message "Please Mount your Gold Key diskette" appears on the screen when executing the RunTime program.

The NPL_SECURITY environment variable is set to a drive other than where the security was installed.

The RunTime program was started from a drive other than where the RunTime and/or Gold Key security files were installed and the NIAKWA_RUNTIME environment variable is not properly set.

The Gold Key security may not be properly installed.

Enter the letter of the drive where the RunTime program was installed. If the Niakwa software is installed on a different drive or directory, make sure that the NPL_SECURITY and/or NIAKWA_RUNTIME environment variable is set.

Check to see that the RunTime program has been installed. This can be done by running the NIAKINST program. This reinstallation displays the available install count on the current Gold Key Diskette. If the count is still one, continue with the installation and retry the RunTime after completion. If the count is zero, continue using the Gold Key as a backup and contact Niakwa to arrange for a replacement RunTime Package.

Problem 15: The message "Please Mount your Gold Key diskette" remains on the screen after executing the RunTime program.

The encrypted serial number in the RunTime Package on the file server does not match the encrypted serial number in the Gold Key diskette. This would occur when attempting to use a different diskette from the one installed from the RunTime program.

Be sure that the Gold Key that is mounted is the original diskette from which the RunTime was installed.

Problem 16: The following message appears on the screen when executing the RunTime program.

```
:  NIAKSER.DAT missing or damaged (Error Code)
RunTime program Cancelled
Contact your distributor for assistance
```

The (Error Code) generated can be one of following:

Error Code	**Description**
Code 0	Cannot find NIAKSER.DAT.
Code 1	Cannot read NIAKSER.DAT.
Codes 2-5	Invalid data in NIAKSER.DAT. File is probably damaged, reinstall the NIAKSER.DAT file from the Gold Key or from the backup that is created on the "upgrade" diskette. Refer to Chapter 2 for details.

The NIAKSER.DAT file could not be located.

Invalid data is contained in the NIAKSER.DAT file.

Be sure that the NIAKSER.DAT file is in the \BASIC2C directory on the current drive or in the directory specified by the NIAKWA_RUNTIME environment variable. If it is not, copy it from the Gold Key to the proper directory.

If the file is in the proper directory, it may be damaged or may contain invalid data. Delete the current NIAKSER.DAT file and copy the NIAKSER.DAT from the Gold Key.

If the above solutions do not work, make a note of the (Code x) and contact Niakwa.

Problem 17: The message "insert working-copy into drive C and press any key" appears on the screen when attempting to load the Gold Key security.

Device drivers are interfering with the transfer of the Gold Key security.

Rename the AUTOEXEC.BAT and CONFIG.SYS files to temporary filename (i.e., AUTOEXEC.TMP and CONFIG.TMP), reboot the computer, and attempt to reinstall the Gold Key security.

If the security successfully installs, rename the AUTOEXEC and CONFIG files back to AUTOEXEC.BAT and CONFIG.SYS and reboot the computer.

NOTE: **This problem is only encountered when installing or recalling the Gold Key security and not when the RunTime is executed.**

If the above solutions do not work, contact Niakwa.

Problem 18: The message "Cannot locate NIAKSEC?.COM authorization program(s)." appears on the screen when executing the RunTime program.

The necessary NIAKSEC?.COM files could not be found.

Copy these six files from the RunTime diskettes to the directory where the other Niakwa RunTime files are stored. If a drive other than the current drive or a directory other than \BASIC2C is used, be sure that the drive and directory are properly specified by the NIAKWA_RUNTIME environment variable. Refer to Chapter 2 for more information.

Problem 19: The message "Block link chain broken" appears.

This message usually indicates that the p-code being loaded is corrupted.

Restore the program being loaded from a backup set.

If this does not resolve the problem, check for hardware problems such as:

- Faulty memory
- Faulty network communications
- Faulty hard drive.

A.3 Upgrade Problems

This section describes general problems which may be encountered in upgrading an existing NPL RunTime to Release IV with a Niakwa NPL Upgrade Package.

Problem 1: When upgrading, a "This upgrade is for a site with x" message appears.

The upgrade being used does not match the original system being upgraded. Contact Niakwa on obtaining the correct Niakwa NPL Upgrade Package for the system to be upgraded. If there is any question of what the system has, use the end-user UPGRTEST program.

Problem 2: When upgrading, an Error #XX message appears.

Causes for this vary. The text for these various upgrade error messages are intended to be self explanatory. Correct the indicated problem or Niakwa on obtaining the correct Niakwa NPL Upgrade Package for the system to be upgraded. If there is any question of what the system has, use the end-user UPGRTEST program.

