

MHT-Search Xtra *Full-text Search Engine for Director*

Instruction Manual

Meetinghouse Technologies

**April 17, 1997
Version 2.0**

Introduction

About MHT-Search

MHT-Search is an extremely fast and powerful full-text search engine Xtra for use with Director 5 and 6. MHT-Search has been featured in commercially available reference titles such as "The War In Vietnam" CD-ROM, where it was used to search thousands of newspaper articles, government documents, weapons specifications, video transcripts, etc.

This professional level Xtra allows for:

- Keyword searching
- Phrase searching
- Multiple keyword and phrase searching
- Boolean operations
- Key ranges to define limited area searching
- Whole word or Partial word search capability (**New** with version 2.0)

The enclosed MHT-Search archive contains the following items for Macintosh computers using System 7.0.1 or higher and Windows 3.1/95/NT computers:

| File | Macintosh Name | PC Name |
|---|---|---|
| <i>MHT-Search Pre-processor</i> | MHT-Search PreProcess (v1.1) | MHTPreProc.EXE (v2.0) |
| <i>MHT-Search Xtra</i> | MHTSearch | MHTSrch.x16, MHTSrch.x32 |
| <i>MHT-Search Demo</i> | MHTSearchDemo.DIR, plus Reflist, Index & Articles Folders | MHTSearchDemo.DIR, plus Reflist, Index & Articles Directories |
| <i>Evaluation and License Agreement in Adobe Acrobat format</i> | MHT Xtras License | MHTXtrasLicense.PDF |
| <i>This manual in Adobe Acrobat format</i> | MHT-Search Manual | MHTSearchManual.PDF |
| <i>"Readme" file</i> | Readme! | ReadMe.TXT |

Cost, Use, & Registration

Please refer to the Evaluation and License Agreement for a complete and formal understanding of the terms and guidelines that apply to the use of MHT-Search.

MHT-Search may be incorporated in commercially distributed products for a one-time license fee of \$495. (Previous licensees of the MHT-Search Xobject may purchase the Xtra for \$295.)

The unregistered version of MHT-Search included on this archive may be evaluated free of charge. Its functionality is restricted only when used in a Projector.

After purchasing a license for MHT-Search, the Registration Code you receive will activate MHT-Search for both Windows and Macintosh Projectors.

If you wish to license MHT-Search, please follow these instructions fully:

- Print out the Evaluation and Title License Agreement PDF's using Adobe Acrobat.
- Complete the "Licensee" sections on the first and last pages, filling in the appropriate information:
 1. Check which Xtra(s) you wish to license.
 2. "By", "Name", "Title", and "Effective Date"
- Mail the completed License Agreement to Meetinghouse Technologies, along with the appropriate license fee (check or money order -- credit card purchases are available via the web site only).
- Upon receipt of the payment, Meetinghouse Technologies will review and sign the License Agreement and fax it back, along with enabling the Registration Code for MHT-Search.

How to Contact Meetinghouse Technologies

- Internet
<http://www.meetinghousetech.com>
- e-mail
mhtxtras@meetinghousetech.com
- US Mail
Meetinghouse Technologies
32 Defense Street
Annapolis, MD 21401
- Phone
(410) 573-9273
- Fax
(410) 573-9302

Technical Issues

File Structure

Searching with the MHT-Search Xtra is accomplished by referencing pre-indexed files, not by searching Director text cast members. These index files are created with a custom tool that processes the original text files. The resulting word-pool structure is then stored outside of the Director movie.

A sample file structure is shown below in Figure 1. The "Index" directory/folder contains the alphanumeric index files that the PreProcessor creates (used by MHT-Search for keyword searches) and the "Articles" directory/folder contains the original text files that the Pre-processor renames (used by MHT-Search for phrase searches). The Pre-processing method is described below.

