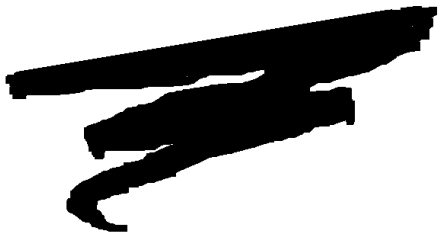
NIAKWA PROGRAMMING LANGUAGE

SUPERDOS SUPPLEMENT



1st Edition - September 1993

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

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

DISCLAIMER OF WARRANTIES AND LIMITATION OF LIABILITIES AND PROPRIETARY RIGHTS

The staff of Niakwa, Inc. (Niakwa) has taken due care in preparing this manual. Nothing contained herein shall be construed to modify or alter in any way the standard terms and conditions of the Niakwa Programming Language (NPL) Support and Distribution License Agreement, the End-User Support Only License Agreement, the Niakwa Software License Agreement and Warranty and any other Niakwa License Agreement (collectively, the "License Agreements") by which this software package was acquired.

This manual is to serve as a guide for use of the Niakwa software only and not as a source of representations or additional undertakings by Niakwa. The licensee must refer to the License Agreements for Niakwa product and service representations.

No ownership of Niakwa software is transferred by any of the License Agreements. Any use of Niakwa software beyond the terms and conditions of the License Agreements, without the written authorization of Niakwa, is prohibited.

PREFACE

P.1 The NPL SuperDOS Supplement

The Niakwa Programming Language (NPL) SuperDOS 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
Bluebird approved IBM-Compatible PC	SuperDOS 5.3 or greater	SuperDOS Protected Mode

P.1.1 Prerequisite Knowledge

This guide assumes at least a basic knowledge of the IBM Personal Computer and the SuperDOS version 5.3 or greater.

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 SuperDOS Supplement discusses the operation of NPL under SuperDOS.

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.

Niakwa NPL Development Package Diskettes
This refers to the physical diskettes delivered as part of the Niakwa NPL Development Package. These diskettes 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 an 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

PREFACE

INTRODUCTION

INSTALLATION

CONFIGURATION

RUNTIME OPERATION

DEVICE SUPPORT

SUPPORTED DISPLAY CHARACTERISTICS

MULTI-USER CAPABILITIES

PLATFORM-SPECIFIC LANGUAGE FEATURES

COMPILER OPERATION

PORTING PROGRAMS AND DATA

MIXED LANGUAGE PROGRAMMING

COMMON PROBLEMS

SUPERDOS ERROR CODES

PERFORMANCE ISSUES

"RAW" DEVICE COMPATIBILITY CHART

CHAPTER 1

INTRODUCTION

1.1 Overview

The NPL Supplement for SuperDOS 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 SuperDOS.

Section 1.5 provides an overview of the NPL SuperDOS Supplement

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 under the SuperDOS operating system. The NPL Development Package consists of the NPL SuperDOS Supplement and four diskettes.

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

Compiler Diskette

B2C	The NPL Compiler program.
DISKID	A file used by SuperDOS for diskette identification purposes.
ENABLED	The "enable" file necessary for allowing the interpretive RunTime program to function.
B2Cx.B	Example EXEC files showing command-line use of the compiler designed to simplify use.
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.
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.
BOOT.SRC	The default RunTime boot program in source format.

Utilities Diskette

DISKID	A file used by SuperDOS for diskette identification purposes.
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

DISKID	A file used by SuperDOS for diskette identification purposes.
W2236INI	A file used in conjunction with the support of Wang 2236DE and 2236DW terminals.
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.
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.

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 Niakwa RunTime is 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 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

Niakwa NPL RunTime Package File Listing

The Niakwa RunTime Package contains the following files:

DISKID	A file used by SuperDOS for diskette identification purposes.
NIAKREG2	A file required by security.
RTI	The Interpretive RunTime entry program.

RTIEXEC	The file necessary for allowing execution of the RunTime program from EXEC files or execution of Business Basic SWAP and CHAIN statements.
RTISHARE	The Interpretive RunTime program.
RTISTOP	The file necessary for terminating the RTISHARE task.
RTP	The Non-interpretive RunTime entry program.
RTPSHARE	The Non-interpretive RunTime program.
ERRORMSG.IDX	File containing the index used when the ERRORMSG.HLP file is accessed.
RTISERR.IDX	File containing the index used when the RTIIERR.HLP file is accessed.
ERRORMSG.HLP	A text file that contains the error messages that are displayed when using the Interpretive RunTime program.
RTISERR.HLP	A text file that contains the SuperDOS error messages that are displayed optionally when using the Interpretive RunTime program.
RTISHELL.SH	A file necessary to allow the RunTime to execute the $SHELL statement.

1.4 SuperDOS Specific Features of NPL

The following are SuperDOS specific features of NPL.

Hardware Support

The SuperDOS RunTime program supports multi-user operation on individual Bluebird approved IBM-compatible PCs under SuperDOS. The SuperDOS operating system on the IBM-compatible PCs has several hardware characteristics which provide a high level of support for several hardware-dependent features of NPL:

- 360K, 720K, 1.2MB, and 1.44MB "raw" diskettes are supported. Refer to Section 5.2 and Appendix D for details.
- Color is supported for specific controllers and monitors for the console and Wyse 370 terminal. Refer to Chapter 6 for details.

1.5 SuperDOS Supplement Overview

The SuperDOS Supplement is designed to provide information on operating NPL on SuperDOS, 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 SuperDOS systems. It also discusses the NPL security under SuperDOS.

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 and math coprocessors.

Chapter 6 discusses display characteristics of NPL on various monitor types for the console monitor, including monochrome, CGA, EGA, VGA and supported terminals. It also discusses color capabilities and keyboard characteristics.

Chapter 7 discusses multi-user issues for SuperDOS systems and how they can coexist under SuperLAN networks.

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

Chapter 9 covers operation of the B2C compiler and conventions used under SuperDOS.

Chapter 10 discusses options for porting programs and data to SuperDOS systems from other platforms.

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

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

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

Appendix C discusses performance issues of NPL under SuperDOS.

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

CHAPTER 2

INSTALLATION

2.1 Overview

This chapter provides instructions for installing the SuperDOS implementation of the Niakwa NPL Development and RunTime Packages on approved SuperDOS compatibles.

Section 2.2 discusses operating system and hardware requirements.

Section 2.3 discusses NPL memory requirements and allocation under SuperDOS.

Section 2.4 discusses installing the NPL Development Software under SuperDOS.

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

Section 2.6 discusses installing the NPL RunTime software.

2.2 OS and Hardware Requirements

The following section discusses the operating system and hardware requirements for the SuperDOS implementation of NPL.

2.2.1 System Requirements

The SuperDOS implementation of the Niakwa Programming Language is designed to operate on Bluebird approved IBM-compatibles PCs, meeting the following requirements:

- SuperDOS version 5.3.
- Bluebird Business Basic Token Processor version 5.3.

NOTE: Release IV of the Niakwa Programming Language only operates under Protected Mode SuperDOS. In addition, software versions listed here are the minimum versions required. Use of current versions of SuperDOS is highly recommended.

2.2.2 Supported Terminals

Of the terminals supported by NPL, only the Wyse 50 and 60 are supported by the native SuperDOS operating system. Other NPL terminals operate correctly within the NPL environment, however, these terminals are not supported or well suited for use with native SuperDOS functions and utilities. If used, these terminals should be configured with automatic passwords to log directly into NPL.

NOTE: The recommended terminal to use for NPL under SuperDOS is the Wyse 60.

Supported NPL terminals include:

- Altos III and V
- DEC VT100 series
- DEC VT200 series
- Wang 2110A

- Wang 2236DE and 2236DW
- Wyse 50
- Wyse 60,150, and 160
- Wyse 370

2.2.3 Printer Support

Both parallel and serial system printers are supported. In addition, local printers are supported for terminals with either parallel or serial printer ports. Refer to Appendix D of the NPL Programmer's Guide for a complete discussion of all Niakwa-supported terminals.

2.3 NPL Memory Requirements

The following sections details the memory requirements for NPL under SuperDOS.

2.3.1 Compiler Requirements

The compiler (B2C) requires a user task size of at least 196K and operates under Protected Mode SuperDOS only.

HINT: Because the NPL compiler is not frequently necessary at most end-user sites, Niakwa recommends that a separate configuration be used when access to B2C is required.

2.3.2 RunTime Memory Requirements

To maximize the number of users who can execute the NPL RunTime, the RunTime under SuperDOS has been split into two tasks, a "shareable" task, (RTISHARE or RTPSHARE) which must be configured as a background task, and a "user" task. The user task contains some non-shareable overhead required by the RunTime, plus the user partition. The following describes of each of these components in detail.

Shareable Task

The shareable task (RTISHARE or RTPSHARE) is a background task which is required by the RunTime. The base size of this task is 230K (for RTISHARE) or 146K (for RTPSHARE), depending on the RTxSHARE program being used. However, due to a RunTime requirement, a minimum of 2 "overflow" areas (see below for details) must be allocated to the task. Therefore, by adding the task base size to the required two overflow areas of 28K each, a minimum task size of 286K (230+ 56) for RTISHARE, or 202K (146+ 56) for RTPSHARE is required.

User Task Non-Shareable

The non-shareable portion of the user task is a 37K block of memory that contains Run-Time and system information which is unique to each user partition. Of this 37K block, 9K must always reside within the user task. The remaining 28K is referred to as an "overflow" area which may be allocated in either the user task or the RTxSHARE task. Refer to section 2.3.3 below for differences between allocating the overflow areas to the RTxSHARE task versus the user task.

User Partition

The user partition is the NPL RunTime partition generated within the user task after the non-shareable portion of the user task has been allocated.

2.3.3 Overflow Area Allocation Examples

This section illustrates the differences between allocating overflow areas for each user task to the RTxSHARE task (versus allocating it to the individual user task). When allocating overflow areas to the RTxSHARE task, the following table illustrates how to determine the required size of RTxSHARE task.

	Interpretive Runtime	**Non-Interpretive Runtime**
Base Shareable Task	230K	146K
Plus	N * 28	N * 28

where N is the number of overflow areas required

The following example illustrate five typical user configurations. Each example shows the amount of NPL memory required by the configuration and provides two methods of allocating available memory.

NOTE: The examples below show memory requirements when using RTISHARE. Memory requirements for RTPSHARE would be identical except that the base shareable task size would be 84K less.

8 Users Running in 55K Partitions

Memory Calculation:	Shareable Task (RTISHARE)		230K
	Non-Shareable Tasks	8 x 37K	+ 296K
	User Partition	8 x 55K	+ 440K
	Total NPL Memory		966K

Once the amount of memory required has been determined, it is then possible to determine how to allocate the memory for the configuration.

Method 1 - Allocating Overflow Areas to the Shareable Task

Shareable Task (RTISHARE) = 454K This is determined by adding the base size of RTISHARE (230K) and the eight overflow areas (8 x 28K) taken from the eight non-shareable tasks.

User Tasks 8 @ 64K = 512K The size of each user task is determined by adding the size of each user partition to the remaining unallocated portion of the non-shareable task (55K + 9K).

Method 2 - Allocating Overflow Areas Within the User Tasks

Shareable Task (RTISHARE) = 286K This is determined by adding the base size of RTISHARE (230K) and the two overflow areas (2 x 28K) required by NPL. These two overflow areas are used by the first two user tasks that access the RunTime.

User Tasks 2 @ 64K
6 @ 92K = 680K Determination of the 64K tasks is the same as in Method 1. The 92K task sizes are determined by adding the user partition size to the entire non-shareable task (55K + 37K), effectively placing the overflow areas of these tasks within the user task.

NOTE: Proper use of Method 2 would require that the /L RunTime startup option be used on the six tasks configured with 92K. This reserves the two required overflow areas for the tasks with 64K Refer to Section 4.4 for a description of all RunTime startup options.

Eight Users Running in 30K Partitions

Memory Calculation:	Shareable Task (RTISHARE)		230K
	Non-Shareable Tasks	8 x 37K	+ 296k
	User Partition	8 x 30K	+ 240K
	Total NPL Memory		766K

Once the amount of memory required has been determined, how to allocate the memory for this configuration can be determined.

Method 1 - Allocating Overflow Areas to the Shareable Task

Shareable Task (RTISHARE) = 454K This is determined by adding the base size of RTISHARE (230K) and the eight overflow areas (8 x 28K) taken from the eight non-shareable tasks.

User Tasks 8 @ 39K = 312K The size of each user task is determined by adding the size of each user partition to the remaining unallocated portion of the non-shareable task (30K + 9K).

Method 2 - Allocating Overflow Areas Within the User Tasks

Shareable Task (RTISHARE) = 286K This is determined by adding the base size of RTISHARE (230K) and the two overflow areas (2 x 28K) required by NPL. These 2 overflow areas are used by the first two user tasks that access the RunTime.

User Tasks 2 @ 39K
6 @ 67K = 692K Determination of the 39K tasks is the same as in Method 1. The 64K task sizes are determined by adding the user partition size to the entire non-shareable task (30K + 37K), effectively placing the overflow areas of these tasks within the user task.

NOTE: Proper use of Method 2 would require that the /L RunTime startup option be used on the six tasks configured with 64K. This reserves the two required overflow areas for the tasks with 39K. Refer to Section 4.4 for a description of all RunTime startup options.

8 Users Running in 75K Partitions

Memory Calculation:	Shareable Task (RTISHARE)		230K
	Non-Shareable Tasks	8 x 37K	+ 296K
	User Partition	8 x 75K	+ 600K
	Total NPL Memory		1126K

Once the amount of memory required has been determined, how to allocate the memory for this configuration can now be determined.

Method 1 - Allocating Overflow Areas to the Shareable Task

Shareable Task (RTISHARE) = 454K This is determined by adding the base size of RTISHARE (230K) and the eight overflow areas (8 x 28K) taken from the eight non-shareable tasks.

User Tasks 8 @ 84K = 672k The size of each user task is determined by adding the size of each user partition to the remaining un-allocated portion of the non-shareable task (75K + 9K).

Method 2 - Allocating Overflow Areas Within the User Tasks

Shareable Task (RTISHARE) = 286K This is determined by adding the base size of RTISHARE (230K) and the two overflow areas (2 x 28K) required by NPL. These two overflow areas are used by the first two user tasks that access the RunTime.

User Tasks 2 @ 84K
6 @ 112K = 840K Determination of the 84K tasks is the same as in Method 1. The 112K task sizes are determined by adding the user partition size to the entire non-shareable task (75K + 37K), effectively placing the overflow areas of these tasks within the user task.

NOTE: Proper use of Method 2 would require that the /L RunTime startup option be used on the six tasks configured with 109K. This reserves the two required overflow areas for the tasks with 84K. Refer to Section 4.4 for a description of all RunTime startup options.

16 Users Running in 55K Partitions

Memory Calculation:	Shareable Task (RTISHARE)		230K
	Non-Shareable Tasks	16 x 37K	+ 592K
	User Partition	16 x 55K	+ 880K
	Total NPL Memory		1702K

Once the amount of memory required has been determined, how to allocate the memory for this configuration can now be determined.

Method 1 - Allocating Overflow Areas to the Shareable Task

Shareable Task (RTISHARE) = 678K This is determined by adding the base size of RTISHARE (230K) and the sixteen (16) overflow areas (16 x 28K) taken from the sixteen (16) non-shareable tasks.

User Tasks 16 @ 64K = 1024K The size of each user task is determined by adding the size of each user partition to the remaining unallocated portion of the non-shareable task (55K + 9K).

Method 2 - Allocating Overflow Areas Within the User Tasks

Shareable Task (RTISHARE) = 286K This is determined by adding the base size of RTISHARE (230K) and the two overflow areas (2 x 28K) required by NPL. These two overflow areas are used by the first two user tasks that access the RunTime.

User Tasks 2 @ 64K
14 @ 89K = 1416K Determination of the 64K tasks is the same as in Method 1. The 92K task sizes are determined by adding the user partition size to the entire non-shareable task (55K + 37K), effectively placing the overflow areas of these tasks within the user task.

NOTE: Proper use of Method 2 would require that the /L RunTime startup option be used on the 14 tasks configured with 92K. This reserves the two required overflow areas for the tasks with 64K. Refer to Section 4.4 for a description of all RunTime startup options.

16 Users Running in 30K Partitions

Memory Calculation:	Shareable Task (RTISHARE)		230K
	Non-Shareable Tasks	16 x 37k	+ 592K
	User Partition	16 x 30K	+ 480K
	Total NPL Memory		1302K

Once the amount of memory required has been determined, how to allocate the memory for this configuration can now be determined.

Method 1 - Allocating Overflow Areas to the Shareable Task

Shareable Task (RTISHARE)	= 678K	This is determined by adding the base size of RTISHARE (230K) and the 16 overflow areas (16 x 28K) taken from the 16 non-shareable tasks.
User Tasks 16 @ 39K	= 624K	The size of each user task is determined by adding the size of each user partition to the remaining unallocated portion of the non-shareable task (30K + 9K).

Method 2 - Allocating Overflow Areas Within the User Tasks

Shareable Task (RTISHARE)	= 286K	This is determined by adding the base size of RTISHARE (230K) and the two overflow areas (2 x 28K) required by NPL. These two overflow area are used by the first two user tasks that access the RunTime.
User Tasks 2 @ 39K 14 @ 67K	= 1016K	Determination of the 39K tasks is the same as in Method 1. The 67K task sizes are determined by adding the user partition size to the entire non-shareable task (30K + 37K), effectively placing the overflow areas of these tasks within the user task.

NOTE: Proper use of Method 2 would require that the /L RunTime startup option be used on the 14 tasks configured with 67K. This reserves the two required overflow areas for the tasks with 39K. Please refer to Section 4.4 for a description of all RunTime startup options.

2.3.4 SuperDOS Memory Considerations

The following should also be considered when configuring the RunTime tasks.

Task Orientation

SuperDOS is a task-oriented operating system with a fixed size memory allocation defined at boot time for each task. Memory allocation under SuperDOS is static. That is, the total amount of memory assigned to all defined tasks cannot exceed the amount of physical memory present on the system. There is no "swapping" or "paging" of active programs as in UNIX or VMS.

Available Memory

WARNING -- Release IV of NPL must be run in Protected Mode. Refer to the SuperDOS installation guide for details on memory configurations.

Protected Mode operation allows for direct addressing of the full amount of the extended memory installed, due to the linear fashion in which Protected Mode operates. Since the entire amount of memory is available for use, there is no restriction as to where either system software, optional tasks or terminal tasks must reside. In addition, any task can be created up to a maximum size of 999K.

MS-DOS Concurrency

When Protected Mode SuperDOS is executed, it completely occupies all system resources, MS-DOS is overwritten and its memory is reclaimed by SuperDOS. Therefore, under Protected Mode there is no support for MS-DOS operations.

Although there is no MS-DOS concurrency in Protected Mode, the ability to perform file transfers from a DOS environment to SuperDOS is still often required. This can be accomplished in two ways:

- To transfer files from SuperDOS to an MS-DOS partition on the SuperDOS server, use the PCXFER utility.

- To transfer files from a SuperDOS system to a remote PC, developers should consider using PC-Connect to interface the remote system with SuperDOS. The PC-Connect file transfer utility is then used to perform the transfer process.

For details on PCXFER, consult the SuperDOS Utilities Guide. Refer to Section 7.8 for details on using PC-Connect with NPL. For information on obtaining PC-Connect, contact Bluebird Systems.

Protected Mode Task Configuration

When operating in Protected Mode, all terminal tasks can be properly configured with enough memory to generate any required size for the user partition without using the overhead area stored in the shareable task (refer to Section 2.3.3, Method 2 examples for details). This is the recommended way to allocate NPL memory when on a system running in Protected Mode. When configuring a system in this manner, the following should be considered:

- The shareable task (RTxSHARE) must be allocated enough memory for a minimum of two overflow areas as required by NPL.
- All user tasks (except for two) should be configured for the required size of the user partition, plus 37K (non-shareable overhead).
- Two of the user tasks may be configured for the required size of the user partition, plus 37K less the 28K overflow area.
- At this point, user tasks which have been configured to not require an overflow area must be suppressed from claiming an overflow area, from the RTxSHARE task. This must be done to ensure that only the 2 user tasks that have been configured to require the overflow areas in the RTxSHARE task, are allowed to do so. This is accomplished by use of the Limited Memory (/L) option of the RunTime. The /L option may be used for those tasks which can operate without the overflow area. Refer to Section 4.4 for details on the use of the /L option.

2.4 Installing the NPL Development Software

This section discusses the installation of the Niakwa NPL Development Package on a SuperDOS-based system. The Niakwa Development Package consists of four diskettes labeled:

- Compiler Diskette
- Utilities Diskette
- Terminal Support Diskette

- SuperDOS BESDK Diskette

The contents of these diskettes are described in Section 1.2. The first three diskettes should be installed on SuperDOS drive 5, in user group 0. 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 NPL RunTime Package to be installed on the host system. The Installation of the NPL RunTime Package is discussed in Section 2.6.

