LS-UTILITY

LS-UTILITY

The LS-UTILITY Disk

Table of Contents

MISOSYS, Inc.
PO Box 239
Sterling, VA 22170-0239
703-450-4181

ADDENDUM TO THE LS-UTILITY DISK

The following points need to be clarified with respect to using MAXLATE/FLT:

On Page 1Ø (third paragraph, starting with "In this example ...") it is stated that "Any character may be used as a separator, with the exception of a quote mark or a digit (Ø-9)". This is incorrect, and should read as:

"Either commas or spaces may be used as separators".

Use of any other character as a separator may cause unpredictable results.

On Page 11 (in the section titled "Practical Use of MAXLATE"), all of the information is correct as printed. A notation should be made however, when using MAXLATE in conjunction with the communications line and the TRSDOS Utility program COMM/CMD. When the communications line is filtered with MAXLATE, and COMM/CMD is used to receive data through MAXLATE (i.e. MAXLATE is established with the INPUT parameter), the translation "stream" should NOT be longer than one character.

In the example of the COMIN/MLT file, the translation stream in the first line of the file (i.e. the two ØD's to the right of the equal sign) will NOT produce the desired results when used with COMM/CMD. The second translation line (i.e. translating a ØA to "no character") will work properly.

This restriction applies only to the INPUT direction when COMM/CMD is used. COMM/CMD will allow multiple-character translation streams for OUTPUT only.

G E N E R A L I N F O R M A T I O N

This section will cover General Information regarding the use of this package. If you are not familiar with using filters in the TRSDOS 6 environment, READ THIS FIRST.

Keyboard Nomenclature

Throughout the documentation, you will see references to individual letters (A-Z), digits (Ø-9) and words that are enclosed in "less than - greater than" symbols <>. This type of representation means that the single key with the "enclosed" caption should be pressed. For instance, when **<ENTER>** is shown in the documentation, you should press the key marked with the legend "ENTER".

Some of the filters require the use of a multiple key sequence to perform a function. KSMPLUS is one such example. To use a KSMPLUS key definition, a two key sequence is necessary, and is of the form <CLEAR><key>. Whenever this type of entry is required, you must depress the <CLEAR> key first, and while holding it down, depress the desired key. If <CLEAR> is not pressed first, or if it is not being held down during the depression of the second key, the "correct" keystroke will NOT be generated.

Another example of a multiple key sequence is when you wish to edit a KSMPLUS key definition. This operation involves a three key sequence of the form <CLEAR><SHIFT><key>. When this type of keyboard input is required, it must be entered in this manner.

1) Press the <CLEAR> key first.
2) While holding down the <CLEAR> key, press the <SHIFT> key.
3) While holding both the <CLEAR> and <SHIFT> keys down, press the desired key.

Installation and Use of Filters

When installing a filter, a two step process must be followed. In the first step, the "filter device" must be established. This is done through the SET command. A "phantom device" is SET to the desired filter. The device specification (referred to as device spec) consists of an asterisk <*> followed by a two character device name. The device name is chosen by you. It is comprised of any two alphabetic characters (A-Z), provided that this device name is not already in use. Examples of device names which are used by TRSDOS (i.e. "already in use") are *KI (the keyboard device), *DO (the video device), *PR (the printer device) and *JL (the job log device).

For example, to use the TRAP/FLT filter, you must first SET a device to the filter. If we wish to name the device *TR, the following command can be used:

```
SET *TR TRAP/FLT (OUTPUT,CHAR=X'ØA')
```

You will notice that there is more on this "command line" than the SET command, the device spec (*TR) and the name of the filter (TRAP/FLT). Following the filter name is an open parenthesis <(> and a list of parameters. These parameters are "passed" to the filter in the command line, and tell the filter what to do. In some cases, parameters are optional, and need not be specified. In other cases, "defaults" will be assigned to the parameters, so that if the parameter is not specified, the default will be used. Information regarding parameter usage can be found in the "syntax block" for each filter/utility in this package. The syntax block is found on the first page of each filter/utility section, and details all necessary information for proper use.

In this specific case (i.e. with TRAP/FLT), two parameters were passed. These parameters are NOT optional and no default values are assumed, so they must be specified. In most cases, parameters can be abbreviated to their first character. Information on allowable abbreviations can be found on the last line in the syntax block (abbr:). The following SET command is the same as the first, except that the parameters are abbreviated.

```
SET *TR TRAP (O,C=X'ØA')
```

Notice that the filter name in this example does not have an "extension" (i.e. the "/FLT" for TRAP was not entered). You need not specify the extension when SETting a device to a filter if the extension of the filter is /FLT (SET assumes a default extension of /FLT if no extension is used). However, if you are REMOVING a filter, the extension (/FLT) must be used.

After the SET has been completed, the next step is to apply the filter to a device. This is accomplished through the FILTER command. In the above example, we have established *TR as a filter device. If we wish to "filter" the printer using TRAP/FLT, we can use the command:

```
FILTER *PR *TR
```

After this has been done, our printer will be "filtered", and will "trap" all line feed characters (X'ØA').

Several additional points need to be made with respect to entering commands and parameters. The examples contained within the syntax blocks are precise in describing command entry. Spaces are necessary where shown. If more than one parameter is specified, a comma <,> must be used as a separator. After entering a SET command, if an error message appears (most commonly Parameter Error), check the entry of the command against that shown in the syntax block. If "multiple" spaces are used (where there should have been only a single space), or a space is used in the place of a comma, an error will occur, and you will need to re-enter the command.

Some filters (such as MAXLATE and KSMPLUS) require the use of a "filespec" when being SET. Filespec is the name of a disk file which contains data that will be used by the filter. Again, the syntax block precisely describes the method by which the filter is SET. In general, the filespec will follow the name of the filter program, and must be separated from the filter name by a space. If parameters are used, they must follow the filespec.

Suppose that you wish to use KSMPLUS2/FLT with a KSM data file named MYKSM/KSM. The following SET command will establish the KSMPLUS filter device (*KM).

```
SET *KM KSMPLUS2/FLT MYKSM/KSM (ENTER=X'25')
```

Of interest in the above command is the ENTER= parameter. With KSMPLUS, the ENTER parameter is used to specify the character which represents an "embedded enter". In this case, a hexadecimal value was passed (X'25'). This value could also have been passed as a decimal value, or as the character enclosed in quote marks. The next two commands will produce results identical to the command above.

```
SET *KM KSMPLUS2/FLT MYKSM/KSM (ENTER=37)
SET *KM KSMPLUS2/FLT MYKSM/KSM (ENTER="%")
```

The valid parameter entries will be listed in the syntax block for each filter/utility in this package.