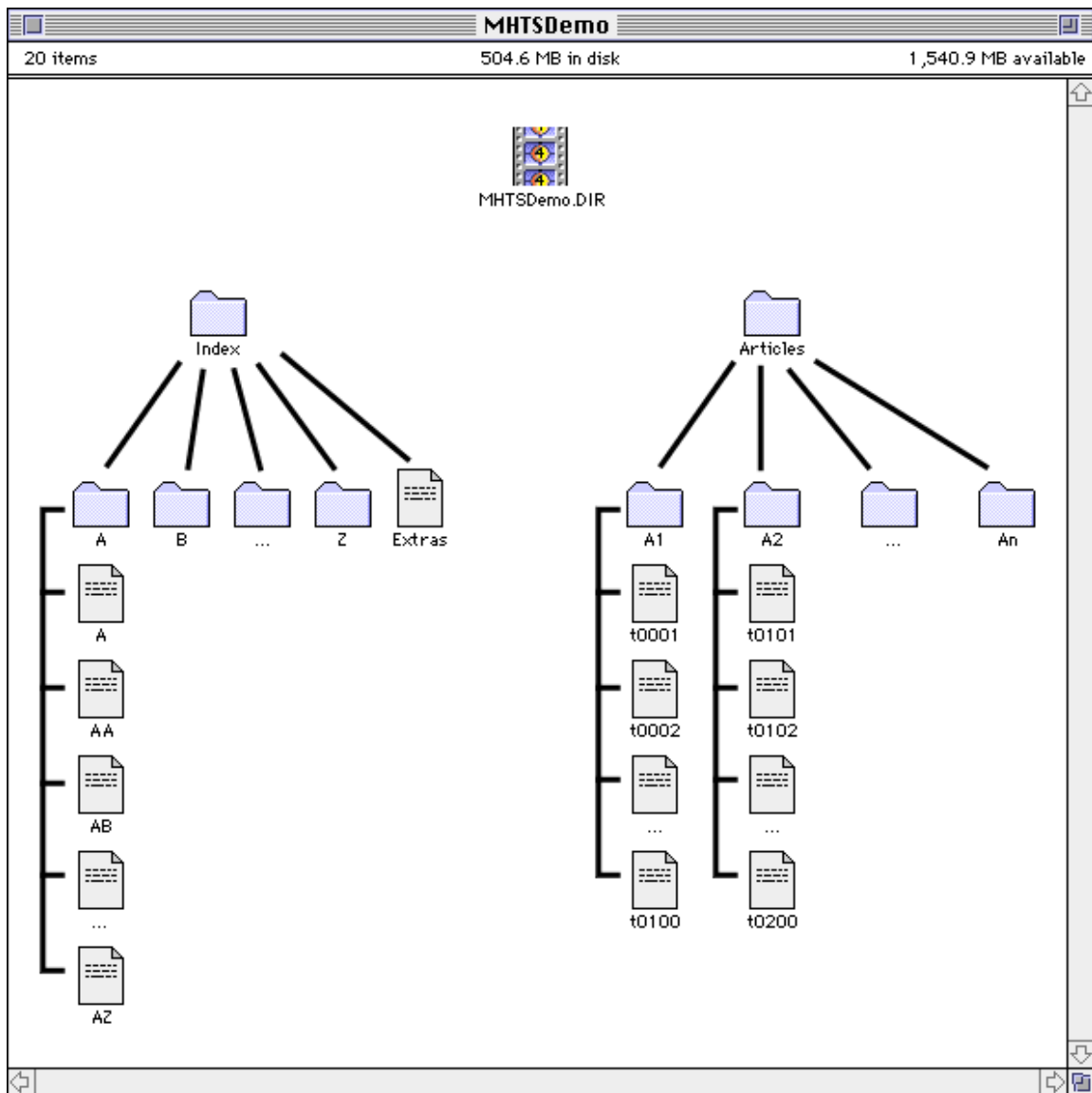


Figure 1: Example File Structure

Keyword Searching vs. Phrase Searching

The MHT-Search index files makes keyword searching extremely fast, but this speed does not fully carry over to phrase searching. Phrase searching complicates matters with the added constraint of spatial order. To compensate for this, MHT-Search breaks the phrase down into separate words and searches for these words with an automatic AND operation. This produces an intermediate list of matched articles containing all the words in the phrase, but not necessarily in the correct order. MHT-Search completes a phrase search by referencing only the renamed original files that satisfy the intermediate list and creating a final list of articles that contain the phrase in the proper spatial order.

