GPD-3

Third Edition
April 1993

How to Create and Modify GEMPACK Header Array Files Using the Program MODHAR

GEMPACK Document No. 3

Copyright 1993 by the Impact Project and KPSOFT

ISSN 1030-2514

This is part of the documentation of the GEMPACK Software System for solving large economic models, developed by KPSOFT and the IMPACT Project, Menzies Building, Monash University, Wellington Road, Clayton, Vic. 3168, Australia.

Abstract

Within *GEMPACK*, Header Array files are the principal means of storing and accessing data for economic models. Because these are binary files, they cannot be edited directly to modify data. The program *MODHAR* is an important tool for creating these files originally and for modifying the data on them.

This document is a user's guide to *MODHAR*. It contains an overview of the program and gives details of the different commands and sub-commands available with *MODHAR*. It also contains a brief discussion of alternative ways of creating and modifying Header Array files.

Document Attributes

Name : How to Create and Modify GEMPACK Header Array Files Using the Program MODHAR

Audience : CGE Modellers

Identifier : GPD-3

History :

Date	*Author(s)*	*Comment*
November 1986	K. Pearson	First edition (GED-18)
		[Title was: "How to Modify Data for Economic Models - The Program MODHAR"]
February 1991	G. Codsi	Second edition
		[No GED Number. Title was "Summary Documentation for MODHAR".]
April 1993	Jill Harrison and Ken Pearson	Third edition (GPD-3)

GemPack ©

CONTENTS

1. Introduction 1

1.1 Header Array Files 1

1.2 An Overview of the Use of MODHAR 2

1.2.1 Creating a New Header Array File 2
1.2.2 Modifying an Existing Header Array File 2

1.3 Alternatives to MODHAR 4

1.4 A Guide to This Document 4

1.4.1 Changes from Previous Versions of MODHAR 5

2. Using MODHAR 6

2.1 MODHAR Commands 6

2.1.1 Commands for Creating a New Header Array File 7
2.1.2 Commands for Modifying an Existing Header Array File 7

2.2 Commands for Adding Arrays to the New file 10

2.3 Operations on Headers or Long Names 13

2.4 Commands for Complicated Modification or Addition 16

2.4.1 MODHAR Sub-commands 16
2.4.2 Examples 17
2.4.3 Writing Arrays to the New Header Array File 19

2.5 Finishing Up 21

2.5.1 Exit Commands 21
2.5.2 History of the New File 22

2.6 Complete Example of a MODHAR Run 22

Appendix A - Text Files 24

References 25

1. INTRODUCTION

1.1 Header Array Files

Data for economic models (for example, input-output tables or parameters such as elasticities) are often held within *GEMPACK* on files called Header Array files. Header Array files contain one or more arrays each containing data values. An individual array of data on a Header Array file is accessed by supplying the unique 4-character identifier (or **Header**) for that array of values.

The data values held by an individual array can be either all real numbers, all integer numbers or all character strings. Depending on the type of data that is to be stored, the number of dimensions allowed varies.

The dimension limits for Header Arrays are :

- For real numbers - up to and including 7 dimensions
- For integer numbers - up to and including 2 dimensions
- For character strings - only one dimensional arrays are allowed, each string being of the same length

Headers for arrays on any one file must be unique since the header acts as a label or primary key to identify the associated array.

Once written, an array contains not just the data for the array itself but also self-describing data, including the type of data values, dimensions of the array and a descriptive "long name" of up to 70 characters.

Header Array files have the advantage that you can access any array just by referring to the header which uniquely identifies the array in the file. There is no need to consider here the format of the arrays or any other details since they are all taken care of automatically by the software.

Headers consist of four characters which are usually letters (A to Z, a to z) or digits (0 to 9). Different arrays must have different headers. The case (upper or lower) of a header is not significant. (For example, you cannot have one array with header 'ABCD' and another on the same file with header 'AbCd'.). Headers starting with letters 'XX' are reserved for internal program use so an error is returned if you choose a header starting with 'XX'.

1.2 An Overview of the Use of MODHAR

Data on Header Array files cannot be edited (that is, modified) using normal text editors. The data on a Header Array file is encoded in binary form to keep the size of the file small. *MODHAR* is therefore provided to enable you to create a new Header Array file that is based on, but differs from, an existing Header Array file. *MODHAR* can also be used to create a new Header Array file.

1.2.1 Creating a New Header Array File

When *MODHAR* is being used to create a new file, the basic method of operation is to read data values from some source into memory and then write them as header arrays to the new file.

Data values can be read in from text files or the terminal as shown in Figure 1. Arrays of data can also be copied from existing Header Array files to the new file. In this mode of operation the data values in the arrays are not usually modified, merely copied unchanged from the source to the new file.