A.4 Compiler Problems

This section describes specific problems which may be encountered in using the compiler, along with possible solutions.

Problem 1: Program B2C not found.

This problem can result from two possible causes:

The B2C.EXE program is not in the NPL RunTime files directory. Install the compiler.

The directory has not been properly set up as an alternate path. Modify the AUTOEXEC.BAT file as described in Chapter 2. It is necessary for the user to reboot the system after this change has been made for it to take effect.

NOTE: This problem may also apply to any batch files used to invoke the compiler. The same solutions apply.

Problem 2: The compiler generates a message "No source programs matching wildcard". This means that the compiler could not find any programs in the SRCLOC specified which matched any of the program names or program name wildcards entered.

The SRCLOC has been improperly specified. When using a batch file to compile programs, be sure that the SRCLOC includes the proper directory designation.

Problem 3: The compiler states "Cannot open SRCLOC as a diskimage file" when compiling from a "raw" format diskette.

Be sure that the SRCLOC is properly specified. For compiling from 320K "raw" format diskettes, the specification is:

```
A:  (or B:)
```

NOTE: The NPL Compiler only supports 320K "raw" format diskettes; attempts to access 360K or 1.2MB "raw" format diskettes result in the error mentioned above.

Problem 4: The compiler states "Cannot open SRCLOC (or OBJLOC) as a diskimage file" when compiling from or to diskimage files.

The compiler cannot locate the specified diskimage file. Be sure that the SRCLOC (or OBJLOC) is specified correctly. If the SRCLOC (or OBJLOC) does not include a directory specification (i.e., is simply a file name--PLATTER1.BS2 for example), be sure that the current directory has been properly specified.

APPENDIX B

DOS ERROR CODES

B.1 Overview

The NPL Error Processor displays an "MS-DOS Error Code" whenever possible. This code indicates the operating system error code received by the RunTime program during execution of the statement producing the error. This code contains information which is quite useful in diagnosing the problem. In the MS-DOS version of the RunTime program, this error code is returned to the RunTime by MS-DOS.

Most MS-DOS errors are associated with NPL errors P48 and I90-I99.

B.2 Error Codes

The following is a list of common MS-DOS error codes. Listed are the two-byte hexa-decimal codes, followed by the short error description. This is followed by a more "mean-ingful" description.

0002 - File not found.
Check the $DEVICE specification for a correctly specified filename.

0003 - Path not found.
Check the $DEVICE specification for a correctly specified pathname.

0004 - Too many open files.
The number of open files has exceeded the FILES parameter in CONFIG.SYS. Increase the FILES parameter to accommodate all open files.

0005 - Access denied.
Check the attribute of the file, it may be set to "R" (read-only).

0006 - Invalid handle.
Check the $DEVICE specification for a correctly specified file handle.

If the help file RTIIERR.HLP AND RTIIERR.IDX are installed, the RunTime automatically accesses these files and displays a literal message describing the MS-DOS error that has occurred. In addition, the MS-DOS Error Code can be examined under program control by use of the $OSERR statement. Refer to the $OSERR statement in the NPL Statements Guide.

APPENDIX D

"RAW" DEVICE COMPATIBILITY CHART

D.1 "Raw" Diskette Chart

This chart lists the "raw" diskette compatibility for all operating system groups currently supported by NPL.

NPL "RAW" DEVICE COMPATIBILITY TABLE									
				5-1/4" Diskettes			3-1/2" Diskettes		
System	Drive Type	$FORMAT	320K	360K	720K	1.2MB	720K (8)	1.44MB (8)	2.88MB (9)
Wang 2200/CS	360K	X	X	X(1)	-	-	-	-	-
	1.2MB*	X(2)	-	-	-	-(3)	-	-	-
MS-DOS	360K	X	X	X(4)	-	-	-	-	-
	720K	X	-	-	-	-	X(4)	-	-
	1.2MB*	X	X	X(4)	-	X(4)	-	-	-
	1.44MB	X	-	-	-	-	X(4)	X(4)	-
	2.88MB	X	-	-	-	-	X(4)	X(4)	X(4)
Intel UNIX &Xenix	360K	(5)	-	X	X	-	-	-	-
	720K	(5)	-	-	-	-	X	-	-
	1.2MB*	(5)	-	X	X	X	-	-	-
	1.44MB	(5)	-	-	-	-	X	X	-
	2.88MB	(5)	-	-	-	-	X	X	X
SuperDOS	360K	X	X	X(4)	-	-	-	-	-
	720K	X(6)	-	-	-	-	X(4)	-	-
	1.2MB*	X	X	X(4)	-	X(4)	-	-	-
	1.44MB	X(6)	-	-	-	-	X(4)	X(4)	-
	2.88MB	X(6)	-	-	-	-	X(4)	X(4)	-
Honeywell XPS-100	720K	(7)	-	-(7)	X	-	-	-	-
Bull DPX/2	720K	(7)	-	-(7)	X	-	-	-	-
NCR TOWER	1.2MB	(10)	X(11)	X(11)	X(11)	-	-	-	-
IBM RS/6000	1.44MB	(5)	-	-	-	-	X	X	-