2.4.1 Installing the Development Software

The following installation instructions assume that 1: is the SuperDOS floppy drive and 5: is the hard drive. It is also assumed that the files are copied to user group 0. If the drive or user group 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 Development Package Software follow the steps below:

1. Login to SuperDOS as "BEGIN" (or any password with unrestricted level 7 access).

2. Insert the diskette labeled "Compiler Diskette" into drive 1: and enter:

```
COPY/D/V/F 5:0 1:0:*
```

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

2.4.2 Installing the BESDK Software

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

2.5 NPL RunTime Security

The following section 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 SuperDOS version of NPL is protected from unauthorized use by the use of a special hardware device which contains encrypted codes accessed by the RunTime. The codes used correspond to the RunTime user limit. For a given user limit, the code for that user limit and all codes for lower user limits must be present on the system.

Developers are allowed to make duplicate copies of the NPL diskettes for internal backup purposes only and are strongly encouraged to do so.

2.5.2 The SuperCODER

The hardware device used is a SuperCODER which attaches to the parallel port of the PC. This device is configured with the multiple codes necessary for successful operation of the SuperDOS system. SuperCODERs are the current security mechanisms shipping with all SuperDOS systems.

NOTE: On all SuperDOS systems staged by Bluebird, SuperCODER codes required for the NPL RunTime are installed on the system before it is shipped.

2.6 Installing the NPL RunTime Software

This section describes the process of installing and configuring the NPL RunTime Package for use under the SuperDOS operating system and define the tasks required for execution of the RunTime.

NOTE: Successful installation and execution of the NPL RunTime requires that the SuperDOS operating system is properly installed and operational. Follow the installation instructions provided by Bluebird Systems for installation of the SuperDOS operating system.

2.6.1 Installing the NPL RunTime Package

The following installation instructions assume that 1: is the SuperDOS floppy drive and 5: is the hard drive. It is also assumes that the files are copied to user group 0. If the drive or user group designations are different on the system where the NPL RunTime Software, is being installed, be sure to use the proper designations.

To install the Niakwa NPL RunTime Software follow the steps shown below:

1. Login to SuperDOS as "BEGIN" (or any password with unrestricted level 7 access.)
2. Insert the NPL Gold Key diskette into drive 1: and enter:

```
COPY/D/V/F 5:0 1:0:*
```

3. Repeat step 2 for the diskette labeled disk (2 of 2).

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

2.6.2 Configuring SuperDOS

SuperDOS is a task-oriented operating system. Each program and port is assigned to a specific task. These tasks are assigned at the startup of the SuperDOS operating system by use of the specifications in the "CONFIG.P" file (Protected Mode). This file contains information that is used to allocate devices, define tasks and automatically execute programs when the system is started. This file can be created and modified by using the SuperDOS SYSGEN or EDIT utility programs (refer to the SuperDOS Utilities Guide for details). For information regarding the CONFIG.P file, refer to Chapter 7 of the SuperDOS Guide to Operations.

2.6.3 Assigning the RunTime Background Task

NPL requires a background task to be defined for RTISHARE or RTPSHARE.

NOTE: RTISHARE and RTPSHARE are mutually exclusive. There must be a separate configuration file for each. In cases where it is desirable to execute RTP most of the time, but RTI is required occasionally, the alternative configuration file could be used.

The background task for RTISHARE or RTPSHARE is configured as follows:

Description:	RTISHARE or RTPSHARE, accordingly.
Memory Size:	Discussed in Section 2.3.
Password:	/RTI or /RTP, accordingly (refer below for details).

In general, the background task size should be configured no larger than required for the particular installation. This allows base memory to be used for other purposes such as cache buffers or a background tape task.

NOTE: If SuperDOS encounters an RTISHARE or RTPSHARE task that does not fit in base memory, when operating in Protected Mode, SuperDOS shifts the task into extended memory and attempts to place subsequent tasks into the unused base memory.

2.6.4 Define Terminal Tasks

For each port to be used as a terminal with NPL, the following parameters should be entered:

Device Type: Terminal

Terminal Kind: 1 for Wyse 50 or Wyse 60
4 for Wang 2236
2 or 3 for all others

Printer Kind: 0

Task Number: Accept default

Memory Size: 64K (refer below for further details on memory size)

Password: All Wang 2236 terminals should be configured with an automatic password. "W2236" is a recommended choice for the password. The password for the 2236 must be set up to execute the program W2236INI. Refer below for details on setting up this password.

Baud Rate: As required

Word Size: 8

Stop Bits: 1 for Wyse 50 or Wyse 60
2 for Wang 2236

Parity: N for Wyse 50 or Wyse 60
O for Wang 2236

Clear to Send: 0

Refer to Chapter 6 for further details on terminal configurations. Refer to the SuperDOS Utilities Guide and Guide to Operations for further details on SYSGEN.

Memory Size for Terminal Tasks

The recommended value of 64K provides a full 55K user partition for any task, given an overflow area is available, or a 27K partition for tasks with no overflow available.

Under Protected Mode, larger task sizes may be configured if larger user partitions are required.

To use the NPL compiler (B2C), a 196K terminal task must be allocated. Refer to Section 2.3 for details on configuring a task for using the NPL compiler (B2C).

2.6.5 Other Configuration Parameters

There are a few other parameters to consider when configuring SuperDOS for use with NPL. These include:

- Number of Lock Entries (third parameter of line 102 in the CONFIG.P file) should be set to at least one for each terminal task configured. The standard SuperDOS default of 32 is acceptable.

- Number of Cache Buffers (forth parameter of line 102 in the CONFIG.P file) should be set to the maximum value which can fit, given other memory requirements. A minimum of one cache buffer per terminal task is essential. Higher values of cache buffers can improve disk performance, sometimes substantially, depending on the nature of the application.

- Dynamic File Buffers (line 108 in the CONFIG.P file - sometimes referred to as "Extent Buffers") should be set to the maximum number of "extended" files opened at any one time. Under NPL, "extended" files are diskimage files where the size has been increased using MOVE END or where there was insufficient contiguous disk space to create the file in one extent using SCRATCH DISK or PCXFER. In addition, any spooled print files may be extended.

NOTE: Failure to allocate sufficient extent buffers can result in severe performance degradation when accessing files with multiple extents. A value of 10 dynamic file buffers is recommended. Refer to Section 5.3.3 for further details on SuperDOS file allocation considerations.

It is recommended that a buffer length of 8 characters be set for ports with Wang 2x36 terminals, when using 550A buffered UARTS. This is accomplished through the addition of a 111 line to the CONFIG file.

For example:

```
111 2,8,1 5,8,1
```

the above would set the transmit length of ports 2 and 5 to 8 characters. All other ports use the default of 16. Higher buffer lengths may result in flow control problems when using Wang 2x36 terminals, particularly when using a local printer. Refer to the SuperDOS Guide to Operations for details.

A RAM Disk (line 114 of the CONFIG.P file) may be configured and used with NPL. This can increase performance significantly. Recommended use of a RAM disk would be to store and access program diskimages or work files.

WARNING -- User data should not be stored in a RAM disk.

2.6.6 Creating Passwords with PASSWORDFM

Once a configuration file has been defined, passwords need must be established for the RTISHARE or RTPSHARE background tasks and for any terminal tasks established with automatic passwords.

To do this, use the SuperDOS PASSWORDFM utility. Passwords for RTISHARE or RTPSHARE should be set up as follows:

Password:	/RTI or /RTP
User Level:	7
Security Flags:	All user groups that need to be accessed.

Searchlist: 5:0

Initial Program Command Line: RTISHARE or RTPSHARE

If the configuration contains Wang 2236 terminals, all non-background passwords should be set up as:

Password: Use the same password used in the configuration file ("W2236" is a good choice for the password).

User Level: As required for the configuration.

Security Flags: As required for the configuration.

Searchlist: 5:0 must be included. Other searchlist groups may be optionally added.

Initial Program Command Line: W2236INI /4

The W2236INI program can be executed two different ways:

- It could be executed on all ports with a defined password with an Initial Program Command Line of W2236INI.
- The W2236INI program can be defined to execute selectively on a port if the terminal kind assigned to that port is the terminal kind specified.

For example:

```
W2236INI /x RTI [command]
          or
W2236INI /x RTP [command]
```

where:

x is the terminal kind number used to execute W2236INI and command is any valid SuperDOS command.

For Example:

```
W2236INI /4 RTI UTILITY
```

The example above is suitable for entry on the initial program command line for any password. This command:

- Sets XON/XOFF characteristics to Wang 2236 mode if the terminal kind for the task is 4.
- Sets XON/XOFF characteristics to normal for any other terminal kind.
- Automatically executes RTI with a boot program name of UTILITY.

Refer to the SuperDOS Utilities Guide for further details on PASSWORDFM.

2.6.7 Terminal Configuration

On most systems, determination of the terminal type is made by examination of native operating system environment variables. However, SuperDOS does not maintain environment variables. For NPL to operate properly, it must be informed of the type of terminal it is working with. This is done by using a file named TERMTYPE.TBL. This file is used to inform the RunTime of the terminal type for any physical port or for any given terminal "kind". The RunTime then uses this terminal type in two ways:

- To determine the filename of the KEYBOARD and SCREEN translation tables. This is done by appending the extension .xxx to the word KEYBOARD and SCREEN where xxx is the first three characters of the terminal type specified.

For example:

```
SCREEN.ALT
KEYBOARD.ALT
```

Refer to Chapter 6 for further information on the use of KEYBOARD and SCREEN translation tables.

- To access tables maintained internally by NPL for various control sequences and other internal logic.

The file TERMTYPE.TBL is not provided on the RunTime diskette. This file can be created by using EDIT (refer to the SuperDOS Utilities Guide for details). Entries can be one of three different types:

port number TERMTYPE	Defines TERM for a specific port.

type number* TERMTYPE	Defines TERM for a specific terminal type number.
* TERMTYPE	Defines TERM for all other terminals.

The following is a example of a TERMTYPE.TBL file:

```
1 WY50
2 WY60
4* W2236
* VT100
```

In the above example:

- Port 1 is defined as a Wyse 50 terminal.
- Port 2 is defined as a Wyse 60 terminal.
- Terminal kind 4 is defined as a Wang 2236. All ports with terminal kind 4 as defined in the CONFIG.P file are treated as Wang 2236 terminals.
- All ports other than Ports 1, 2 or any port defined as terminal type 4 are treated as VT100 terminals.

NOTE: The first line which applies to a terminal always defines the terminal type.

HINT: The Wyse 50 and Wyse 60 should be setup as terminal kind 1 and the Wang 2236 terminals be setup as terminal kind 4. Other terminals may be assigned to terminal kinds 2 or 3. With this configuration the terminal "kind" can be used in TERMTYPE.TBL instead of setting up an entry for every port.

NOTE: In cases where it is necessary to distinguish between Wyse 50 and Wyse 60 terminals, TERMTYPE.TBL must be set up by port.

2.6.8 Other Files Required

There are several files in the Development Package that developers may want to consider installing at end-user sites:

- Keyboard files.
- Screen Translation files.

- Downloadable font files.

Default keyboard and screen translation files for all supported terminals are provided in the NPL Development Package. In addition, downloadable font files are provided for those terminals which support them. For all terminals other than the system console, it is necessary for appropriate keyboard, screen translation files, and font files (if required), to be installed on the end-user's system. These may be copies of the default files provided by Niakwa or versions which have been modified. Refer to Chapter 6 for further details on terminal usage.

NOTE: Execution of the interpretive RunTime (RTI) requires that the ENABLED file be present on the end-user's system. Refer to Section 3.5.1 for details.

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 SuperDOS file system and how to access them from NPL. The method of setting up NPL applications to operate in a SuperDOS file system are discussed this chapter.

Section 3.2 discusses the SuperDOS file system.

Section 3.3 discusses the location of all 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 an EXEC file to invoke the RunTime Package.

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

3.2 SuperDOS's File System

SuperDOS divides each physical disk drive into 64 logical sections or "user groups". The user groups are logical rather than physical groups and are used instead of the "directories" provided by some other operating systems (i.e., MS-DOS). Instead of directories, SuperDOS allows for locating files in individual user groups on a particular drive. File specifications then refer to files located on a particular drive/user group.

By installing the application software in different drive/user groups, multiple application systems can be organized, This allows for ease of operation (i.e., separate backups of individual systems or user groups).

3.2.1 File "Search" Considerations under SuperDOS

Based on the way SuperDOS locates files in the system, programs or data files which contain duplicate names, even when located in different drive/user groups, could pose serious problems.

This is due to the fact that when a file name is referenced, without specifying the exact drive/user group in which the file resides, SuperDOS begins scanning the drive/user groups indicated in the searchlist (found in the Password File) for the individual user. If the file name is not located in the first drive/user group, SuperDOS then looks in the next "group" in the list and continues until the file is found.

Depending upon which drive/user group combinations are listed in the searchlist and the order in which they are listed, it is conceivable that the wrong file may be found first. Therefore, it is strongly recommended that all files on the system which may have duplicate names (PLATTER1.BS2, for example) are located and renamed to avoid confusion. Alternatively, keep searchlists short and always make sure that "private" user groups precede shared or public ones in the searchlist.

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 5:0: drive/user group (disk drive 5, user group 0). The 5:0 user group should be in every user's searchlist, preferably as the last item. Refer to Chapter 2 for details on installation of the NPL Development and RunTime Packages.

3.4 Installation of Application Software

The steps described in this section are for installing an example NPL application at an end-user site (as opposed to a development system) under SuperDOS. 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 a SuperDOS system.

HINT: It is recommended that each NPL application be placed in its own user group under SuperDOS.

To be able to access programs and data in a particular drive/user group under SuperDOS without having to specify the actual drive/user group designation, the drive/user group must be found in the searchlist. The SuperDOS PASSWORDFM utility, is required to specify the drive/user groups to be included in the searchlist. Refer to the SuperDOS Utilities Guide, PASSWORDFM, for details.

If the application is already in SuperDOS format, perform the following steps:

1. Determine the user groups in which the application program and data files are to be installed (refer below).

2. Copy the application files into their appropriate user groups.

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 drives/user groups.

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:

5:25	Contains the accounts payable programs, stored in a diskimage file named APPROGS.BS2
5:26	Contains the accounts payable application data, stored in a diskimage file named APDATA.BS2.
5:27	Contains the accounts receivable application programs, stored in a diskimage file named ARPROGS.BS2.
5:28	Contains the accounts receivable application data, stored in a diskimage named ARDATA.BS2

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 SuperDOS commands (this assumes the 5:0: user group exists on the host system):

```
COPY /D/V/F 5:0 1:0:FILENAME
```

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 SuperDOS commands:

On the development system:

```
COPY /D/V/F 1:0:  5:0:FILENAME
```

On the end-user system:

```
COPY /D/V/F 5:0: 1:0:*
```

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 a user group on the user's searchlist.

If the Interpretive RunTime is invoked without the presence of the ENABLED file in a drive/user group specified in the SuperDOS searchlist, the message:

```
"Interpreter not enabled"
```

appears, and RunTime execution is canceled.

Installing ENABLED in the end user's 5:0: drive/user group enables the Interpreter for all applications.

HINT: It is also possible to install the ENABLED file only in the specific application's user group. 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 keyboards of many terminals used by NPL are not completely compatible with the NPL character set. These keyboard differences are resolved by the use of a simple lookup 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 using the NPL Keyboard Translation Tables Editor. This utility creates a disk file named KEYBOARD.TBL. NPL first searches for the KEYBOARD.TBL file in the searchlist. If the file is not found, the built-in default values are used for keyboard translation.

The NPL Development Package provides a series of KEYBOARD.xxx files. These files are used by NPL for use with the various terminals supported by the RunTime. These files are located on the NPL Terminal Support Diskette.

To modify any of the default keyboard translation tables for use by the end-user, follow these steps:

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 Chapter 13, of the NPL Programmer's Guide, for details on the use of the NPL Keyboard Translation Tables Editor utility. Refer to Appendix D of the NPL Programmer's Guide for details on supported terminals. Also, refer to Chapter 6 of this Supplement for details of the IBM-compatible PC keyboard characteristics, for the console terminal under the SuperDOS implementation of NPL.

WARNING--If reinstallation of the Terminals Support Files Diskette is necessary (for new versions), any changes made to the above files are overwritten. Consequently, a backup copy of all customized files should be made at the time of their creation and stored in a safe place.

3.5.3 Screen and Font Files

The Wang 2200 character set is not fully supported on all the terminal screens supported by NPL. Compatibility is achieved through a simple lookup table, SCREEN.xxx. This file translates various characters into hex codes which generate an equivalent, or at least similar, display character. Refer to Chapter 6 for the appropriate file names for the screen translation files, for a particular terminal. This table can be dynamically modified by using the $SCREEN statement or it can be permanently modified by use of the NPL Screen Translation Tables Editor utility. In addition, a number of NPL-supported terminals support the use of downloadable fonts. If modification is required, it can be performed by the use of the NPL Font Table Editor utility included in the NPL Utilities. Refer to Chapter 6 of this Supplement for details on downloadable fonts. Refer to Chapter 13 and Appendix D of the NPL Programmer's Guide for details on the NPL Utilities and supported NPL Terminals, respectively.

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.

NOTE: The VT 220 series of terminals and the Wyse 370 terminal have an additional file that needs to be downloaded for proper use of these terminals. This is performed by downloading the VTKEYS.VT2 and the WYKEYS.WY3 respectively. Refer to Chapter 6 in this Supplement and Appendix D of the NPL Programmer's Guide for details.

WARNING--If reinstallation of the Terminals Support Files diskette is necessary (for new versions), any changes made to the above files are overwritten. Consequently, a backup copy of all customized files should be made at the time of their creation and stored in a safe place.

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 NPL has built-in default control codes (refer to Section 4.6 for actual default values) which should work well with the IBM-compatible 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 PRINTCTL.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 PRINTCTL.TBL file in the SuperDOS searchlist. If the file is not 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, RTISERR.HLP and RTISERR.IDX, which should be copied to the drive/user group where the RunTime is installed. The ERRORMSG files are used to display NPL error code messages when an error occurs. The RTISERR 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 RTISERR.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 RTISERR.HLP files are modified, they must be processed by the NPL Indexed Help File Processor utility (refer to Section 13.16 of the NPL 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. Consequently, a backup copy of all customized files should be made at the time of their creation and stored in a safe place.

3.5.6 Additional End-User Security (#GOLDKEY)

The GOLDKEY.OBJ program allows developers to determine the #GOLDKEY number for any NPL 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

SuperDOS systems are multi-user NPL environments. Additional user tasks can be added to a system by modification of the SuperDOS configuration file (CONFIG.P). For details refer to the SuperDOS Guide to Operations. Refer to Section 2.3 for details on properly configuring user tasks for use with NPL.

3.7 Configuring User's Workstations

SuperDOS systems are multi-user NPL environments. NPL supports a number of terminals for use with the NPL RunTime. Refer to Chapter 6 for details on configuring terminals for use with NPL under SuperDOS.

3.8 Required Access Privileges

SuperDOS users should be given rights to the user groups that they must access. Refer to the SuperDOS Guide to Operations for details on granting access privileges to user groups.

3.9 Executing the RunTime from an EXEC File

Under all versions of SuperDOS, it can be advantageous to invoke the RunTime program from an "EXEC file". An EXEC file is a SuperDOS text file (type "T" 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 an EXEC 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 user group 5:10 and that the name of the "BOOT" program is ARBOOT. A typical EXEC file for executing this application could be called ARBOOT.B and would contain the following startup line:

```
RTP 5:20:ARBOOT (using the non-interpretive RunTime)
```

or

```
RTI 5:20:ARBOOT (using the interpretive RunTime)
```

To execute this EXEC file, enter the following SuperDOS command:

```
EXEC ARBOOT.B
```

3.9.1 Automatic Login using an EXEC File

A program can be automatically executed at login by indicating the name of the program to execute on the initial program command line in the SuperDOS password file. However, only one program or command may be executed at a time in this manner.

Multiple commands or programs can be specified in the EXEC file and then executed automatically through a password login. For example, to download fonts to a Wyse 60 terminal and then immediately invoke the RunTime, an EXEC file, named ARWY60.B, could appear as:

```
TYPE 5:0:WYFONT.WY6
RTI ARBOOT
```

If working with a Wang 2236 terminal, an EXEC file, named ARW2236.B, could appear as:

```
W2236INI
RTI ARBOOT
```

To execute the above EXEC files, it is necessary to set up different passwords and modify the Initial Program Command Line defined by the PASSWORDFM utility as follows:

```
EXEC ARWY60.B
```

or

```
EXEC ARW2236.B
```

3.10 Executing the RunTime from a Menu System

Under SuperDOS it is very simple to create EXEC files 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.

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

3.10.1 Creating the Menu

One method of creating a menu system is through the use of a user-written NPL program which creates the menu to be displayed.

A more user friendly approach to starting up an application is to start the application program(s) from a simple menu file. This can be created using the SuperDOS EDIT text editor.

For example, assume that a given installation has two applications: AR, located in the 5:17 drive/user group; and AP, located in the 5:18 drive/user group. Assume also that there are corresponding EXEC files for each application named AR.B and AP.B located in the 5:0 drive/user group:

AR.B contains:

```
RTP 5:17:ARBOOT
```

AP.B contains:

```
RTP 5:18:APBOOT
```

NOTE: RTI can also be used in both of the above EXEC files.