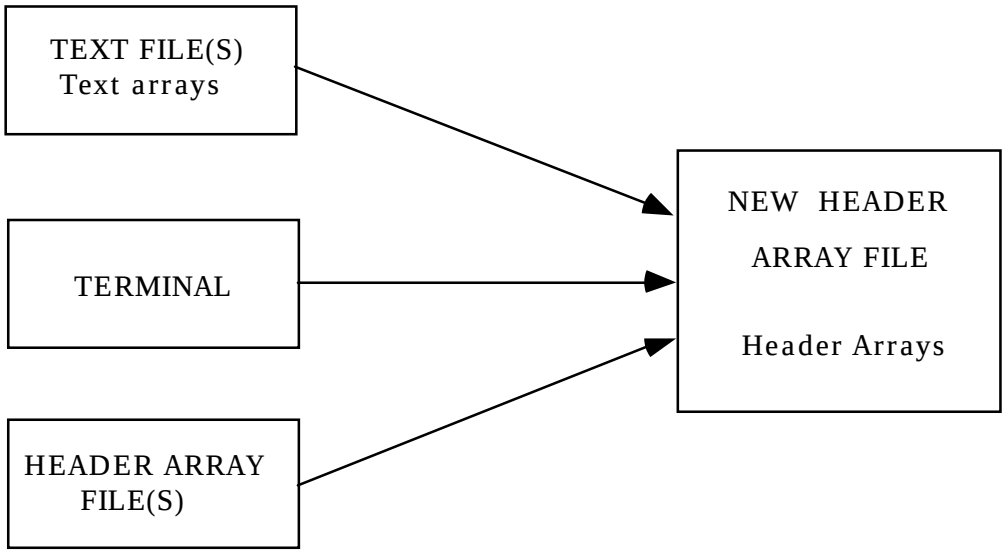

Figure 1: Creating a New Header Array File

1.2.2 Modifying an Existing Header Array File

One very common need for *GEMPACK* users is to modify data on an existing Header Array file. The arrays on the file cannot be edited directly; instead they are copied from the old existing file to a work array in memory. Here the data values in the array can be changed and finally can be written as Header Arrays to the new file. As shown in Figure 2, extra arrays can also be added from extra Header Array files (not just the old file) and from extra text files.

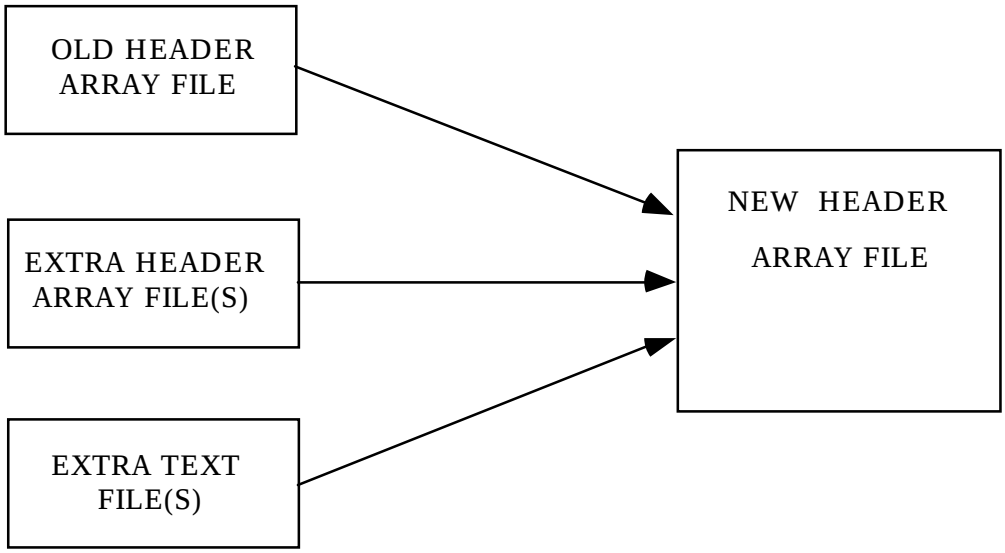

Figure 2: Modifying an Existing Header Array File

Arrays on the old file which are not modified can be conveniently copied directly to the new Header Array file as a block.

Note that unlike some other packages (such as word processors or spreadsheets), *MODHAR* does not actually modify the existing Header Array file. Rather, *MODHAR* leaves it unchanged and creates a new Header Array file containing the changes you make. (For this reason, the new file should be given a different name from that of the existing one.)

1.3 Alternatives to MODHAR

If you need to modify data values in one or more arrays, there are at least two methods to be considered as alternatives to using *MODHAR*.

Spreadsheet method

Input the arrays to a spreadsheet. If the original arrays are on a Header Array file, the program *SEEHAR* is available to convert them to a form suitable for input to a spreadsheet (see section 8.2.2 of GPD-1* for more details.) Edit the arrays in the spreadsheet then output them as text arrays using the comma-separated values (CSV) format (see Appendix C of GPD-1). They can then easily be converted to a Header Array file using *MODHAR*.

TABLO Method

Write a *TABLO* Input file to read values of the arrays from Header Array files (or text files) into arrays defined as COEFFICIENTs. Use FORMULAs to manipulate the data in various ways then write the arrays to a Header Array file. (See section 3.4.3 of GPD-1 for an example.)

1.4 A Guide to this Document

The program described here is version 5.0 (February 1993) of *MODHAR*, which was first made available with Release 5.0 of *GEMPACK* (April 1993).

Section 2 of this document describes the different commands and sub-commands available for using *MODHAR* and gives some examples of their use. Appendix A contains a brief discussion of *GEMPACK* text data files.

---

* References to GEMPACK documents identify the document by GEMPACK Document (GPD) number, rather than by author or date. References are always to the version of the document which is current at the date of issue of the cross-referencing document. The GEMPACK documents referenced are listed in a separate section at the end of the Reference section of this document. Comments from readers on this or any of the GEMPACK documents, either pointing out errors, inaccuracies, omissions or obscurities, or making other suggestions for improvements, will be welcomed. Please address such comments to one of the authors at the Impact Project.

The numbering of GEMPACK Documents has been re-started with Release 5 of GEMPACK, when the abbreviation "GPD" was first used. Previous editions of these documents did not have the same numbers as the current editions. Pre-Release-5 documents are numbered "GED-xx".