Key Ranges

In some cases, it might be appropriate to have Key Ranges associated with different categories of text to allow the user to limit their search. For instance, the MHT-Search Demo contains articles on Careers and articles on College Majors. A user might want to only search for a word in the Careers articles. MHT-Search easily provides this Key Range searching capability.

A Key Range describes a range of articles (represented numerically by their order in the Articles Listing File) which are group together. This must be a contiguous range of article numbers and therefore it is important to group articles in some meaningful order for the PreProcessor. For more details on Key Ranges, see the PreProcessor section, and the section on the Xtra's methods.

MHT-Search Limitations

The following is a list of limits for the search engine:

| | |
|--|---|
| Maximum number of input article-files: | 2,500 |
| Maximum string length for a word: | 255 (truncates characters which follow) |

Pre-processing of Text Files

The first step in using the MHT-Search Xtra is creating the necessary support files with the PreProcessor. The newest version of the PreProcessor (2.0) is a 32-bit Windows program. It will only work on Windows 95, and Windows NT (3.51 or 4.0). The original PreProcessor is still available and works on early versions of Windows, and on the Macintosh (see Appendix A for instructions on use).

The new PreProcessor has several enhancements including:

- 1) HTML tags stripping - this will take in HTML files and filter out any HTML tags.
- 2) Delimiter stripping - this will change any desired characters into a white-space character (delimiter).
- 3) One program - this version of the PreProcessor is one seamless executable.
- 4) Speed - this version of the PreProcessor is slightly faster than the previous version.

Part A) GETTING STARTED

- 1) In order for the PreProcessor to process the text files for searching, the user must create a Articles Listing File. This is a text file that contains a list of the files to be processed. It is important that this list is accurate and in some meaningful order. The order of the files is important because MHT-Search will return the number of the file it matched, not the name. Therefore, the order in which the filenames appear in the Articles Listing File determines the number associated with them in the search engine.

Here is an example Articles Listing File:

```
CAR0101.TXT(first line, first text file entry)
CAR0102.TXT
CAR0103.TXT
...
CAR0918.TXT
CAR0919.TXT
CAR0920.TXT(next to last line, last text file entry ... NOTE: this line needs a hard return after it)
(last line ... nothing)
```

If you want to have Key Ranges included in the search process (allowing the user to limit the search to certain types of text files ... Careers, Majors, etc.) then divide this Articles Listing into groups of similar text files and separate them with a “|” (pipe symbol). This will serve as a visual cue for the Key Range breakdown. It is important to realize that to set up Key Ranges, the user need only call the appropriate Xtra methods. The pipe symbol acts as a visual cue to the creator of Articles Listing File, so he / she can see where the groups of data should be broken down into Key Ranges.

Here is an example Articles File with Key Range markup:

```
...
CAR0916.TXT
CAR0918.TXT
CAR0920.TXT (last file in the Careers group)
| (pipe symbol)
MAJ0101.TXT (first file in the Majors group)
MAJ0103.TXT
MAJ0105.TXT
...
```

and so on. Do not include a “|” at the beginning or end of the Articles Listing File.

- 2) Collect all the text files you need to process and store them in one directory/folder. It is possible to use relative paths to the file in the Articles Listing File. This enables the user to have the files in different directories.

Here is an example where the second career file is in the subdirectory “newdir”:

```
CAR0918.TXT
newdir\CAR0919.TXT
```

- 3) Launch the PreProcessor.
- 4) The main window will open:

Part B) Selecting Article Type

The PreProcessor can process HTML or generic text documents for searching. The PreProcessor will remove any HTML tags (<>) from the articles if the **HTML Document** radio button is highlighted.

