PKGUT1

"THE COMPLETE DISK UTILITY PACKAGE"

USER'S GUIDE

Table of Contents

I. Introduction

PKGUT1 is the first utility package of its kind on the micro computer market today.

Originally advertised as being written in 8080 assembler language, these disk utilities have been rewritten in North Star BASIC to provide the user with an easier method of system interface and execution. There are no addresses to change and no relocations are necessary. PKGUT1 may be used with any micro computer system incorporating the North Star Micro-Disk System and utilizing North Star BASIC version 6 release 3.

PKGUT1 was written with the non-technical user in mind and will therefore prompt the user as he or she goes along with questions as to what type of function processing is to be done.

PKGUT1 was written to fill an outstanding need in the North Star Disk user community for a set of programs that would perform the basic utility functions required for disk data files. All programs access North Star BASIC data (type 3) disk files. COMPIT and CHANGIT can access any file format (machine language, assembler source, etc.) in certain modes of operation,if these files are given a file type of 3.

II. System Characteristics

A) Input

All input statements take the device Ø default value which is passed in register A by BASIC to the CIN routine in the North Star DOS. This is sufficient for the majority of users. If you would like to use a different or alternate input device, you should modify your CIN routines accordingly with respect to your own system configuration.

B) Output

All program prompt messages take the device Ø default on their PRINT statements which is passed in register A to the COUT routine in the DOS.

COMPIT and CHANGIT may produce large amounts of printing, if desired. Thus, these programs prompt the user for an optional output device number (Ø thru 7 as described in the North Star BASIC manual) for this print data. Again, this value (Ø - 7) is passed to the COUT routine in the DOS and should be dealt with accordingly with respect to your own system configuration.

C) BASIC numerical precision

All programs assume the standard BASIC numerical precision of 8 significant digits. If your version of North Star BASIC is using a different precision, you must modify the programs accordingly: The numerical precision is set in line 65 of each program in a variable 'P'.

I.E. For COMPIT line 65 reads:

```
65 P=8\I=100
```

Let us say your version of BASIC is using a precision of 12. Then line 65 should be changed to read:

```
65 P=12\I=100
```

SAVE the program.

Suggestion!, SAVE the utility program on a different disk then the original. Hard disk errors seem to always occur at the worst times and you don't want to destroy your original copy.

D) Maximum string lengths

Utility programs COMPIT,SORTIT and CHANGIT assume a maximum string length of 100 bytes. This is set in variable 'I' in line 65 of these programs.

This length should be sufficient for most applications without taking up too much core memory. If your requirements are such that your strings are larger than 100 bytes, you should modify line 65 to set 'I' to your maximum string length according to the outline above.

E) Memory requirements

The following are the approximate memory requirements for each utility program:

PACKIT	3k
SORTIT	4k
COMPIT	5k
CHANGIT	8k

The above names are the disk filenames and should be used to 'LOAD' your programs from disk.

<u>A Reminder</u>

North Star BASIC allows you to change the region where it allocates programs and data. You must modify the appropriate 'LXI' instruction in the BASIC interpreter if you would like to do this. Consult your North Star Disk Operating System for further instructions.

F) Initial Command Format

All programs await an initial command prompted by a ' UTILITY READY ' message. Thereafter, each program will prompt the user with questions as to what type of processing he would like to perform.

The initial command format is the same for all of the programs and is described now. Brackets denote an optional operand, an underscore, '_', denotes the default value and b̸ denotes a blank.

$$\text{CC}\not{b}\text{XXXXXXXX}\left[,\begin{matrix}\underline{1}\\2\\3\end{matrix}\right]\not{b}\left[\text{YYYYYYYY}\left[,\begin{matrix}\underline{1}\\2\\3\end{matrix}\right]\right]$$

where:

CC=	a two character command. Must start in column 1.
XXXXXXXX=	1-8 character source disk file name as described in the North Star DOS guide.
$,\begin{matrix}\underline{1}\\2\\3\end{matrix}=$	optional disk unit # as described in N* DOS guide.
YYYYYYYY=	optional 1-8 character target disk file name.