1.4.1 Changes from Previous Versions of MODHAR

MODHAR now includes simplified procedures (via the new commands 'at' and 'ah') for adding arrays from text files, extra header array files and from spreadsheets.

The format of *GEMPACK* text data files has been improved to allow you

- to add several arrays from the same text file.
- to include on text files the header and long name associated with each array, and read them in with the array. This makes it a quick and simple procedure to read in all the arrays on a text file at once. Alternatively you can skip over arrays on the text file if you don't want them included on the new Header Array file.
- specify the order of data on text files as row-order, column-order or spreadsheet. The aim of this is to make it easy to transfer arrays of data between spreadsheets and Header Array files since one of the most convenient ways of editing data is on a spreadsheet.

2. USING *MODHAR*

MODHAR is a command-driven program and is easy to use. The menus of possible actions clearly state the tasks that you can currently perform.

MODHAR commences with the usual sort of GEMPACK options screen (as described in section 5.3 of GPD-1). Then you are asked if the new file to be created is to be based on an old (that is, an existing) file.

(i) If you are modifying an old file, respond 'y', and you will then be asked for the name of the file you wish to modify.

(ii) Otherwise (that is, you are creating a new file), respond 'n'.

Note that any references in the rest of this document to the "old file" are only relevant in case (i) above.

2.1 MODHAR Commands

The *MODHAR* Command Menu offers the following commands:

ah	**a**dd one or more arrays from **H**eader Array file
at	**a**dd one or more arrays from **t**ext file
da	**d**elete one or more **a**rrays from old file
ra	**r**estore one or more deleted **a**rrays from old file
mw	**m**odify data from the original file and **w**rite it on the new file
aw	**a**dd new data and **w**rite it on the new file
ex	**ex**it, saving the new Header Array file
q	**q**uit, without saving the new Header Array file
lh	**l**ist the **h**eaders, their status and size of associated arrays
oh	show **o**ne **h**eader, its status and size of associated array
ch	**c**hange a **h**eader name
eln	**e**xamine a **l**ong **n**ame
cln	**c**hange the **l**ong **n**ame associated with a header
?	help

MODHAR commands fall into four groups

- Commands for simply adding arrays to the new file **ah**, **at.**
- Commands for carrying out operations on headers or long names **lh**, **oh**, **ch**, **eln**, **cln**, **da**, **ra.**
- Commands for complicated modification or addition of data on arrays from the old file, terminal or other files **mw**, **aw.**
- Exit commands **ex** or **q**.

2.1.1 Commands for Creating a New Header Array File

When *MODHAR* is being used to create a new Header Array file, it is usually just a matter of transferring data from elsewhere (text files, other Header Array files or input from the terminal) to the new Header Array file. This process is illustrated in Figure 3 where you can see that this can be accomplished using the *MODHAR* commands **at**, **ah** and **aw**. Once all the desired data has been added, you use the exit command **ex** to complete the task.

2.1.2 Commands for Modifying an Existing Header Array File

Here you may need to use the full range of *MODHAR* commands, as illustrated in Figure 4.

