NORTH STAR PASCAL SYSTEM REFERENCE MANUAL, REVISION 2

ADDENDUM REVISION A, (JUNE, 1979)

(describes PAS-PRI-S, PAS-AUX-S, PAS-PRI-D, PAS-AUX-D)

TABLE OF CONTENTS

I. INTRODUCTION.....1

II. THE PRIMARY SYSTEM: PAS-PRI-S, PAS-PRI-D.....3
  A. Introduction.....3
    1. The Bootstrap Diskette, PASNS:.....3
    2. The Compiler Diskette, COMP:.....5

  B. Personalization of Pascal.....7
    1. CREATING THE WORKING DISKETTES.....7
    2. I/O PERSONALIZATION OF THE P-MACHINE SIMULATOR....7
      a. Getting Started.....7
      b. The User-Area JUMP TABLE.....9
      c. Sample Input/Output Routines.....12
      d. Installing the Input/Output Routines.....16

    3. CHOICE OF SIMULATORS.....17
    4. CONSOLE TERMINAL CONFIGURATION.....17
    5. CHOICE OF EDITORS.....19

III. IMPLEMENTATION NOTES.....21
  A. Pascal Disk Blocks.....21
  B. Program Development with Single Density.....21
  C. Terminals with Short Line Lengths.....24
  D. Binary Arithmetic.....24
  E. Unused Drives.....24
  F. Data Transfer Between Single-Density and
      Double-Density Diskettes.....24

IV. THE AUX PACKAGE: PAS-AUX-S, PAS-AUX-D.....28
V. A BRIEF PASCAL BIBLIOGRAPHY.....30

I. INTRODUCTION

The programming language Pascal was designed to encourage the use of modern "structured, modular programming" techniques among computer science students in academic situations, but has nevertheless become popular in professional environments as well. Both this ADDENDUM and its parent document, the NORTH STAR PASCAL SYSTEM REFERENCE MANUAL, assume that the reader is familiar with the Pascal language itself, and so concentrate on the particular use and operation of the North Star Pascal system in preparing and executing programs. For those who need to learn more about programming in Pascal, a bibliography of suggested references is included at the end of this ADDENDUM.

North Star Pascal, Version 1, is an implementation of the "UCSD Pascal" Version I.5/II.0 software system, as configured for operation on 8080 and Z80 microcomputer systems equipped with North Star's MICRO DISK SYSTEM (MDS), including the North Star HORIZON computer. The software system was created at the Institute for Information Systems, University of California, San Diego, and the North Star version is being maintained jointly by the Institute and North Star Computers, Inc.

Version 1 is essentially the complete UCSD Pascal system, containing all tools necessary for program development using the Pascal language: a text editor, Pascal compiler, object code linker, 8080 and Z80 assemblers to facilitate the interface of machine-code routines to Pascal code, and several utilities. The system operates independently of North Star's DOS, and includes its own file-handling and disk-management functions. However, it contains minor changes from the standard UCSD system, designed to make operation more convenient when 5-1/4" floppy disks are used for mass-storage of programs and data. This ADDENDUM discusses the relevant changes and their effects on system operation as described in the SYSTEM REFERENCE MANUAL. Also discussed in this ADDENDUM are the procedures necessary for interfacing North Star Pascal to an arbitrary input/output configuration, and suggested program development strategies for single-density and double-density systems.

HARDWARE REQUIREMENTS

In order to run the North Star Pascal system, your computer must meet the following standards:

1. Its CPU must be one of the "8080 family" of processors, which includes the 8080, Z80, and 8085.

2. It must be equipped with a single-density or double-density North Star MDS Micro-Disk-System. The single-density version of the Pascal system, PAS-S, supports up to three disk drives, and the double-density verison, PAS-D, supports a maximum of four. Note that program development under the single-density version of North Star Pascal requires at least two disk drives on-line. Using the double-density version, program development may be pursued with only one drive on-line, but dual-drive operation is far more convenient, and for that reason is recommended by North Star.

3. The bootstrap PROM on the North Star MDS Controller Board must be located at E800H in memory. (For single-density controllers only, the PROM must be configured to load the North Star DOS into RAM starting at 2000H.)

4. Program development using the Pascal compiler requires at least 48K contiguous RAM, and 56K is recommended because the compiler will run faster, larger assembly-language programs may be assembled and linked together, etc. Once compiled, applications programs may run in as little as 20K of RAM, depending upon their individual storage requirements.

A single-density or double-density dual-drive North Star HORIZON with Z80 processor and 48K RAM starting either at 0000H or 2000H is sufficient for program development using the Pascal system.

Before reading further in this ADDENDUM, read the entire NORTH STAR PASCAL SYSTEM REFERENCE MANUAL at least once to familiarize yourself with general system features and concepts. Return to the ADDENDUM before attempting to use the system, for detailed information about system configuration and operation in the North Star environment.

Every effort has been made to insure that the information presented here is accurate and complete at press time. If you find any errors or omissions, please notify North Star in writing at the following address:

NORTH STAR COMPUTERS
ATTN PRODUCT DOCUMENTATION DEPT.
2547 NINTH STREET
BERKELEY CA 94710

II. THE PRIMARY SYSTEM: PAS-PRI-S, PAS-PRI-D

A. Introduction

The North Star Pascal program development system is available in two parts, the PRIMARY package and the AUXILIARY package. The PRIMARY package contains all the software necessary to permit program development using Pascal. The AUXILIARY package includes the two assemblers and several utilities intended for use by advanced programmers.

This section describes the PRIMARY package, which comes in two versions, PAS-PRI-S for single-density disk operation, and PAS-PRI-D, for double-density operation. Note that the single-density version works only with North Star's first, single-density-only disk controller. The double-density version (which will not read or write single-density disk information) will work only with the newer (current model) double-density controller. System software for PAS-PRI-S is divided between two (2) 5-1/4" single-density floppy diskettes, named PASNS: and COMP:. All system software for PAS-PRI-D fits on one double-density diskette, PASNS:. With either system, PAS-PRI-S or PAS-PRI-D, you receive a copy of the NORTH STAR PASCAL SYSTEM REFERENCE MANUAL and a copy of this ADDENDUM as documentation.

The following discussion describes the contents of the PASNS: and COMP: diskettes in the PAS-PRI-S single-density system. For users of the PAS-PRI-D double-density system, note that all system software described as being on the single-density PASNS: and COMP: diskettes is contained on the one double-density PASNS: diskette in the PAS-PRI-D package. Because of the greater information storage capacity of double-density diskettes, it is not necessary to have a separate COMP: diskette for PAS-PRI-D.

1. The BOOTSTRAP diskette, PASNS:

The BOOTSTRAP diskette for either PAS-PRI-S or PAS-PRI-D is marked

PASNS:

This diskette contains the operating system software and other programs which will be described shortly. The bootstrap diskette must always be inserted in drive #1 (this corresponds to "device #4" in UCSD Pascal teminology) whenever you "bring-up" the system. (This goes for both single-density and double-density systems!) Sometimes, you may have occasion to remove the bootstrap diskette from the device #4 drive during the execution of certain programs or system functions. Under normal operating conditions, the system will remind you to re-insert the bootstrap diskette after the program or function is done. You will see the following words on your terminal:

PUT IN PASNS:

The device #4 drive will turn on, and the message will be repeated until the appropriate diskette has been inserted and the drive door closed. Do not be afraid of re-inserting the bootstrap diskette while the device #4 drive is operating, as it will not harm the diskette.

The bootstrap diskette contains the following software files:

SYSTEM.NSTAR2
SYSTEM.NSTAR0

These are the "p-machine simulator" programs, and are the heart of the North Star Pascal system. All Pascal programs are compiled into "p-code" which is a machine code for a pseudo-microprocessor that is ideal for executing Pascal programs. In order for this code to run on typical microprocessors, a program which SIMULATES the "p-machine" must be written for and executed on the computer. This program makes the processor on which it runs appear to be the ideal p-machine. All system software in the Pascal system, with the exception of the p-machine simulator itself and low-level I/O drivers, is in the form of p-code, so one of the sumulators must always be running when you use the system. SYSTEM.NSTAR2 expects system RAM to begin at 2000H, and loads there, while SYSTEM.NSTAR0 loads at 0000H. Except for their origins in memory, SYSTEM.NSTAR2 and SYSTEM.NSTAR0 are identical. While the system is initially configured to use only the 2000H-based simulator, you may re-configure it to use the 0000H-based simulator instead. The procedure for this is described elsewhere in this ADDENDUM. Note that the selected simulator is automatically loaded and executed whenever you "bring up" the Pascal system, and that the standard North Star bootstrap PROMs are able to bootload either the 0000H-based simulator or the 2000H-based one.