<u>Example</u>: We want to sort a disk data file named SOURCE on disk unit 1 to a file named RESULTS on disk unit 2: Using SORTIT, the user would type the following initial command:

```
SR SOURCE RESULTS,2
```

All programs are terminated by typing 'END' instead of the initial command.

III. Using the Utilities

All programs can only access type 3 disk files. These data files must have the North Star BASIC data format (either string, numeric, or usually both) with two exceptions:

1) COMPIT may access any file format in the byte to byte compare mode,if these files are given a file type of 3.

2) CHANGIT may access any file format in the binary print mode, if the file is given a file type of 3.

The file type may be changed back and forth by using the DOS TY command.

A) COMPIT

COMPIT is a file comparison utility and may be invoked in 3 different modes of operation by 3 different corressponding initial commands.

1) Record for Record   Command=RR

This command will read sequentially through both files comparing record for record. Files may contain numerics, strings, or both. Only records which do not compare are printed. If either file hits end of data before the other, all remaining records from the other file are printed and counted as no compares. A summery at the end gives:

1) Total records read from file 1
2) Total records read from file 2
3) number of successful compares
4) number of unsuccessful compares

2) Byte for Byte   Command=BB

This command will read sequentially through both files comparing each byte from one file to the other. Files may contain any form of data as long as the file type is 3.

A special note must be observed here. All BASIC files have a software end of file marker where the data ends on disk. Thus, a file defined with a length of 2 (256 byte) blocks might have only one record in it(string or numeric) followed by the software end of file marker. Normally, the utility programs stop when this end of file marker is reached. In BB mode, COMPIT will ask for the number of bytes you want to compare and will keep reading until that number is reached. If the number of bytes you specify to read is larger than the defined file length, an "out of bounds" error will occur when the defined file length is reached.

Only bytes which do not compare will be printed. The number of bytes read and the number which did not compare are printed at the end of the run.

3) Key Comparison   Command=KY

This is a special form of file comparison technique for files containing strings with key fields. Files may contain strings or strings and numerics, but only string records are processed. All numerics are read and bypassed.

The user will be asked to enter the start and end key field positions in each string record. The first position is always counted as 1 not Ø.

For example, the following string record has a key field,part number, starting in position 1 and ending in position 6.

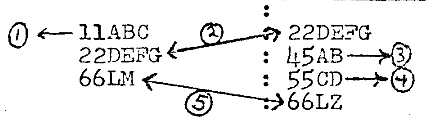
```
record=   ØØ4791WIDGETƀƀƀƀ,DOEƀƀƀƀƀƀƀ,JOHNƀ,Ø969ØØ
position= 1234567890123456789012345678901234567890
          |____| |_____1____| |__2__________3___| |___4__|
          PART   PART NAME    [illegible]          EMPLOYEE #
           #
```

You would enter 1,6 when you are asked for the key position, if you wanted to compare on this field. Alternately, you could key this file on employee number in field 35,40.

Files should be sorted in key field before this type of comparison is done. Key comparison does not compare record for record but works like this:

COMPIT will first compare both records on key. If they are not equal, the record with the lower key value is considered a nomatch record and is printed out. Next, only the file containing the nomatch record will be read. Thus, additional records on a file will not through the comparison out of phase. If the records match on key, then the rest of the record will be compared. If one of the files hits end of data then the remaining records in the other are considered nomatches. For example: <u>key pos.=1,2</u>

<u>FILE A : FILE B</u>



1) 11ABC is read from file A and 22DEFG is read from file B. Keys do not match. 11ABC is printed as a nomatch. Only file A is read next.

2) 22DEFG from file A & B matches on key. Compare is done successfully. Nothing is printed, File A and file B are read next.

3) 66LM is read from file A, 45AB from file B. File B record is a nomatch. Only file B is read next.

4) 55CD is read from file B. It is a nomatch. only file B will be read next.

5) 66LM and 66LZ match on key, but do not compare. Both are printed as nocompares.