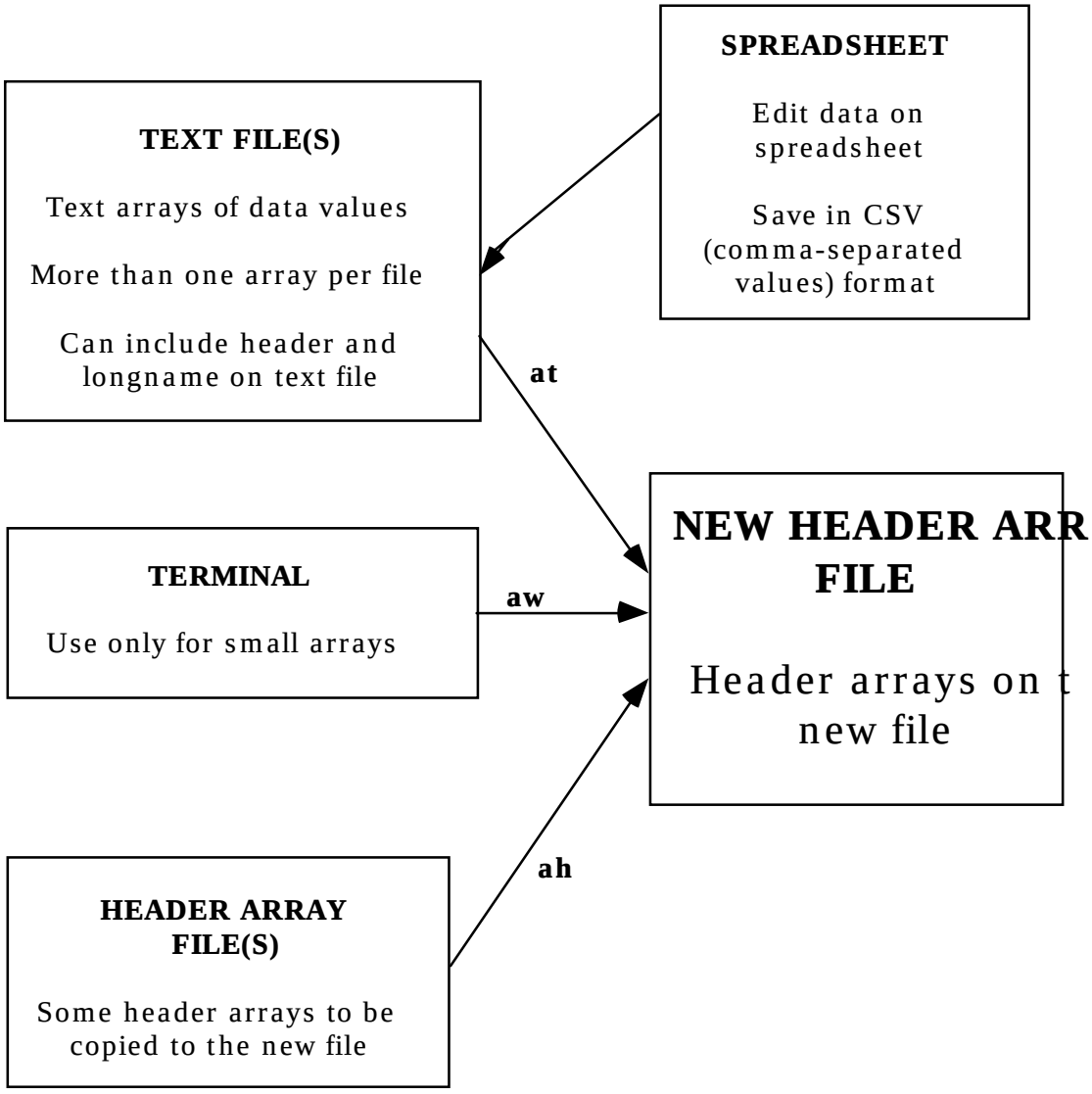

Figure 3: Commands for Creating a New Header Array File

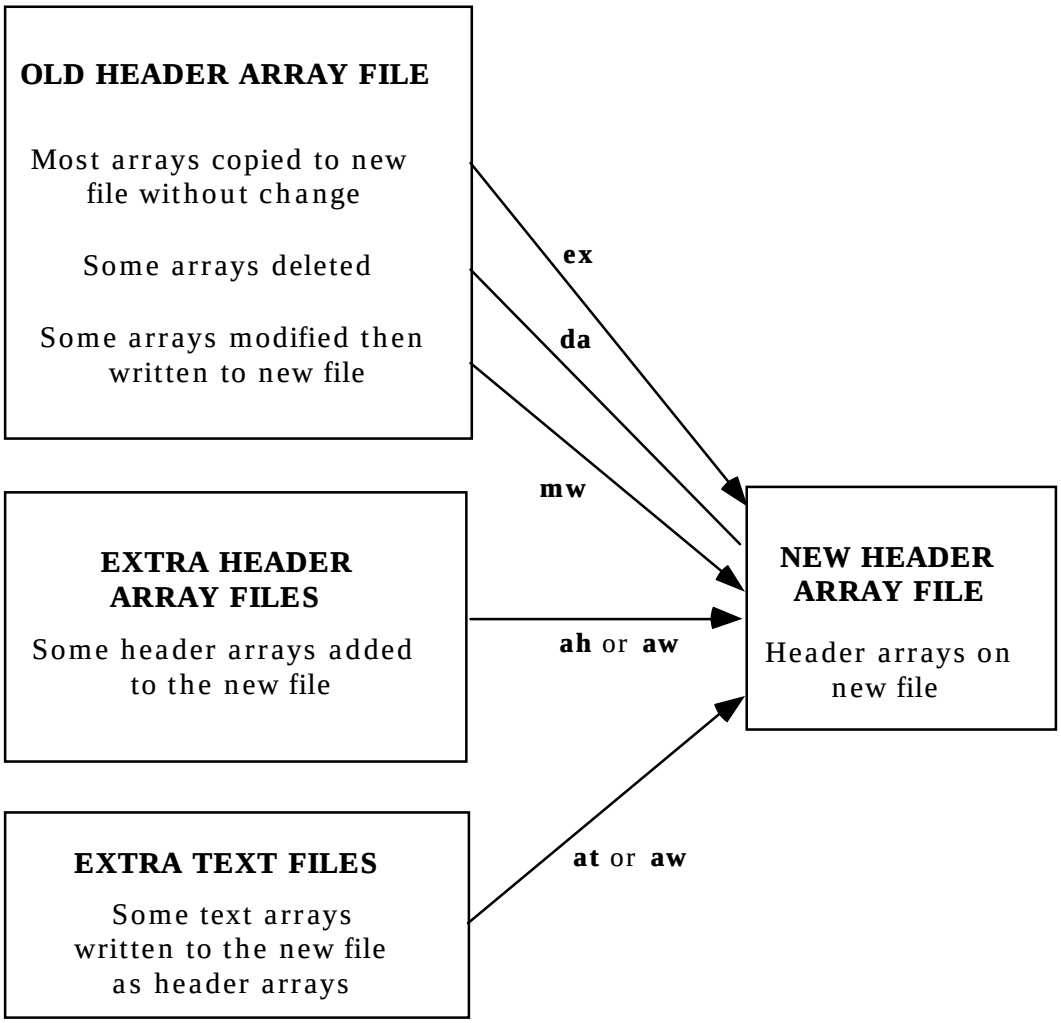

Figure 4: Commands for Modifying an Existing Header Array File

2.2 Commands for Adding Arrays to the New File

ah	**add one or more arrays from a Header Array file**

Menu choice **ah** provides a simple procedure for adding complete arrays from an extra Header Array file. This extra file has arrays in header array format (so is a binary file). (Menu choice **at** provides a similar procedure for text files.)