C A L C / F L T

CALC/FLT is a keyboard filter which allows Hex/Decimal/Binary conversions to be performed, as well as Hex addition and subtraction. The syntax is:

```
To Install CALC/FLT --

SET *ds CALC/FLT
FILTER *KI *ds


To Remove CALC/FLT --

RESET *KI
RESET *ds
CALC/FLT (REMOVE)

*ds is any valid TRSDOS device spec

abbr: REMOVE=R
```

I M P O R T A N T N O T I C E

CALC/FLT is designed as a keyboard filter only!! Use of this filter on any other device will cause unpredictable if not disasterous results.

CALC/FLT will convert Hex/Decimal/Binary numbers and display the representation of the converted number in the "base" specified (e.g. CALC/FLT will allow you to input a Hexadecimal number and will display the corresponding Decimal value). In addition, Hex addition and subtraction may be done.

After installation, CALC/FLT may be invoked by the depression of the key sequence <CLEAR><SHIFT><C>. Upon activating CALC/FLT, this prompt will appear on the top line of the video display:

Bd,Bh,C,D,H,M or exit

Now, a calculation/conversion command may be entered. The commands for CALC/FLT are:

Bdxxxxx	- Convert Decimal number xxxxx to Binary.
Bhxxxx	- Convert Hex number xxxx to Binary.
Cxxxxxxxx	- Convert Binary number xxxxxxxx to Hex.
Dxxxx	- Convert Hex number xxxx to Decimal.
Hxxxxx	- Convert Decimal number xxxxx to Hex.
Mxxxx-yyyy	- Subtract Hex number yyyy from Hex number xxxx.
Mxxxx+yyyy	- Add Hex number yyyy to Hex number xxxx.
<CLEAR><SHIFT><C>	- Exit CALC/FLT and return to current procedure. The screen display will be restored.

Any command entered at the CALC/FLT prompt may be either upper or lower case. Commands cannot contain extraneous spaces or invalid characters. If an invalid conversion/calculation command is entered, an error message will be displayed. After a command has been entered, the result will be displayed. Pressing <ENTER> will return you to the CALC/FLT command prompt.

The following rules and restrictions describe the entry of numeric values and the corresponding values returned:

Binary Data Entry - Up to 8 binary digits (Ø's or 1's) may be entered.

Hex Data Entry - Up to 4 hex digits may be entered. When converting to Binary, either 8 or 16 Binary digits will be displayed, depending on the magnitude of the Hex value entered. If you are converting a Hex number to Decimal and the value entered is greater than X'7FFF', two Decimal values will be returned. They will represent the signed and unsigned decimal representation of the Hex value. If an Addition calculation causes a "carry", or a Subtraction calculation causes a "borrow", the value returned will be represented as "modulo 65536".

Decimal Data Entry - Up to 5 decimal digits may be entered. If a convert to Binary is done, the number entered must be in the range of Ø to 65535. When converting to Hex, the decimal value entered should be in the range of -32768 to 65535.

The following table shows example CALC/FLT commands and the returned results.

CALC/FLT command	Result
Bd255	11111111
Bh1Ø1	ØØØØØØØ1 ØØØØØØØ1
CØ1ØØ11	ØØ13
D1FF	511
D8ØØ1	32769 -32767
H-256	FFØØ
H1Ø24	Ø4ØØ
H65535	FFFF
MFØØØ-FF	EFØ1
M1ØØ-1Ø1	FFFF
M7FFF+7FFF	FFFE
M1Ø+FFF1	ØØØ1

When removing CALC/FLT from high memory, the following points should be noted. The procedure for removal must be followed in the exact sequence shown in the syntax block. Doing a Reset of the CALC/FLT device prior to resetting *KI will cause a system "hang up". If CALC/FLT is the last module in high memory, removing it will restore all memory used.

If CALC/FLT is "trapped" in memory (i.e. some other high memory filter or driver was installed after the installation of CALC/FLT), an informative message will appear indicating that this is the case. If it is desired to re-install CALC/FLT after it had been trapped, the normal installation procedure should be used, and CALC/FLT will re-use its original memory assignment. Please note however, that the orginal device name that was assigned to CALC/FLT should NOT be removed from the device table. This is due to the fact that the re-installation procedure requires the device name (and location of the device in the device table) to be the same as it was during initial installation.

K S M P L U S / F L T

KSMPLUS/FLT is a keyboard filter which adds several features to the standard KSM/FLT found in TRSDOS 6. The main enhancements are: the ability to edit the contents of a currently active KSM key and the ability to define the function and shifted function keys. Additional commands have also been added to display the current date and time, repeat the last DOS command and send a top-of-form character to the printer. The syntax for installing and removing KSMPLUS is as follows:

```
To Install KSMPLUS/FLT:

SET *ds KSMPLUSn/FLT filespec (parm,parm)
FILTER *KI *ds


To Remove KSMPLUS/FLT:

RESET *KI
RESET *ds
KSMPLUSn/FLT (REMOVE)

*ds is any valid TRSDOS device spec

n represents the KSMPLUS module to use (1, 2 or 3).

filespec is any KSM-type data file. If no file extension
  is used with filespec, a default extension of /KSM
  will be assumed.

Optional Parameters:

SPACE=ddd     Amount of additional space to reserve in
              the KSM data area for dynamic key
              definition. It may be entered as a decimal
              value or as a hexadecimal value in the
              form X'nnnn'. Not valid with KSMPLUS
              module 1. The default is 32.

ENTER=x       Character to use as the embedded <ENTER>
              in a KSM key definition. It may be entered
              as a character within quotes, as a decimal
              value or as a hexadecimal value in the
              form X'nn'. The default is the semi-colon
              character <;>.

abbr: SPACE=S, ENTER=E, REMOVE=R
```

I M P O R T A N T N O T I C E

KSMPLUS/FLT is designed as a keyboard filter only!! Use of this filter on any other device will cause unpredictable if not disasterous results.