SYSTEM.PASCAL

This is the Pascal operating system, and includes the command processor with which you interact to initiate editing, compiling, linking, and execution of your programs. The operation of this system is explained in detail in the SYSTEM REFERENCE MANUAL.

SYSTEM.FILER

This is a separate part of the operating system, which allows you to maintain data and program files on diskette. It is entered through the operating system's F(iler command. See section 1.2 of the SYSTEM REFERENCE MANUAL for operating details.

SYSTEM.LIBRARY

This is a collection of special routines which may be linked into your Pascal programs after they are compiled and before they are executed. See the section 1.8 (LINKER), and section 3.3.2 of the SYSTEM REFERENCE MANUAL for more information about the SYSTEM.LIBRARY. Note that this file must always remain on the bootstrap diskette. If it is removed, it may not be possible to prepare certain Pascal programs for execution, especially those which involve input/output of real numbers, since the routines which accomplish this must be linked into compiled programs from the SYSTEM.LIBRARY before execution.

SETUP.CODE

SETUP is an interactive program which permits you to re-configure your Pascal system to observe the screen control conventions of your particular console terminal. See section 4.3 of the SYSTEM REFERENCE MANUAL for further information.

SYSTEM.MISCINFO

This data file contains configuration information for your system, including much of that which permits cursor-controlled operation on video terminals. The SYSTEM.MISCINFO file supplied with your Pascal system makes the system treat your terminal as if it has no cursor-control (except for standard carriage-return and linefeed functions). Single character input deletion is accomplished by striking the underline (_) key, and deletion of an entire input line occurs when the terminal's at-sign (@) key is pressed. You will need to use the SETUP and BINDER programs in order to adjust your system for convenient video screen operation with full cursor-control, screen and line clear functions, backspacing, etc.

SOROC.MISCINFO

For those whose console terminal is a SOROC IQ120, use of this special MISCINFO file will re-configure your system to use the full cursor-control capabilities of the terminal. In particular, striking the left-arrow vector key will delete a single input character at a time, backspacing and erasing that character, while the RUB key must be depressed to cancel an entire input line, erasing that line from the screen. See the PERSONALIZATION section in this ADDENDUM for complete details.

2. The compiler diskette, COMP:

The second diskette in the (single-density) package is marked

COMP:

and contains, among other programs, the Pascal compiler. You will use this diskette when you wish to compile a Pascal program. (Remember that there is no "second diskette" in the double-density system. All software listed under COMP: is included on PASNS: in PAS-PRI-D.)

The COMP: diskette includes the following software:

SYSTEM.EDITOR

The SYSTEM.EDITOR is initially the screen-oriented editor described in section 1.3 of the SYSTEM REFERENCE MANUAL. If you do not have a cursor-controlled video terminal, you should make YALOE.CODE (a conventional line-oriented editor) into the SYSTEM.EDITOR. See the PERSONALIZATION section of this ADDENDUM for more information on your choice of text editors.

SYSTEM.COMPILER

This is the Pascal to p-code compiler as described in section 1.6 of the SYSTEM REFERENCE MANUAL.

BOOTER.CODE

BOOTER is a program which copies the special bootstrap code from the bootstrap disk to another copy of the bootstrap disk. Bootstrap code is contained in several "invisible" Pascal disk blocks which are not accessible through or indicated by the disk directory. When the F(iler's T(ransfer function is used to copy the contents of the bootstrap diskette over to another disk, the "invisible" blocks are NOT copied, since they are not reflected in the directory. Note that, at present, only North Star's DOS can duplicate an entire diskette regardless of its contents, but that the DOS cannot be invoked in the midst of a Pascal session. In order to copy the bootstrap diskette using Pascal, it is necessary to use the T(ransfer function to copy all files in the directory to the new disk, then X(ecute the BOOTER program which then transfers the "invisible" blocks. See section 4.4 in the SYSTEM REFERENCE MANUAL for more information.

BINDER.CODE

This program injects a special procedure, GOTOXY, into the Pascal operating system in order to facilitate cursor-controlled operation. You will need to use this program if your video terminal is NOT a Lear Siegler ADM-3A or a SOROC IQ120. Before you can use BINDER, you must write and compile a version of the GOTOXY procedure which is appropriate for your terminal. BINDER will then make the resulting code file a part of the system. For examples on how your GOTOXY procedure should be written, see section 4.7 of the SYSTEM REFERENCE MANUAL. Note that this procedure must be compiled using the {$U-} compile-time option, which is explained in section 1.6.1 of the SYSTEM REFERENCE MANUAL. Also, your version of GOTOXY may NOT itself be called "GOTOXY", but may be referenced by that name once it has become part of the Pascal system.

SYSTEM.LINKER

Pascal programs may be compiled separately, and then may be linked together, before being executed, to form new software packages. Machine-code routines may also be linked into Pascal code before execution. This makes it possible to have "libraries" of often-used routines which may be linked into compiled Pascal programs whenever necessary. See sections 1.8 and 3.3.2 of the SYSTEM REFERENCE MANUAL for more information.

YALOE.CODE

YALOE is "Yet Another Line Oriented Editor" and is intended to be used as the SYSTEM.EDITOR when no cursor-controlled video display is available as console device. See the PERSONALIZATION section of this ADDENDUM for the procedure which must be followed to convert YALOE.CODE to SYSTEM.EDITOR if the screen-oriented editor is inappropriate for your system. Information about YALOE is contained in the REFERENCE MANUAL, section 1.4.

B. Personalization of Pascal

NOTE: The personalization process described here assumes your familiarity with the North Star DOS and MONITOR, which should have been supplied as standard software with your HORIZON computer or MICRO-DISK-SYSTEM. You will need to use the DOS and MONITOR to effect personalization of your Pascal system. If you are not familiar with the DOS or MONITOR, refer to the appropriate sections of the NORTH STAR SYSTEM SOFTWARE MANUAL for further details.

1. CREATING THE WORKING DISKETTES

The Pascal diskettes you receive from North Star are write-protected. They are your FACTORY MASTERS, and YOU SHOULD NEVER ATTEMPT TO WRITE DATA ON THEM, OR TO REMOVE THE WRITE-PROTECT TAB. The factory masters should be used to create a set of WORKING DISKETTES, which will remain unprotected, and which will be used in routine operation of the system. Upon receipt of your Pascal factory masters, create a set of working diskettes by using the CD command (or utility) in the North Star DOS to copy the contents of each factory master onto a new diskette. Be sure to copy the label information for each factory master onto the label of the appropriate duplicate diskette. Then, retire the write-protected factory masters to a safe place where they may be kept until needed to generate more duplicates. Official warranty policy for North Star Pascal requires return of your factory masters before warranty replacement or update is possible, so RETAIN YOUR FACTORY MASTERS IN THEIR ORIGINAL CONDITION!

2. I/O PERSONALIZATION OF THE P-MACHINE SIMULATOR

a. Getting Started

The standard Pascal system, as shipped, is pre-configured to operate on a HORIZON computer system, using the standard serial port as CONSOLE: device, and the second serial port as both PRINTER: and REMOUT:. (Note that the REMOTE: device described in the SYSTEM REFERENCE MANUAL has been changed to two devices, REMOUT:, which is device #8, and REMIN:, which is device #7. Only REMOUT: is available in North Star Pascal, Version 1.)

If you have a HORIZON as described above, you may bootload the system without any modifications being necessary. In this case, skip to the CONSOLE TERMINAL CONFIGURATION section. However, if your computer does not follow the HORIZON's input/output conventions, you will need to "personalize" your system to use your particular I/O devices.

As implied before, the North Star Pascal p-machine simulator has been written in 8080 machine code compatible with 8080, 8085 and Z80 machines. The first 1.5K of the simulator is devoted to device input/output code, but the first 1K of that contains mostly routines which handle disk I/O. The "User I/O Area" of the Pascal system begins at SYSORG+400H, where SYSORG is the origin of the simulator, 0000H for SYSTEM.NSTAR0, and 2000H for SYSTEM.NSTAR2. Thus, the user area begins at 400H for SYSORG=0000H and 2400H for SYSORG=2000H. Within this area, you have 467 bytes into which you may write I/O drivers for your particular console, printer, etc.