If the user is processing generic text files, then the **Text Document** radio button should be highlighted.

In either case, the user can specify a set of **Delimiters** to use in the process of the articles. A Delimiter is a character that separates or delimits information (words, numbers, etc.). Since MHT-Search is a white-space delimited search engine, the PreProcessor will convert any user specified delimiters into white-space. To specify delimiters, simply fill in the **Delimiters** edit field. Each character typed in will be considered a delimiter.

Part C) Pre-processing the Files

The next step is to specify the location of the Articles Listing File, and any other Path's that are needed for processing the files. Text can be typed into the edit fields, or the buttons above the edit fields can be selected.

Each button will bring up an Explorer style File Dialog box which can be used for selecting files.

NOTE: When selecting a path for **Index Files**, it might be necessary to paste or type in the path directly because the file dialog box does not support selection of folders.

How to Generate the Index Files:

The Index Files are the key to the fast searching capability of MHT-Search. The user must generate the Index Files or MHT-Search will not function properly.

- 1) Fill in the **Articles Listing File** edit box by either typing in the full path to the Articles Listing File, or by selecting the **Articles Listing File** button.
- 2) Fill in the **Path to Articles** edit box by either typing in the full path to the articles to be processed, or by selecting the **Path to Articles** button.
- 3) Fill in the **Path for Index Files** edit box by either typing in the full path where the Index Files should be saved to, or by selecting the **Path for Index Files** button.
- 4) Select the **Generate Index Files** button.

The Results:

The PreProcessor first creates the path to save the index files in if it doesn't already exist. After reading in the chosen files, the PreProcessor divides them into alphabetic index files and an "Extras" file (see Figure 1: Example File Structure). The alphabetic index files are stored in their appropriate sub-directory/sub-folder (A-Z), while the "Extras" file, which contains any search words that begin with numeric or punctuation characters, is stored at the same level as the 26 alphabetic sub-directories/sub-folders.

The final step in the PreProcessor is the creation of the Article files with the t0000 format. MHT-Search needs the articles in this file format in order to do any phrase searching. Instead of the user copying these articles and renaming them, the PreProcessor will perform these functions as well as stripping out any **Delimiters** or **HTML** tags (see Part B.)

How to Copy The Articles:

- 1) Fill in the **Articles Listing File** edit box by either typing in the full path to the Articles Listing File, or by selecting the **Articles Listing File** button.
- 2) Fill in the **Path to Articles** edit box by either typing in the full path to the articles to be processed, or by selecting the **Path to Articles** button.
- 3) Fill in the **Path to Copy Articles** edit box by either typing in the full path where the copied articles should be saved to, or by selecting the **Path to Copy Articles** button.
- 4) Select the **Copy the Articles** button.

The Results:

The PreProcessor first creates the path to copy the renamed files to if it doesn't already exist. After reading in the chosen files, the PreProcessor renames them in a sequential order based on the Articles Listing. The naming scheme is:

File 1: t0001
File 2: t0002
File 3: t0003
...
File n: tn

The renamed files are divided into groups of 100 and stored in the appropriate "a" sub-directory/sub-folder within the directory/folder specified in Step 1:

Files t0001 - t0100: Sub-Directory/Sub-Folder "a1"
Files t0101 - t0200: Sub-Directory/Sub-Folder "a2"
...
Files (tn - 99) - tn: Directory/Folder "a<n/100, rounded to the next highest integer>"
(e.g., 238 original files would be stored in "a1" - "a3", with "a3" holding files t0201-t0238)

MHT-Search Methods and Lingo Examples

The Director movie included with this archive, "MHTSDemo.DIR", contains a fully realized execution of MHT-Search. There are 100 searchable articles, divided into 2 Key Ranges. All the scripts are thoroughly commented to assist in understanding how to use MHT-Search. The individual methods are listed below with their appropriate input values, a description, and sample Lingo from "MHTSDemo.DIR" for using that method.

METHOD NAME:

MHTSearchRegister

Input:

RegistrationCode:

The executable-specific code that enables MHT-Search

Description:

Registers MHT-Search.