After choosing **ah**, you will be asked for the name of the extra Header Array file. *MODHAR* displays the list of headers on the extra file and you are given the choice of adding

[a] all of the headers on the file
[L] some of the headers (list ones to add)
[m] most of the headers (list ones to omit)
[n] none of the headers.

From the choice [a] (which is the default here), the program adds ALL the arrays on the extra file to the new file. However, if the header of one of the arrays on the extra file is the same as that for an array already on the new file or for an array on the old file (but not deleted), the program will omit this array as adding it would cause duplicate headers on the new file.

For the choice [L], you supply a LIST of headers of arrays on the extra file to add to the new file. All other arrays on the extra file will be ignored. Again the program checks for duplicates and will tell you whether that header is already on the new or old file. If a duplicate is found, this array will not be added to the new file. Giving a carriage-return instead of a header ends the list of headers.

For the choice [m], the programs add MOST of the arrays on the extra files. You list the headers that you want to OMIT. A carriage-return ends the list.

For the choice [n], NONE of the arrays on the extra file are added. This gives a way out if you have chosen the wrong filename or if you just want to see what is on another file apart from the one you are modifying.

Example using the LIST option

The following example gives responses for adding arrays with headers 'EEEE' and 'FFFF' to the new file from an extra Header Array file.

User input

ah	**!**	**add one or more arrays from Header Array file**
extra_header.har	**!**	**name of extra file**
L	**!**	**list ones to add**
EEEE	**!**	**header of array to add**
FFFF	**!**	**header of array to add**
	!	**carriage_return to end list to add**
lh	**!**	**list headers**

End of User Input

Headers 'EEEE' and 'FFFF' have been written to new file. The long names on the new file are the same as for these arrays on the extra file.

at	**add one or more arrays from a text file**

Menu choice **at** provides a simplified way of converting one or several arrays of data on a text file to header arrays and writing them to the new file in header array (that is, binary) form.

After choosing **at**, you supply the name of the extra text file and then get a choice of adding

[a] all of the data arrays on the file
[s]some of the arrays
[n] none of the arrays

For the choice [a] (which is the default here), all of the arrays are added. If a valid header, and (optionally) a long name, are given in the "how much data" information (see Appendix A) for an array, *MODHAR* will convert the text arrays read in to header array format and write them to the new file with the given header as label.

If the header is not read in from the text file with the array, or is not a valid choice, or duplicates a header on the old or new file, you will be asked to provide a valid header for this array.

For the choice [s], *MODHAR* reads in the arrays one by one and displays the type, dimensions and, if given on the text file, the header and long name.

You have the choice of

1. accepting the header name with a carriage-return,
2. giving a different header for this array, (type in the new header eg ABCD)
3. skipping over this array (type XXX)
4. ending the reading of data from this file (type XXQ).

For 1 and 2, once a valid header is given, you are asked for the long name (if one is not specified in the text file) and then the array is written as a header array to the new file while for 3, no array is written. After this *MODHAR* goes on to read the next array till the end of file is reached or XXQ is input to end reading this text file.

For choice [n], none of the arrays are added and you are returned to the main menu.

Example using the SOME option

The following example gives responses for adding headers 'GGGG' and 'HHHH' to the new file from an extra text file.

User input

at	**!**	**add one or more arrays from text file**
extra_text.txt	**!**	**name of extra file**
s	**!**	**add some of the arrays**
	!	**carriage_return to add 'GGGG'**
HHHH	**!**	**no header specified on the text file - input header**
second array	**!**	**long name for array**
XXQ	**!**	**XXQ to end list to add**
lh	**!**	**list headers.**

End of User Input

Headers 'GGGG' and 'HHHH' have been written to new file.

The text file was as follows:

Text file

```
2 3 real row_order header "GGGG" longname "first array";
4.0 2.0 1.0
1.0 6.0 8.0
6 integer ;
1 2 3 4 5 6
3 strings length 5 header "xyza";
abcde
pqr
11223
```

End of text file

The appropriate format for the text arrays on the text file is described briefly in Appendix A of this document (and described fully in Appendix C of GPD-1).

2.3 Operations on Headers or Long Names

Initially *MODHAR* assumes that all arrays on the old file will be copied to the new file. To ensure headers on the new file are unique, *MODHAR* checks for duplicates in the new header list assuming

- all (non-deleted) arrays on the old file will be included on the new file.

Of course,

- headers of all arrays already written to the new file are included in the list of new headers.

(Initially, the "new" header associated with each array on the old file is its header on that file, but this can be changed using the **ch** command.)

lh	**list the headers, their status and size of associated arrays**
oh	**show one header, its status and size of associated array**

Menu option **lh** shows the complete list of headers while **oh** gives the same information for just one header. Status refers to whether the header is on the old file, the new file, an extra file or in memory during modification (**mw**). Old and new header names are given. In using **oh**, the header given should be the new header name of the array in question.

The following table is a sample of the output from **lh**.