A summery is printed at the end of the run giving:

1) # of unmatched records from file A(first file in initial c
2) # of unmatched records from file B
3) # of nocompare conditions encountered on matched records
4) # of compare conditions encountered on matched records
5) # of <u>string</u> records read from file A
6) # of <u>string</u> records read from file B

B) SORTIT

SORTIT is a disk sort utility which gives it an advantage over in-core sorts. Specifically, the number of records you may sort is not limited by the amount of core memory you have. SORTIT is a copy operation. A source file and a target file must be specified in the initial command. SORTIT must have a disk work file to use. You must create a work file on disk using DOS with the filename of SORTWORK (SORTIT will ask you what unit it is on) and a length equal to your source file. The second file name you specify in the initial command is your target file which you must also create on disk using the DOS CR command with a length equal to the source file's. All files may of course be on the same unit.

File may contain either numerics or strings not both. SORTIT will ask you for the type of data. If you are sorting string data, SORTIT will ask you for the start and end fields that you would like to sort on. Upon completion, SORTIT will tell you the number of records that were sorted.

C) PACKIT

PACKIT will compress disk data files enabling you to store more information on a disk. Files may contain, numerics, strings, or both. PACKIT works in three modes of operation.

1) Pack   Command=PK

Pack is a copy operation. The initial command must specify both a source and a target filename. The method for packing a file is as follows:

a) Create a target file with a length equal to you source file's, by using the DOS CR & TY commands.
b) Execute the initial pack command.
c) Upon completion of the packing operation, PACKIT will tell you the number of blocks (256 bytes) that your target file length may be changed to.
d) Using the DOS, delete your target file (the DE DOS command only removes the file name from the disk directory, data is not touched) and re-create it with the new file length.

2) Verify   Command=VR

Verify requires only one filename in the initial command and will tell you if a particular file is in the packed format or not. In addition, verify will tell you the number of blocks required to unpack the file.

3) Unpack   Command=UN

Unpack works exactly the same way as the pack command. The target file length you specify is the length of your original unpacked source file and may be retrieved via the verify command. You do not have to delete any files, since you know before hand what your target file length is going to be.

NOTE: PACKIT works by squeezing out blanks and zeros in strings and numerics respectivly. PACKIT'S only restriction is that no string may contain more than 255 contiguous blanks.

D) CHANGIT

CHANGIT is a multi-function utility program. CHANGIT has the ability to print data records, change data records or dump data records. Records may be chosen for processing based upon your selection criteria. Records may be changed on a global level or a local level based upon your selection criteria, Files may contain strings, numerics, or both. Print and dump are one file functions. That is you specify one file name in your initial command. Change is always a copy operation, That is you specify two file names, a source and a target, in the initial command. Change was implemented in this manner in order to protect your source data from errors in replying to the computer prompts. After the change has been done and you are satisfied with your target file, you may delete the source file if desired by using the DOS.

You will be prompted as to what type of processing you would like, so there is no need to memorize the different options. Let us first look at what the different options mean, and then we will explain each function.

1) Selection

You will be asked if you would like to select records from your data file, Selection may be performed on either numeric or string type records. If you choose to select string records, you must enter either the starting string position for CHANGIT to begin looking for your selection data or Ø. Zero tells CHANGIT to search the <u>entire</u> string record for your selection data and will replace that data with your replacement data wherever and only wherever it finds it in your record. You will not be asked for a replacement starting position. If you enter a selection starting position, CHANGIT will ask you for a starting position for your replacement data. CHANGIT will replace the data at that position with your replacement data on each and only each record that contains the selection data at the selection position. CHANGIT reads sequentially through your source file. If a particular record meets your selection criteria, it is changed accordingly (or printed or dumped) and written to your target file. If it does not meet your selection criteria or if it is of a different record type (I.E. numeric if you are selecting strings) CHANGIT will just copy it over to your target file (or for print/dump it will be ignored).

If you choose to select numerics, simply enter the number you would like changed and the replacement number..

2) SKIP/STOP

Skip tells changit to skip a certain number of records before processing. Stop tells CHANGIT to stop processing after a certain number of records have been processed. These options enable the user to print, dump or change any number of records from his or her file.