There are three KSMPLUS/FLT modules included with this package. They are named KSMPLUS1/FLT, KSMPLUS2/FLT and KSMPLUS3/FLT. Here are the functions of each KSMPLUS/FLT module:

KSMPLUS1/FLT - Includes 4 special key sequences. These will allow date or time strings to be generated, a top of form character (X'ØC') to be sent to the printer, and a repeat of the last DOS command. Also, the number of KSM keys has been expanded from 26 (<CLEAR><A> through <CLEAR><Z>) to 32. The extra 6 KSM keys are accessible via the function and "shifted" function keys (<F1> through <F3>).

KSMPLUS2/FLT - Includes all features of KSMPLUS1/FLT, plus the ability to re-define any KSM key.

KSMPLUS3/FLT - Includes all features of KSMPLUS2/FLT, plus a "video restore" is done after a key re-definition.

KSMPLUS1/FLT

KSMPLUS1/FLT adds the following key functions to the standard TRSDOS KSM/FLT.

<CLEAR><SHIFT><X> - Repeat last DOS command.
<CLEAR><SHIFT><T> - Send a top of form (X'ØC') to the printer.
<CLEAR><SHIFT><Z> - Generate and pass through the keyboard driver an 8 character Date string in the form mm/dd/yy.
<CLEAR><SHIFT><S> - Generate and pass through the keyboard driver an 8 character Time string in the form hh:mm:ss.
<F1> through <F3> - Additional KSM key definitions. Up to 32 KSM keys may be defined. If more than 26 keys are defined, the 27th through 32nd KSM key definitions can be invoked by use of the <F1> through <F3> keys (for KSM key numbers 27, 28 and 29, respectively) and their "shifted" counter-parts (for KSM key numbers 3Ø-32). If no assignment is made to a KSM key, its function will remain unchanged.

Using TRSDOS KSM, 26 KSM keys (<CLEAR><A> through <CLEAR><Z>) are available. With KSMPLUS, up to 32 key definitions can be made. If used, the data definitions for the function keys must follow the data definition for the <CLEAR><Z> key (i.e. they must follow the 26th carriage return in the KSM data file). Please note that if no data is defined for a KSM key, the key depression sequence will return the standard key value. For example, if there is no KSM data assigned to the <A> key and <CLEAR><A> is pressed, the value X'C1' (decimal 193) will be returned.

To create a KSM data file, it is recommended that a text editor (such as LS-LED) be used. If it is necessary to use the TRSDOS BUILD command, the following points should be noted. Normally, when the BUILD command determines that the file to be built has an extension of "KSM", prompts will appear for each of the KSM keys A-Z. Once the definition for the <Z> key has been made, the BUILD operation will be terminated. When using BUILD to create a KSMPLUS data file with key assignments for the function keys, use BUILD in the normal manner to define the keys A-Z. Now use BUILD again with the same filespec and the APPEND parameter. KSM key prompts will appear again, starting with A. Entries made for the keys A-F will correspond to the function and shifted function keys of KSMPLUS.

The KSMPLUS <CLEAR><SHIFT><X> command differs from the TRSDOS <CTRL><R> command (repeat last DOS command) in that the command will be displayed, but not automatically executed. All characters of the last DOS command will be passed through the keyboard driver and displayed on the video, with the cursor positioned after the last character of the command. It is only valid as the first keyboard entry at the TRSDOS Ready prompt.

KSMPLUS2/FLT

In addition to the features noted in KSMPLUS1/FLT, KSMPLUS2/FLT will respond to the key sequence <CLEAR><SHIFT><E>. This will cause the "Edit KSM Key" mode to be invoked. After this key sequence, the top two lines of the video will be cleared, and a plus sign <+> will appear in the upper left corner. This indicates that the edit mode is active.

One of two actions may be performed at this point. Depressing the key sequence <CLEAR><SHIFT><E> again will terminate the edit mode, and you will be returned to the currently active process (i.e. the currently running program or TRSDOS Ready).

Pressing any alphabetic key (A through Z) by itself (without the <CLEAR> key), or a numeric key 1-6, will select the key to be re-defined. The plus sign will be replaced by the character pressed, followed by an equal sign <=>. At this stage, the given KSM key definition may be edited. The numeric keys 1-6 will allow editing of the function and shifted function keys, with 1 corresponding to <F1> and 6 corresponding to <SHIFT><F3>. During any part of the editing, the key sequence <CLEAR><SHIFT><E> may be depressed to terminate the edit and leave the KSM key definition unchanged.

While editing a KSM key, any keyboard character/sequence with the exception of <ENTER>, <LEFT ARROW> and <CLEAR><SHIFT><E> may be entered, and may be assigned to the KSM key (the <BREAK> key is a valid character in most cases). <ENTER> will terminate the edit and assign the information after the equal sign to the KSM key. A KSM key can be "eliminated" by entering the KSM key edit mode, pressing the desired key (A-Z, 1-6), and pressing <ENTER> (i.e. assign "No data" to the KSM key).

The <LEFT ARROW> key may be used as a backspace.

If a KSM key sequence is pressed during a key edit, the characters assigned to that key will be entered for the key re-definition. For example, to add data to the <A> KSM key (which already has data assigned to it):

1) Depress <CLEAR><SHIFT><E> to enter KSM edit.
2) Press <A> to edit the <A> KSM key.
3) Press <CLEAR><A> to assign the "current" data to the <A> KSM key.
4) Type in any additional information.
5) Press <ENTER> to save the new definition.

Some consideration must be made regarding the SPACE parameter (see the installation procedure in the syntax block). When a KSM key is edited, all keys entered are stored in a temporary buffer whose length is dictated by the SPACE parameter. The size of the SPACE buffer changes according to any edits which are made.

For example, suppose the SPACE parameter was defined to be 50, and you wished to "add on" to a key definition whose length was 45. Although there are 50 characters of "free space", the maximum amount of characters that could be added on to the current definition would be 5 (using the above procedure). Furthermore, if such an addition was made, and it was desired to "add on" again to the same KSM key, only the original 45 characters of the key definition could be displayed and entered. This is because the first addition caused the "free space" to shrink to 45 characters.

Two ways around this situation are: 1) The SPACE parameter can be specified to be "large" enough to hold the longest KSM key definition, taking into account any edits that might be made. In some cases this may not be practical, since the larger the SPACE buffer, the more memory KSMPLUS will take up.

2) First assign "No data" to one (or more) KSM keys which will no longer be used. This has the effect increasing the available SPACE buffer. Once enough space has been freed up, the addition can be made to the KSM key in question.

As a final note concerning the use of KSMPLUS2, the maximum length of an edited KSM key is 156 characters. There is no restriction to the length of a key definition loaded from a disk file.