A simple menu file can be created which displays available options to the operator. For example, use EDIT to create a file named MENU.DAT in the 5:0 drive/user group (other drive/user group combinations can be used). The contents of this file are:

Enter the name of the program you wish to run

```
EXEC AP.B - Accounts Payable
EXEC AR.B - Accounts Receivables
```

NOTE: The program names listed correspond to the names of the EXEC files established for the application.

3.10.2 Displaying the Menu

This menu file can be displayed on the screen by use of the SuperDOS TYPE command.

For example:

```
TYPE 5:0:MENU.DAT
```

displays the menu.

The command required to display the menu should by added to the actual EXEC files used to invoke the application so that the menu is displayed when the user exits a particular application.

For example, the AR.B file can be modified to:

```
RTP  5:17: ARBOOT
TYPE  5:0: MENU.DAT
```

This addition redisplays the menu after the RunTime is exited.

NOTE: RTI can also be used in the above EXEC file.

The same changes would be made to all EXEC files used.

In addition, the system can be configured so that a menu is automatically displayed whenever the system is booted. This can be accomplished by first "creating" a menu through a simple NPL program and then specifying that program in the Password File. Upon logging on to the system, that program automatically executes, thus displaying the menu.

For example, assuming that a menu "program", called MENU.OBJ in the 5:0 default user group, has been created, enter the following at the Initial Program Command Line in the Password File:

```
RTI 5:0:MENU  (or)  RTP 5:0:MENU
```

CHAPTER 4

RUNTIME OPERATION

4.1 Overview

This chapter covers the general operation of the NPL RunTime under SuperDOS. 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 SuperDOS.

4.2 RTPSHARE Versus RTISHARE

Included in the RunTime Package are two RunTime programs: One program is Non-interpretive, (RTPSHARE), and the other Interpretive (RTISHARE).

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 2.3 for details on memory requirements.

Included with the RunTime are two programs, RTI and RTP, which are entry programs under SuperDOS for the RTISHARE and RTPSHARE SuperDOS tasks. Since only one of the RunTime programs can be in operation at a given time, either RTPSHARE or RTISHARE must be defined in the SuperDOS configuration file at boot time. Refer to Chapter 2 for details on setting up these tasks.

NOTE: The entry programs RTI and RTP are identical. Thus, if RTPSHARE is loaded at boot time, specifying RTI to invoke the RunTime brings up the Non-interpretive RunTime. Likewise, if RTISHARE is loaded, specifying RTP invokes the Interpretive RunTime.

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 3.5.1 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 SuperDOS Prompt

The general form of starting the RunTime from a SuperDOS 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 operating system-dependent).

4.3.2 Starting from EXEC Files

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

Refer to Section 3.10 for additional information on starting the RunTime from an EXEC 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.

Refer to Section 3.11 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 allows the RunTime program to begin operation in a background task. Whenever the /B option is used, the /T option must also be used to specify a port to which the background partition is assigned.

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

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

where:

xx = a defined SuperDOS port number

For example:

```
RTI /B /T=10
```

starts the RunTime as a background task assigned to port 10.

Refer to Section 8.3 for details on background partitions.

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

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} [/D=nnn]

{RTI} [/D=nnn}
```

where:

```
nnn =  the number of device equivalence table
       entries.
```

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.

- Open file limits are controlled by the SuperDOS configuration file, CONFIG.P. Refer to the SuperDOS Guide to Operations for details.

4.4.3 /G (Graphics Mode)

The /G option is not supported by NPL under SuperDOS.

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} [/H]

{RTI} [/H}
```

where:

H = the number of 4K entries allocated.

For example:

```
RTP /H15
```

allocates a 60K handle table (4K x 15).

NOTE: Values greater than /H15 result in an "Insufficent Memory For Task" message. For most applications, a value of one (1) or two (2) for the handle table is typically sufficient.

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 not supported under SuperDOS.

4.4.6 /L (Leave Overhead Memory)

The /L option can be used to supress the use of an overhead memory area by this user task. By invoking this option, the overflow area (28K for RTI or RTP) is deducted from the user tasks own memory, as opposed to the RTISHARE task, effectively reducing the task's partition by the size of the overflow area. This allows reserving overflow areas so that they could be used by other tasks when a particular application can properly function with less memory.

```
{RTP} [/L]

{RTI} [/L]
```

For example:

```
RTI /L
```

Forces the task to allocate an overflow area using its own memory, instead of using an overflow area from RTISHARE.

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 is not supported under SuperDOS

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 not supported under SuperDOS.

4.4.10 /S (Separate Program Segments)

The /S option performs no operation under SuperDOS.

4.4.11 /T (Terminal/Port Number)

The /T option is used to assign a port number for background partition operation. The /T option must be used in conjunction with the /B option.

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

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

where:

```
     xx = a numeric-constant which indicates which terminal
          number to use (overrides default terminal number
```

For example:

```
RTI /B /T=10
```

starts the RunTime as a background task assigned to port 10.

Refer to Section 8.3 for details on background partitions.

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

4.4.12 /U (UMB Memory)

The /U option is not supported under SuperDOS.

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 NPL RunTime, all unused task memory is considered available to the RunTime. The RunTime then dynamically allocates this memory as needed by the application. The amount of available memory varies from one system to the next.

The following elements affect the maximum size of the shareable RunTime task (RTPSHARE or RTISHARE):

- Revision of SuperDOS being used.
- The number of file buffers, cache buffers initialized in the system's CONFIG.P file.
- The RunTime that is executing, RTPSHARE or RTISHARE.

The following elements affect the size of the user partition generated in each terminal task:

- Size of the terminal task.
- The RunTime options included 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
   8               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 SuperDOS 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 SuperDOS. 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

The SuperDOS operating system provides its own methods for terminating program execution. The interrupt key termination method is one of these. Upon depression of the interrupt key, the system performs a HALT, invoking immediate mode, when operating under the Interpretive RunTime. Under the Non-interpretive RunTime, the 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.

Refer to Section 6.8 of this Supplement and Appendix D of the NPL Programmer's Guide for for a description of the interrupt key sequence for specific NPL supported terminals.

As discussed above, depressing the interrupt key while in the Interpretive RunTime, Immediate Mode is invoked. At this point the interpreter may be exited by entering the $END command. Refer to Chapter 2 of the NPL Programmers Guide for a complete discussion of Immediate Mode capabilities.

4.7.4 Terminating the RunTime Shareable Task

In cases where it is desirable to stop the RunTime shareable task (RTISHARE or RTPSHARE), the special program RTISTOP should be executed. This program properly shuts down the shareable task.

Execution of this program prevents new tasks from starting the RunTime and causes the shareable task to be stopped if no terminal tasks are currently executing the RunTime. If other tasks are using the RunTime, then task numbers are reported and RTISTOP must be run again when the tasks have left the RunTime.

The abort function of the SuperDOS MMI utility should never be used to shut down the RunTime shareable task. If this is done while a user is using the RunTime, it may cause unpredictable results, including a general system crash.

CHAPTER 5

DEVICE SUPPORT

5.1 Overview

This chapter discusses the devices supported under the SuperDOS implementation of NPL. Included in this discussion are any special requirements or implications for the programming of NPL code under the SuperDOS operating environment.

Section 5.2 discusses supported diskette devices.

Section 5.3 discusses 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 a Bluebird approved IBM-compatible PC as the console terminal under SuperDOS are discussed in Chapter 6.

5.2 Diskette Devices

NPL under SuperDOS 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.

NOTE: $FORMAT DISK is not supported in any format. Attempts to execute $FORMAT DISK result in an I93 error. As a result, diskettes must be preformatted before use with NPL under SuperDOS.

Support of raw diskette access under Protected Mode SuperDOS is limited to SuperDOS Revision 5.0 or greater.

The 320K raw format is not supported under NPL for SuperDOS.

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

The 2.88MB raw format is not supported under NPL for SuperDOS.

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 displays the naming conventions and revision of the RunTime required for the diskette devices supported under the Release IV SuperDOS implementation of NPL.

Media Size	Type Format	Naming Convention
5-1/4" inch media	360K	1: 360 = Y
	1.2MB	1: 1.2 = Y
3-1/2" inch media	720K	1: 720 = Y
	1.44MB	1: 1.4 = Y

5.2.2 Supported Media

The following section discusses the various NPL supported media.

5-1/4" Media

320K Media

320K raw diskettes are not supported under the Release IV version of NPL.

360K Media

The $DEVICE clause "1: 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)="1: 360=Y"  :REM 360K raw media on drive A
```

1.2MB Media

The $DEVICE clause "1: 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)="1: 1.2=Y"  :REM 1.2MB raw media on drive A
```

3-1/2" media

720K Media

The $DEVICE clause "1: 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)="1: 720=Y"  :REM 720K raw media on drive A
```

1.44MB Media

The $DEVICE clause "1: 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)="1: 1.4=Y"  :REM 1.44MB raw media on drive A
```

2.88 Media

2.88 raw diskettes are not supported under NPL for SuperDOS.

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 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
   : $DEVICE(/D10)="1: 360=Y"          :REM try 360K format
   : S=1440                            :REM 360K media size
   : DATALOAD BAT/D10,(0)A$
   : ERROR $DEVICE(/D10)="1: 1.2=Y"    :REM 360K failed, try 1.2MB
   : S=4800                            :REM 1.2MB media size
   : DATALOAD BAT/D10,(0)A$
   : ERROR $DEVICE(/D10)="1: 720=Y"    :REM 1.2MB failed, try 720K
   : S=2880                            :REM 720K media size
   : DATALOAD BAT/D10,(0)A$
   : ERROR $DEVICE(/D10)="1: 1.4=Y"    :REM 720K failed, try 1.4MB
   : S=5760                            :REM 1.4MB 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., 1:), if access to different types is required. Attempting to define, for example, /D10 as "1:" and /D20 as "1: 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, and 1.44MB 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 performs a read of the 512-byte sector, 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 (360K= 1440 sectors, 1.2MB= 4800 sectors, 720K= 2880, 1.44MB= 5760) which may be used (after subtracting 1 sector) as the END= parameter to a subsequent SCRATCH DISK statement.

NOTE: A value of zero is placed in these bytes by previous SuperDOS implementations of the RunTime.

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 (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 (when supported by the operating system). 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.

NOTE: 320K raw format is not supported by NPL under SuperDOS. The above information, on performance, is provided only as a reference only for developers who use 320K raw access on other platforms.

Media Defects

All types of raw diskettes used by NPL must be defect-free.

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.

Data Integrity

A SuperDOS diskette cannot be read from or written to NPL (unless the diskette was formatted for SuperDOS and access was to a named SuperDOS diskimage file on the diskette), and SuperDOS applications cannot read or write 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 SuperDOS diskettes, which could result in corrupting the data on the diskette. Similarly, it is possible that SuperDOS applications could corrupt the NPL diskettes if direct access to the media is made.

Compatibility

5-1/4" Media

320K

320K raw format diskettes are not supported under the Release IV version of NPL for SuperDOS. Refer to Chapter 10 for information on porting data to other platforms (i.e., MS-DOS) if it is necessary to use 320K raw format diskettes to transfer data to/from a SuperDOS system.

360K

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

Compatibility with Wang 2275 and DS 360K drives is also supported.

1.2MB

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

3-1/2" media

720K

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

1.44MB

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

2.88MB

2.88MB raw diskette access is not supported under NPL for SuperDOS.

NOTE: The drive/media compatibility issue described earlier in this chapter applies to 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 SuperDOS filename in any valid SuperDOS drive/user group designation. Any type of physical disk media supported by the Super-DOS operating system on a Bluebird approved 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 5.3.7).

All features of diskimage files described in Chapter 7 of the NPL Programmer's Guide, are fully supported on the SuperDOS implementation of NPL. Refer to Section 5.3.5 of this Supplement and Section 7.3 of the NPL Programmer's Guide for more information pertaining to performance issues of NPL diskimages.

5.3.1 Naming Conventions

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

```
$DEVICE(/XXX)="[drive:group:]diskimage" [clause]
```

where:

XXX	Any valid NPL disk device address.
drive	Any valid SuperDOS drive designation. If no drive:group is specified, the search path is used to locate the specified file.
diskimage	The name of the diskimage file.
clause	Optional clause which modifies access to the diskimage depending on the clause in use.

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

NOTE: SuperDOS is case-insensitive. Therefore, 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 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 SuperDOS systems using the SuperDOS disk cache option. Refer to the SuperDOS Guide for Operations for details on configuration and use of this option.

WARNING -- 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.

Use of disk caching for write operations with a hard disk can potentially cause problems. This is do 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 actually 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 system.

A second problem with write caching is that errors that occur during the physical write to the disk cannot be detected by the program.

5.3.3 File Allocation Considerations

NPL diskimage files use the SuperDOS "Extendable" file format. This means that the diskimage file size can be changed (either increased or decreased) without having to recreate the file.

Diskimage files are SuperDOS text type "T" files. A SuperDOS header sector is present for each diskimage file. Therefore, NPL sector 0 corresponds to SuperDOS sector 1 of the file. The physical size of a diskimage file is always 1 (512 byte) sector greater than otherwise expected.

All SuperDOS files require that certain parameters be defined when a file is created. These parameters include:

- Primary Extent Size (PES). This identifies how much contiguous disk space is required to initially create the file.
- Secondary Extent Size (SES). This identifies how much contiguous disk space is required to build secondary extents, should the file exceed the primary extent size.

NPL Diskimage File Creation

NPL diskimage files are created using the NPL SCRATCH DISK command. NPL must set the SuperDOS PES and SES values at time of diskimage creation.

This is done by one of two methods:

- The built in defaults of the RunTime.
- The $DEVICE clauses PES and SES.

NOTE: The SCRATCH DISK statement does not create the diskimage file if the file previously exists and can be extended to the required size. In this case, the PES and SES for the original file remain in effect.

Built-In Defaults

NPL attempts to use the END parameter specified in the SCRATCH DISK statement as the default PES parameter for the diskimage. If the required space is unavailable in a single contiguous space, NPL uses half the size of the diskimage as the default PES. If this also fails due to lack of contiguous space, NPL returns an error.

The RunTime uses the following computation as the default SES parameter:

```
The value of 1/66th of the total size of the file, or 10 (512-byte)
sectors (5K), whichever is larger (refer below for an example calcula-
tion).
```

The default NPL parameters can be overridden by use of the PES and SES clauses on the $DEVICE statement. Refer to the discussion of PES and SES clauses in Section 5.3.4.

NPL Diskimage File Expansion

The size of a diskimage file can be extended (increased) by use of the MOVE END statement. The amount of disk space required is allocated using a series of secondary extents, each extent containing the number of sectors defined by the SES at the time of file creation. If the RunTime is unable to allocate the necessary number of secondary extents for file expansion, NPL returns a P48, Illegal Device Specification , error.

SuperDOS file allocation places certain limitations on the size of a file imposed at the time the diskimage file is created. The maximum size of a diskimage file under SuperDOS is computed by the formula:

```
Primary extent size (PES) + (100* secondary extent size (SES))
```

This means that a diskimage file which was originally created with an END value of 10,000 in the SCRATCH DISK statement, using built in defaults for PES and SES sizes, has a maximum END value in the MOVE END statement of:

```
10,000 + (100 * (1/66 *10000)) = 25,200
```

NOTE: Odd numbered PES and SES values are intrinsically rounded up to the next even number. NPL uses the full sizes of the sectors allocated (i.e., it does not restrict itself to the half-sectors if an extent size is defined with an odd number.

In most cases, 2.5 times the original size as an approximate number. Smaller diskimage files have a larger expansion capability since they can always expand up to 1000 (512 byte) sectors. The above example uses the default values of the RunTime for determining the SES. Refer to section 5.3.4 below for a discussion on the PES and SES clauses.

When MOVE END is used to DECREASE the size of a diskimage file, only full secondary extents are released. If the file is reduced into the primary extent, the primary extent size is reduced. If the file subsequently grows, the primary extent size is not increased.

5.3.4 The PES and SES Clauses

The PES= and SES= clauses of the $DEVICE statement, are used to specify the primary and secondary extent sizes (PES and SES respectively) in units of 256-byte blocks (to be used when a file is created). Under SuperDOS, the numeric-value specified by the PES and SES clauses overrides the built-in default logic for determining the extent size to use at the time of creation.

NOTE: The extent sizes (both primary and secondary) for a file can only be set at the time of file creation. If the file already exists, existing extent sizes are used and the PES or SES clause specified is not used.

For example:

```
$DEVICE(/D11)="PLATTER1.BS2 PES=10000 SES=1000"
SCRATCH DISK T/D11,LS=10,END=10000
```

The primary extent size for PLATTER1.BS2 is 10000 256-byte blocks (or 2,560,000 bytes) and the secondary extent size is 1000 256-byte blocks (or 256,000 bytes).

However, when using the built-in default logic for SuperDOS file creation as shown below in the example:

```
$DEVICE(/D11)="PLATTER1.BS2"
SCRATCH DISK T/D11,LS=10,END=10000
```

The primary extent size for PLATTER1.BS2 is 10000 256-byte blocks (or 2,560,000 bytes), but the secondary extent size would be (1/66 * 10000) or 151 256-byte blocks (38,656 bytes).

NOTE: The primary extent size of the diskimage is the lesser of the PES value or the END value specified on the SCRATCH DISK statement.

Use of these clauses helps reduce the fragmentation of the diskimage by reserving larger contiguous extents for future file expansion. this reduces the total number of extents necessary. Refer to Section 5.3.5 below for a discussion of performance issues.

In the event that the specified primary or secondary extent size cannot be allocated, NPL automatically attempts to use its built-in default logic to determine the primary or secondary extent size. If the built in default logic cannot be used, NPL returns a,P48 error code, Illegal Device Specification.

5.3.5 Performance Issues

Access to diskimage files stored in a single primary extent without using any secondary extents is much faster than access to a diskimage file which is using secondary extents. Every effort should be made to create and maintain diskimage files in a single primary extent. Therefore, when creating a diskimage file that will later be expanded, it is better to specify a larger END value in the SCRATCH DISK statement than to use MOVE END later.

The SuperDOS "DIR" utility, specified with the "/L" switch, allows viewing of the file and the ability to determine if secondary extents are in use. If they are in use, then one of the following methods can be used to create the file with one primary extent.

If there is enough contiguous disk space to create a second file, (the SuperDOS FREE utility can be used to determine this) the following method should be used:

- Create another diskimage file of the same size using the SCRATCH DISK command. If further expansion is expected, then create the file at the anticipated size required.
- Using the NPL utility 2CCOPY, copy contents of the original diskimage into the new diskimage.
- Delete the original diskimage, either by using the NPL $FORMAT DISK statement or by using the SuperDOS DEL utility.
- Rename the new diskimage file to the old diskimage file name using the SuperDOS F.RENAME utility.

On systems where there is not enough contiguous disk space to create a second file, the following method should be used:

- Backup the diskimage either to a series of diskettes using the NPL 2cbckp utility, 2CBCKP or to tape using the SuperDOS tape backup utility.
- Delete the original diskimage, either by using the NPL $FORMAT DISK statement or by using the SuperDOS DEL utility.
- Using the SuperDOS CRUNCH utility, free up contiguous disk space on the hard drive.

NOTE : A complete backup of the entire hard disk should be made prior to using CRUNCH.

- Recreate the diskimage file, which was deleted in step 2, for the same size using the SCRATCH DISK command. If further expansion is expected, create the file at the anticipated size required.

- Restore the diskimage from the series of diskettes created in step 1, using the NPL 2CRCVR utility, or from tape using the SuperDOS tape restore utility. (use switches in restore to ensure the file is restored to the new location).

WARNING -- Back up all files before deleting any diskimages.

HINT: Although it is highly recommended that files with secondary extents be reduced to a single primary extent, there are some situations where this is not possible. In these cases, make sure that an adequate number of "extent buffers" have been specified in the CONFIG.P file. An extent buffer maintains the diskimage file header record (which keeps track of the file sectors that reside in an extended area) in memory while the file is opened. This means that when access to a portion of the diskimage that resides in an extent is requested, the header record is read from memory instead of disk, offering a substantial performance increase.

Refer to Chapter 13 of the NPL Programmer's Guide for details on the operation of: 2CCOPY, 2CBCKP and 2CRCVR. Refer to the NPL Statements Guide, SCRATCH DISK, MOVE END and $FORMAT DISK, for proper syntax and usage of these commands. Refer to the SuperDOS Guide to Operations for details on setting extent buffers, and use of the DEL, F.RENAME, FREE and DIR utilities.

5.3.6 Implicit $BREAK Implications

The SuperDOS time slicing algorithm does not automatically release the remainder of the time slice for a task which completes a disk I/O operation. Therefore, to provide greater overall system efficiency, NPL a $BREAK following the completion of every disk I/O operation. This is controlled by byte 14 of $OPTIONS. The default value for SuperDOS is HEX(01), meaning that an implicit $BREAK is performed. It is recommended that this not be modified. Refer to Section 7.5.2 for details on $BREAK.

5.3.7 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.