You should boot the North Star DOS now, and insert the working copy of the PASNS: diskette into drive #2 (Pascal device #5). Use the LI 2 command to get a listing of the diskette directory. You should see the following:

```
PASNS:      0    4 S    0
V1.R1.SD    0    0 S    0
PASCAL      0    0 S    1  E800
USERIO.2 xxx     2 S    1  2400
USERIO.0 xxx     2 S    1   400
```

Note that the above illustrates the PASNS: directory for the single-density system, PAS-PRI-S. For PAS-PRI-D, the entry "V1.R1.SD" would be "V1.R1.DD", and the column of "S"s, denoting single-density, would be a column of "D"s, for double-density. Otherwise, the North Star format directories for the two PASNS: diskettes are the same. In place of "xxx" in the listing will be the actual North Star disk addresses where the USERIO files happen to be on the factory master diskette. The Pascal system itself does not create, maintain, or use the North Star format diskette directory. North Star provides DOS format directories on its Pascal system diskettes for your convenience only. The presence of these directories alerts DOS users to the fact that diskettes bearing them are Pascal diskettes and are not to be used under DOS (except in the cases of disk initialization, duplication, or personalization, as described below).

Another reason for including a DOS format directory on Pascal system diskettes (especially the bootstrap diskette) is to facilitate personalization of Pascal under DOS. Notice the two "files", USERIO.2 and USERIO.0. The areas on the diskette named by these files correspond to the USER I/O areas for the simulators SYSTEM.NSTAR2 and SYSTEM.NSTAR0, respectively. For either simulator to run on your computer system, the low-level I/O routines contained in its USER I/O area must be appropriate for your computer configuration. Note that changing one version of the simulator so that it will run on your system will not change the other. For both SYSTEM.NSTAR2 and SYSTEM.NSTAR0 to run on your system, you will have to configure both USERIO files to reflect the I/O requirements of your computer and peripheral devices.

As shipped from the factory, the PASNS: bootstrap diskette is configured to bootload SYSTEM.NSTAR2 and ignore SYSTEM.NSTAR0. Thus, you must make certain to modify the DOS "file" USERIO.2 before you can bring the system up. If you intend to use the 0000H-based SYSTEM.NSTAR0, you should also modify USERIO.0 as well. To load a user I/O area into RAM using North Star DOS, use one the following commands, depending on which area you wish to manipulate:

```
LF USERIO.2,2 xxxx
```

or

```
LF USERIO.0,2 xxxx
```

The above assumes that the bootstrap diskette is in the secondary drive (Pascal device #5). The actual address in RAM memory where you wish the code to be loaded should be substituted for the "xxxx" shown above. This address must be in hexadecimal. For example, to load the user I/O area for the 2000H-based simulator from disk into RAM starting at location 4400H, type:

```
LF USERIO.2,2 4400
```

For sake of discussion, it will be assumed that the user area has been loaded into 4400H during all personalization steps described here.

Now, use the North Star MONITOR to modify the user area so that it contains routines appropriate to your computer's configuration. (To get a feel for the type of things you will be doing, see Chapter G, "INSTALLING THE INPUT/OUTPUT ROUTINES", of the GETTING STARTED section of the North Star SYSTEM SOFTWARE MANUAL. This process is roughly analogous to, but NOT THE SAME as what you must do to install I/O routines into Pascal.) Remember that the I/O routines will actually begin in memory at the origin of the simulator + 400H.

b. The User-Area JUMP TABLE

You may place your I/O routines at any arbitrary locations in the user area, as long as the USER AREA JUMP TABLE reflects the locations you choose. The jump table is a 45-byte portion at the very beginning of the user area. The 467-bytes reserved for user I/O routines follow the jump table in memory, and together, they occupy a 512-byte block in RAM, which corresponds to two contiguous North Star disk blocks (256 bytes per North Star block).

The jump table contains 15 sequential 8080 JMP instructions, each one corresponding to a different routine in the user area. The first byte in each 3-byte JMP instruction is usually a C3H, corresponding to an 8080 JMP. The next two bytes give the location of the routine itself, and it is this pair of bytes which must be changed to conform to the location you choose for a given routine. Note that a correct jump table must be present at the beginning of ANY user I/O area, or low-level I/O functions will fail and the system will crash.

Below is a description of the jump table, along with specifications you must follow when writing the corresponding I/O routines. Note that all routines are responsible for returning to any code which calls them. For all but one of the routines, this should be done by executing one of the RET family of instructions. The case of NSMSIZ is unique, and is described in detail.

SYSORG+400H: JMP CONONL

This routine is called frequently to determine whether or not the CONSOLE: device is on-line or off-line. If the CONSOLE: is available, CONONL must return a 00H in the accumulator, otherwise 09H should be returned. No registers except the accumulator may be modified. (In this

and all routines described here, the condition flags need NOT be saved or restored.) Two other routines, PTRONL and REMONL, perform on-line status reporting for the PRINTER: and REMOUT: devices, respectively, and must adhere to the same specifications as CONONL.

SYSORG+403H: JMP CONINP

CONINP waits until a character is available from the CONSOLE: device, then returns it in the accumulator. No registers except the accumulator may be modified.

SYSORG+406H: JMP CONOUT

CONOUT waits until a character may be sent to the CONSOLE:, then puts out the contents of the C register. No registers except the C register and accumulator may be modified.

SYSORG+409H: JMP CONST

This routine checks input status of the CONSOLE: device. If a character is ready, the value FFH (TRUE) must be returned in the accumulator. If no character is ready, 00H (FALSE) should be returned in the accumulator. No registers except the accumulator may be modified.

SYSORG+40CH: JMP PTRONL

Reporting of on-line status for device #6 (PRINTER:) is done here. See CONONL (SYSORG+400H) for specifications.

SYSORG+40FH: JMP PTRINP

PTRINP waits for an input character to be available from the PRINTER:, and then returns the character in the C register. (This is useful for implementing buffered-printer protocol schemes, etc.) The value 00H must be returned in the accumulator (indicating a successful I/O operation). No registers other than C and the accumulator may be modified.

SYSORG+412H: JMP PTROUT

A single character, the contents of the C register, is sent to the PRINTER: device. The value 00H is returned in the accumulator to denote a successful I/O operation. No registers other than C and the accumulator may be modified.

SYSORG+415H: JMP REMONL
SYSORG+418H: JMP REMINP
SYSORG+41BH: JMP REMOUT

These are, respectively, on-line status reporting for, character input from, and character output to the REMOUT: device (Pascal system device #8). The specifications are the same as PTRONL, PTRINP, and PTROUT, respectively, except that a different device is accessed by these routines.

NOTE: The JMP instructions for PTRINP, and PTROUT may be replaced in the jump table by RET instructions, provided that PTRONL always returns 09H (device off-line) in the accumulator. PTRINP and PTROUT will not be called by the system unless PTRONL reports that the device is on-line. The same thing holds for REMONL, REMINP, REMOUT.

SYSORG+41EH: JMP NSMSIZ

This routine returns a 16-bit value on the stack which corresponds to the highest contiguous WORD (16-bit) location in Pascal system RAM. The routine which is supplied with the standard Pascal system as shipped from the factory searches the computer's memory space to find this upper limit, and considers that limit be reached when it ecounters no memory, ROM, or write-protected RAM. Note that if memory-mapped I/O devices (such as video display boards, etc.) are contiguous with system RAM, the standard "memory sizing" routine will consider them as part of available RAM! If this is the case in your system, you will have to install an alternate memory sizing routine, such as one which returns a constant value as the upper limit.

Remember that the value returned must point to an EVEN address (word boundary). For example, if system RAM extends to BFFFH, the value returned by NSMSIZ should be BFFEH. All registers may be used by this routine. The technique used for RETurning with the stack pointer pointing at the appropriate value is to load the value in the HL register pair, do an XTHL (exchange the value at the top of the stack -- assumed to be the RETurn address -- with the value in HL), then execute a PCHL instruction, which is a JMP to the location represented by the value in HL. If you do not use this technique, your memory sizing routine may crash the Pascal system. (See SAMPLE INPUT/OUTPUT ROUTINES for an example of this technique in use.)