KSMPLUS3/FLT

In addition to the features noted in KSMPLUS2, KSMPLUS3/FLT will restore the top two lines of the video upon completion of an edit, and will return the cursor to where it was prior to the edit. With KSMPLUS2/FLT, the top two lines of the video will be lost when a KSM key is edited.

Removal and Re-installation of All KSMPLUS modules

When removing KSMPLUS/FLT from high memory, the procedure for removal must be followed in the exact sequence shown in the syntax block. Doing a Reset of the KSMPLUS/FLT device prior to resetting *KI will cause a system "hang up". If KSMPLUS/FLT was the last module placed in high memory, removing it will restore all memory used.

If KSMPLUS/FLT is "trapped" in memory (i.e. some other high memory filter or driver was installed after the installation of KSMPLUS/FLT), an informative message will appear indicating that this is the case. If it is desired to re-install KSMPLUS/FLT after it had been trapped, the normal installation procedure should be followed, and KSMPLUS/FLT will re-use its original memory assignment. Please note however, that the orginal device name that was assigned to KSMPLUS/FLT should NOT be removed from the device table. This is due to the fact that the re-installation procedure requires the device name (and location of the device in the device table) to be the same as it was during initial installation.

When replacing KSMPLUS1 data files (when KSMPLUS1 is "trapped" in memory), the same restrictions which apply to TRSDOS KSM should be noted (i.e. the data length must be less than or equal to the originally installed data length). When replacing trapped data files for KSMPLUS modules 2 and 3, the "replaced" data length must be less than or equal to the quantity - original data length + SPACE. In the case of re-installation, SPACE need not be specified.

Only one of the KSMPLUS modules may be resident in memory. If you mistakenly try to install two KSMPLUS modules, an error message will be displayed.

In terms of the memory consumed by the KSMPLUS modules, the following list gives a rough approximation for each module.

```
KSMPLUS1   - 250 bytes + data length.
KSMPLUS2   - 500 bytes + data length + SPACE.
KSMPLUS3   - 750 bytes + data length + SPACE.
```

Example

To install a KSM data file named MYKEYS/KSM with KSMPLUS module 2:

```
SET *ds KSMPLUS2 MYKEYS
FILTER *KI *ds
```

M A X L A T E / F L T

MAXLATE/FLT is an input/output filter which performs a translation of specified characters to character streams. Individual characters may be translated into multi-character sequences. The syntax is:

```
SET *ds MAXLATE/FLT filespec (parm)
FILTER *fd *ds

*ds is any valid TRSDOS device spec.

*fd is the device to be filtered.

filespec is the file containing the translation table.
    If filespec contains no extension, /MLT will be the
    default.

Parameters: INPUT or OUTPUT

    Used to specify the direction of the translation.
    Exactly one must be specified. The direction must
    correspond to that of the filtered device (if one
    directional) or to the direction of the translation
    (if the filtered device is bi-directional).

abbr: INPUT=I, OUTPUT=O
```

MAXLATE/FLT is a translation filter which may be applied to any type of device. Individual characters may be translated into multiple character sequences. The maximum length for each translation stream is 254 characters. The translations to be performed are defined in a translation table.

Setting Up a Translation Table

To facilitate the creation of translation tables, it is recommended that a text editor (such as LS-LED) be used. If a text editor is not available, the TRSDOS BUILD Library command may be used.

Any translation defined in the table must follow the format K=T, where K is the key to be translated and T is the resultant translation stream. For the most part, the translation table can be built in a free format, where the characters may be specified as either hexadecimal values or the actual character contained within quote marks. One important restriction is that every defined translation must be terminated by a carriage return. The best way to illustrate the format of a translation table is through the use of examples.

Suppose that it was desired to set up a translation table for the Printer Device (*PR). The purpose of the translation is to send to the printer the character stream "Ø (zero)" each time the character <Ø> was encountered, and the character stream - "O (oh)" for all occurrences of the capital letter <O> (oh). The following is a sample translation table that would accomplish this.

```
3Ø = "Ø (zero)"
4F = "O (oh)"
```

The above translation table illustrates the format to be used in building a translation table (K=T). In the first line, the <3Ø> is the hexadecimal value associated with the character <Ø>. The translation stream appears to the right of the equal sign, and is contained within quote marks. Notice that there are spaces separating the various elements of each translation line (i.e there is a space between the equal sign and the first quote mark). Although not required, the translation table may be constructed in this manner to allow easier reading. However, the space appearing between the <Ø> and the open parenthesis <(> will become part of the translation, since it is within the quotes.

The first line of the above translation table could also have been represented as:

```
"Ø" = 3Ø,2Ø,28,7a,65,72,6f,29
```

In this example, the translated character is contained within quotes, while each character in the translation stream is represented in hexadecimal format. Notice that commas are used as separators in the translation stream. Any character may be used as a separator, with the exception of a quote mark or a digit (Ø-9).

Let us consider one more example of establishing a translation table. Suppose once again that we wish to set up a translation table for the printer. This time we are blessed with the luxury of having a printer that will backspace (when the character X'Ø8' is sent to the printer). One of our objectives is to print a slash </> through each zero <Ø> encountered. Furthermore, we know that the character X'ØF' will cause our printer to do strange and mysterious things, and so we do not wish to send that character to the printer. The following table could be set up to accomplish this.

```
"Ø" = "Ø" Ø8 "/"
ØF =
```

In the first line of the translation table, notice the intermixing of character strings in quotes and hexadecimal values. The free format of MAXLATE translation tables will accomodate this type of setup. Also notice that the backspace character (X'Ø8') is specified as two hexadecimal digits. In this case, the leading <Ø> is required, as all hexadecimal specifications must have two digits.

The second line of the translation table demonstrates the method of "filtering out" unwanted characters. Any time a X'ØF' is encountered, "no character" will be sent to the printer. Please note the distinction between "no characater" and a "null" character. If it was desired to send the character X'ØØ' to the printer in place of X'ØF', the following translation line should be used.

```
ØF = ØØ
```

Installation of MAXLATE

Once a translation table has been established, follow the command sequence shown in the syntax block. Let us assume that we have a translation table stored on disk in a file named PRT/MLT. To set up a MAXLATE translation, the following command sequence could be used.

```
SET *PM MAXLATE/FLT PRT (OUTPUT)
FILTER *PR *PM
```

From the above commands, *PM is the filter device which we are creating. Since MAXLATE uses as a default extension "MLT", we do not need to specify the extension for the translation file. The OUTPUT parameter is required, since the printer is an output device and we wish to filter characters going "from" the computer "out" to the printer. The FILTER command applies the MAXLATE filter to the printer device.