No.	Header Old	Header New	Status	Data type	Size old	Size new	Size change?
1	AAAA	ABCD	On old file	real	2 ∞ 2	2 ∞ 2	no
2	BBBB	BBBB	On old file	int	6	6	no
3	CCCC	CCCC	On old file	real	2 ∞ 5	2 ∞ 5	no
4	DDDD	DDDD	On old file	real	2	2	no
5	EEEE	EEEE	Deleted	int	3 ∞ 4	3 ∞ 4	no
6	FFFF	FFFF	Deleted	char	3 strs,len 5	3 strs,len 5	no
7	GAMM	GAMM	ON NEW FILE	real	1 ∞ 2	1 ∞ 2	no
8	PROD	PROD	ON NEW FILE	real	2 ∞ 2	2 ∞ 2	no
9	AAAA	AAAA	ON NEW FILE	int	6	6	no
10	FFFF	FFFF	ON NEW FILE	int	4	4	no
11	(TM)	GGGG	ON NEW FILE	real	2 ∞ 4 ∞ 7	2 ∞ 4 ∞ 7	no
12	CCCC	HHHH	IN MEMORY	real	2 ∞ 5	6 ∞ 5	YES

The original Header Array file contained arrays with headers AAAA, BBBB, CCCC, DDDD, EEEE, FFFF. Header AAAA was changed to ABCD using **ch** and arrays EEEE and FFFF were deleted using **da**.

Command **ah** and an extra Header Array file were used to add GAMM, PROD, and a new version of AAAA and FFFF.

Array GGGG was added from the terminal using command **aw** (see section 2.4). Command **aw** is currently being used to modify array CCCC from the old file to a new array with header HHHH; the size of this array has been increased from size 2 ∞ 5 to 6 ∞ 5.

ch	**change a header name**
eln	**examine a long name**
cln	**change a long name associated with a header**

Menu option **ch** is used to change a header on the old file to a new different header. At the end when the array corresponding to this header is copied to the new file (see section 2.5), this array will have the new header associated with it.

Option **eln** is used to display the long name associated with a header and **cln** is used to change the long name to a new value. A “long name” associated with a Header Array is a character string of up to 70 characters which can be used to describe the array in more detail than the 4 character header.

da	**delete one or more arrays from the old file**

To delete one or more arrays, choose the menu item **da**. You will be asked for a list of headers associated with arrays on the old file which you wish to delete. (These headers should be the names on the old file). The associated arrays are marked 'deleted' and at the end of the program (when you exit) are not copied across to the new file. (They are, of course, not actually deleted from the old file, which is never changed by MODHAR.)

Only arrays on the old file can be deleted. (Arrays already written to the new file cannot be deleted.)

The header associated with a new array (added via **at**, **ah** or **aw** commands) cannot be the same as the new header associated with one of the un-deleted arrays on the old file. (This is because *MODHAR* must not allow duplicate headers on the new file.) In such a case you can either delete the old array using **da** or else change the new header associated with the old array using **ch**.

So, in summary, there are two reasons you might have for using **da**.

(i) To mark an old array as deleted so it will not be transferred to the new file when you exit *MODHAR* .
(ii) To clear the header name so you can associate it with another array of data.

Example of case (ii) above

For an old file containing arrays with headers 'AAAA', 'BBBB', 'CCCC', we wish to remove the old arrays 'AAAA' and 'CCCC' and insert from an extra Header Array file, new arrays with headers 'AAAA', and 'CCCC'.

After opening the old and new files, give the following responses.

User input

da	**!**	**delete one or more arrays from the old file**
AAAA	**!**	**header on old file to delete**
CCCC	**!**	**next header to delete**
	!	**carriage_return to end the list**
ah	**!**	**add one or more arrays from Header Array file**
extra_header.har	**!**	**name of extra file**
L	**!**	**list ones to add**
AAAA	**!**	**header on extra file to add**
CCCC	**!**	**next header to add**
	!	**carriage return to end the list**
lh	**!**	**list headers to see the result**

End of User Input

ra	**restore one or more deleted arrays on the old file**

If an array has been deleted from the old file, you can restore it using the menu choice **ra**. You will be asked for a list of headers to restore to the list of headers of arrays to be copied across to the new file from the old file on exit.

As with all cases when a header is added to the list of headers to be added to the new file, *MODHAR* checks for duplicates which might occur on the new file (including those to be copied from the old file). If the program finds a possible duplicate, you will be asked to supply a new header for the array you are restoring.

2.4 Commands for Complicated Modification or Addition

mw	**modify data from the old file and write it on the new file**
aw	**add new data and write it to the file**

If you choose either **mw** or **aw**, a (secondary) *MODHAR* Sub-command Menu becomes active, offering the Sub-commands shown in section 2.4.1 below.

Note that you should use command **aw** rather than **at** or **ah** if

(a) you want to modify the data read before writing it to the new file, or
(b) you want to enter the new data from the terminal.

2.4.1 MODHAR Sub-commands

m	**m**odify the data (before writing it)
ed	**e**xamine part of the **d**ata (held in memory)
is	**i**ncrease the **s**ize of the array
w	**w**rite (all of) the array to the new file
wp	**w**rite **p**art of the array to the new file
a	**a**bort this Sub-command, and go to the Command menu
lh	**l**ist the **h**eaders, their status and size of associated arrays
oh	show **o**ne **h**eader, its status and size of associated array
ch	**c**hange a **h**eader name
eln	**e**xamine a **l**ong **n**ame
cln	**c**hange the **l**ong **n**ame associated with a header
?	help

For users unfamiliar with the operation of *MODHAR*, use of the help command **?** lists the current menu to the terminal screen. The use of the command **lh** (available in both menus) to list the current headers on both the original, old Header Array file (if one is being used) and the new Header Array file will also help clarify *MODHAR's* operation to new users.

MODHAR gives a range of prompts depending on which Subcommand has been chosen. Generally, you name the header of an array to be modified. *MODHAR* reads the array into a work array in memory. You can then examine the data in this array **ed,** increase its size **is** or modify the data **m** by replacing parts of it or scaling its values.

When the modification of the array is complete, it is written to the new file using **w** or **wp**. Once an array is written to the new file it cannot be further modified.