SuperDOS incorporates buffering techniques for disk I/O which could adversely affect operations of NPL programs which use removable media. SuperDOS 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 SuperDOS 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 SuperDOS 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 SuperDOS file:

```
$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.

Use of the Niakwa Utilities with Diskimage Files on Removable Media

The 2CBCKP, 2CRCVR and 2CCOPY utilities provided with the NPL Development Software support the use of diskimage files on diskettes.

When using 2CBCKP, 2CRCVR or 2CCOPY with diskimage files on removable media, it is necessary to create the diskimage file with SCRATCH DISK in advance. In addition, when specifying the size of the output diskette in 2CBCKP and 2COPY, use the size originally specified when the file was created. Use of a larger file size results in the file being extended to secondary extents, thus decreasing performance significantly.

5.3.8 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 SuperDOS CONFIG.P file is necessary to set up a RAM disk under SuperDOS. Refer to the SuperDOS documentation for details of setting up a RAM disk.

HINT: RAM disks can significantly increase the performance of NPL-based applications

5.4 Supported Monitors/Controllers

NPL for SuperDOS supports monochrome and color controllers and monitors for the console terminal. Chapter 6 discusses the use of these controllers and monitors in detail.

5.4.1 File Naming Conventions

Under the SuperDOS implementation of NPL, the terminal type for a Bluebird approved IBM-compatible PC monitor on the console terminal is determined automatically by the RunTime.

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

Monitor Type	Value of byte 3 of $MACHINE
ASCII	C
Monochrome	M
Color	C

The terminal type is stored in byte 9 of $MACHINE. The following are valid monitor types:

Terminal Type	Value of byte 9 of $MACHINE
HEX(00)	Wang PC
HEX(01)	Wang 2210
HEX(02)	Altos III
HEX(03)	VT100/VT200
HEX(04)	IBM PC (using /R)
HEX(05)	Wyse 60,150,160
HEX(06)	Wyse 50
HEX(07)	Wang 2236
HEX(08)	Altos V
HEX(0B)	Wang 370

5.4.2 Using Emulation Products

Many emualtion products exist that allow remote control of the RunTime under Super-DOS. These products can be usd to allow a developer to check the operation of a Run-Time application at a remote location.

NOTE: Niakwa does not offically support the use of any emulation product. These products can have an effect on the keyboard and screen translation 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 EDKEYBOA may be needed, but this can effect 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 of the NPL Programmer's Guide on the NPL Utilities.

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

Byte 31 of $OPTIONS

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

5.4.3 NPL Plotter Drivers

Niakwa's Plotter Driver software contained with the Niakwa Scientific and Communications Drivers Packages is not available under SuperDOS.

5.5 Printers

NPL under SuperDOS provides support for print output to either parallel ports, serial ports, local printers attached to supported terminals or SuperDOS text files (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 the responsibility of the programmer. Section 5.5.5 discusses several special considerations when configuring new printers.

5.5.1 Naming Conventions

The following section discusses NPL naming conventions for printers under SuperDOS.

Parallel Printer Ports

For printing to a parallel printer port, the special device name "> Px" should be used in the $DEVICE statement.

For example:

```
$DEVICE (/215)=">P1"
```

directs all print output sent to the NPL print address /215 to the parallel printer where x is the port number (i.e., P1,P2).

Serial Printer on Non-Allocated Serial Ports

For printing toa non-system serial printers, the special device name "> n" should be used in the $DEVICE statement.

For example:

```
$DEVICE (/216)=">5"
```

directs all print output sent to the NPL print address /216 to the serial printer, where n is the SuperDOS port number (i.e., 5).

Local Printers

Local printers can be accessed under NPL by using a special designation of "> 0 LCL= Y".

For example:

```
$DEVICE (/204)=">0 LCL=Y"
```

directs all printer output sent to the NPL print address /204 to the local printer port. Both parallel and serial ports are supported, depending on the terminal.

NOTE: Local printers can be accessed only by the terminal to which they are attached.

SuperDOS Text File

For directing print output to a SuperDOS text file, the device equivalence should be set to the name of the SuperDOS text file. Drive designation and user group may be included.

For example:

```
$DEVICE (/217)="5:4:SPOOL.DAT"
```

directs all print output sent to the NPL print address /217 to the file SPOOL.DAT on hard drive 5 in user group 4.

Directing output to a SuperDOS text file can be useful in two ways.

- It can be used for generating ASCII files which can be used as input to other SuperDOS applications.
- It can 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 SuperDOS text files. Refer to $DEVICE in the NPL Statements Guide for details on the functionality of ERR= Y.

5.5.2 PES and SES Considerations

The file allocation considerations which apply to diskimage files also apply to SuperDOS text files (refer to Section 5.3.3). The use of the PES and SES clauses for specifying primary and secondary extents are applicable for creation of print class devices (ASCII text files). If these clauses are not implemented when creating an ASCII text file using a print class device, the maximum size of the file is .5MB.

For example:

```
$DEVICE(/204)="SPOOL.DAT"
```

The PES for the SPOOL.DAT text file is ten (10) 512 byte blocks (the 5K SuperDOS default) and the SES is ten (10) 512 byte blocks (the 5K SuperDOS default). Based on the PES and SES values for this file, the maximum size it could obtain can be computed using the formula discussed in Section 5.3.3, expanding, as follows:

```
10 + (100 * 10) = 1010 512K byte blocks (or .5 MB).
```

NOTE: The above value should be adequate in most cases, however if there is a need to have a larger text file created, then the PES and SES clauses should be used.

For example:

```
$DEVICE(/204)="SPOOL.DAT PES=1000 SES=1000"
```

The PES for SPOOL.DAT is be 1000 256 byte blocks (or 500 512k bytes), and the SES would be 1000 256 byte blocks (or 500 512k bytes).

Therefore, based on the PES and SES values for this file, the maximum file size can be computed using the formula discussed in Section 5.3.3, as follows:

```
500 + (100 * 500) = 50500 512K byte blocks (or 25.25 MB).
```

The PES and SES clauses only apply when the file is initially created. Therefore, if ASCII files already exist at a smaller size, they must deleted and recreated with the PES and SES clauses specified.

NOTE: If the amount of print output exceeds the size of the ASCII text file to which it is directed, the additional output is truncated without any warning message. This situation can be avoided by use of the ERR= Y $DEVICE clause. Refer to the NPL Statements Guide, $DEVICE and Chapter 7 of the NPL Programmer's Guide for details.

5.5.3 Printer Not Ready Condition

Attempting to access a non-existent or de-selected parallel printer results in the terminal hanging until the printer is ready. Attempting to access a non-existent or de-selected serial printer results in output being discarded. In either case, a "Printer Not Ready condition" cannot be detected in advance.

5.5.4 Special Considerations

SuperDOS 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, NPL 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 SuperDOS print functions (double spacing results). However, NPL does provide a mechanism for suppressing the insertion of the line feed with every carriage return so these printers can be used with NPL applications.

The ALF (Auto Line Feed) Option

NPL 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)=">P1 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 concerning the use of the ALF $DEVICE clause.

5.6 Mouse Support

A mouse device is not supported under the SuperDOS implementation of NPL.

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 SuperDOS for NPL.

5.7.1 Naming Conventions

Serial ports under SuperDOS can be accessed through the special device designation "> x" in the $DEVICE statement, where x is the physical port number.

For example:

```
$DEVICE(/211)=">5"
```

or

```
$DEVICE(/211)=">5 TMO=Y"
```

sends subsequent I/O operations directed to address 211 to the SuperDOS port number 5. 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 under NPL.

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. Refer to Section 7.9 of the NPL Programmer's Guide for more details on accessing serial ports under NPL.

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 SuperDOS implementation of NPL 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 or the Niakwa Scientific and Communications package are not supported under NPL for SuperDOS.

The C620 $GIO microcommand 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)=">10 TMO=Y"    :REM read from port 10
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 SuperDOS "SYSGEN" utility can be used to define the baud rate, number of data bits, stop bits, and parity for serial ports. These parameters are then established when SuperDOS is loaded and may not not be altered dynamically. Consult the SuperDOS Utilities Guide for more information regarding the "SYSGEN" command.

- Any port used for serial input must be configured as an unassigned port (i.e., no task assigned to the port).

- The input buffer for serial ports under SuperDOS is 78 characters. 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).

- 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.

5.7.4 NPL Communications Drivers

The Niakwa's 2227 communication driver software is not supported under SuperDOS

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 SuperDOS 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 functions if available.

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

Applications which 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 the 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 SuperDOS, the Device Equivalence Table contains the following default entries:

2200 Address	SuperDOS Equivalent
/B10 or /D10	1:
/310 or /D11	PLATTER1.BS2 in the current directory
/D12	PLATTER2.BS2 in the current directory
/215	> P1
/204	> P1

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

This chapter discusses the terminals supported on the Bluebird approved IBM-compatible PCs under the SuperDOS operating system. This chapter also discusses the color support available on the SuperDOS console monitor and how the NPL screen handling features are affected by the various controllers supported by NPL.

NOTE: NPL color support is specific to the IBM Color Graphics (CGA) controller. Refer to Section 7.4 of the NPL Programmer's Guide for a complete discussion of color implementation within NPL applications.

Section 6.2 discusses operating system considerations.

Section 6.3 summarizes supported terminals and controllers.

Section 6.4 discusses the console terminal characteristics using a monochrome controller.

Section 6.5 discusses the console terminal characteristics using the color graphics controller.

Section 6.6 discusses color support under NPL.

Section 6.7 discusses graphics support with NPL under SuperDOS.

Section 6.8 discusses keyboard characteristics.

6.2 Operating System Considerations

The following section discusses the characteristics of SuperDOS 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. The HALT key function, local printer support and terminal configuration requirements are also discussed.

6.2.1 Determination of Terminal Type

On the SuperDOS implementation of NPL, the terminal type is determined by a file called TERMTYPE.TBL. The information in the TERMTYPE.TBL file contains the terminal type for any physical port or for any "terminal kind".

For example:

```
2 WY60          defines port 2 as a Wyse 60 terminal
3 VT100         defines port 3 as a VT100 terminal
4 WY60          defines port4 as a Wyse 60 terminal
4* W2236        defines terminal "kind" 4 as a Wang 2236
* WY50          defines all other terminals are Wyse 50
```

The "terminal kind" is established by SuperDOS configuration files and is used by SuperDOS in conjunction with the TERMDATA file for accessing terminals from SuperDOS utilities.

NOTE: The SuperDOS TERMDATA file is not used by NPL, only the "terminal kind".;

If the TERMTYPE.TBL file contains no entry which applies to a particular port as terminal kind, NPL assumes a WY50 (Wyse 50) as a default.

The TERMTYPE.TBL file is required to allow the RunTime to load the correct SCREEN.xxx and KEYBOARD.xxx files. Refer to Section 2.6.7 for details on creating and modifying the TERMTYPE.TBL file.

6.2.2 Location Of Terminal Files

NPL searches for the screen translation table file (SCREEN.xxx) and keyboard translation table file (KEYBOARD.xxx) in the order specified in the user's searchlist. For example, if a user has a searchlist of 5:0, 5:1, and 5:2, the RunTime searches for the screen translation table file in the 5:0 user group first, the 5:1 user group second, and the 5:2 user group third. The same search path is used to find the font files (XXFONT.XXX) and KEYS files (VTKEYS.VT2) when it is necessary to use these files. Refer to Section 6.2.3 for details on downloading FONT and KEYS files.

NOTE: It is recommended that the terminal files supplied with the NPL Development Software be located within the 5:0 user group and that 5:0 should be indicated in each user's searchlist.

Refer to Section 6.3 for a list of the SCREEN and KEYBOARD files associated with each supported terminal. Refer to the NPL Statements Guide, $KEYBOARD, $SCREEN for details on using the $KEYBOARD and $SCREEN system variables. Refer to Chapter 13 of the NPL Programmer's Guide for details on the use of the NPL utilities, used to modify the SCREEN and KEYBOARD tables, and the FONT files.

6.2.3 Downloading FONT and KEYS Files

Several of the terminals supported under the NPL implementation of SuperDOS allow for the use of "downloadable" fonts. In addition, the VT220 series and the Wyse 370 supports the use of a "downloadable" KEYS file. It is necessary to use the appropriate FONT and KEYS files for terminals that support downloadable files.

NOTE: For the SCREEN.XXXX and KEYBOARD.XXXX files to work correctly the appropriate FONT and KEYS files must be downloaded.

The appropriate files can be downloaded to the terminal being used, by use of the SuperDOS TYPE command.

For example, to download the Wyse 60 font file, enter the following from the SuperDOS command prompt at a Wyse 60 terminal:

```
TYPE 5:0:WYFONT.WY6
```

Refer to Section 6.3 for a list of the FONT and KEYS files associated with each NPL supported terminal.

6.2.4 HALT Key Functionality

For the SuperDOS implementation of NPL, the HALT key function is active at all times unless disabled using $OPTIONS byte 13. The HALT key sequence for all supported terminals under SuperDOS is (CTRL-C).

NOTE: The Wang 2X36 terminals use the standard HALT key on the keyboard to generate the proper interrupt.

6.2.5 Local Printer Support

Local printers may be accessed under NPL by using a special device designation "> 0 LCL=Y".

For example:

```
$DEVICE(/204)=">0 LCL=Y"
```

directs all print output sent to the NPL print address /204 to the local printer port on the terminal. Both serial and parallel local printers are supported. Refer to Appendix D in the NPL Programmer's Guide for specific features of each NPL supported terminal.

NOTE: Local printers are accessed only by the terminal to which they are attached.

6.2.6 Terminal Configuration Requirements

Under SuperDOS, supported terminals must be set for 8 data bits and XON/XOFF. The baud rate, stop bits and parity must match the SuperDOS configuration for the selected serial port. Refer to Appendix D in the NPL Programmer's Guide for the suggested configuration for each supported terminal.

NOTE: SuperDOS serial ports will support communication rates up to 38400 baud. However, baud rates higher than 9600 may have restricted cable length requirements.

6.2.7 Wang 2X36 Terminal Configuration

SuperDOS supports the use of the Wang 2X36 terminal. This terminal requires the execution of a program named W2236INI to set up proper XON/XOFF codes for use with the Wang 2236 terminal. If this program is not used, flow control problems may occur.

General Form for Executing W2236INI:

```
W2236INI /x [command]
```

where x is a number that refers to the specific terminal kind, and command is any valid SuperDOS command.

For example:

```
W2236INI /4 RTI UTILITY
```

sets XON/XOFF for the port in use to Wang mode if it is defined as terminal kind 4, or to standard mode if defined as any other terminal kind. In addition, it executes RTI with a boot program name of UTILITY.

This program can also be executed from an automatic password. Refer to Section 2.6.7 for details.

NOTE: Previously installed Wang terminal cables can be used. However, a gender swapper is necessary at the system end of the cable. Special cables for use with 2236 terminals are available from Bluebird Systems.

6.3 Supported Terminals

The following section discusses the terminals supported under NPL for SuperDOS.

6.3.1 NPL Supported Terminals

The following table lists the terminals that NPL supports under SuperDOS. This table also contains the "terminal kind" name which should be used within the TERMTYPE.TBL file and the names of the auxiliary files supplied for each terminal. Refer to Appendix D of the NPL Programmer's Guide for a complete description of the supported terminal characteristics. Refer to Section 2.6.8 for details on the TERMTYPE.TBL file and for a detailed listing of all auxiliary files supplied with the NPL Development Package.

Terminal Name	"Terminal kind"	Keyboard File Name	Screen File Name	Font File Name	KEYS File Name
Altos III	ALTOS3	KEYBOARD.ALT	SCREEN.ALTOS	N/A	N/A
Altos V	AL5	KEYBOARD.AL5	SCREEN.AL5	VTFONT.AL5	N/A
DEC VT100	VT100	KEYBOARD.VT1	SCREEN.VT100	N/A	N/A
DEC VT200 Series	VT220	KEYBOARD.VT2	SCREEN.VT220	VTFONT.VT2	VTKEYS.VT2
IBM Console Monitor	N/A	KEYBOARD.TBL	SCREEN.TBL	N/A	N/A
Wang 2110A	W2110	KEYBOARD.W21	SCREEN.W2110	N/A	N/A
Wang 2236 DE/DW	W2236	KEYBOARD.W22	SCREEN.W2236	N/A	N/A
Wyse 50	WY50	KEYBOARD.WY5	SCREEN.WY50	N/A	N/A
Wyse 60,150, 160	WY60	KEYBOARD.WY6 KEYBOARD.WPC	SCREEN.WY60	WYFONT.WY6	N/A
Wyse 370	WY370	KEYBOARD.WY3	SCREEN.WY370	W370FONT.80 WY370FONT.132	WYKEYS.WY3

NOTE: The Wyse 60 terminal is recommended for use with NPL applications under SuperDOS. Refer to Appendix D of the NPL Programmer's Guide for details.

The Wyse 60, 150, and 160 support several different kinds of keyboards. Niakwa provides support for the ASCII and PC styles only. The file KEYBOARD.WY6 contains the default keyboard translation for the ASCII style keyboard. KEYBOARD.WPC contains the default translation for the PC style keyboard.

The PC style keyboard, KEYBOARD.WPC, if used on the Wyse 60, 150, or 160 terminals, must be renamed to KEYBOARD.WY6 before entering the RunTime.

NPL contains built-in defaults for keyboard and screen translation for the SuperDOS console. Therefore, KEYBOARD.TBL and SCREEN.TBL are not provided on the Terminal Support diskette included in the NPL Development Package.

The following chart shows the features supported by the above terminals.

Terminal Features Notes

On terminals that support a single attribute, this attribute will be used for any Niakwa Programming Language attribute or combination of attributes.

Bright, Blink, Reverse and Underline attributes are supported in any combination. Certain restrictions may apply.

For terminals that support downloadable fonts, Niakwa provides a font file that provides complete emulation of the standard Niakwa Programming Language character set. These font files may be modified by the developer. On terminals without downloadable fonts, the full Niakwa Programming Language character set cannot be emulated. Typically, pixel graphics (HEX(CO) through HEX(FF)) and special characters may not be available.

The above terminal features chart is intended as an aid in determining the capabilities of a particular terminal within the Niakwa Programming Language. Please note that limitations may exist that are unique from one terminal to the next. Developers should refer to the appropriate terminal's documentation for details on any restrictions that may apply.

6.3.2 Use with Native Operating System Functions and Utilities

The terminals listed in Section 6.3.1, operate properly while executing the RunTime. However, some of those terminals do not support (and are not well suited) for use with SuperDOS native operating system functions and utilities. Below is a summary of the terminals which are well suited for native operating system use, and those that are not well suited.

Terminals that are supported and well suited for use with the SuperDOS operating system functions and include:

Console
Wyse 50
Wyse 60
Wyse 150
Wyse 160

Terminals that are not supported or well suited for use with the SuperDOS operating system functions and utilities:

Altos III
Altos V
DEC VT100
DEC VT200 series
Wang 2110A
Wang 2X36 DE/DW
Wyse 370

6.4 Monochrome Terminal Characteristics

The following section discusses the console terminal characteristics using a monochrome controller.

6.4.1 Graphics Availability

The /G (graphics) RunTime startup option is not supported under SuperDOS.

6.4.2 Screen Character Set

Downloadable Fonts

The console terminal does not support the use of downloadable fonts.

Alternate Character Set (Pixel Graphics)

Use of the alternate character set is not supported by the console terminal.

Non-English Character Sets

Use of non-English character sets through the $OPTIONS Font Designator is not supported on the console terminal.

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

The console terminal 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

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

- Inverse video does not work in conjunction with underline.
- Underlines are only avaible when these attributes are combined.

6.4.5 Cursor Appearance

The direct video IBM screen does not have the ability to display a "steady" cursor. A steady cursor appears as a thicker, blinking cursor.

6.4.6 Support for 132-Column Mode

The console terminal does not support 132-column mode output.

6.4.7 Keyboard Characteristics

The console should be configured to use SuperDOS keyboard drivers, not MS-DOS drivers (refer to the SuperDOS Guide to Operations, Line 103 of the system generation file). The SuperDOS keyboard drivers remap the console keyboard to Wyse 50 equivalent keys. This means that keyboard equivalences arel not the same as the standard IBM-compatible PC values. Refer to Section 6.8 for default keyboard equivalences table.

6.4.8 Use With Native Operating System Functions and Utilities

The console terminal is supported (and well suited) for use with SuperDOS native operating system functions and utilities.

6.5 Color Console Terminal Characteristics

The following sections discuss the console terminal characteristics for the Color Graphics Controller

6.5.1 Graphics Availability

The /G (graphics) RunTime startup option is not supported under SuperDOS.

6.5.2 Screen Character Set

Downloadable Fonts

The console terminal does not support the use of downloadable fonts.

Alternate Character Set (Pixel Graphics)

Use of the alternate character set is not supported by the console terminal.

Non-English Character Sets

Use of non-English character sets through the $OPTIONS Font Designator is not supported on the console terminal.

Screen Translation

Screen character translation is performed using an internal lookup table. NPL 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).

6.5.3 Box Graphics

The console terminal 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.

6.5.4 Attributes Support

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

- 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 Section 6.6 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 NPL 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 NPL Programmer's Guide for details on $OPTIONS bytes 22 and 23.

6.5.5 Cursor Handling

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

6.5.6 Color Support

Programmable specification of foreground, background, underline and perimeter colors is supported. Refer to Section 6.6 for details.

6.5.7 Support For 132-Column Mode

The console terminal does not support 132-column mode output.

6.5.8 Keyboard Characteristics

The console should be configured to use SuperDOS keyboard drivers, not MS-DOS drivers (refer to the SuperDOS Guide to Operations, Line 103 of the system generation file). The SuperDOS keyboard drivers remap the console keyboard to Wyse 50 equivalent keys. This means that keyboard equivalences are not the same as to the standard IBM-compatible PC values. Keyboard equivalencies for the console terminal when using a CGA controller are identical to those when a monochrome controller is in use. Refer to Section 6.8 for details on keyboard characteristics of the console terminal.

6.5.9 Use With Native Operating System Functions and Utilities

The console terminal is supported (and well suited) for use with SuperDOS native operating system functions and utilities.

6.6 Color Support under NPL

Under the SuperDOS implementation 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 Section 7.4.7 of the NPL Programmer's Guide.

6.6.1 Supported Controllers and Monitors

Under SuperDOS, NPL supports the use of color on the SuperDOS console when the console is an IBM Color display using an IBM Color Graphics Adapter (CGA) controller.

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 provide 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.

6.6.2 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.

6.7 Graphics Support under NPL

The /G (graphics mode) is not supported under the SuperDOS implementation of NPL. "True" box graphics is only supported on the Wang 2236 terminal. Refer to Appendix D, of the NPL Programmer's Guide for details.

6.8 Keyboard Characteristics

The standard IBM-compatible PC (console terminal) 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 should 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 SuperDOS implementation of NPL, keyboard characteristics, and the effects of the CAPS LOCK and NUM LOCK keys.

6.8.1 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 console terminal keyboards supported under the SuperDOS implementation of NPL.

CONTINUE

There is no CONTINUE key or equivalent on the keyboards supported on the PC.

6.8.2 Keyboard Characteristics

The console should be configured to use SuperDOS keyboard drivers, no MS-DOS drivers (refer to the SuperDOS Guide to Operations, Line 103 of the system generation file). The SuperDOS keyboard drivers remap the console keyboard to Wyse 50 equivalent keys, this means that keyboard equivalences are the same as the standard PC values. The default values for the keyboard used with the SuperDOS implementation of NPL are built-in to the RunTime programs. If modifications are made to these defaults, they are saved to disk in a file called KEYBOARD.TBL.

Default equivalences for commonly used keys are:

HALT	CTRL-C (not modifiable)
EXECUTE	HOME
CANCEL	END
HELP	ESC,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.)
INS/LINE	Inserts a linefeed within a line of text.
DEL/LINE	Deletes characters from the current cursor position to the end of line.
INSERT	Inserts a blank space within a line of text or toggle between Insert or Overstrike mode. This behavior depends on the value of byte 44 of $OPTIONS.
DELETE	Deletes a character at the current cursor position.
PREV/PAGE	Causes the screen display to shift to the previous screen (if more than 23 lines of text exist) or cause cursor movement to the first line of text.
NEXT/PAGE	Causes the screen display to shift to the next screen (if more than 23 lines of text exist) or cause cursor movement to the last line of text.
CTRL-P	Recalls the previous command for the "multi-command" keyboard buffer. Refer to Chapter 5 of the NPL Programmer's Guide for more information.

CTRL-N	Recalls the next command from the "mulit-command" keyboard buffer. Refer to Chapter 5 of the NPL 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.
ESC,1	Inserts one space at the current cursor position.
ESC,2	Moves cursor 5 spaces to the right.
ESC,3	Moves cursor one space to the right.
ESC,4	Moves cursor one space to the left.
ESC,5	Moves cursor 5 spaces to the left.
ESC,6	Recalls the current line.

NOTE: The numeric values used in combination with the ESC key are from the main keyboard area, not the numeric keypad.

6.8.3 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.

SuperDOS Compatible Default Keyboard Equivalences Tale		
NPL Code Key	**NPL Virtual Key**	**Console Terminal Key**
08	BACKSPACE	BACKSPACE
0D	RETURN	ENTER
5F	UNDERLINE	UNDERLINE
81	CLEAR	?
82	EXEC	7-HOME
84	CONTINUE(LOAD)	?
A1	LOAD	CTRL-X
E5	SHIFT-ERASE	CTRL-W
FF'A0'xx	UNDERLINE(DEAD KEY)	?
'00 ... '09	SF '0 ... '9	F1 ... F10
'0A ... '0F	SF '10 ...'15	ESC,0 ... 5 or *CTRL-(F1...F6)
'10 ... '19	SHIFT SF '0...'9	SHIFT F1 ... F10
'1A ... '1F	SHIFT SF'10...'15	ESC,SHIFT 0 ... 5 or *CTRL/SHIFT-(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-E
'49	DELETE	.-DEL
'4A	INSERT	0-INS
'4C	EAST	6-EAST
'4D	WEST	4-WEST or BACKSPACE
'4F	RECALL	CTRL-R
'50	SHIFT-CANCEL	?
'52	SHIFT-PREV-SCREEN	CTRL-P
'53	SHIFT-NEXT-SCREEN	CTRL-N
'55	SHIFT-SOUTH	ESC, SOUTH
'56	SHIFT-NORTH ESC, NORTH	
'59	SHIFT-DELETE (LINE DEL)	SHIFT .-DEL
'5A	SHIFT-INSERT (LINE INS)	SHIFT 0-INS
'5C	SHIFT-EAST (EAST-5)	ESC, EAST
'5D	SHIFT-WEST (WEST-5)	ESC, WEST
'5F	D TAB	CTRL-T

SuperDOS Compatible Default Keyboard Equivalences Tale		
NPL Code Key	**NPL Virtual Key**	**Console Terminal Key**
’7C	GL	CTRL-G
’7D	SHIFT-GL	?
’7E	TAB	TAB
’7F	SHIFT-TAB	SHIFT TAB
’E1	HELP	ESC, ESC
’F0	EDIT	1-END

NOTE: A question mark (?) indicates that no key is assigned to the NPL code.

Entries preceded by a "*" (CTRL-(F1-F6) and SHIFT/CTRL-(F1-F6)) are supported only on SuperDOS 5.0 or greater.

"Console Terminal Key" represents default values under SuperDOS.

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.

6.8.4 Keyboard Lock Status

The console terminal keyboard under SuperDOS 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.

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.

CHAPTER 7

MULTI-USER CAPABILITIES

7.1 Overview

Multi-user capabilities are available on approved SuperDOS compatibles with the Super-DOS version of the NPL RunTime. This chapter provides a summary of these capabilities.

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 SuperLAN.

Section 7.8 discusses PC-Connect.

7.2 Device Sharing and "Hogging"

Device sharing and "hogging" is commonly performed by use of the NPL statements, $OPEN and $CLOSE. In some cases, $GIO statements may also be used. The $OPEN and $CLOSE statements, when directed to disk devices, are fully implemented and functional on the IBM and compatibles under SuperDOS. This is also true when directed to PRINT class devices (refer to Section 5.5 for the discussion on printer handling). $GIO statements which perform device "hogging" are not supported.

Refer to the NPL Statements Guide, $OPEN and $CLOSE, for details on the exact syntax of these statements.

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.

HINT: Under SuperDOS 5.0 and later, placing lock files in a RAM Disk will substantially improve performance.

NOTE: Applications which can handle a multiplexed 2200 environment, will likely handle record locking without already having the global partition since in a multiplexed 2200 environment, the global partition cannot be used for record locking.

7.4 Terminal Identification

Under SuperDOS, NPL has the ability to uniquely identify individual terminals through the use of system variables #TERM, #PART, and #ID. This allows multi-user applications to share information, while maintaining data integrity, by having control over individual terminals.

7.4.1 #TERM and #PART

Terminal identification (#TERM) is determined by the physical port number as specified in the system configuration file (CONFIG.P).

The /T RunTime startup option is used to directly set the #TERM value for background partitions. Refer to Section 4.4 for details.

The partition number (#PART) refers to the physical area of memory, or "task", assigned to a particular port through the system configuration file (CONFIG.P). When a program is being executed, it is entirely contained within the task area.

7.4.2 #ID

#ID is typically used in multi-user networks to distinguish between users. Under SuperDOS, it always returns a value of 0.

7.5 Inter-Task Communications

The following section discusses inter-task communication options for NPL under SuperDOS.

7.5.1 $PSTAT

NPL applications can use $PSTAT to pass information between partitions.

The $PSTAT statement is a special instruction which returns various status information for the partition specified in the expression. The argument for the function must be specified as a numeric-expression equal to the partition number.

An alpha-variable can be set to the contents of $PSTAT. This variable contains the following:

Byte	Contents
1-8	User-defined area
9	Operating system type
10	RunTime revision number
11	Bank # - will be set to #PART in packed (##) format
12-13	Always contains SPACEK in packed (##.##) format
14	Programmability ("P" or " ")
15	Terminal number in packed (##) format
16	Partition status
17-24	Blank
25	ERR function value (numeric portion of the last error encountered (in hex))
26-28	Partition # assignments, respectively
29	Device-address

For example:

```
0010 Q$ = $PSTAT(#PART)
0010 B$(),STR(A$,1,30) = $PSTAT(#PART)
```

Refer to the NPL Statements Guide, $PSTAT, for details on the exact syntax and use of this statement. Refer to section 8.3 for details on use of $PSTAT with background partitions.

7.5.2 $BREAK

$BREAK under SuperDOS works essentially identical to the operation on the Wang 2200. The remainder of the task's timeslice is released. Depending on system load, $BREAK may return immediately (no other tasks busy) or may cause a substantial delay (many other tasks busy).

NOTE: The special case $BREAK! is equivalent to a $END and causes an exit from the Run-Time Program.

Refer to the NPL Statements Guide, $BREAK for details.

7.5.3 DATE and TIME

The DATE and TIME functions operate normally with the following exception:

- The TIME function is only accurate to even seconds.

Refer to the NPL Statements Guide, DATE and TIME for details on these functions.

7.5.4 System Messages

The broadcast message ($MSG) may be assigned and inspected from any terminal. Refer to the NPL Statements Guide, $MSG, for details.

7.6 User Count

Under SuperDOS, each terminal using the NPL RunTime counts as one user against the total user limit. There are no charges against the user count for operating pseudo-ports or tasks assigned to port zero. This allows operating background partitions, PC-Connect multiple tasks, and Flip Screen multiple tasks without affecting the current user limit. Refer to Section 8.3 for details on background partitions. Refer to Section 7.8 of this Supplement and the SuperDOS manuals for details on PC-Connect. Refer to the SuperDOS manuals for details on Flip Screen.

7.7 SuperLAN

Bluebird Systems' SuperLAN allows networking capabilities to SuperDOS and NPL users (Release 5.1 or greater of SuperDOS and Release III or greater of NPL are required).

Use of SuperDOS on SuperLAN allows NPL developers to do the following:

- Read and write files that physically reside on other CPUs (called "nodes") in the network.
- Output information to ports (printers) that are physically connected to other nodes on the network.

- Generate a unique terminal identification based on each node number.

7.7.1 #ID under SuperLAN

#ID is used to return unique node identification based on the node ID number. The network controller board determines the unique node ID number for each CPU.

NOTE: #ID is determined by the RTISHARE (or RTPSHARE) task, not by the individual user task.

In some instances, #ID returns a value of 0 because the RTISHARE (or RTPSHARE) task loaded faster than the SuperLAN task. To avoid this problem, it is recommended to delay the RTISHARE (or RTPSHARE) task by a number of seconds specified in the RTISHARE (or RTPSHARE) task line of the CONFIG.P file.

For example:

```
4    0   38400H     /RTISHARE  xx
```

where xx divided by 10 is the number of seconds that the task will be delayed. Niakwa recommends that xx be equal to or greater than sixty (60) representing a six (6) second delay.

7.7.2 File Designation on SuperLAN

Use the following format for $DEVICE statements for diskimage files residing on remote nodes:

```
$DEVICE(/XXX)="Volume Number:Group:diskimage [clause]"
```

where:

XXX	Any valid NPL disk device address.
Volume Number	Identical to the drive designator under SuperDOS, it is an identifying number from 20 through 99. The user assigns the volume (drive) when the SuperLAN setup procedure is executed by means of the ISUPLAN program.

Group	Any valid SuperDOS user group.
Diskimage	The name of the diskimage.
Clause	Optional clause that modifies access to the diskimage depending upon the clause in use.

For example:

```
$DEVICE(/D11)="20:10:TESTDATA.BS2"
```

7.7.3 Printer Designation under SuperLAN

Either parallel or serial printers may be accessed under SuperLAN as in SuperDOS. $DEVICE statements for system printers are the same as under SuperDOS.

For example:

```
$DEVICE(/XXX)="Pn"
```

where:

XXX	any valid NPL print address.
n	SuperDOS system printer number.

To be able to print on a printer attached to a remote node, the local system printer must be assigned to the remote port number using the "Port" (Pn) command of the MMI utility.

For example:

```
MMI Pn=SSS
```

where:

SSS	The SuperLAN port number, from 128 through 255, which identifies a specific CPU and a specific port on that CPU. The user assigns these port numbers when the SuperLAN setup procedure is executed by means of the ISUPLAN program.

For example:

```
MMI P3=132
```

For printing to non-system serial printers, the SuperLAN port number is used directly in the $DEVICE statement:

For example:

```
$DEVICE(/XXX)="SSS"
```

where:

XXX	Any valid NPL print address.
SSS	The SuperLAN port number.

For example:

```
$DEVICE(/215)="132"
```

7.7.4 $OPEN/$CLOSE under SuperLAN

$OPEN/$CLOSE statements perform under SuperLAN in the same manner as under SuperDOS, with the additional ability to reserve/release a diskimage or printer located on a different CPU. No changes are required to NPL applications to use $OPEN/$CLOSE once the diskimages and/or printers are defined as described above.

NOTE: Access to files or devices across the SuperLAN network is slower than access to local files and devices. In addition, heavy traffic on the network can degrade the performance of the node where the files or devices are located. Therefore, it is recommended that systems be configured so that local files and local devices are accessed wherever possible, particularly for intensive update or report generation procedures.

The SuperDOS MMI utility is used to monitor active SuperLAN nodes and programs running on the remote nodes.

For example:

MMI RE	Displays all active network systems by node ID and the number of tasks on each system.
MMI TA= z	Displays the tasks (programs) running on the node ID equal to z.

7.8 PC-Connect

Bluebird Systems' product, PC-Connect, allows SuperDOS users to establish up to six (6) concurrent SuperDOS tasks from a stand-alone PC while maintaining the ability to access other MS-DOS applications running under Microsoft Windows 3.1 or greater.

Features of PC-Connect include:

- True Wyse 60 emulation.
- File transfer support to and from the MS-DOS workstation and the SuperDOS host.
- Support of communications line speeds up to 19200 baud.
- Support of Microsoft's Dynamic Data Exchange (DDE), allowing MS-Windows applications to share data.
- Local printer support.
- Foreign language keyboard selection.

Niakwa has evaluated PC-Connect and recommends its use with NPL, when using the PC-Connect Wyse 60 emulator.

The following sections describe the use of PC-Connect under NPL.

7.8.1 Wyse 60 Configuration under PC-Connect

For the Wyse 60 emulator to operate appropriately under NPL, two terminal configuration options must be selected after attaching to a SuperDOS task. This is done by selecting the Control Panel of the SuperDOS tasks window and choosing "CONFIGURE", and then "TERMINAL".

From the terminal configuration window, verify the selection of the "WANG" character set and the selection of "APPLICATION KEY MODE". Once these options have been selected, accept the changes and exit.

NOTE: Terminal configuration changes are dynamically updated and effective immediately. If either of these options is not selected, NPL applications will not operate properly.

Keyboard Characteristics under PC-Connect

The PC-Connect Wyse 60 emulator treats the remote PC as a Wyse 60 and not a SuperDOS console. The KEYBOARD.PCC file is necessary under PC-Connect. This file is included with the NPL Development Software on the Terminal Files Diskette. In order for the NPL RunTime to use this file, it is necessary to rename the file to KEYBOARD.WY6 or load it manually using the NPL $KEYBOARD statement. Refer to the NPL Statements Guide, $KEYBOARD for details.

NOTE: Renaming the KEYBOARD.PCC file to KEYBOARD.WY6 could cause possible problems for users on actual Wyse 60, 150, and 160 terminals because certain keyboard mappings would be changed. A solution to this problem is to load the keyboard files for either the PC-Connect workstations or the Wyse terminals by program control using $KEYBOARD.

The difference between the Wyse 60 keyboard and the standard PC style keyboard, under PC-Connect, is with the function keys, specifically function keys F11 through F16. The following table illustrates function key selection under PC-Connect.

PC KEY	NPL Virtual Key	NPL Value
F1...F10	SF'0..'9	'00...'09
ALT+ F1 ...F6	SF '10 ... '15	'0A ... '0F
SHIFT+ F1 ... F10	SHIFT SF '0 ... '9	'10 ... '19
ALT+ SHIFT+ F1 ... F6	SHIFT SF '10 ... '15	'1A ... '1F

Screen Character Set under PC-Connect

All screen character sets and attributes are supported. The Wyse 60 downloadable font "WYFONT.WY6" provided with the NPL Development Software for SuperDOS Niakwa, is emulated by selecting the "WANG" character set, when configuring PC-Connect.

NOTE: It is not necessary to download WYFONT.WY6 to PC-Connect workstations.

Terminal attributes are emulated using different colors for each attribute or combination of attributes. The user may choose to redefine the default color sequence for one or all attributes within the ATTRIBUTE Selection window which is selected from the Terminal Configuration window of PC-Connect. Alternatively, users may simply choose to disable the attributes, if they so choose.

All other screen characteristics are supported as documented in Section 7.6 of the NPL Programmer's Guide.

Local Printer Support under PC-Connect

Local printing is supported under PC-Connect. Applications operating under PC-Connect may now be configured to access a printer attached to the remote PC.

NOTE: NPL applications may use the same device equivalence as used for any local printer under SuperDOS.

7.8.2 #TERM and #PART under PC-Connect

#TERM and #PART values are generated appropriately. #PART always reflects the value of the TASK number in use. #TERM returns the pseudo port value of the terminal assigned by PC-Connect. Refer to Section 7.4 for details on #TERM and #PART.

7.8.3 Foreign Language Keyboard Selection Under PC-Connect

PC-Connect provides foreign character sets for the following countries:

France	Spain
Finland	Italy
Switzerland	United Kingdom
Denmark	Sweden
Norway	Germany
U.S.A.	The Netherlands (using U.S.A. keyboard)

These character sets are contained in a series of .MAP files located in the PCC subdirectory. The character set selected by PC-Connect is determined by the "country code" defined in the MS-Windows WIN.INI file (refer to the MS-Windows documentation for details on the WIN.INI file). The country code in this file can be overridden by defining a country code directly in the PCC.INI (refer to the PC-Connect User Guide for more details).

7.8.4 File Transfer under PC-Connect

PC-Connect allows a user to easily transfer files between multiple SuperDOS host systems or even locally on the PC. Refer to the PC-Connect documentation for details on implementing file transfer.

7.8.5 NPL Configuration Requirements under PC-Connect

Once PC-Connect is configured on the SuperDOS system, configuration of the NPL RunTime is necessary. PC-Connect allows multiple sessions (up to 6 tasks) to execute from one PC under MS-Windows 3.1 or greater. On the SuperDOS system, a single WTM task interfaces with the PC and controls communications between the tasks (pseudo ports) it has been assigned. When the NPL Runtime is executed, the RunTime uses the Terminal Kind value of the WTM task to locate an appropriate entry in the TERMTYPE.TBL file. This should be set to WY50 or WY60 depending on which PC-Connect emulator is selected. All pseudo ports assigned to WTM echo this terminal type. Refer to Section 6.3 for details on terminal determination.

Executing multiple NPL sessions under PC-Connect is supported under NPL. NPL respects the task and logical pseudo port numbers assigned by WTM. No further configuration is necessary for NPL. Refer to the PC-Connect documentation for details on setting up and configuring PC-Connect.

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 NPL under SuperDOS.

Section 8.2 discusses the environment-specific statements.

Section 8.3 discusses background partition support.

Section 8.4 discusses memory management.

8.2 SuperDOS-Specific Statements

This section discusses the NPL language features that are specific to the SuperDOS operating system.

8.2.1 $MACHINE

The $MACHINE system variable contains information about the hardware environment in which the RunTime is executing. In the SuperDOS implementation, the following platform specific values are set:

Byte	Description
Byte 1	RunTime Version :"S" for SuperDOS.
Byte 2	Hardware Manufacturer Code - HEX(00) for SuperDOS.
Byte 3	Monitor Type - Blank (ASCII)
Byte 4	Graphics Enabled - "G" = graphics enabled - (if using a Wang 2236 terminal) " " (blank) = "true" box graphics are not available.
Byte 5	Hardware Model Code. Under SuperDOS - the number of overflow areas (in binary) setup in the memory share module.
Byte 7	RunTime type in use - "I" = Interpretive version; "P" = Non-Interpretive version.
Byte 8	Display width in binary (80 or 132 column)
Byte 9	Current Terminal Type: HEX(00) Wang PC HEX(01) Wang 2210 HEX(02) Altos III HEX(03) VT100/VT200 HEX(04) IBM PC (using /R) HEX(05) Wyse 60,150,160 HEX(06) Wyse 50 HEX(07) Wang 2236 HEX(0B) Wyse 370
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 Chapter 6 for details on color support for NPL under SuperDOS.

Byte	Description
Byte 21	Contains the maximum number of entries (in binary) allocated to the handle table (in K) during a RunTime session.

NOTE: $MACHINE is a 64-byte variable. The above bytes are specific to the SuperDOS operating environment. For a complete description of all bytes within the $MACHINE system variable, refer to the NPL 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 SuperDOS version of NPL include:

Byte	Description
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 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 14	Indicates whether an implicit $BREAK should be performed after each disk I/O operation. The default is HEX(01).

Byte	Description
Byte 16	Switch for using a math co-processor. A value of HEX(00), the default, indicates that the math co-processor should not be used. A value of HEX(01) indicates that the math co-processor should be used.
Byte 17	Indicates whether dynamic selection of the 80 column or the 132 column mode of terminal display is to be performed by the RunTime.
Bye 18 & 19	Indicates support of a larger part of the 2200 character set as "normal" characters during line entry (LINPUT and INPUT operations) and command entry. The main purpose of this is to support the use of non-English character sets, which may require more than the standard range (HEX(10) through HEX(7F)). Byte 18 sets the low value of the extended range (default HEX(00)), while byte 19 sets the high value of the extended range (default HEX(00)).
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 - no color.
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 - no color.
Byte 31	Controls certain features for terminal emulators which do not provide 100% support of the terminal being emulated. This feature is provided strictly as a convenience for the users who must use emulation products. These emulation products are not supported for use with NPL. Each defined bit can be used to suppress an NPL feature for the terminal in use. Defined bits include: HEX(00) Default - No features supplied. HEX(01) The HELP display highlights only the current field. HEX(02) The terminal has no local printer capability. HEX(04) 132 column mode is not supported.

Byte	Description
Byte 32	Cursor style selection. For terminals where the cursor style must be set under program control, this byte may be used to instruct the Non-Interpretive RunTime as to which cursor style to set. Possible values include: HEX(00) Default. Use current cursor style or built in defaults. HEX(01) Set cursor style to line. HEX(02) Set cursor style to block. When a non-zero value is specified the cursor style is set permanently to the specified style and is not reset when the Non-Interpretive RunTime is exited.
Byte 39	Used to specify that no implicit $OPEN is to be performed on DATA LOAD BA and DATA LOAD BM when the platter has 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 NPL 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, and Section 7.5 for details on the exact syntax and use of this statement.

8.2.4 $SHELL

All NPL tasks have $SHELL (INVOKE) capabilities. This allows a temporary exit from the NPL environment to allow for interfacing with SuperDOS functions and commands.

When a $SHELL operation is executed, the entire current used task memory area is saved to disk in the SuperDOS swapper area. This means that $SHELL operations may be somewhat slower than on other operating systems and take longer, if large NPL programs or variables are currently defined in memory.

The use of $SHELL for interactive sessions is equivalent to execution of the SuperDOS SHELL command. Therefore, for interactive sessions, all syntax supported by the SuperDOS SHELL command is supported. Refer to the SuperDOS Utilities Guide for details on the SuperDOS SHELL command.

A six (6) byte return code is set by $SHELL if a return-variable is specified. This six (6) byte return code consists of three two byte binary fields. These fields correspond to inter-program communications codes set by various SuperDOS utilities using the STMA 2,1; STMA 2,2; and STMA 2,3 Business Basic statements respectively.

For example, return codes can be used to test for successful completion of a SuperDOS utility such as SDCOM. Refer to SuperDOS documentation for information on which inter-program communications codes are used by various utilities and the possible return codes. Refer to the Bluebird Systems Business Basic Statements Guide for further information on the Business Basic STMA statements.

8.2.5 $GIO Microcommand C620

This microcommand provides limited capabilities to input data from the serial port. Refer to Section 5.7, for details.

8.2.6 $GIO Microcommand 7600

This microcommand provides a method of determining whether specific security (PAL) codes are present on SuperCODER security device. This allows application developers to use the existing security mechanism provided by Bluebird Systems when operating under SuperDOS.

When directed to the NUL device (/000), the 7600 microcommand tests for the presence of the PAL code specified in the first four bytes of the specified register variable. If the PAL code is not present, byte 8 of the register is set to zero. If the PAL Code is present, byte 8 is set to a non-zero value.

The following is an example program which would perform a security check using the 7600 microcommand.