NOTE: If you are not sure whether a device is OUTPUT, INPUT or bi-directional, use the following TRSDOS Library command.

DEVICE (B=Y)

Along with other information, a device table will be displayed, listing all currently active devices. The following symbols will appear after each device name, and will show the capabilities of the device.

=>	Device is capable of Output only
<=	Device is capable of Input only
<=>	Device is bi-directional

If at any time you wish to temporarily suppress the translation process, perform a RESET of the device being filtered (in the above example, RESET *PR). Please note that the translation filter will be resident in memory, but inactive. To re-establish the translation process, simply re-use the FILTER command (the SET command is not required).

Practical Use of MAXLATE

The implementations of and uses for MAXLATE are wide and varied. It would be difficult to cover all possible instances where MAXLATE could be a viable answer to a problem. Let's look at one possible use of the MAXLATE filter with the communications line (*CL).

Let us suppose that is it desired to send and receive files through the communications line (*CL). We will be dealing with ASCII document files. Our main concern is centered on the "termination" of a physical line. On our end, each physical line is terminated by a carriage return (X'ØD'). On the other end, each line is terminated by a carriage return and a line feed (X'ØD' and X'ØA', respectively). When we send a document, we wish to have all carriage returns changed to a carriage return and a line feed. When we receive a document, we wish to have two carriage returns at the end of each physical line (for double spacing purposes).

This problem suggests the use of two MAXLATE translations; one of which installed on the comm line as an input device, the other handling output. We need to create two translation table files. The first one (named COMIN/MLT) will handle the translations as files are received. The second (named COMOUT/MLT) will perform translations on files which are sent. The listings below depict the two translation files.

COMIN/MLT	COMOUT/MLT
ØD = ØD ØD	ØD = ØD ØA
ØA =	

Once the translation files have been created, it is but a matter of establishing our filter devices and filtering the comm line. If we pick the device names *CI (input filter) and *CO (output filter), the following commands may be used to achieve our final goal.

```
SET *CI MAXLATE COMIN (I)
SET *CO MAXLATE COMOUT (O)
FILTER *CL *CI
FILTER *CL *CO
```

P R C O D E S / F L T

PRCODES/FLT is a printer filter which will allow several printer control functions to be performed. These functions include sending a zero, a "backspace character" sequence and a slash </> every time a <Ø> (zero) is encountered, and allowing a toggle for bold faced and underlined printing. This filter will function only on printers capable of true backspacing. The syntax is:

```
SET *ds PRCODES/FLT (parm,parm)
FILTER *PR *ds

*ds is any valid TRSDOS device spec.

Parameters:

BACK="xxxx"  Specifies the character sequence to use
             for a backspace character. Must be
             specified as an ASCII string enclosed in
             quotes, and will represent the Hex
             value(s) used to generate a backspace.
             The default is "Ø8".

BOLD=n       Character used to toggle bolded print on
             and off. It may be entered as a decimal
             number, as the character enclosed in quotes
             or as a hexadecimal value in the form X'nn'.
             The default is X'7F' (decimal 127).

STRIKE=n     Decimal value which determines the number
             of times to strike a character for bold
             printing. The default is 3.

UNDER=n      Character used to toggle underlined print
             on and off. It may be entered as a decimal
             number, as the character enclosed in quotes
             or as a hexadecimal value in the form X'nn'.
             The default is X'5F' (decimal 95).

abbr: BOLD=B, STRIKE=S, UNDER=U.
      BACK cannot be abbreviated.
```

PRCODES/FLT is a printer filter designed to produce **BOLDED** and <u>UNDERLINED</u> print effects. Additionally, every time a <Ø> (zero) is encountered, PRCODES will backspace and print a </> (slash) through the zero. Due to the manner in which PRCODES functions, your printer MUST have backspace capabilities. PRCODES should NOT be used with printers that are incapable of backspacing or use backspace merely to delete the last character in the print buffer.

The BACK Parameter

Due to differences in printer specifications, the BACK parameter is included in PRCODES to allow the specification of a character sequence which will produce a printed "backspace". If your printer recognizes the character X'Ø8' as a backspace, the BACK parameter need not be specified.

On some printers, a multi-character sequence is required to generate a backspace. The following example will illustrate the proper use of the BACK parameter for such printers. Please refer to your printer manual for exact specifications.

Let us assume that in order to print a backspace, your printer needs to receive the 2 character sequence of Escape and Backspace (X'1B' and X'Ø8, respectively). The following command must be used to establish the PRCODES filter device.

```
SET *ds PRCODES (BACK="1BØ8")
```

The character sequence used to produce a backspace must be enclosed within quote marks. For each "printed" character in the sequence, a two-character hexadecimal value must be specified (in the case of the "Ø8", the leading <Ø> is required). Extraneous characters (e.g. spaces used as separators) CANNOT appear in the backspace "string".

Use of Bold and Underlined print

As the default, PRCODES uses the characters X'7F' and X'5F' as "toggles" to turn on/off Bold and Underlined printing. The first occurrence of either of these characters will "turn on" the desired function. The function will remain "on" until the "toggle" character is encountered again, at which time the function will be "turned off". Both Bold and Underline may be active at the same time. The toggle characters can be generated through the keyboard using the following key sequences.

```
X'7F'  -  Bold ON/OFF       - <CLEAR><SHIFT><ENTER>
X'5F'  -  Underline ON/OFF  - <CLEAR><ENTER>
```

If toggle characters other than the defaults are desired, the appropriate parameter (i.e. either BOLD= or UNDER=) must be specified when the PRCODES filter device is established.

As an example, suppose that you wish to use PRCODES to print text containing underscore characters (X'5F'). It is desired to have these characters printed "as is" (normally an underscore would toggle underlining on and off). In this case, a different character must be used as a toggle for underlining. If the "vertical bar" character (X'7C') was chosen as the toggle character to replace the underscore, the following command could be used to establish the PRCODES filter device.

```
SET *ds PRCODES (U=X'7C')
```

The STRIKE Parameter

The STRIKE parameter is used to tell PRCODES the number of times to "overstrike" a character for bold printing. If not specified, the default will be 3 (which is the strike value used to print this document). Theoretically, a value up to 255 could be used. In practice however, the strike value should not exceed 1Ø, since repeated overstriking can literally "punch" a hole in the paper.