SYSORG+421H: JMP NSCLOK

The real-time clock option of the Pascal system is not yet available. For now, this routine should RETurn, with the "not-on-line" value of 09H in the accumulator. No other registers may be modified.

SYSORG+424H: JMP MACINT

This routine is called by the Pascal system at bootstrap-load time just before the memory sizing routine is called, and provides for one-time machine and I/O device initialization each time the Pascal system is re-booted. For example, the HORIZON motherboard is initialized, memory-parity is enabled, and the two serial I/O ports are reset in the MACINT provided in the user area of both p-machine simulators on the factory master diskette. All registers may be used.

SYSORG+427H: JMP NSOFTEN
SYSORG+42AH: JMP NSXTRA

These jump-vectors are reserved for future expansion, and are implemented as "jump-to-self" loops in the user-area of the standard Pascal release.

c. Sample Input/Output Routines

The following is an assembler listing of the HORIZON user-area routines supplied on the standard Pascal release bootstrap diskette. You may use them as models in writing your own personalization routines.

```
0000|                              .PROC HRZIO
Current memory available:    9144
0000|
0000|                         ; ***********   User Area Routines for N* Horizon   ***********
0000|
0000|                              .ORG    1024.
0400|
0400|                         ; Define Constants
0400|
0400| 00FF                    TRUE    .EQU    0FFH
0400| 0000                    FALSE   .EQU    00
0400| 0400                    NSJTST  .EQU    $               ; Start of N* jump table.
0400| 0600                    STRTSR  .EQU    NSJTST+512.     ; Memory-sizing search will
0400|                                                          ;  start here.
0400|
0400| 0009                    NOTRDY  .EQU    9               ; Device off-line code.
0400| 00E8                    DCTRLB  .EQU    0E8H            ; High byte of disk controller
0400|                                                          ;  address.
0400|
0400|                         ; Jump Table
0400|
0400| C3 ****                        JMP     CONONL
0403| C3 ****                        JMP     CONINP
0406| C3 ****                        JMP     CONOUT
0409| C3 ****                        JMP     CONST
040C|
040C| C3 ****                        JMP     PTRONL
040F| C3 ****                        JMP     PTRINP
0412| C3 ****                        JMP     PTROUT
0415|
0415| C3 ****                        JMP     REMONL
0418| C3 ****                        JMP     REMINP
041B| C3 ****                        JMP     REMOUT
041E|
041E| C3 ****                        JMP     NSMSIZ
0421|
0421| C3 ****                        JMP     OFFLIN          ; NSCLOK is off line.
0424|
0424| C3 ****                        JMP     MACINT          ; Machine initialization.
0427|
0427|                         ; The following are reserved for future expansion:
0427|
0427| C3 2704                 NSOFTN  JMP     NSOFTN
042A| C3 2A04                 NSXTRA  JMP     NSXTRA
042D|
042D|                         ; Console drivers
042D|
 401* 2D04
042D| 042D                    CONONL  .EQU    $               ; CONSOLE: on-line?
042D| AF                             XRA     A               ; Return device on-line.
042E| C9                             RET
042F|
```

```
0404* 2F04
042F| DB 03          CONINP  IN      3               ; Check status.
0431| E6 02                  ANI     2
0433| CA 2F04                JZ      CONINP          ; Loop on no character.
0436| DB 02                  IN      2
0438| E6 7F                  ANI     7FH             ; Return character in acc.
043A| C9                    RET
043B|
0407* 3B04
043B| DB 03          CONOUT  IN      3               ; Check status.
043D| E6 01                  ANI     1
043F| CA 3B04                JZ      CONOUT          ; Loop on not ready.
0442| 79                    MOV     A,C
0443| D3 02                  OUT     2               ; Output character in reg C.
0445| C9                    RET
0446|
040A* 4604
0446| DB 03          CONST   IN      3               ; Check status.
0448| E6 02                  ANI     2
044A| CA ****                JZ      $01
044D| 3E FF                  MVI     A,TRUE          ; Input is ready.
044F| C9                    RET
044B* 5004
0450| 3E 00          $01     MVI     A,FALSE         ; No character avail.
0452| C9                    RET
0453|
0453|
0453|                       ; Printer driver
0453|
040D* 5304
0453| 0453           PTRONL  .EQU    $               ; PRINTER: on-line?
0453| AF                    XRA     A               ; Return device on-line.
0454| C9                    RET
0455|
0410* 5504
0455| DB 05          PTRINP  IN      5               ; Character input.
0457| E6 02                  ANI     2
0459| CA 5504                JZ      PTRINP
045C| DB 04                  IN      4
045E| 4F                    MOV     C,A             ; Return character in C.
045F| AF                    XRA     A
0460| C9                    RET
0461|
0413* 6104
0461| DB 05          PTROUT  IN      5               ; Character output.
0463| E6 01                  ANI     1
0465| CA 6104                JZ      PTROUT
0468| 79                    MOV     A,C
0469| D3 04                  OUT     4
046B| AF                    XRA     A
046C| C9                    RET
```

```
046D|
046D|                          ; Remote driver
046D|
0416* 6D04
046D| 046D                     REMONL  .EQU    $               ; REMOUT: on-line?
046D| AF                              XRA     A               ; Return device on-line.
046E| C9                              RET
046F|
0419* 6F04
046F| DB 05                   REMINP  IN      5               ; Character input.
0471| E6 02                          ANI     2
0473| CA 6F04                        JZ      REMINP
0476| DB 04                          IN      4
0478| 4F                              MOV     C,A             ; Return character in C.
0479| AF                              XRA     A
047A| C9                              RET
047B|
041C* 7B04
047B| DB 05                   REMOUT  IN      5               ; Character output.
047D| E6 01                          ANI     1
047F| CA 7B04                        JZ      REMOUT
0482| 79                              MOV     A,C
0483| D3 04                          OUT     4
0485| AF                              XRA     A
0486| C9                              RET
0487|
0487|                          ; Vector to this routine if a given device isn't
0487|                          ;  available.  ALL I/O drivers (input, output,
0487|                          ;  initialization and status) should use this
0487|                          ;  when a device is off-line.
0487|
0422* 8704
0487| 3E 09                   OFFLIN  MVI     A,NOTRDY
0489| C9                              RET
048A|
048A|                          ; Dynamic Memory Sizing
048A|                          ; Note sequence for returning to calling routine:
048A|                          ;       XTHL
048A|                          ;       PCHL
048A|                          ; It's crucial that a return is done this way,
048A|                          ;  else system will probably crash.  When this
048A|                          ;  return sequence is executed, HL must be hold-
048A|                          ;  ing pointer to last WORD of contiguous memory
048A|                          ;  available.
048A|
041F* 8A04
048A| 21 0006                 NSMSIZ  LXI     H,STRTSR        ; Start sizing
048D| 2E FF                          MVI     L,0FFH          ;  on 256 boundary-1.
048F| 7E                       $01     MOV     A,M             ; Get current contents of
0490| 47                              MOV     B,A             ;  test loc and save it.
0491| 2F                              CMA                     ; Complement A.
0492| 77                              MOV     M,A             ; Stuff it back in.
0493| BE                              CMP     M               ; Is it the same ?
0494| C2 ****                        JNZ     ENDSRCH         ;  No: then found end.
0497| 70                              MOV     M,B             ;  Yes: Put byte back.
0498| 24                              INR     H               ; Move on to deader pastures.
0499| C2 8F04                        JNZ     $01             ; Stop when we wrap around.
```

```
0495* 9C04
049C| 25                    ENDSRCH DCR     H               ; Go back to last memory bank.
049D| 2D                            DCR     L               ; Point to last memory WORD.
049E| E3                            XTHL                    ; Put onto stack,
049F|                                                       ;  get return address,
049F| E9                            PCHL                    ;  and return.
04A0|
0425* A004
04A0| 04A0                MACINT  .EQU    $               ; Initialize HORIZON machine.
04A0| F5                            PUSH    PSW
04A1| E5                            PUSH    H
04A2| AF                            XRA     A
04A3| D3 06                        OUT     6               ; Does the motherboard.
04A5| 3E 40                        MVI     A,40H           ; Parity enabled on RAM.
04A7| D3 C0                        OUT     0C0H
04A9| 21 00EC                      LXI     H,DCTRLB*100H+1024.
04AC|                                                       ; Disc controller board addr
04AC|                                                       ;  + 1K.
04AC|
04AC| 04AC                RAMINT  .EQU    $               ; Initialize HORIZON RAM.
04AC| 7E                          MOV     A,M
04AD| 77                          MOV     M,A
04AE| 2C                          INR     L
04AF| C2 AC04                     JNZ     RAMINT
04B2| 24                          INR     H
04B3| 7C                          MOV     A,H
04B4| FE E8                      CPI     DCTRLB
04B6| C2 AC04                     JNZ     RAMINT
04B9| 3E 41                      MVI     A,41H
04BB| D3 C0                      OUT     0C0H
04BD|
04BD|                          ; Now initialize serial ports/USARTS
04BD|
04BD| 3E 03                      MVI     A,3             ; 2 stop bits, 16*clock,
04BF|                                                       ;  8 data bits, no parity.
04BF| D3 03                      OUT     3               ; 1st serial port.
04C1| D3 05                      OUT     5               ; 2nd serial port.
04C3| 3E 40                      MVI     A,40H
04C5| D3 03                      OUT     3
04C7| D3 05                      OUT     5
04C9| E3                          XTHL                    ; Let parameters sink in --
04CA| E3                          XTHL                    ;  waste a little time.
04CB| 3E CE                      MVI     A,0CEH
04CD| D3 03                      OUT     3
04CF| D3 05                      OUT     5
04D1| 3E 27                      MVI     A,27H
04D3| D3 03                      OUT     3
04D5| D3 05                      OUT     5
04D7| DB 02                      IN      2               ; Reset rda.
04D9| DB 04                      IN      4
04DB|
04DB| E1                          POP     H
04DC| F1                          POP     PSW
04DD| C9                          RET
04DE|
04DE|                          .END
```