```
0010 DIM A$10                    :REM status variable must be 10 bytes or more
    :X=12340001                  :REM sample PAL code in decimal
    :A$=BIN(X,4)                 :REM convert code value to binary
    :$GIO/000(7600,A$)           :REM status check addressed to /000,
                                 :REM result to register 0 treated
                                 :REM specially by RTP
    :IF VAL(STR(A$,8))=0 THEN 30 :REM non-zero value means 'installed'
0020 PRINT "PAL code is installed"
    :STOP
0030 PRINT "PAL code is not installed"
    :STOP
```

The $GIO 7600 microcommand queries the null address and stores the result in the "bit bucket".

NOTE: When executed on any other Niakwa supported operating system or the Wang 2200, byte 8 returns a zero value.

Contact Bluebird Systems directly for allocation of PAL numbers for the SuperCODER.

8.3 Background Partition Support

Background partitions are supported under the SuperDOS version of NPL. The remainder of this section discusses the implementation and use of background partitions under SuperDOS.

8.3.1 Configuration Requirements

1. Each background partition requires that a SuperDOS task be configured to the required size (task size requirements are the same as for foreground tasks). The SuperDOS task must be configured as a background task (no terminal assigned).

2. RTP (or RTI) may be started in the background partition by an automatic password or EXEC file. The /B and /T= xx startup options must be specified. /B is used to inform the RunTime that it is operating in a background partition. The /T option is used to assign the background partition to a specific port (xx).

For example:

```
RTI /B /T=4 MYBOOT
```

starts the RunTime in the background, assigns the task to SuperDOS port number 4 and executes the MYBOOT.OBJ boot file.

3. Multiple background partitions may be assigned to the same port.

8.3.2 Operation of the Background Partition

1. Write attempts to the screen by the application when running in a background partition are stored in an internal buffer. When the partition is switched to foreground, the buffered screen contents display.

2. Values for byte 16 of $PSTAT (terminal status byte) are maintained as follows:

   - "A" for partitions that are in the foreground.
   - "D" for partitions in the background that are not waiting for a terminal to be attached.
   - "W" for background partitions that are waiting for a terminal to be attached. This state is caused by execution of a non-polling keyboard input request by the background partition.

3. #PART is equal to the SuperDOS task number and varies between foreground and background partitions assigned to the same port.

4. #TERM is equal to the physical port number to which the background partition is assigned to (as specified with the /T option). #TERM is identical for all partitions assigned to a given terminal, whether in the foreground or the background.

5. The value of Byte 15 of $PSTAT (terminal number) is equal to the physical port number assigned to the partition, as is #TERM.

6. $RELEASE TERMINAL is supported with restrictions as described in Section 8.3.3.

7. $IF ON, directed to addresses 005 or 001, returns a status of "BUSY" if a terminal is not attached to the partition or a status of "READ" if a terminal is attached to the partition. This is the same as the operation of $IF ON on the Wang 2200.

8. Terminal type is determined based on the SuperDOS port number specified with the /T option and is identical to the terminal type of the foreground partition.

8.3.3 Restrictions

1. Background partitions cannot be assigned to terminal number zero. Each background partition must be associated with a specific terminal and may be accessed only from that terminal.

2. $RELEASE PART performs no operation. On the Wang 2200, $RELEASE PART releases the current partition to terminal number zero so that it may be accessed by another terminal. Since, as described above, NPL background partitions must be associated with a specific terminal, this statement performs no operation.

3. Printing to local printers from a background partition is not supported. Attempting to access the local printer results in a P48, Illegal Device Specification, error. Printing to system printers or ASCII text files from the background partition is fully supported.

4. Limitations for $RELEASE TERM

   - $RELEASE TERM TO x is successful only when the background partition specified is waiting for keyboard input.

   - The HALT key may not function properly when a background task is in the foreground.

8.4 Memory Management

This section discusses Release IV memory allocation as it pertains to the SuperDOS implementation of NPL. For additional information concerning RunTime memory usage, refer to Chapter 3 of the NPL Programmer's Guide and Chapter 2 and Section 4.5 of this Supplement.

8.4.1 SuperDOS Considerations

Under SuperDOS, the partition size is fixed and determined by the configuration file (CONFIG.P). All NPL program code and defined variables reside within this section of memory (defined as the "user partition"). Under Release IV, this "partition" includes all modules currently in memory.

Available memory in the RunTime partition can be determined with the use of the SPACEF and SPACEW functions. Refer to the NPL Statements Guide, SPACEF and SPACEW for details.

CHAPTER 9

COMPILER OPERATION

9.1 Overview

This chapter provides an overview of the general operation of the NPL Compiler (B2C) on a Bluebird approved IBM-compatible PC running under SuperDOS. 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 SuperDOS operating system.

Section 9.4 discusses the print devices available under the NPL Compiler.

Section 9.5 discusses the use of exec files to operate the NPL Compiler.

NOTE: Under Release IV NPL under SuperDOS, the compiler is not capable accessing 320K raw diskettes. This is do to the fact the Release IV version of NPL operates under SuperDOS Protected mode. Therefore it is not possible for the compiler to compile directly to or from 320K raw format diskettes. Files either intended for input to the compiler or generated as output from the compiler must be directed to diskimage files.

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 5:20:2200DISK.BS2 PROG1 PROG2
```

the designation of "B2C" at beginning of the SuperDOS command line, refers to the name of the NPL compiler and causes SuperDOS 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 SuperDOS 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 NPL 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 SuperDOS conventions and their implications to NPL.

9.3.1 Supported Characters

The file-naming conventions of SuperDOS are not entirely compatible with the names of files within NPL diskimages. Specifically, within diskimages, filenames may be any eight characters, with no restrictions. SuperDOS also allows eight-character filenames, but the permitted character set is restricted. Only the letters A-Z, digits 0-9, a period "," and an underline "_" are allowed.

Consequently, situations may arise where it may not be possible to specify the exact form on an input program name on the command line. For example program names with embedded blanks are legal in NPL diskimages but are not allowed under the SuperDOS operating system. The problem has no general solution, but 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 SuperDOS 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 SuperDOS 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 SuperDOS 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 SuperDOS user group, it replaces the embedded blanks in the filenames from within the diskimage file with underlines. Conversely, when the compiler needs to place an SuperDOS filename into a diskimage it replaces underlines with spaces.

For legibility, groups of codes may be separated by a decimal point (".").

For example:

```
/TRANSLATE 5F20.513C
```

Defines the following equivalences (where D.I. stands for Diskimages):

SuperDOS (HEX)	5F	51
SuperDOS (display)	_	Q
Filenames in D.I. (display)		<
Filenames in D.I. (HEX)	20	3C

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 SuperDOS 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 SuperDOS does not permit entry of lower case names in the command line 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 user group 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 PLATTER2.BS2 /OBJLOC PLATTER1.BS2 *
```

compiles all programs in the diskimage file PLATTER2.BS2 in the user group specified in the user's search list, placing compiled output in diskimage file PLATTER1.BS2 in the default user group.

```
B2C /SRCLOC PLATTER2.BS2 /OBJLOC PLATTER1.BS2 AR??001
```

compiles all programs with the first two characters "AR" and characters 5-7 as "001" in the diskimage file PLATTER2.BS2 in the user group specified in the user's search list, to diskimage file PLATTER1.BS2 in the default user group 5:0.

When wildcard-scanning a user group, 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 following discussion discusses pathnames under SuperDOS.

User groups

The NPL Compiler (B2C) assumes that all input and output files reside in the default user group. This is modifiable by specifying an user group in the compiler command line.

For example:

```
B2C /SRCLOC 5:00:PLATTER1.BS2 /OBJLOC 5:20:PLATTER2.BS2 *
```

locates the 5:00 user group and compiles all the programs found in PLATTER1.BS2, into PLATTER2.BS2 in the 5:20 user group.

NOTE: The user group 0:0 is special and is reserved to mean the default user group.

Raw Diskette Support

Under Release IV NPL under SuperDOS, the compiler is not capable accessing 320K raw diskettes. This is do to the fact the Release IV version of NPL operates under SuperDOS Protected mode. Therefore it is not possible for the compiler to compile directly to or from 320 raw format diskettes. Files either intended for input to the compiler or generated as output from the compiler must be directed to diskimage files. Refer to Chapter 10 for details on porting to and from a Wang 2200.

The NULL Device

Output for any of the compiler operations can be discarded by specification of the null device as the output parameter. Under SuperDOS, the null device must be specified as NULL.

For example:

```
B2C /SRCLOC 5:10:PLATTER1.BS2 /OBJLOC NULL /LSTLOC 5:20:PLATTER2.BS2
/LSTFORMAT 2200 *
```

creates files in 2200 atomized format (5:20:PLATTER2.BS2), directly from a diskimage file containing already compiled programs (5:10:PLATTER1.BS2). The NULL 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 > P1	Direct listing to parallel printer. Other system printers may be used if configured (i.e., > P2, > P3). Access to serial ports not configured as printers and not used by other tasks is also allowed (i.e, > 4 directs output to port 4).
/LSTLOC NULL	Suppresses listing.

/LSTLOC 5:0:	Store source listing for each file in the specified user group. Filenames are derived from the input program name, with an extension of .LST or .SRC, depending on the /LSTFORMAT designation.
/LSTLOC 5:20:filename	Concatenates all program listings into the specified filename (with a .SRC file extension) in the specified user group.

9.5 Exec Files

Exec files are functions which allow the user to set up and execute files containing SuperDOS commands and parameters. Exec files can be used to invoke the NPL Compiler.

For example:

```
B2C /SRCLOC PLATTER1.BS2 /OBJLOC PLATTER2.BS2 /LSTLOC 5:0:
 /LSTFORMAT .SRC AR*
```

can be set up as a exec file called COMPAR.B, and then be executed from the command processor by simply entering the name of the exec file:

```
EXEC COMPAR.B
```

In this case, the exec file would perform in exactly the same way as the example command line from which it was taken.

9.5.1 Example Exec Files

Niakwa has provided some files for your convenience (some developers may want to set up their own). These examples may be invoked from the SuperDOS command processor by name. These exec files were designed to meet the most frequent requirements of the NPL developer.

Following are the four exec files supplied by:

B2CA.B

```
B2C /DISPLAY ON /SRCLOC 5:20:PLATTER2.BS2 /OBJLOC PLATTER1.BS2
```

The above exec file compiles input programs from a diskimage file PLATTER2.BS2 on the 5: drive in the 20: user group, creating p-code files in UNSCRAMBLED format and places them in the diskimage file PLATTER1.BS2 (located using user groups in the user's search list), displaying any compiler warning messages on the screen.

B2CB.B

```
B2C /DISPLAY ON /SRCLOC 5:20:PLATTER1.BS2 /OBJLOC NULL /LSTLOC 0:0
/LSTFORMAT .SRC  /TRANSLATE 5F20=@=/=@=$
```

The above exec file compiles input programs from a diskimage file PLATTER1.BS2 on the 5: drive in the 20: user group, creating ASCII text files in the first user group in the user's search list, which may be edited and resubmitted to the compiler.

NOTE: Conversion of filenames between filenames within NPL diskimages and SuperDOS filenames is performed on the character pairs " " (NPL) and "_" (SuperDOS), and "/" (NPL) and "@" (SuperDOS), and "$" (NPL) and "@" (SuperDOS) when creating the text files.

B2CC.B

```
B2C /DISPLAY ON /REM$ ON /OBJLOC PLATTER1.BS2 /TRANSLATE 5F20=@=/=@=$
```

The above exec file compiles input programs in ASCII format from the first user group in the user's search list, creating pcode files in UNSCRAMBLED format are placed in the diskimage file PLATTER1.BS2 (located in the user's search list), compiling any statements which start with a "REM $ PC", and displaying any compiler warning messages on the screen.

NOTE: 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.

B2CD.B

```
B2C /DISPLAY ON /LSTLOC 5:20:PLATTER1.BS2 /OBJLOC NULL /WARNINGS OFF
/TRANSLATE 5F20=@=/=@=$
```

The above exec file compile input programs in ASCII format from the first user group in the user's search list, creating output programs in 2200 atomized format directly to the diskimage file PLATTER1.BS2 on the 5: drive in the 20: user group, which may then be ported over to the 2200.

NOTE: If the SuperDOS filenames contain "_" or "@" characters, the names of the files within the diskimage are automatically replaces any "@" characters with the "/" character, and replaces the "_" with a " ".

9.5.2 Example Compiles

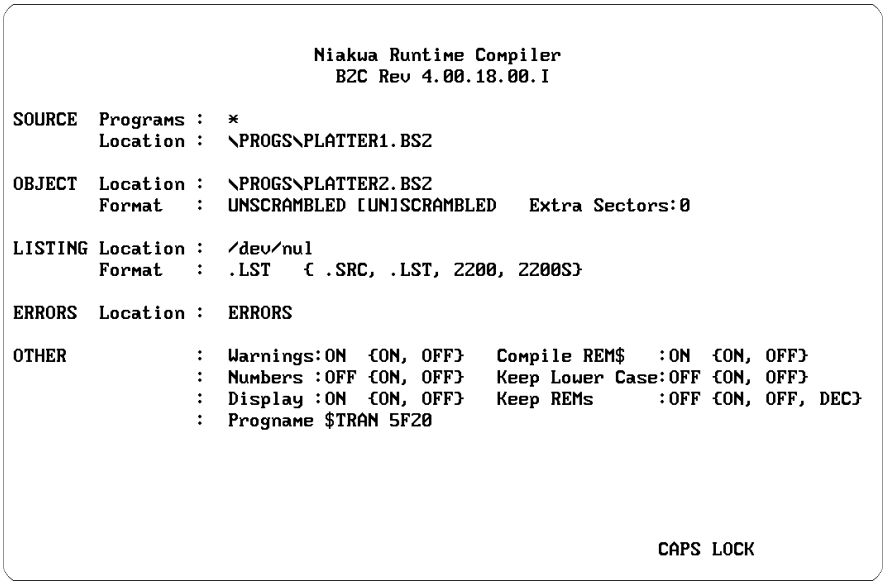
Chapter 14 of the NPL Programmer's Guide describes each of the options available for use when compiling programs. The remainder of this chapter provides several examples of commonly used compilations, indicating both the particular compiler command line and the compiler display. A brief discussion of 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 5:20:PLATTER1.BS2 /OBJLOC 5:20:PLATTER1.BS2
/ERRLOC ERRORS /REM$ ON *
```

Display:



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 5: drive in the 20: user group.
OBJECT Location:	indicates a diskimage file, PLATTER2.BS2, on the 5: drive in the 20: user group, receives the compiled programs.
Format:	the default UNSCRAMBLED indicates that program encryption is not to take place.
Extra Sectors:	the default 0 indicates that extra free sectors are not to be allocated to the programs.
LISTING Location:	NULL 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 default user group which will receive all warnings and errors during the compilation.

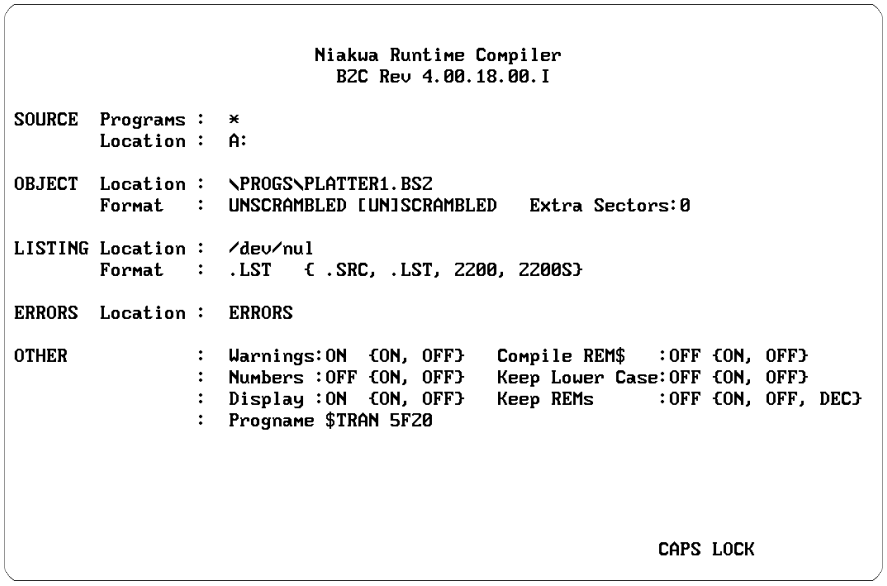
OTHER Warnings:	ON indicates that all warnings encountered during the compile will be displayed on the screen.
Numbers:	the default OFF indicates that additional object code to keep track of multiple program statements is not generated.
Display:	ON indicates that the compiler display screen will be shown 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 SuperDOS 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 SuperDOS files and programs stored in NPL diskimage files.

From a 2200 To a Compiled Diskimage

Command line:

```
B2C /DISPLAY ON /SRCLOC 5:20:PLATTER2.BS2 /OBJLOC 5:20:PLATTER1.BS2
/ERRLOC ERRORS *
```

Display:



Discussion:

The above example indicates the compilation procedure which would be used for the initial porting of programs from a Wang 2200 diskimage (located on the SuperDOS system's hard drive) to a diskimage file on the system's hard drive.