Use of PRCODES with TRSDOS Printer Options

If it is desired to use PRCODES with the TRSDOS printer filter (FORMS/FLT), the printer device should be filtered with PRCODES first, followed by FORMS. The following commands will illustrate the proper sequence of installation.

```
SET *ds PRCODES/FLT
SET *PF FORMS/FLT
FILTER *PR *ds
FILTER *PR *PF
```

If PRCODES is to be used with the TRSDOS print Spooler, the Spooler should be installed first.

If all three are to be present at the same time, they should be installed in the order of the Spooler, followed by PRCODES, followed by FORMS.

When using PRCODES and FORMS, the best results will be obtained when the "Maximum Characters per Line" parameter of the FORMS filter is OFF (i.e. in its default state). If a value is specified for this parameter, the actual number of characters printed will be less than the maximum when Bold and Underline "toggles" are encountered.

Miscellaneous

Use of PRCODES with printers that accept control sequences to enable/disable Bold and Underline should present no problem. If you are using such a printer, you may wish to examine the possibility of using the internal printer capabilities to produce the desired result. This type of application may be approached by using the MAXLATE filter. For more information, refer to the MAXLATE/FLT section of this document.

If you wish to temporarily suppress PRCODES (i.e. you do not wish to print bold, underline or slashed zeroes) simply perform a RESET of the printer (*PR). To re-enable PRCODES, use the FILTER command as shown in the syntax block (the SET command need not be done).

NOTE

Performing a RESET of the PRCODES filter device will cause the filter to be permanently resident in high memory. Futhermore, it will not be usable, and you will not be allowed to re-establish the filter. For these reasons, it is recommended that the PRCODES filter device NEVER be RESET.

R D T E S T / C M D

RDTEST/CMD is a utility program which will perform a read test on a formatted diskette. Since the read test is non-destructive, the diskette may contain data. The entire diskette will be read, and any read errors that are found will be reported. The syntax is:

```
RDTEST :d (parm)

:d is the drive containing the diskette to be tested.
   If not specified, it will be prompted for.

Parameters:

P   Send any informative messages to the printer as well
    as to the video. If not specified, any message will
    be sent to the video only.

T   Specify number of consecutive times to perform a
    read test on a diskette. The default is 1.

abbr: None
```

RDTEST will read all sectors on a diskette, starting at cylinder Ø, and proceding to the highest numbered cylinder. As the read test is being done, the cylinder/sector number which is currently being tested will be displayed on the video.

Any read error encountered will be reported, and the display will consist of the cylinder number and sector number of the "bad read". Please note that RDTEST will read the entire diskette, including any "locked out" cylinders. If the T parameter is used, the diskette will have that many entire read tests performed on it. If at any time it is desired to terminate the read test, press the <BREAK> key.

R E A D 4 Ø / C M D

READ4Ø/CMD is a utility/driver program designed to allow 5 1/4 inch diskettes which have been formatted in a 35/4Ø track drive to be read by an 8Ø track disk drive. When used, a high memory disk driver will be installed to perform the operations. The syntax is:

```
READ4Ø :d (parm)

:d is the 8Ø track drive to use for the read. It MUST be
   a 5 1/4 inch - 96 Track/Inch (96 TPI) drive. It can
   be any enabled drive, with the exception of :Ø.
   If not specified, it will be prompted for.

Parameters:

TABLE     Display a table of all drives and associated
          driver information. If specified, READ4Ø will
          NOT be installed in high memory.

REMOVE    Remove the READ4Ø driver from high memory.
          :d cannot be specified when removing READ4Ø.

abbr: TABLE=T, REMOVE=R
```

I M P O R T A N T N O T I C E

READ4Ø/CMD will install a special disk driver into high memory to allow reading 35/4Ø track diskettes in an 8Ø track drive. Only one drive may have a READ4Ø driver attached to it. Once installed, the drive in question will be software Write Protected, to assure that no "writes" are performed during the duration of the READ4Ø operation. Under NO circumstances should any write be performed to a 4Ø track diskette in an 8Ø track drive using READ4Ø. Attempting to do so WILL cause the diskette in question to be unusable. Furthermore, the 4Ø track diskette to be read should be of compatible format with TRSDOS 6 (such as an LDOS 5.1 diskette). READ4Ø will allow you to read a MODEL III TRSDOS 1.2/1.3 diskette, using the TRSDOS CONV utility. Any diskette that needs to be REPAIRed cannot be used with READ4Ø (unless it is REPAIRed on a 4Ø track drive prior to being used with READ4Ø).

Under normal operating conditions, 4Ø track diskettes (i.e. diskettes that were formatted on either 35 or 4Ø track drives) cannot be read on an 8Ø track drive, and vice versa. READ4Ø will allow reading a 4Ø track diskette in an 8Ø track drive (but NOT vice versa).

It is a simple matter to use READ4Ø. All that is needed is to enter a command similar to the one shown in the syntax block. Please note that the drivespec entered should correspond to an 8Ø track drive. The drive in question must be enabled prior to installing READ4Ø.

Once READ4Ø has been installed, you may place a 4Ø track diskette in the 8Ø track (READ4Ø) drive, and perform any read operation (such as DIR, BACKUP, COPY or CONV if the diskette is of MODEL III TRSDOS 1.2/1.3 format).

READ4Ø has a built in I/O error monitor. If a read error is detected on the READ4Ø drive, an informative message will appear, and you will be prompted for the desired action to be taken. At this point you may:

<A> - Abort the operation, and return to TRSDOS Ready.

<C> - Continue the operation. The error will be returned to the currently active process, and will be handled accordingly.

<I> - Ignore the error. A return will be made to the currently active process without an error indication. If an Ignore is done during a file move operation, the integrity of the data in the destination file may be suspect.

<R> - Retry the operation. The disk operation causing the error will be retried. If the retry is successful, a return will be made to the currently active process. If it is not successful, you will be returned to this prompt.

NOTE

While READ4Ø is active, if an 8Ø track diskette is placed in a READ4Ø drive, the drive may be seen as containing "No Disk". This is normal. If such a situation occurs and a 4Ø track diskette is then placed in the drive, it may not initially be readable by READ4Ø, and the first access of it may invoke the I/O monitor noted above. At this point, the prompt should be answered with <C> (continue), and the 4Ø track diskette will then be "logged" in for proper operation.