A projector will not be able to successfully call any methods without a proper registration code.

Sample Lingo for Registering MHT-Search:

```
MHTSearchRegister("your code here");
```

For MHT-Search to be registered properly, this command must be included before any other method calls when run from a projector. In authoring mode, the Xtra will work without a registration code.

METHOD NAME:

MHTSearchSetIndexPath

Input:

IndexPath:

Absolute path to the Index Files

Description:

Sets the absolute path to the Index Files used for keyword searching.

Sample Lingo for Setting the Index Path:

```
MHTSearchSetIndexPath("c:\myFile\index\is\here")
```

METHOD NAME:

MHTSearchSetArticlesPath

Input:

ArticlesPath:

Absolute path to the Articles Files

Description:

Sets the absolute path to the Articles Files used for phrase searching.

Sample Lingo for Setting the Articles Path:

```
MHTSearchSetArticlesPath("c:\myArticles\are\here")
```

METHOD NAME:

MHTSearchCountMatches

Input:

None

Description:

Returns the total number of matches as an integer.

If an error occurs, a negative number will be returned.

Sample Lingo for Counting the Number of Matches:

```
set gNumMatches = MHTSearchCountMatches()
```


METHOD NAME:

MHTSearchGetMatch

Input:

PointerPosition:
0 or 1

Description:

Allows for *sequential access* of the Matches Array.

Returns an integer if a match was found. This integer represents the position of the matched file in the Reference List, minus any lines containing Key Range pipes ("|"). (*see the "Pre-processing of Text Files" section for more information on Key Range definition*)

Returns a zero if there were no matches.

Returns a negative number if an error occurred.

Sample Lingo for Getting Matches (GetMatch method):

(not used in the MHT-Search Demo)

Sending mGetMatch a value of "1" for PointerPosition will return the next match in the Array:

```
repeat with i = 1 to gNumMatches
    set gMatchList = gMatchList & MHTSearchGetMatch(1) & RETURN
end repeat
```

Sending mGetMatch a value of "0" for PointerPosition will reset the Array pointer and return the first match in the Array:

```
set gMatchList = gMatchList & MHTSearchGetMatch(0) & RETURN
```

The Array pointer is reset automatically whenever a new instance of MHT-Search is created.

METHOD NAME:

MHTSearchGetAt

Input:

PointerPosition:
Integer between 1 and total # of matches

Description:

Allows for *direct access* of the Matches Array

Returns an integer if a match was found. This integer represents the position of the matched file in the Reference List, minus any lines containing Key Range pipes ("|").

Returns a zero if there were no matches.

Returns a negative number if an error occurred.

Sample Lingo for Getting Matches (GetAt method):

Sending mGetAt a value of "n" will return the "nth" match:

```
repeat with i = 1 to gNumMatches
    set gMatchList = gMatchList & MHTSearchGetMatch(i) & RETURN
end repeat
```

METHOD NAME:

MHTSearchAddWordPhrase

Input:

WordPhrase:

A keyword or phrase

Description:

Adds a string of characters (max. 255) to a 2-slot buffer in MHT-Search.

Returns an integer representing the buffer slot where it was added.

Returns a negative number if an error occurs.

Sample Lingo for adding a keyword/phrase:

```
set result = MHTSearchAddWordPhrase( gSearchWrd1)
```

METHOD NAME:

MHTSearchDeleteWordPhrase

Input:

WordPhraseToDelete:

A keyword or phrase

Description:

Deletes a slot in the buffer based on a match with WordPhrase.

Returns the remaining number of filled buffer slots if the keyword or phrase was deleted.

Returns a negative number if no match was found.

Sample Code for deleting a keyword/phrase:

```
set result = MHTSearchDeleteWordPhrase(gSearchWrd1)
```

METHOD NAME:

MHTSearchAndOr

Input:

BooleanType:

0, 1, or 2

Description:

BooleanType = 0 indicates no Boolean operation

BooleanType = 1 indicates an AND search.