If selection has been chosen, SKIP and STOP apply only to records meeting the selection criteria, For example, let us say you are selecting on a particular string and you specify 2 for the skip option and 3 for the stop option. Also assume that 10 records in your file of 30 records contain that string. In this case, all records not meeting your selection criteria will be copied without change (20 of them). The first two occurences of a record containg your selection string will be copied without change (skip 2) , the next 3 containg your selection string will be changed with your replacement data and copied (stop 3) , the remaining 5 records containing your selection string will be copied without change.

If selection is not choosen (print/dump only), then skip and stop apply to all records.

3) Binary print

After you enter the print or dump commands, CHANGIT will ask you if you would like to do a binary print. There is no selction involved in a binary print nor can you specify the skip/stop options. Binary print works somewhat similar to the byte for byte comparison of COMPIT. You must tell CHANGIT how many bytes you would like printed. CHANGIT will read each byte of the file and print its decimal equivalent. Files need not be BASIC data files. You may binary print any type file (I.E. BASIC source files), so long as the file is given a filetype of 3. Remember, byte commands do not look for the software end of file marker and you should be careful not to request more bytes printed then there are contained in the defined length of the file. You will get an 'out of bounds' error if you do.

A) Change   Command=CH

As stated before, change always implies a copy operation. Two filenames, a source and a target, are required in the initial command. You must specify your selection criteria. You also have the option of printing the changed records or not. If you choose to print them, you can enter an output device number different than the default value.

CHANGIT will give you a summery of the number of records read and replaced after it has finished processing.

You may limit thenumber of records changed by using the skip and stop options. If these are not specified, all records containing your selection data will be changed.

B) Print Command=PR

Print is a one file operation. You have the option of either selecting records for printing or not. Files may contain numerics,strings or both and you have the option of printing numerics, strings, or both. If you choose selection and opt to print both, all non-selection type records will be printed and only those selection type records meeting your criteria will be printed. Each record is printed along with its record number.

You may specify skip and stop parameters, If you do and do not specify selection, skip and stop will apply to all records, If you choose to select, skip and stop apply only to the selected record types. You can of course print to an optional output device.

C) Dump Command=DU

The dump operation gives you a binary print in decimal of each byte of a record. Each record is printed with ist record number. Dump <u>does</u> stop on software end of file. Strings, numerics and both may be dumped. Records may <u>not</u> be selected when using the dump command. This is because dump gives you a true picture of what is on disk,by transferring the record bytes directly from disk to your output device. This includes the <u>entire</u> record along with BASIC'S encoded record type and length values before each record for strings, and before and after each record for numerics.

Dump differs from the binary print option in that

it:

1) Dumps each record along <u>with</u> its record number.
2) You may specify skip and stop options.
3) Dump stops on BASIC'S software end of file.

IV. Initial Command Summery

A). COMPIT

BB	=	Byte for Byte comparison
RR	=	Record for Record comparison
KY	=	Key comparison

B). SORTIT

SR	=	Sort

C). PACKIT

PK	=	Pack operation
VR	=	Verify operation
UN	=	Unpack operation

D). CHANGIT

CH	=	Copy file with selected records changed.
PR	=	Print operation
DU	=	Dump operation

DDSC #1

THE DOS MOVER

Relocation Routines For Moving The Standard

NORTH STAR DISC OPERATING SYSTEM

I. Purpose

This software package is designed to allow relocation of the standard issue of North Star DOS (at 2000H), including I/O patches, to any new starting address. These routines will only work on those systems using the standard DOS at 2000H and the control ROM at E900H. These relocated versions of DOS are not replacements for the standard DOS. They will not work directly with the standard issue of North Star Basic (without patching) since it uses the I/O jump table of the DOS at 2000H. However, they are likely to be very useful for those who wish to use their disc system with software that would normally overlap the standard DOS at 2000H. Such programs include most of the versions of Basic available for 8080/Z-80 based systems as well as many machine language games and utilities.

II. Requirements

A. Hardware requirements

A 8080/Z-80 based computer system with a North Star Microdisc system or a Horizon system with control ROM at E900H configured for the standard DOS at 2000H. During relocation, there must be continuous RAM from 2000H to 3FFFH.