d. Installing the Input/Output Routines

Once you have modified the user-area which has been in RAM during this discussion with the appropriate changes to the jump table and your own I/O personalization routines, return to the DOS and execute either the command:

```
SF USERIO.2,2 4400
```

or

```
SF USERIO.0,2 4400
```

depending upon which version of the simulator you are personalizing.

If you have not had to personalize your working bootstrap diskette, or if you have followed all instructions in section 2 correctly, your diskette should be ready to bootstrap-load the Pascal system into your computer.

Simply insert the bootstrap diskette into the primary drive and cause your computer to begin execution at E800H. (Standard HORIZONs will do this automatically at reset or whenever they are turned on.) If you are in the DOS, and the DOS format directory contains the "PASCAL" file as listed above, you may instead type:

```
GO PASCAL
```

After a few seconds of disk activity, you should be greeted with the following message:

```
Welcome PASNS:, to

U.C.S.D. Pascal System I.5

Current date is 13-Mar-79


Command: E(dit, R(un, F(ile, C(omp, L(ink, X(ecute, A(ssem...
```

Note that the "Current Date" will be 12-Apr-79 for double-density systems. If you do not get this response, try re-booting (forcing your computer to execute at E800H). If you did not personalize your diskette prior to bootstrap-loading, either the diskette is defective, or you do not have a standard HORIZON configuration and need to personalize the system as described in section 2. If your diskette is defective, go back to section 1, CREATING THE WORKING DISKETTES, and start everything over again, using new diskettes for your working copies.

If you did personalize your diskette prior to bootstrap, your personalization routines may be incorrect. Please double-check the

correctness of your I/O personalization routines, and, when you are satisfied that they are correct, re-personalize the working bootstrap diskette according to section 2.

3. CHOICE OF SIMULATORS

If your system has come up as described above, you are using the Pascal p-code simulator which is resident starting at 2000H in memory (SYSTEM.NSTAR2). Because many computer systems cannot contain the 48K RAM required for program development unless memory starts at 0000H, and also because no system may run with the recommended 56K of RAM unless contiguous memory begins at 0000H, the simulator SYSTEM.NSTAR0, with origin at 0000H, is included on the bootstrap diskette. If you are satisfied with the system at 2000H, you may R(emove the SYSTEM.NSTAR0 file from your working bootstrap diskette. However, if you would rather use the 0000H-based simulator, you must first personalize its I/O routines if necessary (see section 2, above), then use the F(iler to R(emove SYSTEM.NSTAR2 or C(hange its name to something else, such as "P.SIMULATR.2000". Once you have done this, you may re-boot the system, and the SYSTEM.NSTAR0 simulator will be used. (At boot-load time, the bootstrap routine scans the diskette's Pascal directory -- NOT the DOS format directory -- for the first SYSTEM.NSTARx file it can find, then loads the contents of that file into the appropriate memory location as indicated by the digit at the end of the file name.)

Note that, if you R(emove either simulator file, the DOS format directory (which lists the locations of the user I/O areas for each file) will NOT be changed to reflect the file's absence. If you use the Pascal system to make changes in a diskette which also has a DOS file directory, and any of those changes affects the accuracy of the information in the DOS format directory, you may want to use the DOS to update or delete the information in that directory.

4. CONSOLE TERMINAL CONFIGURATION

Your Pascal system was shipped configured for a terminal with no cursor-control. If you have a cursor-controlled video screen, you may want to reconfigure the system to take advantage of the special features of your terminal. For example, features on your terminal may make it possible for instant erasure of a line of input in response to striking the line-delete key, or backspacing and erasure of the last-inputted character in response to the character-delete key.

If your terminal is a SOROC IQ120, the reconfiguration process is brief. When the system asks for a "Command" (as it does at bootstrap-load time), strike the "F" key. After a second of disk activity, you should see a command-prompt line which begins with the word "Filer" instead of "Command." You are now in the File Maintenence portion of the Pascal operating system. Strike the "C" key, and you will be asked for the name of a file whose name is to be CHANGED. Type

SOROC.MISCINFO<CR>

where <CR> indicates striking the RETURN key. Then, you will be asked for

the new name for the file. Type

SYSTEM.MISCINFO<CR>

The system will reply

PASNS:SYSTEM.MISCINFO exists...remove it ?

and wait for you to strike a key. Strike "Y" (for "yes"). When the system replies that the change has been made, you should re-boot the system. You should see a difference in the greeting procedure. For one thing, the screen will be cleared before the greeting is printed on the screen. This is a hint to you that everything is operating smoothly, and that the system is now using the cursor-control and other facilities of your SOROC terminal.

The SYSTEM.MISCINFO file contains information which allows your Pascal system to take advantage of whatever special screen-control facilities your console terminal has. Initially, the SYSTEM.MISCINFO file tells the system to assume that only a non-cursor-controlled device is available. By changing the SOROC.MISCINFO file to SYSTEM.MISCINFO, you have changed the system's expectations and abilities.

North Star could not include special MISCINFO files for every terminal on the market. If you own a terminal other than a SOROC IQ120, you must (in the Command mode) X(ecute the program SETUP, which will assist you in creating a NEW.MISCINFO file suited to the particular features of your cursor-controlled console terminal. See section 4.3 of the PASCAL SYSTEM REFERENCE MANUAL for more details about SETUP. Before you quit SETUP, be sure that you have "Updated the Disk" as part of the QUIT procedure. When you return to Command mode at SETUP's conclusion, change NEW.MISCINFO to SYSTEM.MISCINFO just as described above for SOROC.MISCINFO.

One more thing may be necessary before cursor-controlled video operation will work on your terminal. The SOROC IQ120 and the Lear Siegler ADM-3A terminal use the same "escape sequences" for repositioning the screen cursor to any specific point on the screen. In brief, an ESC code (ASCII 27) and the "equals" (=) character are sent to the terminal, followed immediately by the Y (row) and X (column) co-ordinates expressed as ASCII characters. ASCII character 32 (space) represents row or column 0, ASCII 33 (!) denotes row or column 1, etc. If your terminal follows the same conventions, you need only re-boot the system after installing the new SYSTEM.MISCINFO file to achieve full screen operation, since the system is already configured to recognize and use this particular cursor-control method.