BooleanType = 2 indicates an OR search.

No logical operation takes place unless both slots in the WordPhrase buffer are filled.

A negative number is returned if something other than the accepted values of BooleanType is passed; the current value of BooleanType stays unaltered if this occurs.

Sample Lingo for AND/OR searches:

```
if gAND then
    set result = MHTSearchAndOr(1)
else if NOT gAND then
    set result = MHTSearchAndOr(2)
else
    set result = MHTSearchAndOr(0)
end if
```

METHOD NAME:

MHTSearchAddKeyRange

Input:

keyRangeName:

Name the programmer wishes to assign to the Key Range.

lowerBound:

The Key Range starting position in the Reference List.

upperBound:

The Key Range ending position in the Reference List.

Description:

Returns the number of Key Ranges currently set.

Returns a negative number if there are no more spaces for Key Ranges or if the bounds are incorrect.

If there are no Key Ranges set then MHT-Search considers all files to be valid.

Sample Lingo for Adding Key Ranges:

```
set result = MHTSearchAddKeyRange("CAREERS", 1, 125)
```

(keyRangeName, lowerBound, upperBound)

METHOD NAME:

MHTSearchDeleteKeyRange

Input:

keyToDelete:

Any of the keyRangeNames defined with the AddKeyRange method.

Description:

Deletes the specified Key Range, if present, and reorganizes the remaining Key Ranges.

Returns an integer representing the number of remaining Key Ranges.

Returns a negative number if there are no Key Ranges that match keyToDelete.

Sample Lingo for Deleting Key Ranges:

(not used in the MHT-Search Demo)

```
set result = MHTSearchDeleteKeyRange("CAREERS")
```

Appendix A

Pre-processing of Text Files with 1.1

The MHT-Search Pre-processor creates the indexed files for keyword searches and the renamed article files for phrase searches. There are versions for Windows and Macintosh computers. The Windows version is called "MHTSPREP.EXE"; the Mac version is called "MHT-Search PreProcess". Since both versions look and operate exactly the same, the following instructions apply to either platform. All examples used in this manual are taken directly from the included MHT-Search Demo.

The index files (see **Part B** below) and the renamed original files (see **Part C** below) can be accessed by both the Mac and the PC versions of MHT-Search.

Part A) GETTING STARTED

The Pre-processor files must be copied to your hard drive to run properly:

Mac Users: Copy all the files in the "Pre-processor" folder.

Windows Users: Copy all the files in the "PREPROC" directory.
Unzip the "PREPDLLS.ZIP" file in the same directory as the three executables.

1) A Reference List of the text files to be processed must be created. A spreadsheet application such as Microsoft Excel or a simple text editor such as Notepad can be used for this. If you use something like Excel (which will make the task easier), be sure to save the file as a .CSV file (comma-delimited). The entries should start on the very first line (cell A1 in a spreadsheet). Each line in the Reference List should contain the name of one text file to be made searchable, followed by a hard return, with no extraneous characters; e.g.,

CAR0101.TXT(first line, first text file entry)

CAR0102.TXT

CAR0103.TXT

...

CAR0918.TXT

CAR0919.TXT

CAR0920.TXT(next to last line, last text file entry ... NOTE: this line needs a hard return after it)

(last line ... nothing)

If you want to have Key Ranges included in the search process (allowing the user to limit the search to certain types of text files ... Careers, Majors, etc.) then divide the Reference List into groups of similar text files and separate them with a "|" (pipe symbol); e.g.,

...

CAR0916.TXT

CAR0917.TXT

CAR0920.TXT(last file in the Careers group)

| (pipe symbol)

MAJ0101.TXT (first file in the Majors group)

MAJ0103.TXT

MAJ0201.TXT

...

and so on. Do not include a “|” at the beginning or end of the Reference List.