B. Software requirements

A recent release of the standard North Star DOS (at 2000H). The DOS MOVER routines are known to work for DOS version 2, release 3 and Horizon DOS version 3.1. A program is provided to test your DOS for compatibility with the relocation routines.

C. Human requirements

It is likely that any person that is familiar with the use of their computer and some simple 8080 machine coding can use this software package.

III. Introduction

The standard issue of North Star Dos (STDOS) is located at 2000H with a control ROM at E900H. This ROM contains the the main disc control routines as well as a bootstrap loader to load the standard DOS. This loader loads the stack at 2114H and then loads the sector at disc address 4 into memory starting at 2000H. It then jumps to 2004H, to a routine that loads the remaining sectors of STDOS. Some of the routines within the ROM use a 4 byte buffer starting at 2000H for keeping track of the disc drive head positions of the various (up to 3) drives on the system. Thus, it is apparent that a relocated version of DOS must not use the ROM routines and it cannot be booted up directly with the ROM (without altering the ROM).

The DOS MOVER routines solve these problems by creating relocated versions of DOS that do not use the ROM. This is done by appending relocated versions of some of the ROM routines to the relocated DOS. Thus these relocated versions of DOS are 11 blocks long (instead of 10 blocks for STDOS). These relocated versions would normally be loaded from STDOS (or another relocate DOS). However, a short boot routine is provided that allows one to boot up indirectly from the ROM bootstrap with a relocated version of DOS.

The DOS MOVER diskette supplied contains 5 files:

1.	DOS	A blank file for the user's DOS.
2.	CHKR	A program to test your copy of STDOS for compatability with the relocation routines.
3.	BOOT	A routine used to make a relocated version of DOS boot up with the ROM bootstrap.
4.	CRDOS1	The DOS relocator creation program.
5.	CRDOS2	Same as CRDOS1 but for the most recent version of STDOS such as used on the Horizon system.

The format of the remaining documentation consists of simple descriptions and examples of the use of the DOS MOVER routines. The appendix contains a more detailed description of the workings of the relocation process. It is suggested that the user read all of the material before using any of software provided. In the examples provided, the underlined commands indicate user inputs to program prompts.

IV. Program Descriptions and Examples

A. CHKR

This program is used to test your copy of STDOS for compatability with the relocation routines. Incompatability can result from very old versions of STDOS, extensively modified versions, or very new versions of STDOS released after your copy of the DOS MOVER was made.

To use the program, load your copy of STDOS and:

1. Run CHKR with the GO command.

2. Answer the prompt with the starting address of STDOS (see the appendix for extended capabilities).

CHKR will then examine selected portions of STDOS and respond with a recommendation as to compatability. It then returns to the DOS that loaded it.

---------- Example, use CHKR to test STDOS -----------

```
        *GO CHKR
        -- DOS CHECKER --
        START ADDRESS OF THE DOS TO BE TESTED : 2000

        THIS DOS CAN PROBABLY BE RELOCATED WITH CRDOS1

        * ←— return to calling DOS
```

CHKR is not infallible. It does not check all of the DOS. This was a deliberate compromise chosen because the relocation routines are usually capable of correctly relocating patches made to the main parts of DOS provided no calls to the ROM routines have been moved or added.

B. CRDOS (CRDOS1 and CRDOS2)

This program examines the standard DOS at 2000H (with your I/O patches) and the standard ROM at E900H and copys selected portions of these programs to RAM beginning at 2A00H and relocates the code to create a relocated version of DOS at 2A00H (it can be run at its present location). It also creates a pointer table driven relocation routine behind this DOS (starting at 3500H) which can be used to create relocated versions of DOS for any start address. This combined DOS and relocator at 2A00H is called RDOS and is to be save and used to generate relocated versions of DOS. This two step process was chosen to avoid copyright problems with North Star. It also provides a very reliable method of generating the relocated versions of DOS. Note that the relocated versions of DOS will automatically have all of the I/O routines and patches that were made in your copy of the standard DOS at 2000H.