Use of READ4Ø Parameters

If READ4Ø was the last module placed in high memory, the space consumed by it may be reclaimed using the REMOVE parameter. As an example, if drive 5 was established as a READ4Ø drive, and it was desired to remove READ4Ø, the following command could be entered.

READ4Ø (R)

After execution of this command, drive 5 would be restored to normal 8Ø track operation. Note that no drivespec was used, since only one READ4Ø drive may be established. Entering a drivespec with the REMOVE parameter will cause a Parameter error.

If READ4Ø is "trapped" in high memory, an attempt to remove it will cause an informative message to be displayed, indicating that it could not be removed from high memory. At this point, READ4Ø will be inactive, and the 8Ø track drive involved will function as normal. READ4Ø may be re-installed and will utilize the same high memory allocation.

If READ4Ø is entered with the TABLE (T) parameter, an informative drive table will be displayed, showing all drives and associated drivers. If the T parameter is specified, READ4Ø will NOT be installed. Any drivespec supplied on the command line will be ignored.

T R A P / F L T

TRAP/FLT is an input/output filter which will "trap" for a specified character. The trapped character will be discarded. The syntax is:

```
SET *ds TRAP/FLT (parm,parm)
FILTER *fd *ds

*ds is any valid TRSDOS device spec.

*fd is the device to be filtered.

Parameters:

INPUT/OUTPUT   Used to specify the direction of the
               trapping. At least one must be used. The
               direction must correspond to that of the
               filtered device (if uni-directional) or
               to the direction of the trapping (if the
               filtered device is bi-directional). Both
               INPUT and OUTPUT may be used to trap for
               the same character in both directions if
               the device is bi-directional.

CHAR=nn        Specifies the character to "trapped".
               nn may be entered as a decimal number, as
               the character enclosed in quotes, or as a
               hexadecimal value in the form X'nn'.

abbr: CHAR=C, INPUT=I, OUTPUT=O
```

TRAP/FLT may be used with any device to trap a character. If multiple characters need to be trapped, TRAP/FLT must be installed as many times as the number of characters to be trapped. Each installation of TRAP will cause a new high memory allocation to be made. If you need to trap several characters, you may wish to use MAXLATE/FLT.

Using TRAP/FLT on an Output Device

One use for the TRAP/FLT would be when listing a file to the screen. If the "control" character X'17' (decimal 23) is passed to the video device (*DO), the "expanded" character mode (40 characters per line) will be enabled. To prevent this from happening, use following command sequence:

```
SET *ds TRAP (C=X'17',OUTPUT)
FILTER *DO *ds
```

Using TRAP/FLT on an INPUT Device

As another example, let us use TRAP/FLT to filter the Keyboard device (*KI), so that the percent key <%> will be ignored when pressed. These commands may be entered to do this (note that in this example, the character to trap is enclosed in quotes).

```
SET *kd TRAP (C="%",I)
FILTER *KI *kd
```

T Y P E I N / C M D

TYPEIN/CMD is a utility program designed to aid in performing "automated" procedures. Unlike TRSDOS Job Control Language (JCL), TYPEIN will allow single key entries to be passed to an application program, and be acted upon. With JCL, commands are acted upon only through the keyboard "lineinput" routine, and must be terminated with a carriage return (the BASIC statement INPUT is an example of such a command). TYPEIN allows single key requests and certain keyboard "scanning" situations to be satisfied (the BASIC function INKEY$ is an example of a keyboard "scan" request, and can be satisfied through TYPEIN). The syntax is:

```
TYPEIN filespec,filespec,... (parm,parm)

filespec is a file containing characters which will be
     acted upon by TYPEIN. Use of filespec is optional.
     If not specified, all characters entered will be
     stored in a "TYPEIN buffer" until <BREAK> is
     pressed, at which time they will be processed.
     More than one filespec may be used. If this is the
     case, the filespecs will be acted upon in the order
     of entry on the TYPEIN command line.

Parameters:

Xn=X'fftt'  -  Optional character translation to be
               performed by TYPEIN. n may be a value
               from 1 to 4 (i.e. up to 4 translations
               may be done). ff is the hex character to
               be translated. tt is the hex character to
               translate ff to.

LINES=n     -  Optional parameter used to specify the
               number of "lines" to be processed from
               the TYPEIN file. Each occurrence of a
               carriage return (X'ØD') signifies the end
               of a line. n lines will be processed by
               TYPEIN from the filespec, starting with
               the first line in the file. n must be a
               decimal number.

abbr: LINES=L
```

N O T I C E

Although a total understanding of JCL processing is not required to use TYPEIN, the basic function of JCL (that being to perform automated tasks) needs to be understood prior to using TYPEIN. Any questions regarding JCL processing and the limitations therein can be answered by consulting your TRSDOS Users manual. TYPEIN is NOT designed to replace JCL; rather, its function is to perform automated tasks which cannot be done with JCL. Also note that TYPEIN will NOT function with any program which scans the keyboard for certain "Abort" keys (in essence throwing away any non-abort entry), since this type of situation will "drain" the TYPEIN buffer/filespec.

The main advantage of using TYPEIN over JCL is that it can deal with keyboard "scans" and single key requests (for technical information, refer to the TRSDOS Technical Reference Manual - SVC's @KBD and @KEY). JCL will function only with programs containing "prompts" which require depression of the <ENTER> key to perform the action. Prompts which act "immediately" after the input of a key (i.e. without the <ENTER> key) cannot be controlled through JCL. These "immediate" prompts may be handled with TYPEIN (in many situations) to allow the procedure to be automated.

There are two methods in which TYPEIN may be used. The first method requires no TYPEIN filespec (i.e. just enter the command TYPEIN). Using this method, TYPEIN will "save" all ensuing keyboard entries without acting upon them. This will continue until the <BREAK> key is pressed, at which time all keystrokes will be processed.

The second method utilizes one or more filespecs containing the keystrokes to be processed. It is similar to JCL, except that keyboard scan requests may be satisfied.

NOTE

TYPEIN will only "process" keystrokes, and no JCL "macros" or enhanced features (such as //IF conditionals) are allowed. TYPEIN operates by installing itself into a temporary "high memory" allocation. For this reason, it should NOT be used to install high memory "drivers" or "filters". Using TYPEIN to perform such operations will cause TYPEIN to become "trapped" in high memory, with the result being wasted memory.