X SUPPORTED

* Any 360K diskette (DSDD) which has been written to in a 1.2MB drive may be unreliable when accessed on a 360K drive. This is a stated hardware limitation of the 1.2MB drive technology.

NOTE: Diskettes which have been written to on 1.2MB drives can always be read successfully on another 1.2MB drive. In addition, diskettes which have been created on a 360K drive can always be read successfully on any 1.2MB drive. This restriction applies only to reading diskettes on a 360K drive which have been written to on a 1.2MB drive.

NOTES:

(1) Using the "PC Interchange" format. A special $GIO microcommand sequence is required to format 360K diskettes. Refer to Section 15.7.1 of the Programmer's Guide for details.

(2) Neither 360K or 320K diskettes can be formatted in the 1.2MB diskette drive on the DS. 1.2MB diskettes created in the native 2200 format (256 byte sectors) on the Wang DS are not supported on any NPL machine.

(3) Although the Wang Documentation specifies that "PC Interchange" format is supported on the 1.2MB diskette, our testing to date has not yielded positive results.

(4) Access to 360K, 1.2MB, 720K, 1.4MB, and 2.88MB "RAW" diskettes under MS-DOS and SuperDOS (excluding 2.88MB) is supported by RTP and RTI, but not by B2C, which requires 320K "raw" formatted diskettes..

(5) $FORMAT DISK is not supported under Intel UNIX or Xenix. Refer to the Intel UNIX Supplement for details.

(6) $FORMAT DISK for 3.5 inch diskettes is not supported under Protected Mode SuperDOS.

(7) Many restrictions apply to the use of 360K "raw" diskettes on the Honeywell Bull XPS-100 and Bull DPX/2. Refer to Section 10.3 of the appropriate UNIX Addendum in the NPL Release III UNIX V Supplement for more information.

(8) Support of 3.5" 720K and 1.44MB diskettes require Release 3.00 or higher.

(9) Support of 3.5" 2.88MB diskettes require Release 4.00 or higher.

(10) The $FORMAT DISK command is not supported for any "raw" diskettes on the NCR TOWER 32.

(11) Only read and write operations of "raw" diskettes are supported. It is possible to format 640K diskettes using the UNIX format command. According to the NCR documentation, 720K diskettes are not supported. This is because the UNIX format command does not always succeed when a 720K diskette is specified. As a result, you may not be able to format 720K diskettes on the NCR TOWER 32 systems, It is not possible to format 320K or 360K diskettes. Therefore, 320K and 360K diskettes must be preformatted on another system.

For information regarding naming conventions and accessing "raw" diskettes within specific hardware environments, refer to Chapter 5 of the appropriate NPL Supplement.

APPENDIX C

CO-PROCESSOR SUPPORT

C.1 Co-Processor Implementation

Use of the 80x87 driver may be enabled by setting byte 16 of the $OPTIONS system variable. Valid values for the $OPTIONS byte are:

HEX(00) - Do not use math co-processor even if available.
HEX(01) - Use math co-processor for transcendentals if available.

Other values are reserved and should not be assigned to this byte.

Applications which now use the math co-processor should be sure to set byte 16 to HEX(01) or they will experience a decrease in performance, since the RunTime default is not to use the co-processor.

For example:

```
0010 DIM X$64
  : X$=$OPTIONS
  : STR(X$,16,1)=BIN(1)        :REM use co-processor
  :$OPTIONS=X$
```

NOTE: The $MACHINE system variable indicates whether a math co-processor is available. Byte 10 of $MACHINE contains the following values:

HEX(00) - No co-processor available
HEX(01) - Co-processor available

Programmers making use of the math co-processor should be aware that there may be differences in precision of results and range of functional domain to some functions. In particular, the 8087-class co-processors are generally accurate with 48-bit precision and have an exponent of 2 in the range +/- 16383. This will not normally present a problem except where results or arguments approach overflow or underflow values.