To use CRDOS (1 or 2), load your STDOS and:

1. Run the recommended (by CHKR) version of CRDOS with the GO command.

2. Note the size of the RDOS that was created.

3. Create a file for RDOS and save it on the disc.

4. Test the relocated DOS at 2A00H (part of RDOS).

---------- Example, CHKR recommended using CRDOS1 -------

```
        *GO CRDOS1
         RDOS : 2A00H - 38FCH ←— the size of RDOS for a simple
                                  port oriented I/O patch as
        *CR RDOS 15 ←             given in the North Star manua

        *SF RDOS.2A00             a more complex I/O patch may
                                  give an RDOS 16 blocks long.
        *JP 2A00  <----  test the relocated DOS.

        ---- DOS AT 2A00H ----

        *LI 0     <----  test self identification feature.

        ---- DOS AT 2A00H ----

        *      <-------  other tests of all commands.
```

After you have verified that the DOS at 2A00H works correctly, reboot your standard DOS (*JP E900) and create a relocated DOS with RDOS. If the DOS did not work correctly, review the possible problems discussed in the appendix.

C. RDOS

This program was created from your copy of STDOS and the standard ROM. It can be used to generate relocated versions of DOS for any new starting address. You should never have to use CRDOS again unless you decide to add new patches to your STDOS that you want transferred to the relocated versions.

To use RDOS:

1. Boot up with STDOS.

2. Create a disc file 11 blocks long to copy the relocated version into.

3. Change the file type to a type 1 file at the specified load address.

4. Load the relocating DOS (RDOS) at 2A00H.

5. Jump to the relocation routine at 3500H.

6. After the prompt, give the desired new start address (the same as used in step 3) and hit carriage return. A control-C will abort the process and return to STDOS at 2028H. Any other non-hex characters will initiate a restart of the relocator. The relocation process is virtually instantaneous. When it is completed, the relocation routine is disabled and the standard DOS is restarted. The program from 2A00H to 34FFH has now been modified so that it will run at the new origin.

7. Save the relocated DOS on the file that you created.

8. Test it by loading it with the GO command.

```
---------- Example, create a DOS at 6B00H ---------------
 ( this is the highest location allowed if you only have
   32K of RAM starting at 0 and you want your DOS to use
   IN, DT, CF, and CD.)

        *CR DOS6B 11 ↵

        *TY DOS6B 1 6B00 ↵

        *LF RDOS 2A00 ↵

        *JP 3500 ↵

        -- DOSXX RELOCATOR --
         NEW START ADDRESS : 6B00 ↵

        *SF DOS6B 2A00 ↵

        *GO DOS6B ↵
        ---- DOS AT 6B00H ----
        *LI 0 ↵     <----------    test self identification feature

        ---- DOS AT 6B00H ----
        *
```

D. BOOT

This program provides a way of booting up with a relocated version of DOS by using the ROM bootstrap routine. This is done by placing the BOOT program at disc address 4 followed by a copy of the relocated DOS. The ROM bootstrap program loads the BOOT at 2000H and it loads the relocated DOS at its specified start address. Note, this operation overwrites the first 256 bytes (and stack area) beginning at 2000H. Thus, this is most likely to be useful when your system is first turned on and there is no useful information storred in the first 275 bytes starting at 2000H.
To use BOOT:

1. Initialize a blank disc and create files for BOOT and the relocated DOS starting at disc address 4.

2. Change the file type of the file for the relocated DOS to a type 1 file at the specified load address (this will allow you to use the DOS independently of the BOOT).

3. Load RDOS and run the relocator.

4. Load BOOT at some convenient location (say 1000H) and patch in the desired start address of the relocated DOS at relative address 37H (1037H).
Example: a relocated DOS at 6B00H requires the patch:

```
            1037:  00
            1038:  6B   .
```

5. Save the patched BOOT on the disc beginning at disc address 4.

6. Save the relocated DOS on the disc beginning at disc address 5.

7. Test the program by jumping to the ROM at E900H. The relocated DOS should load and identify itself.

```
----------- Example, create a selfbooting DOS at 6B00H --------

    *IN            <--------- put in the blank disc
    *CR BOOT 1
    *CR DOS6B 11
    *TY DOS6B 1 6B00
    *LF RDOS 2A00  <-- put in the disc with RDOS and BOOT
    *JP 3500
    -- DOSXX RELOCATOR --
     NEW START ADDRESS : 6B00
    *LF BOOT 1000
                     <---------patch the copy of BOOT and
    *SF BOOT 1000              put in the initialized disc
    *SF DOS6B 2A00
    *JP E900       <-------- test self boot
    ---- DOS AT 6B00H ----
    *
```