If your terminal handles X,Y cursor positioning differently than the above terminals, however, it will be necessary for you to prepare a Pascal procedure such as "IQ120XY" in section 4.7 of the PASCAL SYSTEM REFERENCE MANUAL. The procedure should accept two integer co-ordinates, X and Y (column, row), and the result should be that the cursor is re-positioned at the proper screen co-ordinates. C(ompile the procedure, and X(ecute the program "BINDER" ("COMP:BINDER" for PAS-PRI-S) to bind the compiled procedure into the Pascal Operating System.

Note that BINDER requires at least 60 free disk blocks before it will operate correctly. It invokes the L(inker program automatically to create a new version of the operating system with your personalized XY procedure linked in. Neither the single-density nor the double-density version of PASNS: has sufficient free disk space for the BINDER's operation, so it will be necessary for you to T(ransfer some files from the PASNS: disk to another, empty Pascal data diskette, then R(emove these files from your working PASNS: diskette, leaving at least 60 free disk blocks before X(ecuting BINDER. When BINDER is finished, re-boot the system, and T(ransfer the files back to the PASNS: diskette.

The SETUP.CODE and SYSTEM.LIBRARY files may be T(ransferred to an alternate diskette and R(emoved from the working PASNS: diskette to make room for BINDER operation. PAS-PRI-S users will also find it necessary to T(ransfer the unused simulator file to an alternate diskette and R(emove the file from the working PASNS: diskette before there will be enough free disk space for the BINDER. DO NOT remove any SYSTEM file (except the SYSTEM.LIBRARY and the unused simulator) from the PASNS: diskette for this procedure. Once the required files have been removed, consolidate the available free disk blocks at the end of the diskette by using the K(runch command in the F(iler. (See section 1.2.5.16 of the SYSTEM REFERENCE MANUAL.) BINDER may now be executed.

Once your special version of GOTOXY has been bound into the system, you should be able to re-boot the system and enjoy full cursor-controlled video console operation. PAS-PRI-S users should note that both the SYSTEM.COMPILER and the BINDER program are on the COMP: disk, so you will have to have the COMP: disk in your second disk drive while compiling and binding your GOTOXY procedure. You cannot use the screen editor to prepare the GOTOXY text until you have screen control fully implemented on your system, so you must X(ecute COMP:YALOE in order to prepare the text file for compilation prior to binding. Do NOT use the E(dit command to invoke the editor until you have completed step 5 below.

5. CHOICE OF EDITORS

Once you have set up screen-oriented operation on your system, or have decided that (because your terminal is a TTY, doesn't have cursor-control, etc.) you will not opt for screen-orientation, you may choose which, of the two editors available on the system, you will use as your SYSTEM.EDITOR. If your system is screen-oriented, you need make no change in the existing SYSTEM.EDITOR file -- it already contains the screen-editor. In this case, you should R(emove YALOE.CODE from the working diskette (the working COMP: diskette if you have PAS-PRI-S), in order to acquire more free storage space on that diskette. If you do not have a screen-oriented terminal, you CANNOT use the screen editor, and must use YALOE instead. To do this, go into the F(iler, C(hange YALOE.CODE to SYSTEM.EDITOR (both of these are on the COMP: disk for PAS-PRI-S), Q(uit the Filer, and you're all set.

After choosing your SYSTEM.EDITOR, you may use the E(dit command to invoke that editor.

For purposes of convenience in program development, single-density users may wish to use the F(iler's T(ransfer command to put their chosen SYSTEM.EDITOR on the bootstrap diskette, rather than keep it on the COMP: diskette. With the bootstrap diskette in the primary drive and the COMP: diskette in the secondary drive (Pascal device #5), go into the F(iler, use the T(ransfer command to move COMP:SYSTEM.EDITOR to *SYSTEM.EDITOR, then R(emove COMP:SYSTEM.EDITOR and Q(uit the Filer.

III. SOME IMPLEMENTATION NOTES ON THE NORTH STAR PASCAL SYSTEM

A. Pascal Disk Blocks

A Pascal "disk block" contains 512 bytes of information. This is equivalent to two North Star standard disk blocks (256 bytes each). Thus, a single density diskette, holding 350 North Star dick blocks of information can contain up to 175 Pascal-style blocks. Double-density diskettes, holding 700 North Star disk blocks of information, can contain up to 350 Pascal-blocks. However, the first track on every diskette (track 0) is "off limits" to the Pascal system, and is reserved for routines and information (such as the DOS format directory and the bootstrap routine) which are specific to the North Star implementation of Pascal. These blocks are not accounted for in the Pascal diskette directory, but all others are (including those which contain the Pascal directory itself!).

The blocks covered in the Pascal directory, and which are normally available to the Pascal system for routine data storage and manipulation, are called the RELATIVE BLOCKS. They begin with the first sector of the second track (track 1/sector 0). There are 170 relative blocks on a single-density North Star Pascal diskette, and 340 on a double-density diskette. When you use the F(iler's Z(ero command to "zero" the directory of a Pascal diskette, make sure that the "number of blocks" is set to 170 or 340, depending on whether you are using the single-density or double-density system.

B. Program Development with Single Density

NOTE: While this section is primarily for the benefit of single-density users, double-density users may find items 4, 5, and 6 useful.

Program development with a single density system is somewhat tricky. The SYSTEM REFERENCE MANUAL describes program development procedures which rely heavily on the "WORKFILE" features of the system. That is, unless told otherwise, the system assumes that all program development will be done using standard "workfiles" which may be manipulated conveniently with special built-in commands, etc. Unfortunately, the system requires that all workfiles exist on the bootstrap diskette, which is inconvenient under single-density operation on 5-1/4" floppies, since there is not much free storage space available on either the bootstrap or COMP: diskettes.

The single-density user may use "workfile mode" successfully if the programs developed under that mode are small, and are infrequently modified. As programs become larger, and are maintained more often, you may find yourself running out of diskette storage space for the workfile in the middle of an editing session, meaning that it will be impossible for you to update your workfile with any changes made during that session! To combat this space limitation, you should use the following strategy:

1. Avoid using commands or features which establish or update a workfile. Do not use the "U" option when quitting the editor. Instead, use the "W" option to write the editor buffer back to the original file (which you will be expected to name).

2. If ever you do find yourself with an unwanted workfile, go into the filer and use the N(ew command to eradicate the workfile. Note that, if there is a workfile in your bootstrap diskette directory, the system will shift automatically into "workfile mode." N(ew takes it out of that mode. This is especially important when using the editor, compiler, or assembler, because you do not get a choice of which files you wish to edit, compile or assemble in workfile mode. The system assumes that you wish to deal with the workfile only. When you are not in workfile mode, the editor asks which file you want to edit, the compiler and assembler ask for the names of source and destination (code) files, etc.

3. A test to see whether or not you are in workfile mode is to go into the F(iler, and invoke the W(hat command. Any response other than "No Workfile" means that you are in workfile mode, and should invoke N(ew to get out of that mode. (Since you probably entered the workfile mode by accident, there is a good chance that important information is in the workfile. You should use the S(ave command in the F(iler to save the contents of the workfile into an appropriate TEXT or CODE file before using the N(ew command, which will erase the workfile.)

4. Before writing a file to diskette from the editor, you must be sure that there is enough free space on the diskette to hold the contents of that file. If that free area is not available, you will not be able to write out the contents of the editor's buffer, and run the risk of losing your changes for that session. If you are editing large files, it is best to go into the F(iler before editing, and use the L(ist or E(xtended-list commands to determine, by looking at the directory, whether sufficient free diskette space is available for a file creation or update. If you have reason to doubt that sufficient space exists, you should use the filer's B(ad blocks command to search the diskettte for any bad blocks. If there are any, you have a bad diskette, and should take steps to transfer all the good blocks onto another, good diskette. If there are no bad blocks, you can use the K(runch command in the filer to reclaim unused diskette storage space (similar to the CO utility in North Star's DOS). If, after K(runching the diskette you still have insufficient space for the file, you will have to update your file onto another diskette where enough free space is available. CAUTION: It is not possible to leave the editor temporarily for the purpose of K(runching a diskette. When you leave the editor, you lose the work for the session unless you first write the editor's buffer to a text file on diskette. So, it is important that you check the status of your diskette BEFORE editing.