The Reference List for this Demo was created as an Excel spreadsheet, "Master.XLS". Note that column A of this spreadsheet contains the article filenames (CAR0101.TXT, CAR0102.TXT, etc.), while column B contains the article titles (ACCOUNTANTS AND AUDITORS, ADMINISTRATIVE SERVICES MANAGERS, etc.). Maintaining this correlation is extremely important for identification of the text files within Director, and is explained inside "MHTSDemo.DIR", handler "ShowSearchMatches", comment "2) a) ii) Adjust 'article' to account for different Key ranges".

When this Reference List was saved as a comma-delimited format, column B was deleted so that only column A is in the .CSV file, "Master.CSV". This is the .CSV file referred to in **Parts B & C**.

- 2) Collect all the text files you need to process and store them in one directory/folder.
- 3) Launch the Pre-processor.
- 4) The main window will open:

You can:

Run the "For Word Searches" Module
(see **Part B** below)

Run the "For Phrase Searching Only" Module
(see **Part C** below)

Choose one of the Utility Buttons
-Open Log
-Help
-Quit
(see **Part D** below)

Part B)"For Word Searches" Module

This Module pre-processes the original text files and creates the indexed files used for keyword searches. You need to complete everything in **Part A** before you can continue.

How-to:

- 1) Click the "Find CSV" button. A file dialog box will open.
Find and open the Reference List file you created in Step 1 of **GETTING STARTED**.
The text-entry field labeled "CSV file (contains article filenames):" will now display the absolute path to the file you just selected; e.g., "MHTSDemo:CSVFiles:Master.CSV" or "X:\csvfiles\master.csv".
- 2) Click the "Locate" button. A file dialog box will open.
Find the directory/folder where you stored the text files in Step 2 of **GETTING STARTED**.
Open one of the files within that directory/folder.

The text-entry field labeled "Location of articles (full pathname):" will now display the absolute path to that directory/folder; e.g., "MHTSDemo:Original Files:" or "X:\original\".

- 3) Click in the third text-entry field, labeled "Path to output search index files":

Manually enter the directory/folder path that you would like the Pre-processor to save the indexed files in; e.g. "MHTSDemo:Index:" or "X:\index\". This path doesn't have to exist; the Pre-processor will create it for you.

- 4) Click the "Gen Index Files" button to start the indexing process.

A window titled "console" will open, and the following will be displayed in the window:

```
# MHTSindx: Search Pre-processor Started ...
```

Listing of files to be word - tree'd

File 1: car0101.txt

File 2: car0103.txt

File 3: car 0105.txt

...

File n: <last filename>

\-/-\-/-\-/-\-/-\-/-\-/-\-/-\-/-\-/-\-/-\-/-

MHTSindx: Reading words from n articles ...

#MHTSindx: Completed word count/sort ... now writing out wordTree

Running Total :: Char :: Cnt (1-ltr + 2-ltr + extras)

WordCnt 19; For \$: 19 (0 + 0 + 19)

```
WordCnt    20;  For ( : 1    ( 0  + 0    + 1)
```

...

[This continues for all words beginning with non alpha-numeric characters]

[Next comes the count for all words beginning with numeric characters]

...

WordCnt 34; For 0: 1 (0 + 0 + 1)

WordCnt 39; For 1: 13 (0 + 0 + 13)

...

[This continues for all words beginning with numeric characters]

[Next comes the count for all words beginning with alphabetic characters]

...

WordCnt 131; For A: 88 (3 + 84 + 0)

WordCnt 161; For B: 13 (3 + 27 + 0)

...

[This continues for all words beginning with alphabetic characters]

[Now the Pre-processor finishes]

```
# MHTSindx: Completed file writing to <Path set above in Step 3>
```

```
# MHTSindx: FREEing tree and list nodes.
```

```
# MHTSindx: Search Pre-processor Finished.
```

Press <RETURN> to exit ...

When the Pre-processor is finished, the title also changes to read "press <<return>> to exit". As prompted, press <RETURN> to close this window and continue.