2.4.2 Examples

Some examples may be useful to indicate the operation of *MODHAR* and the types of questions *MODHAR* may ask the user.

Source of data Example 1

When you want to add an array to the new Header Array file, a series of *MODHAR* prompts will ask you where the array data comes from. The possibilities are :

- directly via input by the user at the terminal,
- from a formatted text file,
- from the old Header Array file upon which the new Header Array file is based or
- from some other Header Array file.

Terminal Input Example 2

When you are typing in the actual values for a new array to be added to the new Header Array file, *MODHAR* will ask a series of questions :

- what type of data is being added - real, integer or character values
- the number of dimensions of data
- the actual dimension sizes
- the values for the array data are prompted for, with a clear indication of their position in the array. For example, for a *6x3x4* real array, you will receive prompts of the form:

```
Please enter the  6 values for positions (1-6, 2, 3)
```

Modifying the data Example 3

When modifying an array that has been just read by *MODHAR* (including an array whose values have just been input by the user), *MODHAR* will ask such questions as:

- how much of the array is to be modified ? - possibilities are one single entry, a few entries, a rectangular sub-array or the whole array
- is the modification a scaling of the current values or a replacement?
- are all entries to be modified in the same way ? i.e. either scale all values by the same factor, or replace all values with the same replacement value
- if the modification is different for different values, are the individual modification (scaling or replacement) values read from the terminal, from a text file, from the old Header Array file upon which the new Header Array file is based or from some other Header Array file ?

Modifying a sub-array on an old Header Array file Example 4

This example gives full responses for a run of *MODHAR* which

- Modifies an array on the old file before writing it to the new file
- First the array dimensions are increased from 2 ∞ 3 to 4 ∞ 4. (The new positions initially contain zero values.)
- Then two new data values are added as a subarray in positions (3–>4,4) from data input from the terminal

The old array 'abcd' was

```
4.0  2.0  1.0
1.0  6.0  8.0
```

After modification the new array will be

```
4.0  2.0  1.0  0.0
1.0  6.0  8.0  0.0
0.0  0.0  0.0  3.0
0.0  0.0  0.0  5.0
```

User input

```
<carriage-return>      !   carriage return to end option selection
y                      !   based on an old Header Array file
old_file.har           !   Name of old Header Array file
new_file.har           !   Name of new file
lh                     !   list headers of arrays on old file
mw                     !   modify an array on old file and write it to new file
abcd                   !   header of array to be modified
is                     !   increase size of array
1  2                   !   increase dimension 1 by 2, that is, from 2 to 4
2  1                   !   increase dimension 2 by 1, that is, from 3 to 4
end                    !   end selection of dimensions to increase
m                      !   modify this array before writing to new file
r                      !   replace some parts of the array data
s                      !   modify a subarray
1  3  4                !   select entries 3 to 4 in first dimension
2  4  4                !   select entry 4 in second dimension
end                    !   end of selection of subarray (3-4,4) of size 2 ∞ 1
g                      !   general matrix used to modify array
t                      !   input modifying matrix from terminal
3.0  5.0               !   enter 2 values since array is of size 2 ∞ 1
?                      !   view menu for list of commands
w                      !   write all the array to the new file
n                      !   don't want to reuse data in memory
ex                     !   exit saving the new header array file
a                      !   all remaining arrays to be transferred
=                      !   use old history without adding to it
```

End of User Input

2.4.3 Writing Arrays to the New Header Array File

Once you have completed all modifications to an array, the array can be written to the new Header Array file. *MODHAR* gives you the choice of writing either the entire array (using the **w** Command) or only a rectangular sub-array of the entire array (using the **wp** Command). Before proceeding onto another array, you can also base an entirely new Array on the most recently written array. In this way, any complicated modifications just performed for that array need not be repeated.

Note that, although there is an **is** Subcommand for increasing the size of an array, no **ds** (“Decrease the size”) Subcommand is needed since you can decrease the size by only writing part of the array.

Writing part of an array using ***wp*** Example 5

This example gives full responses for a run of MODHAR which

- examines part of an array using the **ed** command
- writes part of an array to the new file using the **wp** command.

Initially the array has dimensions 4 ∞ 3 ∞ 4 ∞ 5 ∞ 2.

On the new file, the sub-array written is

1 , 1 -> 3 , 2 , 2 -> 5 , 1 -> 2

The new array has dimensions 1 ∞ 3 ∞ 1 ∞ 4 ∞ 2.

User input

```
<carriage-return>          !   carriage return to end option selection
y                          !   based on old file
old.har                    !   old file name
new.har                    !   new file name
lh                         !   list  headers on old file
mw                         !   choose modify and write option
JJJJ                       !   header of array to be modified
ed                         !   examine data
s                          !   examine a submatrix
2  1  3                    !   second dimension elements 1-->3
3  2  2                    !   third dimension element 2
4  2  5                    !   fourth dimension elements  2-->5
5  1  2                    !   fifth dimension elements 1-->2
end                        !   end of selection of submatrix
?                          !   list menu of options
wp                         !   write part of the array to the new file
2  1  3                    !   write second dimension elements 1-->3
3  2  2                    !   third dimension element 2
4  2  5                    !   write fourth dimension elements 2-->5
5  1  2                    !   write fifth dimension elements 1-->2
end                        !   end selection - written 1 ∞  3 ∞ 1 ∞  4 ∞  2 array
n                          !   don't reuse data in memory
lh                         !   list headers
ex                         !   exit saving the new file
a                          !   all remaining arrays to be transferred
Jill                       !   name as part of history
12 -1-93                   !   date (Only prompted for on some machines)
Example of selecting part of an array
**end
y                          !   accept history for new file
```