5. Usually, during program development and execution, the bootstrap diskette is in the primary drive (device #4), and the COMP: or AUX: diskette is in the secondary drive (device #5). When compiling or assembling large programs whose source and object won't fit together on either of those disks, it may be necessary to put source on a separate diskette, and compile or assemble the object to that diskette also. To do this, you must give the A(ssem or C(omp command in the Command mode, and when the chosen program asks for a source file name, you must remove the bootstrap diskette from device #4 and insert the source/destination program development diskette. To name the files on that diskette, you must prefix each name with either "#4:", as in "#4:PROG1", or with the name of the diskette volume, as in "DEVELOP:PROG1", assuming the name of the diskette is "DEVELOP:". (Advanced users will

probably want to enter the F(iler and use the P(refix command to set the default diskette to the name of their development volume before entering the editor, compiler, or assembler. Then, they won't have to retype the volume name for every file name on that diskette.) When the compilation or assembly is complete, the operating system will remind you to re-insert your bootstrap diskette before you can re-enter Command mode.

If you must remove the bootstrap diskette in order to facilitate an assembly or compilation, be warned that, if a fatal error occurs during the operation and the bootstrap diskette is not available for system re-initialization and recovery purposes, you will be stuck in an endless loop of p-machine execution errors (typically errors 2 and/or 3 -- see TABLE 1, PASCAL SYSTEM REFERENCE MANUAL). If this occurs, you will be forced to re-insert the bootstrap diskette in device #4 and initiate re-booting by hand. In less severe circumstances, the system will remind you to replace the bootstrap diskette.

When using the E(ditor, NEVER remove the diskette which contains the editor program itself. The diskettes containing the C(ompiler, L(inker, or A(ssembler must also remain in their respective drives when any of them is in use. Failure to observe this rule may result in p-machine execution errors, and (especially in the case of the editor) loss of data before it can be written to diskette.

At some point, it may be necessary for you to remove BOTH system diskettes in order to transfer a file from one development diskette to another development diskette, or to a fresh diskette. To do this, you must first be in the F(iler. Once you are in the F(iler, it is OK to remove ALL diskettes from the drives and replace them with new ones for the T(ransfer, if you wish. The F(iler is self-contained, and doesn't care what diskettes are in the drives. However, to leave the F(iler, it is necessary to re-insert the bootstrap diskette, and the system will remind you to do so.

6. As of this release (Version 1), North Star Pascal isn't totally independent of the North Star DOS, because Pascal requires that new diskettes be initialized to North Star format (all spaces over the entire diskette) before being used in the Pascal system. At press time, there is no Pascal system function which will initialize a diskette correctly for North Star Pascal use, but one will be available soon through the North Star Software Exchange. Check latest copies of the North Star NEWSLETTER for announcements of Pascal software from NSSE. Until the diskette initialization utility is available, it will be necessary to initialize a diskette in the DOS first, using the IN command, before that diskette may be used by the Pascal system.

The system will copy Pascal data diskettes in a very convenient fashion, but a complete copy of a bootstrap diskette (including boot information on the "invisible" blocks) now requires the use of the separate "BOOTER" program. Another utility which will be offered through NSSE is a complete diskette copy program, which will be sufficient for copying both data diskettes and bootstrap diskettes.

C. Terminals with Short Line Lengths

North Star does not recommend that terminals with line lengths shorter than 80 columns be used for cursor-controlled operation under Pascal version 1, since the menu-prompting scheme for the Command mode and the F(iler are rather heavily oriented to screens at least 80 characters wide. However, the screen-editor will work properly with terminals whose line lengths are shorter than 80 columns. In any case, Pascal may still be used in the line-oriented mode (with YALOE as the SYSTEM.EDITOR) with all terminals. Future releases of North Star Pascal will support at least terminals with line length of 64 characters or greater.

D. Binary Arithmetic

In keeping with all known implementations of UCSD Pascal, North Star Pascal permits 7.2 digit precision in real number computations, using 32-bit binary floating-point methods. Future releases will support extended-precision real arithmetic using North Star's packed Binary Coded Decimal (BCD) representation. These later releases will also permit use of North Star's Hardware Floating Point Board (FPB-A) to perform real arithmetic.

E. Unused Drives

Although North Star Pascal supports up to three single-density drives or four double-density drives, it is possible to operate the system with only one or two drives. (As noted before, dual single-density drives are the practical mimimum requirement for program development.) As a consequence of system architecture, there will be occasions when the system will "look" for drives which may not be on-line (notably, at bootstrap load time, when the system initializes its peripherals). When this happens, the drive motors will be on, but no drive will appear to be selected. The "search" process for a nonexistent drive takes about 10 seconds with a single-density controller and 1 second with the double-density controller for each drive which is not on-line. As an example, if you have a dual-drive single-density system, note that the drives appear to become dormant midway through the bootstrap initialization disk activity. After a few seconds of motor activity but no disk selection, a drive is finally selected, and the system proceeds to give you the greeting mentioned earlier. The dormancy period is normal, and occurs because the system is seeking the third drive, which is not available in your system. Except for the delay caused by the "dormant" periods, normal system operations remain unaffected.

F. Data Transfer Between Single-Density and Double-Density Diskettes

The PAS-S system cannot read or write double-density data; The PAS-D system cannot read or write single-density data. However, an entire volume of Pascal data may be transferred from one diskette to another of the opposite density using the CF utility in the North Star dual-density DOS (Version 6, Release 5.0 or later).

Nominally, CF will handle only DOS-format files, using the DOS directory. CF will not use the Pascal directory. Therefore, it is necessary for CF to treat the entire Pascal data area on a diskette as if it were a DOS

file. To use CF in transferring Pascal information between diskettes of opposite densities, you must use DOS to CReate an entry in the DOS directory of each Pascal diskette which corresponds to the entire Pascal data area on that diskette. "Creating" a DOS-style file on a Pascal diskette does not alter any information on that diskette (except in the DOS-directory area itself, of course). It is done only to give the CF utility a "pointer" to the Pascal data region on the diskette. Once the DOS file entries exist, it is a simple matter to invoke CF and transfer information from one diskette to another, switching densities along the way.

The Pascal data region of a single-density diskette is only half as large as that of a double-density diskette. This means that it will be possible to transfer a single-density Pascal volume in its entirety to a double-density diskette. However, all the information on a double-density Pascal volume will not fit on a single-density diskette. It is possible, however, to prepare a double-density Pascal volume which can contain only as much information as a single-density one. This half-capacity double-density volume may then be transferred to a single-density data diskette using the method mentioned above. To prepare a half-capacity double-density volume, first use the DOS to initialize an unused diskette to double-density. Then, boot-up Pascal and use the F(iler's Z(ero command, to initialize the Pascal directory of the diskette. When Pascal asks for the number of blocks, enter 170, the number of blocks on a single-density diskette, instead of the usual 340 for a double-density diskette.

As a result of the above procedure, you will have a double-density Pascal volume whose entire contents may be transferred to a single-density diskette. To transfer from double-density to single-density, first use the Pascal F(iler's T(ransfer command to copy files you choose from regular-capacity double-density Pascal volumes to the special half-capacity volume. Because the Pascal system is aware of the decreased storage capacity of the special volume, it will not permit you to put more information on that volume than a single-density diskette can hold. When all the files you wish to transfer to single-density have been copied onto the half-capacity diskette, boot-up the double-density DOS, make sure that both the half-capacity double-density diskette and the destination single-density diskette have appropriate DOS files on them which refer to their respective Pascal data areas, and then use the CF utility to transfer the volume from double-density to single-density.

Note that after an entire volume transfer using CF, the resulting single-density or double-density diskette (depending on which direction the transfer went) will be an exact duplicate of the original volume which was transferred. In particular, the volume names will be the same. Also, if a single-density volume is transferred to double-density, the resulting double-density volume will be a half-capacity volume, and the diskette storage area after the 170th Pascal block will be inaccessible using normal Pascal disk accessing methods.

Following are examples of transfers between single-density and double-density volumes. The first transfers single-density to double-density, and the second transfers double-density files to a single-density volume using an intermediate, half-capacity double-density Pascal volume. Both procedures assume that the disks involved in the

transfer either have good information of the appropriate density on them, or have been initialized using the DOS IN command to the appropriate density before the transfer process begins. Also, both procedures require at least two disk drives to be on-line. If you have only a single-drive system, transfer of Pascal information between diskettes of different densities is quite difficult, and is not recommended.