If the filenames in the Reference List do not exactly match the actual filenames, or if a file doesn't exist, the Pre-processor will stop and display:

#MHTSindx: ERROR, Missing file #<number of the problem file - 1>

If this should happen, make a note of which file caused the interruption and close the Pre-processor. Verify that the text file exists and that its name matches the name in the reference list file. Once the problem is fixed, delete the partial index files that were just created and start again at Step 4.

The Results:

The Pre-processor first creates the path to save the index files in if it doesn't already exist. After reading in the chosen files, the Pre-processor divides them into alphabetic index files and an "Extras" file (see Figure 1: Example File Structure). The alphabetic index files are stored in their appropriate sub-directory/sub-folder (A-Z), while the "Extras" file, which contains any search words that begin with numeric or punctuation characters, is stored at the same level as the 26 alphabetic sub-directories/sub-folders.

Part C) "For Phrase Searching ONLY" Module

This Module copies the original text files into a new folder and renames them according to the Reference List. You need to complete everything in **Part A** and **Part B** before you can continue.

How-to:

- 1) Click in the text-entry field labeled "Path to copy articles:".
Manually enter the directory/folder path that you would like the Pre-processor to copy the renamed files to; e.g. "MHTSDemo:Articles:" or "X:\articles\". This path doesn't have to exist; the Pre-processor will create it for you.
- 2) Click the "Copy Articles" button to start the copying and renaming process.
A window titled "console" will open, and the following will be displayed in the window:

```
# MHTScopy: Duplication of files Started ...
# MHTScopy: Starting file duplication process

Copying File   1: <path to the original files><filename> to <path from Step 1>a1:t0001
Copying File   2: <path to the original files><filename> to <path from Step 1>a1:t0002
Copying File   3: <path to the original files><filename> to <path from Step 1>a1:t0003
...
Copying File   n: <path to the original files><filename> to <path from Step 1>ax:tn
\-\-\-\-\-\-\-\-\-\-
# MHTScopy: Completed copying n files to <path from Step 1>
#MHTScopy: Finished
```

Press <RETURN> to exit ...

When the Pre-processor is finished, the title also changes to read "press <<return>> to exit". As prompted, press <RETURN> to close this window and continue.

If the filenames in the Reference List do not exactly match the actual article filenames, or if a file doesn't exist, the Pre-processor will display:

```
Error opening src file <path to the original files>:<problem file>
#MHTScopy: ERROR, during copy of file '<problem file>'
```

If this should happen, make a note of which file caused the interruption and close the Pre-processor. Verify that the text file exists and that its name matches the name in the reference list file. Once the problem is fixed, delete the remaining renamed files that were just copied and start again at Step 2.

The Results:

The Pre-processor first creates the path to copy the renamed files to if it doesn't already exist. After reading in the chosen files, the Pre-processor renames them in a sequential order based on the Reference List. The naming scheme is:

File 1: t0001
File 2: t0002
File 3: t0003
...
File n: tn

The renamed files are divided into groups of 100 and stored in the appropriate "a" sub-directory/sub-folder within the directory/folder specified in Step 1:

Files t0001 - t0100: Sub-Directory/Sub-Folder "a1"
Files t0101 - t0200: Sub-Directory/Sub-Folder "a2"
...
Files (tn - 99) - tn: Directory/Folder "a<n/100, rounded to the next highest integer>"
(e.g., 238 original files would be stored in "a1" - "a3", with "a3" holding files t0201-t0238)

Part D) Utility Buttons

1) Open Log

Clicking this button opens the log files from the "For Word Searches" and "For Phrase Searching ONLY" Modules. A window titled "<path to log files><log file>" will open. At the bottom left are two radio buttons:

- Index LOG
Displays the log file from the "For Word Searches" Module
- Copier LOG
Displays the log file from the "For Phrase Searching ONLY" Module

Click the "Close" button at the bottom right to return to the Main Pre-processor Screen

2) Help

Clicking this button opens the MHT-Search Pre-processor QuickHelp -- an electronic reference card that gives you brief reminders on how to use the application.

3) Quit

Oddly enough, clicking this button quits the Pre-processor.