APPENDIX

I. Similarities of STDOS and the relocated versions of DOS.

1. The jump table and read-after-write flag are in the same relative positions (relative address: 04H-2BH).

2. Most of the code and tables from relative address 82H to 09FFH is in the same place. This includes the 256 byte I/O area (relative address: 0900H-09FFH).

3. A modified version of the control ROM is loaded in the 256 byte area above the I/O area (rel. add.: 0A00H-0AFFH). Thus, this version takes up 11 blocks on the disc.

4. The 2560 byte area used for disc copy, test, and initialization was shifted up by 256 bytes to relative address: 0B00H-15FFH.

5. The head position buffer (4 bytes) usually found at rel. add. : 0-3 has been shifted up to relative add.: 5AH-5DH. This was done to put a jump at relative position 0. This allows the DOS GO command to be used to load and start one of the relocated versions.

6. A sign on message has been added that identifies the start address of the relocated DOS. This feature will be invoked by a jump to its start address or an attempt to list the directory of drive 0 (*LI 0)- which gives the usual error message in the standard DOS. Also, every time that the sign on message is printed out, the head position buffer is initialized to 59H.

II. Comments on the Relocation Process

A. CRDOS

The relocation routines in CRDOS are similar to those found in a disassembler. They operate by scanning the code and comparing suspected instruction bytes to tables of 3 and 2 byte (8080) instructions. Those 3 byte instructions which appear to reference an address within the standard DOS are relocated and their positions are saved in a table starting at 35BCH. This process is not infalible. However, the relocation is done in sections and other routines make up for deficiencies that are known to occure when used on a copy of the standard DOS. This method is quite insensative to the nature of the I/O patches or even patches made to internal positions of the DOS. These patches will be handled correctly provided the following rules are observed:

1. The patches must not involve moving or adding calls or jumps to the control ROM at E900H.

2. They must be done <u>cleanly</u> by using NOPs (00H) in place of fragments of the previous code that are no longer used. Such fragments can throw the relocator "out of step" as it scans thru the DOS.

3. If the I/O routines (starting at 2900H) are separated by any extra space, insure that either all the extra space is filled by NOPs or that at least 3 NOPs preceed the beginning of each of these separated routines. This insures that either the relocator never gets out of step or that it at least gets back in step before it encounters any valid machine code.

4. Any 3 byte 8080 instructions used in your patches will be relocated if the 2 byte operand appears to reference an address within the DOS (2002H-29FFH). There is a way to correct errors made in this way by patching RDOS. However, this is usually unnecessary. Note, this method does not relocate (re-address) calls or jumps to external monitor ROMS (which is the desired response).

5. The patch should not involve moving any part of the DOS command symbol table. However, the command names and jump addresses can be changed without any problems with the relocator.

6. The use of some Z-80 codes not valid for the 8080A is likely to get the relocator out of step. However, errors made in this way can often be corrected by patching RDOS. Horizon DOS does not usually contain any of these instructions.

Most people will probably not have to worry about the above details if they made their own I/O patches from the original copy of STDOS supplied with their system. This copy usually has the I/O area filled with NOPs and no unusual or "unclean" patches in the DOS. The major problem is that some people may never have received a copy of STDOS. Our initial testing of this product uncovered that some computer stores were copying their own patched versions of STDOS over the original versions on the disc shipped with the system. This is often done without telling the customer that it was done or the precise nature of the patches made- even though the label on the disc suggests that it is the original version from North Star. If THE DOS MOVER does not work on your DOS, even though you believe that the above rules were obeyed, we suggest that you ask for some help at the computer store that sold you your system. They may know what part of their patches to fix or supply you with a copy of the original STDOS.