SINGLE-DENSITY TO DOUBLE-DENSITY

* Choose a single-density Pascal diskette and a double-density diskette in good condition.

* Boot up double-density DOS (Release 5.0 or later). Type the following command:

```
LF CF 2D00
```

  This readies the CF utility for later use.

* Put the single-density Pascal diskette in drive #1 and the double-density diskette in drive #2.

* Type the following commands:

```
CR SDVOLUME 340 10 S
     {Create DOS single-density dummy file on drive #1}

CR DDVOLUME,2 340 10 D
     {Create DOS double-density dummy file on drive #2}
```

  Note that either diskette may have been used for transfers like this before. Therefore, prior to using the CR command, you might check the directory of each diskette to see whether or not the SDVOLUME and DDVOLUME files already exist. (They must each be 340 North Star blocks in length, starting at disk address 10 on their respective diskettes.) If they do exist, avoid creating them.

* Now, the CF utility may be used. Type the following:

```
JP 2D00 SDVOLUME DDVOLUME,2
     {Enter the CF utility}
```

  The CF utility will ask if you wish to write in Single or Double Density. Strike "D" for double. When the CF command finishes, the diskette in drive #2 will be a half-capacity double-density Pascal volume (170 Pascal storage blocks available, instead of 340). The information contained on it will be identical to that contained on the original single-density volume. Double-density Pascal may now be booted-up, and will be able to read files from the newly copied diskette.

DOUBLE-DENSITY TO SINGLE-DENSITY

* First, prepare the half-capacity diskette. (If you already have one of these, skip to the next step.) Insert an initialized double-density diskette in device #5 (North Star drive #2), and enter the F(iler command level. Use the Z(ero command to initialize the Pascal directory of the diskette in the secondary drive (Pascal device #5) so that it contains only 170 free blocks of information. (When the system asks "# of blocks?" enter 170.)

* Still in the Pascal F(iler, use the T(ransfer command to copy the Pascal files you choose onto the half-capacity diskette.

* Leave Pascal by booting-up the double density DOS. Type the following command:

```
LF CF 2D00
```

  This readies the CF command for later steps.

* Put the half-capacity double-density diskette in North Star drive #1 and a single-density diskette in North Star drive #2. Check to see whether or not the required DDVOLUME and SDVOLUME files exist on the two diskettes. If not, create them:

```
CR DDVOLUME 340 10 D
     {Create DOS double-density dummy file on drive #1}

CR SDVOLUME,2 340 10 S
     {Create DOS single-density dummy file on drive #2}
```

* Type the following command:

```
JP 2D00 DDVOLUME SDVOLUME,2
     {Enter CF utility}
```

  The CF command will ask if you wish to write in Single or Double Density. Strike "S" for single-density. When the utility is finished, the diskette in drive #2 will be a single-density Pascal volume with the same name as, and containing all the files and information on the double-density half-capacity diskette in drive #1. This newly generated diskette may now be read by single-density North Star Pascal systems.

IV. THE AUX PACKAGE: PAS-AUX-S, PAS-AUX-D

The Auxiliary package, PAS-AUX-S, is designed for use by advanced programmers. It includes two assemblers, for 8080 and Z80 operation, with which you may create machine-language routines which may be linked into your compiled Pascal programs. A series of utility programs round out the package. PAS-AUX-S is shipped on one diskette (single-density or double-density) which includes the following files:

Z80.ASSMBLER

This is the Z80 (Zilog Mnemonics) version of the U.C.S.D. Adaptable Assembler described in section 1.9 of the PASCAL SYSTEM REFERENCE MANUAL.

Z80.OPCODES
Z80.ERRORS

These are data files for the Z80.ASSMBLER -- the assembler WILL NOT OPERATE without them.

8080.OPCODES
8080.ERRORS

These are data files for the SYSTEM.ASSMBLER listed below.

MARKDUPDIR.CODE

MARKDUPDIR marks a Pascal data diskette so that a duplicate directory will be maintained automatically on it by the Pascal Operating System. (Note that the System Filer's Z(ero command, which initializes a diskette directory, gives you the option of specifying single or duplicate directories. MARKDUPDIR need only be used when it is desired to maintain duplicate directories on a diskette which previously contained only one directory.) See section 4.8 of the SYSTEM REFERENCE MANUAL for more information.

COPYDUPDIR.CODE

In the event that the main directory on a diskette is crashed, it may be regenerated from the duplicate directory (if there is one) by use of COPYDUPDIR. See section 4.8 of the SYSTEM REFERENCE MANUAL for further details.

RELOC.CODE

This program relocates a machine-language CODE file produced by the system assembler to any desired base address, and produces a pure-code, relocated object file as output. The user must be careful to specify the EXACT, COMPLETE file name for CODE and OBJECT files (including prefix if the file is not on the default diskette, and any suffixes such as .CODE, .OBJ, etc.) or the program will fail.

ABSWRITE.CODE

ABSWRITE allows you to transfer the contents of a file to an absolute disk location (ignoring the directory). There are 175 absolute blocks on a single-density Pascal-formatted 5-1/4" floppy and 350 absolute blocks on a double-density floppy. Absolute numbering starts with block 0. Generally, when ABSWRITE asks if you wish to "skip the first block", you should reply with NO.

RELWRITE.CODE

RELWRITE permits you to move the contents of a file to a point on the diskette starting at an arbitrary RELATIVE block number. Track 0 on any North Star Pascal diskette is "invisible" to the system, and contains bootstrap code on bootstrap diskettes (i.e., PASNS:). Pascal is aware of disk storage space starting at track 1. (This corresponds to absolute block 5 on single-density diskettes, absolute block 10 for double-density diskettes.) There are 170 relative data blocks on a single-density Pascal-formatted 5-1/4" floppy, 340 relative blocks on a double-density floppy.

SYSTEM.ASSEMBLER

This is the 8080 (Intel Mnemonic) version of the UCSD Adaptable Assembler. Note that this version of the Assembler gives you the option of generating either a non-relocatable code file (with absolute addressing) or a relocatable code file (with relative addressing information usable by LINKER and RELOC).

PATCH.CODE

This utility program facilitates patching of data on diskette, and is described in section 4.5 of the PASCAL SYSTEM REFERENCE MANUAL.

V. A BRIEF PASCAL BIBLIOGRAPHY

The Pascal programming language itself has not been covered in either the NORTH STAR PASCAL SYSTEM REFERENCE MANUAL, or this ADDENDUM; the assumption has been that anyone who reads these documents with an eye toward learning the mechanics of the North Star Pascal Program Development System is already well-grounded in Pascal. For those who need first to learn Pascal programming, several excellent references are available.

Algaic, Suad, and Arbib, Michael A., THE DESIGN OF WELL-STRUCTURED AND CORRECT PROGRAMS, Springer-Verlag: 1978.

Bowles, Kenneth L., MICROCOMPUTER PROBLEM SOLVING USING PASCAL, Springer-Verlag, 1978.

Bowles, Kenneth L., BEGINNER'S MANUAL FOR THE UCSD PASCAL SOFTWARE SYSTEM, BYTE Publications: 1979.

Gabrielson, Mike, "Pascal Bibliography", DR. DOBB'S JOURNAL, Vol 4:2 (32), Feb 1979, pp. 29-30.

Grogono, Peter, PROGRAMMING IN PASCAL, Addison-Wesley: 1978.

Jensen, Kathleen, and Wirth, Niklaus, PASCAL USER MANUAL AND REPORT (second edition), Springer-Verlag: 1975.

Mickel, Andy (Ed.), PASCAL NEWS, Pascal Users' Group, University Computer Center, 227 Experimental Engineering Building, University of Minnesota, 208 SE Union Street, Minneapolis MN 55455 USA. (Published quarterly; yearly subscription: $6.00 U.S.)

Schneider, G.M., Weingart, S., and Perlman, D., AN INTRODUCTION TO PROGRAMMING AND PROBLEM SOLVING WITH PASCAL, Wiley: 1973.

Wilson, I.R., and Addyman, A.M., A PRACTICAL INTRODUCTION TO PASCAL, Springer-Verlag: 1979.

Wirth, Niklaus, ALGORITHMS + DATA STRUCTURES = PROGRAMS, Prentice-Hall: 1976.

Wirth, Niklaus, SYSTEMATIC PROGRAMMING -- AN INTRODUCTION, Prentice-Hall: 1973.