Explanation of Options:

SOURCE Programs: the '*' indicates that all programs in the specified SRCLOC are to be compiled.

Location: 5:20:PLATTER2.BS2 indicates that the input programs are located on the 5: drive in the 20: user group.

OBJECT Location:	indicates a diskimage file, PLATTER1.BS2, ON the 5: drive in the 20: user group, 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:	NULL 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 default user group 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 SuperDOS 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.

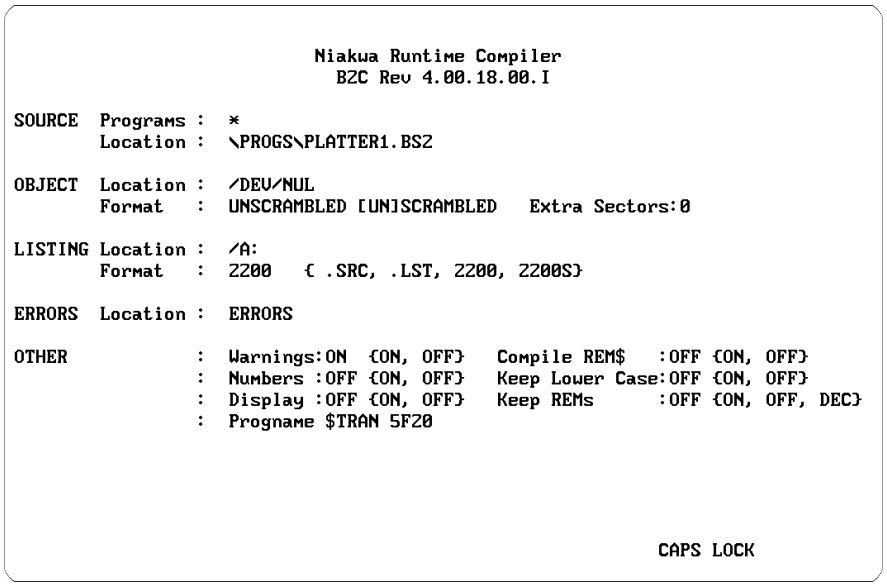
Progname $TRAN: the default 5F20 is irrelevant since it only pertains to filename translations between programs stored as stand-alone SuperDOS files and programs stored in NPL diskimage files.

From a Compiled Diskimage To 2200 Atomized Code

Command line:

```
B2C /DISPLAY ON /SRCLOC 5:20:PLATTER1.BS2 /OBJLOC NULL /LSTLOC
5:20:PLATTER2.BS2  /LSTFORMAT 2200 /ERRLOC ERRORS *
```

Display:



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, if supported by SuperDOS or as the example, to the diskimage file, specified by LSTLOC, on the hard disk. This diskimage then could 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, on the 5: drive in the 20: user group.
OBJECT Location:	NULL indicates that the compiler will suppress generating a second set of compiled programs.
Format:	the default UNSCRAMBLED is irrelevant since an OBJLOC of NULL is specified.
Extra Sectors:	the default 0 is irrelevant since an OBJLOC of NULL is specified.
LISTING Location:	5:20:PLATTER2.BS2 indicates that the compiler will generate output programs directly to the diskimage file on the 5: drive in the 20: user group.
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 default user group 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. 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 NULL 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 NULL is specified.
Keep Lower Case:	the default OFF is irrelevant since it only pertains to file-name translations between programs stored as stand-alone SuperDOS files and programs stored in NPL diskimage files.
Keep REMs:	the default OFF is irrelevant since an OBJLOC of NULL is specified.
Progname $TRAN:	the default 5F20 is irrelevant since it only pertains to file-name translations between programs stored as stand-alone SuperDOS 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 5:20:PLATTER2.BS2 /OBJLOC 5:20:PLATTER1.BS2
/LSTLOC 5:21: /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, Z200, Z200S}

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 SuperDOS directory. It assumes that the diskimage file 5:20:PLATTER2.BS2 exists and contain programs in the Wang 2200 atomized format.

Explanation of Options:

SOURCE Programs: the '*' indicates that all programs in the specified SRCLOC are to be compiled.

Location: 5:20:PLATTER2.BS2 indicates that the input programs are located on the 5: drive in the 20: user group.

OBJECT Location:	indicates a diskimage file, PLATTER1.BS2, on the 5: drive in the 20: user group 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 5:21 user group.
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 default user group 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, SuperDOS is case insensitive. Therefore, no distinction can be made in the SuperDOS 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.

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 a Compiled Diskimage To ASCII Text Files

Command line:

```
B2C /DISPLAY ON /SRCLOC 5:20:PLATTER1.BS2 /OBJLOC NULL /LSTLOC 5:21:
/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, Z200, Z200S}

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 SuperDOS 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, on the 5: drive in the 20: user group.
OBJECT Location:	NULL indicates that the compiler will suppress generating a second set of compiled programs.
Format:	the default UNSCRAMBLED is irrelevant since an OBJLOC of NULL is specified.
Extra Sectors:	the default 0 is irrelevant since an OBJLOC of NULL is specified.
LISTING Location:	indicates that source program listings will be generated in the 5: drive in the 20: user group.
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 default user group 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 NULL 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 NULL is specified.
Keep Lower Case:	the default OFF is irrelevant because even though ASCII .SRC files are being created, SuperDOS is case insensitive. Therefore, no distinction can be made in the SuperDOS files between upper and lower case.
Keep REMs:	the default OFF is irrelevant since an OBJLOC of NULL 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 User Group

Command line:

```
B2C /DISPLAY ON /SRCLOC 5:20 /OBJLOC 5:20 /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, Z200, Z200S}

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 SuperDOS directory. For example, a simple boot program may appear as:

```
10 $DEVICE(/D11)="5:21:PLATTER1.BS2"
20 $DEVICE(/D12)="5:20: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 on the 5: drive in the 20: user group.
OBJECT Location:	indicates that the compiler will create an object file with an extension of .OBJ on the 5: drive in the 20: user group.
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:	NULL 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 default user group 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 SuperDOS 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.

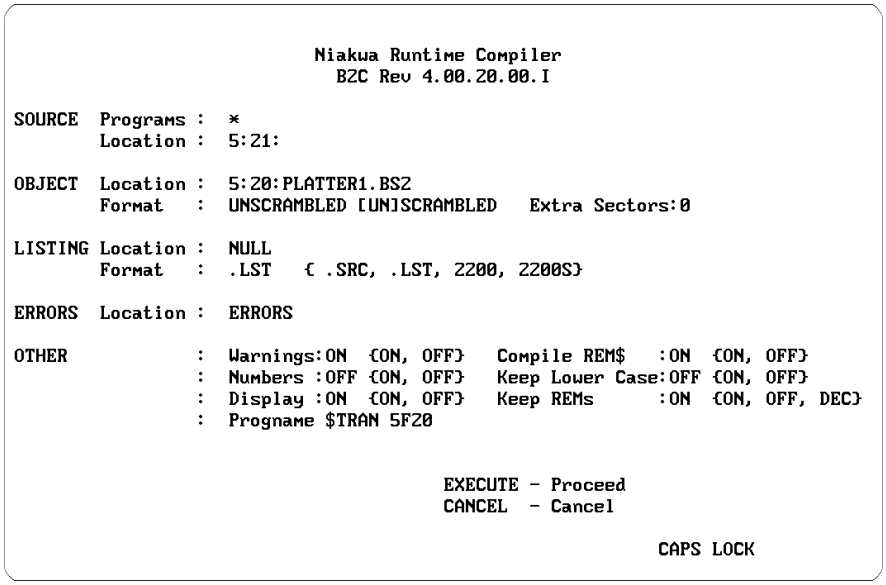
Progname $TRAN: the default 5F20 is irrelevant since it only pertains to filename translations between programs stored as stand-alone SuperDOS files and programs stored in NPL diskimage files.

From ASCII Text Files To A Compiled Diskimage

Command line:

```
B2C /DISPLAY ON /SRCLOC 5:21: /OBJLOC 5:20:PLATTER1.BS2 /ERRLOC ERRORS
/REM$ ON /KEEPREMS ON *
```

Display:



Discussion:

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 on the 5: drive in the 20: user group.
OBJECT Location:	indicates a diskimage file, PLATTER1.BS2, on the 5: drive in the 20: user group, 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:	NULL 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 default user group 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. 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 SuperDOS .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 SuperDOS.

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. The following sections discuss these options.

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 or from a SuperDOS 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.

Porting directly to the MS-DOS partition on the SuperDOS system from various other systems, including the Wang 2200, is possible using serial communication programs operating under MS-DOS.

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 SuperDOS version of NPL accepts 5-1/4" 360K, 1.2MB and 3-1/2", 720K, 1.44MB, "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 platforms.

NOTE: SuperDOS protected mode does not allow for the formatting of "raw" diskettes. Therefore it is necessary to pre-format "raw" diskettes under the MS-DOS version of NPL before using them under SuperDOS.

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 SuperDOS	**$DEVICE(/D10)="1: 360= Y"**	**Instructs NPL to treat 1: as a "raw" 360K device.**
Under MS-DOS	**$DEVICE(/D10)="A: 360= Y"**	**Instructs NPL to treat A: 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

The following section discusses the methods available to port NPL applications for SuperDOS from specific environments. Several of the methods described below require to porting the application to the MS-DOS partition first and then porting it to the SuperDOS environment.

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 an UNIX-Based System

One product Niakwa has used for the transfer of files to PC's from UNIX systems is Reflections 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 Reflections 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 a MS-DOS based partition on Super-DOS systems from UNIX-based systems by using the UNIX doscopy command. Once on the MS-DOS partition they can be transferred to the SuperDOS partition using Super-DOS Utilities. Refer to Section 10.3.3 for details. Most UNIX flavors can read and write MS-DOS format 5-1/4" and 3-1/2" diskettes using the UNIX doscopy command (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 another SuperDOS-Based or MS-DOS 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. Real mode is not supported by Release IV of NPL under SuperDOS, but real mode can be used to transfer the files. SuperDOS 6.0, which runs only in protected mode, has a different 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 Section 2.6 for details

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.

Refer to Chapter 10 of the NPL MS-DOS Supplement for additional details on transferring NPL data and programs between MS-DOS systems.

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. There are both benefits and penalties which may occur as a result of using mixed language 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 SuperDOS. 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 SuperDOS support.

Section 11.5 discusses support of Metaware High C under SuperDOS.

Section 11.6 discusses support of Microsoft Macro Assembler under SuperDOS.

Section 11.7 discusses support of Metaware Professional Pascal under SuperDOS.

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 in the text in Chapter 16 of the NPL Programmer's Guide.

The examples are provided mainly to allow a simple test of the 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.
INSTALLS.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 Metaware Professional Pascal. Release IV features are not supported under this language.
RTPALL.INC	For programmers that have BESDK libraries in Microsoft Macro Assembler. Release IV features are not supported under this language.
\INCLUDE\SUP	This directory is specific to the MS-DOS implementation of BESDK.
RPTPARM.OBJ	Compiled version of RTPPARM.C using Metaware High C.
RTPPARM.C	C subroutines to provide the rtpfn_getparminfo() function used to check function declarations.
QLBHEAD.OBJ	Compiled version of QLBHEAD.C using Metaware High 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.
\SUPCEXAM	Contains example files for SuperDOS implementations using Metaware High C.

\SUPMEXAM Contains example files for SuperDOS implementations using Microsoft Macro Assembler.

\SUPPEXAM Contains example files for SuperDOS implementations using Metaware Professional 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 Metaware High C programs
ASM	for Microsoft Macro Assembler programs
PAS	for Metaware Professional 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 masm, hc, pp, 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., LINK) are set to null values.
- When using Metaware High C, the directory in which this product has been installed is defined by the environment variable HIGHC. Add a line to the AUTOEXEC.BAT file in the form "SET HIGHC= D:\DIRECTORY" to ensure this information is always available. The CONFIG.S file in this directory must contain appropriate SuperCoder information for the hc cross compiler product.

- When using Metaware Professional Pascal, the directory in which this product has been installed is defined by the environment variable PP. Add a line to the AUTOEXEC.BAT file in the form "SET PP=D:\DIRECTORY" to ensure this information is always available. The CONFIG.S file in this directory must contain appropriate SuperCoder information for the pp cross compiler product.

11.3 Installation of the BESDK

The installation of the BESDK Files Diskette is performed by the use of a simple batch file INSTALLS.BAT. To install, enter:

```
INSTALLS "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:

```
INSTALLS 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 SuperDOS Support

The following section discusses extended call support under SuperDOS.

11.4.1 Environments

External libraries are supported for SuperDOS using an MS-DOS cross-assembly environment -- it is necessary to compile and link under MS-DOS, then transport the resulting program or quick library to SuperDOS (using PCFILE) to use it. Standalone files (.EXE extension) should be copied to type "R" SuperDOS files, while the quick library (.QLB extension) should be copied to type "T" files. The PCFILE utility uses these as default file types.

NOTE NPL Release IV supports NPL external calls only under Protected Mode SuperDOS.

Examples assume Microsoft LINK version 5.01.21 or later. Earlier versions may also work but are not tested.

11.4.2 Quick Library--Stack and Heap Allocation

When loaded as a quick library, the image is loaded into the high part of task memory, with approximately 8K available for stack and heap allocations (this is the minimum requirement for the standard C startup). Additional space for stack and heap (or less, for MASM quick libraries) may be reserved by modifying the "MIN_BSS" field of the quick library using the "EXEHDR" utility.

For example:

```
ren mylib.qlb *.exe
exehdr mylib.exe /min 400
ren mylib.exe *.qlb
```

adjusts the file so that 0400H paragraphs (16K) are reserved for stack and heap in the task space above the image (renaming is required because EXEHDR only operates on files with an .EXE extension). Increasing the allocation for the external library reduces the space available to the NPL partition. Examples assume Microsoft EXEHDR version 2.01 or later. Earlier versions may also work but are not tested.

The entry point of the module (as far as Microsoft LINK is concerned) may be anything and is ignored when running under SuperDOS. The first 600H bytes of the code area are not loaded. The usual convention is to include a small MS-DOS compatible warning message printing program that is located at offset 0 of the code area and make this the entry point, in case the .EXE program is accidentally run under MS-DOS. Entry to the image when running under SuperDOS is made at 600H in the first segment of the image. There must be a symbol '_mwINIT' at this location and this symbol must be public. Linking with the standard C startup and support library meets all these conditions.

11.5 Metaware HIGH-C (DOS Cross-Compiler)

Writing external subroutines in Metaware High C for SuperDOS is relatively straightforward. Examples assume Metaware High C version 1.4 or later. Earlier versions may also work but are not tested.

11.5.1 General

The cross-compiler (when making programs) or SuperDOS High C support library (when running programs or using external libraries which contain C code) requires the installation of SuperDOS security codes (on SuperCODER) to enable use of these functions. The cross-compiler requires that an appropriate config.s file be available in the current directory when compiling.

Use the big model (-mm big) option on all "hc" compile commands, or ensure that this is declared in a pragma in the profile file used.

For example, "hc.pro" contains the line:

```
pragma Memory_model(Big);
```

make sure the include files provided with the BESDK are available either by a "-ipath directory" option to the "hc" command or ensure that this is declared in a pragma in the profile file used. For example, "hc.pro" contains the line:

```
pragma ipath("directory");
```

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. When linking C standalone modules or quick libraries, the library module CINIT.OBJ must be the first module in the linkage order. The linker entry point of an image linked using Metaware High C is 0H in the first code segment, which is a MS-DOS-compatible warning routine. The SuperDOS entry point is 600H in the first code segment, and is labeled _mwINIT.

The RTP subroutine should be referenced as a far external with the name "RTP" (linker symbol "RTP").

11.5.3 Calling Conventions for BESDK Subroutines

Test RTP Subroutines

Declare all GOSUB' routines using "_cc(_FAR_CALL | _CALLEE_POPS_STACK)" calling conventions. The "rtpdeffn.h" include file for SuperDOS defines "rtpdeffn_ext" as equivalent to this designation.

RTPEXT Subroutine

The RTPEXT subroutine should be defined as a far procedure with the name "RTPEXT" (Linker symbol "RTPEXT"). When called, the (far) 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 "_cc(_FAR_CALL | _CALLEE_POPS_STACK)" designation on declarations of all subroutines which will be called using the GOSUB' interface. The "rtpdeffn.h" include file for SuperDOS defines "rtpdeffn_ext" as equivalent to this designation.

Formats of strings in NPL do not have a zero-terminator and are not variable length. If strings are to be used by C library routines you must make copies which have trailing spaces removed and a zero terminator added.

11.5.4 Linkage of Test Program

Programs written in High C 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 standalone should include:

- CINIT.OBJ (from SuperDOS High C support)
- The mainline
- The RTP test subroutine
- 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\SUP\) must also be included.

- The High C support library (HCBE.LIB) (name may vary)

NOTE: CINIT.OBJ must be the first input file specified to LINK.

For example:

```
LINK cinit mymain myrtp myrtpext mysub myproc rtpparm,mymain,,hcbe
```

Linkage of Quick Library

Programs written in High C must be specifically linked to produce an executable file. All input files to Microsoft 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 standalone should include:

- CINIT.OBJ(from SuperDOS High C support)

- The mainline (i.e., MYMAIN.OBJ)

- Quick library header (QLBHEAD.OBJ)

- Your RTPEXT subroutine (i.e., MYRTPEXT.OBJ)

- Your GOSUB' subroutines (i.e., MYSUB.OBJ)

- Any FUNCTION or PROCEDURE subroutines. If these are used, the rtpparm.obj module (located in the \INCLUDE\SUP) must also be included.

- The High C support library (HCBE.LIB)(name may vary)

NOTE: CINIT.OBJ must be the first input file specified to LINK.

For example:

```
LINK /QUICKLIB /PACKCODE cinit mymain qlbhead myrtpext mysub myproc my-
callbk rtpparm,mylib,,hcbe;
```

produces "MYLIB.QLB".

Protected Mode Support

Quick libraries written using the C support libraries operate in Protected Mode.

NOTE: The /PACKCODE option of LINK is required when the external library (or stand-alone) is operating under Protected Mode. This reduces the number of logical code segments produced by the linker.

11.6 Microsoft Assembler (DOS Cross-Assembler)

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 have not been tested.

External libraries written in Microsoft Macro Assembler do not support the FUNCTION or PROCEDURE interfaces or callbacks to NPL.

11.6.1 General

Make sure include files provided with the BESDK are available either in a directory specified by the INCLUDE environment variable or by a "/Idirectory" option for the "masm" command.

Do not use any logic which treats segment numbers as physical address pointers (i.e., phys address= segment*16).

All code segments should be designated as combine class "CODE". This is used by the LINK program to distinguish code from data when the /packcode option is specified. All such segments are non-modifiable (read-only) when running in Protected Mode.

11.6.2 Mainline

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 you must use explicit segment names, 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 will be 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.

Assumptions should not be made about the location of the Task Information Block (TIB) relative to segments in the image. When the RTP loads the quick library, the total load size of the image (including all memory below it in the task area, which is reserved for NPL) is placed in the "size of loaded program" fields of the TIB (word at offset 350, longword at offset 362). At the 600H entry point, the segment registers ES: and DS: point to the TIB.

At entry the stack in use is in the TIB segment ending at 600H. It is necessary to define a separate stack area and change SS:SP to start using it before calling the RTP subroutine. The area between 200H and 600H of the TIB segment is used as a scratch area by the RTP and is not available to the externals except during image startup.

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 RTPREQ structure (defined in include file RTPREQ.INC) is on the stack as a far pointer. The last field (RTPREQ_VAR) of this structure is the variant part. If RTPREQ_TYPE has the value RTPREQ_TYPE_RTPDEF then the variant part is a RTPDEF structure (defined in include file RTPDEF.INC).

GOSUB' Subroutines

Each subroutine should be defined as a far procedure. When called, the parameters will be on the stack below the return address. The first parameter of the GOSUB' is pushed first, the last parameter pushed last.

A string parameter is passed as:

```
PUSH            SEG <string>
PUSH            OFFSET <string>
PUSH            SIZE <string>
```

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.

11.6.4 Linkage of Test Program

Programs written in MASM must be specifically linked to produce an executable file. All input files to Microsoft LINK must be the result of previously run assemblies or libraries of files.

The files required for production of the standalone 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 standalone should include:

- The mainline(i.e. MYMAIN.OBj)
- Quick library header (i.e.,QLBHEAD.OBJ)
- The RTPEXT subroutine (i.e., MYRTPEXT.OBJ)
- The GOSUB' subroutines (i.e, MYSUB.OBJ)
- Any libraries referenced by the above modules. (i.e., LIBS.LIB)

For example:

```
LINK /QUICKLIB mymain qlbhead myrtpext mysub,mylib,,libs;
```

produces "mylib.qlb".

Quick libraries written entirely in Microsoft Macro Assembler must meet the following conditions:

- The number of segments defined should be kept to a minimum. Linking with the /PACKCODE option will usually suffice to place all code segments into a group without program modifications. At entry to the image, the DGROUP segment will have data (modifiable) access. All other segments are code (read-only) access. SuperDOS system calls are available to change the access class of segments. If the image contains too many segments to load, NPL will exit with a diagnostic at startup time.

- No attempt should be made to modify the Local Descriptor Table for any descriptors that are (numerically) lower than the first code segment of the image, when using assembly language modules. Failure to comply with this requirement can crash the system.

Some additional restrictions which will continue to apply to Protected Mode code are:

- Since segment addresses are no longer directly related to physical addresses, no operations which use this dependence will work.
- Modifications to code segments are prohibited.
- Execution of code located in data segments is prohibited.

11.7 Metaware PASCAL (DOS Cross Compiler)

External subroutines can be written in Metaware Professional Pascal. It is usually necessary to use extensions to standard Pascal are required to support the required functionality for the external calls. In particular the standard packages "Loopholes" and "Other_languages" must be available to the example include files Examples files assume Metaware Professional Pascal version 2.7 or later. Earlier versions may also work but have not been tested.

11.7.1 General

Note that use of the cross-compiler (when making programs) or the SuperDOS Professional Pascal support library (when running programs or using external libraries which contain Pascal code) requires the installation of SuperDOS security codes (on the SuperCODER) to enable use of these functions. The cross-compiler requires that an appropriate CONFIG.S file be available in the current directory when compiling.

Use the big model (-mm big) option on all "pp" compile commands, or ensure that this is declared in a pragma in the profile file used. For example, "pp.pro" contains the line:

```
pragma Memory_model(Big);
```

Make sure the include files provided with the BESDK are available either or by a "-ipath directory" option to the "pp" command or ensure that this is declared in a pragma in the profile file used. For example, "pp.pro" contains the line:

```
pragma ipath("directory");
```

11.7.2 Mainline

Declare RTP as an external function returning type CARDINAL result. It is not necessary to pass parameters to RTP in this case. The value returned by RTP can be passed back to MS-DOS by use of the halt(x) standard function.

The starting label of user code is called _mwmain ("_mwmain" to linker). The "_mwmain" symbol is generated for any PP module which has a body associated with the "program" header.

NOTE: Substantial startup code from the Pascal library is executed before reaching the "_mwmain" routine.

When linking Pascal standalone modules or quick libraries, the library module init.obj must be the first module in the linkage order. The linker entry point of an image linked using Metaware Professional Pascal is 0H in the first code segment, which is a MS-DOS compatible warning routine. The SuperDOS entry point is 600H in the first code segment, and labeled "_mwINIT".

The RTP subroutine should be referenced as a far external with the name "RTP" (linker symbol "RTP").

11.7.3 Calling Conventions for BESDK Subroutines

Test RTP Subroutines

Declare RTP as function named "RTP" returning cardinal.

Declare GOSUB' subroutines with argument types "x:rtpstr_pointer, len:cardinal" for a string, and "x:rtpnum_pointer" for a numeric.

Call GOSUB' subroutines with arguments in format "Address(x[1]), length(x)" for a string, and "Address(x)" for a numeric.

Precede all GOSUB' routine declarations using the following pragma:

```
pragma Calling_convention([far_call,callee_pops_stack]);
```

Follow all GOSUB' routine declarations using the pragma:

```
pragma Calling_convention();
```

RTPEXT Subroutine

Declare RTPEXT as a function named "RTPEXT" returning an integer.

Assign the address of the GOSUB' procedure to rtpdef_pointer using the Address(x) function.

Assign the address of the LIST' description string using the retype(address(x[1]),rtpstr_pointer) function. Strings used for this purpose should be not be declared as local variables to RTPEXT, since this places them in a volatile area (the stack).

The RTPEXT subroutine should be defined as a far procedure with the name "RTPEXT" (Linker symbol "RTPEXT"). When called, the (far) address of the rtpdef structure (defined in include file rtpdef.ppi) is the only parameter. The first field of this structure is a rtpreq structure (defined in include file rtpreq.ppi).

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, Professional Pascal requires a duplicate function declaration with the "external" designation replacing the body.

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.

Precede all GOSUB' routine declarations using the following pragma:

```
pragma Calling_convention([far_call,callee_pops_stack]);
```

Follow all GOSUB' routine declarations using the pragma:

```
pragma Calling_convention();
```

Formats of strings in NPL do not have appropriate format for use by Pascal. If strings are to be used by Pascal library routines you must make copies which have trailing spaces removed and appropriate length definitions.

11.7.4 Linkage of Test Program

Programs written in Professional Pascal 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 standalone should include:

- Init.obj (from SuperDOS Professional. Pascal support)
- The mainline
- The RTP test subroutine
- The RTPEXT subroutine (optional, but recommended)
- The GOSUB' subroutines
- The Professional Pascal support library (PPBE.LIB) (name may vary)

NOTE: INIT.OBJ must be the first input file specified to LINK.

For example:

```
LINK init mymain qlbhead myrtpext mysub,mylib,,ppbe;
```

Linkage of Quick Library

Programs written in Professional Pascal must be specifically linked to produce an executable file. All input files to Microsoft 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 stand-alone should include:

- INIT.OBJ (from SuperDOS Professional Pascal support)
- The mainline(i.e., MYMAIN.OBJ)
- Quick library header (QLBHEAD.OBJ)
- The RTPEXT subroutine (i.e., MYRTPEXT.OBJ)
- The GOSUB' subroutines (i.e., MYSUB.OBJ)
- The Professional Pascal support library (PPBE.LIB) (name may vary)

NOTE: OBJ must be the first input file specified to LINK.

For example:

```
LINK /QUICKLIB /PACKCODE init mymain qlbhead myrtpext my-
sub,mylib,,ppbe;
```

produces "MYLIB.QLB".

NOTE: The /packcode option of LINK is required. This reduces the number of logical code segments produced by the linker.

11.8 Flow Control for External Libraries

The flow of control (in chronological order) for RTP 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 startup routine for the language in use.

3. The C startup routines in the external library execute, and eventually get into the external library mainline (i.e.,main()), which calls RTP() as a subroutine.

4. The RTP subroutine completes NPL startup, 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' (i.e., 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 (i.e.,GOSUB'100 exists, has 3 parameters with types 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, a NPL error is generated on the GOSUB' statement. Otherwise, NPL evaluates parameters and calls the external subroutine (mysub) whose address was provided by RTPEXT.

10. The external subroutine (mysub) executes and returns to NPL. NPL execution proceeds until returning to step 5 (another GOSUB') or the RTP ends ($END, Killed from Help, etc.). In the second case, proceed to the next step.

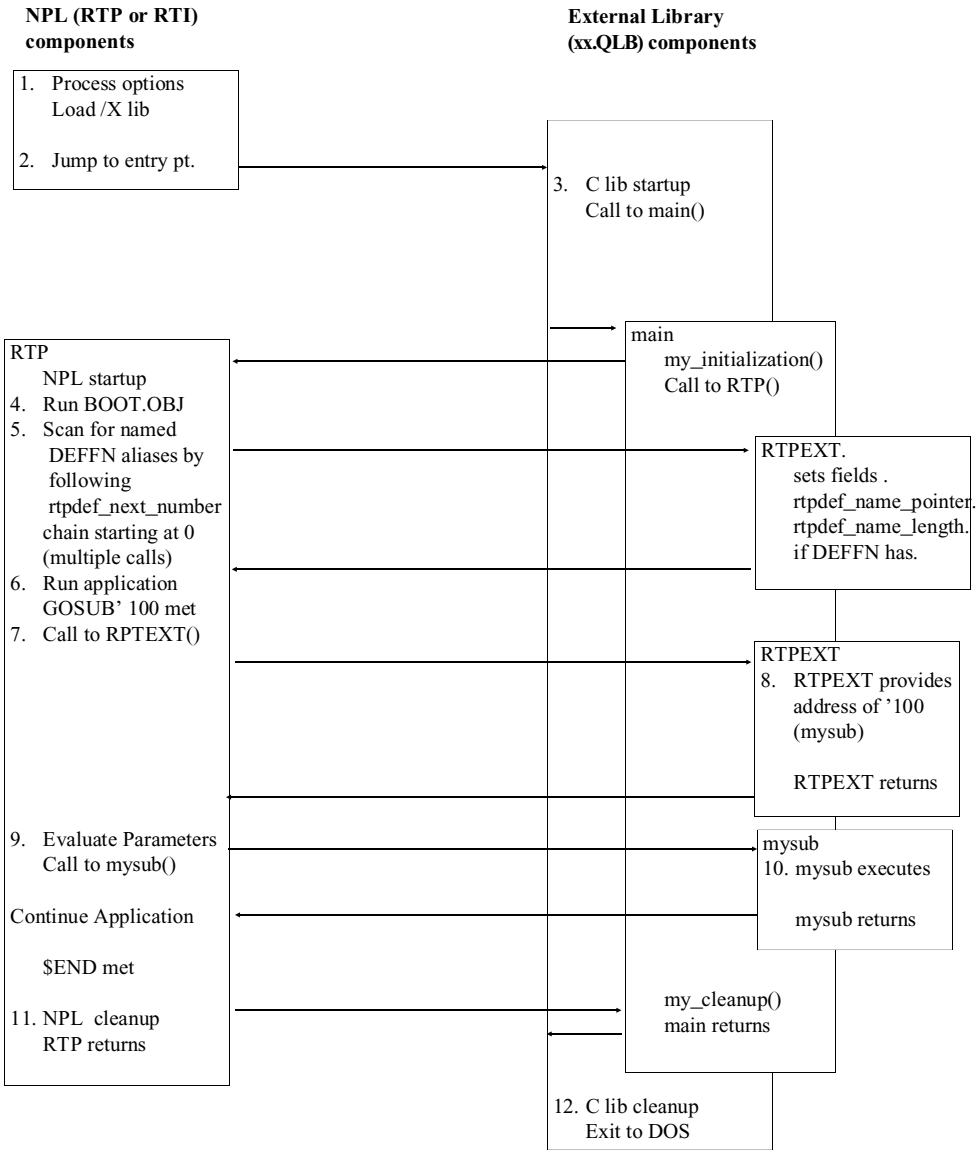
11. NPL does its cleanup, then the RTP subroutine returns to the caller (in the external library mainline).

12. The external library mainline does C library shutdown, and eventually exits back to MS-DOS.

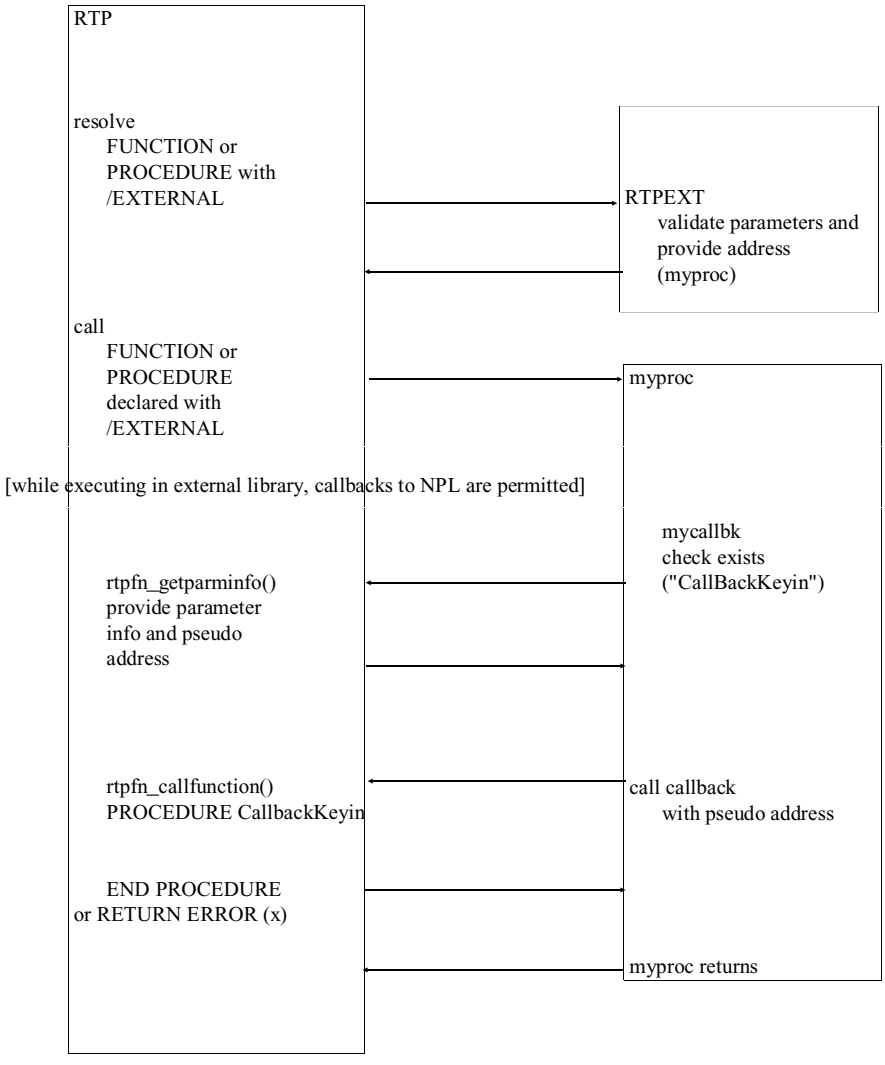
RTPEXT can be called by LIST' to find out information about DEFFN' subroutines, without actually calling the subroutines.

NOTE: The subroutines should not depend on the above flow control order to work (i.e., a subroutine should not expect that RTPEXT is 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:


NPL (RTP or RTI) components
External Library (xx.QLB) components
1. Process options
Load /X lib
2. Jump to entry pt.
3. C lib startup
Call to main()
main
my_initialization()
Call to RTP()
RTP
NPL startup
4. Run BOOT.OBJ
5. Scan for named DEFFN aliases by following rtpdef_next_number chain starting at 0 (multiple calls)
6. Run application
GOSUB' 100 met
7. Call to RPTEXT()
RTPEXT.
sets fields .
rtpdef_name_pointer.
rtpdef_name_length.
if DEFFN has.
RTPEXT
8. RTPEXT provides address of '100 (mysub)
RTPEXT returns
9. Evaluate Parameters
Call to mysub()
mysub
10. mysub executes
mysub returns
Continue Application
$END met
11. NPL cleanup
RTP returns
my_cleanup()
main returns
12. C lib cleanup
Exit to DOS

The following diagram shows the execution flow for the FUNCTION/PROCEDURE (in-derface):



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 gives some explanation to the errors which are most likely to occur:

Code	Meaning
01	Cannot locate the specified filename at all, or read access is not allowed. Is the file name specified properly? Is the file in the correct user group? Is the file type "T"?
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, or is not the correct format.
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 your mainline contains an ORG directive, or if you are 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 SuperDOS operating system.

Section A.2 discusses problems and errors generated by the RunTime.

Section A.3 discusses problems and errors generated by the Compiler.

A.2 Runtime Problems

This section describes specific problems you may encounter in using the RunTime along with possible solutions.

Problem 1: The message "The token processor (run time package) is not resident" appears when trying to invoke either RTP or RTI.

NOTE: This message may be delayed for up to 30 seconds before displaying.

The memory shareable module, RTPSHARE (or RTISHARE) is not properly loaded in the system configuration file or has been stopped using RTISTOP. Check the CONFIG.P file and reinstall or restart, if necessary. Refer to Chapter 2 for installation requirements of RTISHARE or RTPSHARE.

Problem 2: A RunTime error P48 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 specific drive/user group specification (i.e., is simply a file name - PLATTER1.BS2 for example), be sure that the searchlist is established properly in the Password File prior to execution of the RunTime.

If the device being accessed is a 2200 format "raw" diskette, check that the $DEVICE statement references the proper SuperDOS device:

```
$DEVICE(/D10)="1: 360=Y"  or
$DEVICE(/D10)="1: 1.2=Y"
$DEVICE(/D10)="1: 720=Y"
$DEVICE(/D10)="1: 1.4=Y"
```

Problem 3: The RunTime generates an immediate D82 error stating that the boot program cannot be found.

This error means that the RunTime could not find the boot program specified. This could be because:

The drive/user group has been improperly defined. Check the EXEC files used to invoke the RunTime.

The boot program exists, but is specified incorrectly. Either modify the EXEC file or re-name the boot program.

Problem 4: Box graphics do not print.

"True"box graphics are supported under SuperDOS only on the Wang 2X36 DE/DW terminals, or when using certain compatible terminal emulation products on an IBM PC with an appropriate controller. On other serial terminals, including the console, you must use character boxes. Refer to the NPL Statements Guide, $BOXTABLE for details.

Problem 5: The message "Interpreter not enabled." appears.

Use the Non-interpretive version of the RunTime (RTP), or install the ENABLED key file in the 5:0 drive/user group.

Problem 6: The message "Authorized user limit exceeded." appears.

Check to see that the 107 lines installed in the configuration file for NPL are correct.

Problem 7: The terminal displays unusual characters or otherwise does not function properly.

Check that TERMTYPE.TBL properly defines the specific terminal type you are using to the RunTime.

If using a Wang 2236 terminal, be sure that the W2236INI program has been executed by an automatic password before attempting to use the terminal.

If using a Wyse 60, 150 or 160 terminal, be sure to download the fonts first.

If using a Wyse 50 or Wyse 60, 150, 160 terminal, be sure that the XON/XOFF handshake is set ON. This can be done through the Wyse terminal SetUp procedure.

Problem 8: A RunTime error I90 is generated trying to access a "raw" format diskette using SuperDOS 5.0 or later.

Protected mode SuperDOS does not support the use of 320K "raw"format diskettes. Either use another "raw" format or use an earlier NPL version under real mode SuperDOS.

A.3 Compiler Problems

This section describes specific problems you may encounter in using the compiler, along with possible solutions.

Problem 1: The message "File not found - it is not in any directory of the searchlist" appears when trying to invoke B2C.

This problem can result from two possible causes:

The B2C program is not found in any directory of the SuperDOS search list. If so, install the B2C program from the Development Package.

The B2C program is resident in a drive/user group not specified in the SuperDOS search-list. If so, edit the Password file to add the specific drive/user group designation to the searchlist.

Problem 2: The message "Program cannot be run - it is too big for this task" appears when trying to invoke B2C.

A task of at least 196K must be allocated for use with the B2C program. Edit the configuration file to allow for a 196K task.

Problem 3: 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. If you are using a EXEC file to compile programs, be sure that the SRCLOC includes the proper drive/user group designation.

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, and contains a valid drive/user group designation in the SuperDOS searchlist.

APPENDIX B

SUPERDOS ERROR CODES

B.1 Overview

The Error Processor will display a "SuperDOS 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 SuperDOS version of the RunTime Program, this "SuperDOS Error Code" contains the error code returned to the RunTime by SuperDOS.

Section B.2 discusses SuperDOS error codes.

B.2 Error Codes

Following are some of the more frequently encountered SuperDOS error codes and their meanings:

23 - File not found
Check the $DEVICE specification for a correctly specified filename.

37 - Action prohibited by protect flags
Check security level established in Password File.

38 - Invalid user group
Check user group access in Password File.

If the help file RTIIERR.HLP AND RTIIERR.IDX are installed, the RunTime automatically accesses these files and displays a literal message describing the SuperDOS error that has occurred. In addition, the SuperDOS Error Code can be examined under program control by use of the $OSERR statement. Refer to the $OSERR statement in the NPL Statements Guide.

Refer to the SuperDOS Guide to Operations, for a complete listing and descriptions of all SuperDOS error codes.

APPENDIX C

PERFORMANCE ISSUES

C.1 Overview

The purpose of this Appendix is to provide suggestions on ways to manage performance when using SuperDOS.

Section C.2 discusses the SuperDOS RAM disk option.

Section C.3 discusses cache buffers.

Section C.4 discusses memory.

Section C.5 discusses the SuperDOS print spooler.

C.2 RAM Disk

SuperDOS supports an optional RAM Disk configuration. Performance can be significantly enhanced by implementing this feature, especially in the areas of program loading and record locking. Refer to the SuperDOS Guide to Operations and Chapter 3 for details.

C.3 Cache Buffers

Configuring the SuperDOS system for the maximum number of cache buffers will significantly increase overall system performance. However, write caching presents a reliability issue. Refer to the SuperDOS Guide to Operations and Section 5.3 for details.

C.4 Memory

Increasing the amount of system memory can allow for increase in performance be allowing more memory to be configured for a RAM DISK. Refer to Section C.2 and the SuperDOS Guide to Operation for details.

C.5 Print Spooler

The Print Spooler feature of SuperDOS is used to increase productivity for individual users and the system as a whole by reducing the time spent waiting for printers to finish or become available for use. Refer to the SuperDOS Guide to Operation for details.

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(6)	-	X(4)	-	-	-	-	-
	720K	X(6)	-	-	-	-	X(4)	-	-
	1.2MB*	X(6)	-	X(4)	-	X(4)	-	-	-
	1.44MB	X(6)	-	-	-	-	X(4)	X(4)	-
	2.88MB	-	-	-	-	-	-	-	-
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 nor 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. Under Release IV, SuperDOS, 320K "raw" formatted diskettes are not supported under B2C.

(5) $FORMAT DISK is not supported under Intel UNIX or Xenix. Refer to the Intel UNIX Supplement for details.

(6) $FORMAT DISK for 3-1/2" or 5-1/4" 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 requires Release 3.00 or higher.

(9) Support of 3.5" 2.88MB diskettes requires 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.