B. RDOS

The relocator part of RDOS works by using a table of over 400 addresses (2 byte pointers) that point to the positions in the relocated DOS (at 2A00H) that require relocation. These points are the operands of 3 byte instructions like LHLD, SHLD, STA, LDA, JMP, CALL, and LXI. This table is at the end of the relocator starting at 35BCH. It was filled by CRDOS as it scanned the code in STDOS. In the event that CRDOS was unable to accomplish the entire relocation correctly (for reasons discussed above), you may be able to fix RDOS. This will require locating the incorrectly relocated code, patching it so that the DOS at 2A00H will work correctly, and adding or disabling pointers in the relocation table. The errors are probably involved with your patches to the code. The pointers associated with the I/O patches typically start at 38F8H unless patches have been made to the main part of DOS. The position of the first empty position in the pointer table is given when CRDOS is finished (38FCH in the examples given earlier). A pointer can be disabled by replacing it by a new pointer to a buffer area in RDOS (whose data is unimportant). Such an address is 3200H. If you must add a new pointer to the table, insert the new pointer at the end of the table and increment the table length word at 351AH:

```
        3519:  LXI B, LEN  .
```

As an example of how the table is addressed, note that the first pointer in the table (at 35BCH) points to the operand of the of the JMP instruction at 2A00H:

```
        35BC: 01
        35BD: 2A

        2A00: C3 5E 2A    JMP 2A5EH  .
```

III. Extended features of CHKR and CRDOS

Both CHKR and CRDOS have an unusual feature that may be of some interest. They can both be used from a relocated version of DOS, although CRDOS must still have an example of STDOS at 2000H. They both must be started from a copy of DOS (possibly a relocated version of STDOS). This can be done either by the GO command or a LF command followed by a JP 2A00. In either of these cases, DOS puts a return address on the stack (DOS origin+82H). CHKR and CRDOS use this address to set their return address as well as to compute the locations of the I/O jump table positions. These features may be of use in a computer store situation where the customer's copy of STDOS (with unusual I/O patches) is located at 2000H. Thus both CHKR and CRDOS can be run from a relocated version of DOS (with the store's I/O) to test and configure a relocating DOS from the customers STDOS. Also, note that CHKR can also be used to test the customer STDOS by loading it at some other location (say 1000H) and then running CHKR from the store STDOS (at 2000H) and giving CHKR the address of the STDOS at 1000H.

IV. Warning on jumping between different versions of DOS

Jumping between different versions of DOS can cause problems because the hardware does not keep track of the drive head positions. This is done by each individual version of DOS in software. When a new version of DOS is first loaded, its head position buffer is filled with 59H which causes an initialization to track 0 (a Home operation) before any other disc operations. If you then jump to another version of DOS that last thought the head was over track 2, it will become confused. You can demonstrate this most directly (and non destructively) by trying to list the directory in such a situation (you get a sort of random dump of characters). Thus, the safe thing to do is reboot between jumps to versions of DOS that have already accessed the disc at least once. Another option is to jump to the system monitor and reinitialize the head buffer with 59H. The relocated versions of DOS do this automatically when the sign on message is printed out.

This Product is Distributed thru the

DIGITAL DELI SOFTWARE COOPERATIVE
(DDSC)

What is DDSC, I here you ask. DDSC is at this moment an experiment. The intention is to provide a way for independent authors of significant software to market their products thru a common organization. We hope to gain consumer support and recognition by setting standards for software quality and documentation as well as reasonable prices. Distribution of many products thru a common organization should lower the distribution costs as well as provide various services that would be too expensive for single authors. By organizing the cooperative around an established computer store (The Digital Deli) we have direct access to the consumer market, a real mailing address and phone number (not just a mail drop), and a resident business consultant who regularly deals with the home computer (or small business computer) market.

This experiment will never get off the ground if the consumer market still believes that software should all be free. If your friends want a copy of this software or a relocated DOS created by it, they should pay for it in the same way that they purchased the parts of their computer. A number of very interesting software projects are in a holding pattern awaiting the results of this experiment. You the consumer will determine by your actions if any of these new products are ever released.

The DIGITAL DELI
80 West El Camino Real
Mountain View, Ca. 94041

(415) 961-2670