Using TYPEIN in the Direct Mode

Essentially all that is required to use TYPEIN in the direct mode is to enter the command TYPEIN at TRSDOS Ready. Once this is done, all keystrokes entered will be "saved" until break is pressed, at which time they will be acted upon.

As an example, let us use TYPEIN to perform a RDTEST (see RDTEST/CMD for more information). RDTEST is a program which can be controlled through a JCL, provided that the drivespec is included on the command line. However, if the drivespec is not supplied on the command line, RDTEST will prompt for it. The type of prompt that RDTEST uses cannot be handled with JCL (since it is an "immediate" prompt and <ENTER> is not required). However, TYPEIN may be used to supply this prompted information.

Suppose the following keystrokes were entered after TYPEIN was activated (note that each line, with the exception of the last, is terminated with a carriage return, and that the last line represents the depression of the <BREAK> key).

```
RDTEST (P)
1
<BREAK>
```

These commands will appear on the screen as they are entered. They will not be executed until the <BREAK> key is pressed, at which time they will be processed in the order entered. If the carriage return after the <1> was not entered (i.e. if <1> was pressed followed by <BREAK>), only the first line would have been processed. If <BREAK> is pressed before "Entering" a line, no characters on that line will be processed.

NOTE: RDTEST is a program which performs a "scan" of the keyboard for <BREAK>, and discards any non <BREAK> character. If additional commands would have been entered in the above example, they would not have been executed, since the TYPEIN buffer would be "drained" by RDTEST.

Use of TYPEIN with a Filespec

The basic concept of using TYPEIN in the direct mode also applies to using it with a filespec. Rather than entering the keystrokes from the keyboard, they are contained in a file.

A typical example might be to run a program which "prints" information (such as Mailing Labels). If the program cannot be automated through JCL, TYPEIN may provide the solution.

Assume that the program in question is a machine language program, named MAILPRT/CMD. In order to print labels, input needs to be taken for the print file to use and the number of times to repeat each label. Each of these prompts is "immediate", and must be answered by entering a digit (1-9). To exit the program, the letter <X> must be input and <ENTER> pressed.

To use the filespec method of TYPEIN, we need to have a disk file containing the program name (so that it will be executed) and the keystrokes needed by the program to perform the printing and return to DOS when complete. Below is a sample of the information contained in the file.

```
MAILPRT
52X
```

If these characters are contained in a file named MAILPRT/TYP, the following TYPEIN command will allow the MAILPRT program to print labels, using print file number 5, repeating each label twice. After printing is completed, the program will return to DOS.

```
TYPEIN MAILPRT/TYP
```

Use of the LINES= parameter

TYPEIN may be instructed to use only a specified number of lines from a file. This is accomplished through the LINES= parameter. The value entered will be the number of lines that TYPEIN will process. All TYPEIN processing will begin with the first line in the file, and continue from there in sequential line order. A "line" is denoted as ending with a carriage return (X'ØD').

Use of Multiple TYPEIN Filespecs

More than one filespec may be used with the TYPEIN command. If this is the case, each file used will be processed sequentially as appearing on the TYPEIN command line. If the LINES= parameter is specified with multiple filespecs, it will signify the total number of lines to be processed. For example, if File1 contains 1Ø lines, File2 contains 15 lines and File3 contains 2Ø lines, the following TYPEIN command will process File1 entirely and the first 5 lines of File2 (no lines will be processed from File3).

```
TYPEIN File1,File2,File3 (LINES=15)
```

Using TYPEIN within a JCL

Since TYPEIN is an executable utility program, use of it may be incorporated within a JCL. Assuming that the MAILPRT/TYP file is as defined previously, the following JCL file could be written to incorporate the use of TYPEIN.

```
.Example of using TYPEIN in a JCL
.
TYPEIN MAILPRT/TYP
RESET *PR
//EXIT
```

The results of using this JCL file (with the TRSDOS Library command DO) will cause all of the previously mentioned TYPEIN processing to take place. After TYPEIN is completed, the JCL processing will continue, and a Reset of the printer device will be done.

Use of TYPEIN with Translations

TYPEIN may be used to perform translations on characters that are passed to it. Up to 4 different translations may be done.

As a sample use, let us consider a situation where we have an ASCII text file (named TEXTA/TXT) which was created on a different machine and downloaded for use on a Model 4. We wish to list the document to the printer, but it contains a carriage return and a line feed (X'ØD' and X'ØA', respectively) as line termination characters. This will cause problems when printing the file, as we only require a carriage return to terminate the line.

One method of dealing with this situation is to use TYPEIN to "re-type" the document into a text editor, translating all line feeds to null characters. Assuming that we are using LS-LED as the text editor, we need to build a TYPEIN file which will run LED and type in the document, translating the line feeds along the way. We wish to name our TYPEIN file RMVLF/TYP. It will look something like this.

```
LED TXTNOLF/TXT
```

The file TXTNOLF/TXT will be the resulting file (created by LED) which will have all line feeds removed.

At present, our TYPEIN file will allow us to enter LED. What is needed now is to re-type the file into LED, using TYPEIN. This can be accomplished by APPENDing the orginal text file (TEXTA/TXT) to the end of our TYPEIN file. To do this, the following command can be used.

```
APPEND TEXTA/TXT RMVLF/TYP
```

The TYPEIN file now contains a command to run LED and the entire document to be typed. It is merely a matter of using TYPEIN to remove the line feeds. The command shown below will perform the desired function.

```
TYPEIN RMVLF/TYP (X1=X'ØAØØ')
```

As the process is being performed, the entire text file will be typed into LED. All line feeds will be changed to nulls (which will be "ignored" by LED).

LS-UTILITY

* * * WARRANTY * * *

This software program(s) is warranted to perform as documented when used on the specified hardware operating under the specified disk operating system as shown on the accompanying documentation. If within 90 days of the date of purchase the program is found to be defective due to a bug in the code, the publisher will, upon request, provide a patch to correct the bug or will update the program diskette with a corrected copy within a reasonable time period after return of the program diskette to the publisher. If within 90 days of the date of purchase the documentation proves defective due to missing pages, the publisher will provide substitutes for the missing pages upon request.

The publisher shall have no liability or responsibility to the purchaser or any other person, company, or entity with respect to any liability, loss, or damage caused or alleged to have been caused by this product, including but not limited to any interruption of service, loss of business and anticipatory profits, or consequential damages resulting from the operation or use of this program.

* * * ATTENTION * * *

MISOSYS, Inc.
P.O. Box 239
Sterling, Virginia 22170-0239
703-450-4181