End of User Input

2.5 *Finishing Up*

If the new Header Array file has been based on an existing, Header Array file, then when you have finished either adding any new arrays or writing any modified arrays to the new Header Array file, *MODHAR* gives you the option of copying, unchanged, some or all of the arrays from the old file that have not yet been transferred to the new file. So, if you wish to copy an existing Header Array file containing 100 arrays, only one of which needs to be modified, then you only need to access explicitly the array to be modified. Once the single modified array has been written to the new Header Array file and you enter the **ex** command to exit *MODHAR*, *MODHAR* will ask if you want to copy some or all of the untouched 99 arrays from the old Header Array file across to the new Header Array file. In this way, you are saved from having to access explicitly any array that requires no modifications.

2.5.1 Exit Commands

ex	**exit, saving the new Header Array file**
q	**quit without saving the new Header Array file**

When you have finished adding, deleting and modifying arrays, menu choice **ex** is the way to exit *MODHAR* if you want to save the newly created Header Array file. Choice **q** aborts the process and deletes the new file.

If the new Header Array file has been based on an existing Header Array file and you give command **ex**, *MODHAR* gives you the option of copying unchanged some or all of the arrays from the old file which have not previously been transferred to the new file and have not been deleted using **da**. The choices are:

[a] ALL remaining arrays to be transferred
[L] LIST headers of arrays to be transferred
[m] MOST of the arrays (List those not transferred)
[p] Respond to PROMPTS to enter which to transfer
[n] NONE of the arrays to be transferred.

[a], [L], [m], and [n] are similar to those in menu item **ah**.

For the choice [a], the program adds ALL the non-deleted arrays on the old file to the new file.

For the choice [L], you supply a LIST of headers of arrays on the old file to add to the new file. All other arrays on the old file will be omitted. Giving a carriage-return instead of a header ends the list of headers.

For the choice [m], *MODHAR* adds MOST of the arrays on the old file. You list the headers that you want to OMIT. A carriage-return ends the list.

For the choice [n], NONE of the remaining arrays on the old file are added.

For the choice [p], *MODHAR* prompts you with the header names of undeleted arrays on the old file and you choose from the following list of actions:

[y]Transfer this array to the new file.
[n] Do NOT transfer this array.
[r]Transfer this and ALL REMAINING arrays
[q] Do NOT transfer this or ANY REMAINING ARRAYS.
[?]Display this menu again.

After transferring the requested arrays, a list of headers is prod- uced showing the status and sizes of the arrays on the old and new files.

2.5.2 History of the New File

Finally you are asked to add to the history by supplying your name and any notes you wish to append to the current history about what changes have been made in this run of *MODHAR*. (On some machines you may also be asked to input the date.)

If you don't want to add to the history in this way, you can choose from

[-]ignore any old history. (Start the history with a fresh slate - type in your name and a new history.)
[=] to take any old history (no new). (Copy the old history to the new file without adding to it.)
[0] to have no history on the new file.

2.6 *Complete Example of a MODHAR Run*

The following example gives all responses for a MODHAR run. If you want to try it yourself, create the following three files (either using option **at** and taking data from a text file, or using option **aw** and giving input from the terminal.)

Initially you need

(1) An old file ‘old_file.har’ containing arrays with headers ‘AAAA’, ‘abcd’, ‘EEEE’ and ‘FFFF’

(2) An extra file ‘moupd.har’ containing arrays with headers ‘prod’ and ‘gamm’

(3) A second extra file ‘extra_header.har’ containing an array with header ‘AAAA’.

User input

```
<carriage-return>       !   carriage return to end option selection
y                       !   based on an old file
old_file.har            !   name of old  file
new_file.har            !   name of new file
lh                      !   list headers on old file
ah                      !   add arrays from extra Header Array file
moupd.har               !   name of extra file
L                       !   list headers of arrays to add
prod
gamm
<carriage-return>       !   carriage return to end list
lh                      !   list headers to see 'prod' and 'gamm' on new file
da                      !   delete arrays on old file
AAAA                    !   header of array to delete
FFFF                    !   header of array to delete
<carriage-return>       !   carriage return to end list
lh                      !   list headers to see deleted arrays 'AAAA' and 'FFFF' on old
file
ah                      !   add arrays from an extra Header Array file
extra_header.har        !   name of extra file
L                       !   list headers to add to new file
AAAA                    !   header to transfer (permitted since old 'AAAA' is deleted)
<carriage-return>       !   carriage return to end list
lh
ra                      !   restore array deleted from the old file
AAAA                    !   this is the original header on the old file
BBBB                    !   necessary to enter a new header because 'AAAA' is on new
file
<carriage-return>       !   carriage return to end list to restore
lh
aw                      !   add a new array and write it to the new file
o                       !   from the old file
cdef                    !   header of array on old file
fghi                    !   'cdef' is on old file so a new header is needed 'fghi'
w                       !   write array to new file
n                       !   don't reuse array in memory
lh
ex                      !   exit saving the new file
a                       !   all remaining (non-deleted) arrays to be copied to new file
Jill                    !   add to history on new file
29/9/92                 !   Date (Only prompted for on some machines)
Example for MODHAR
**end                   !   end of history
y                       !   confirm history before adding
```

End of User Input

After running this example, on the new file are arrays with headers 'BBBB' (was 'AAAA' on old file), 'abcd', 'cdef', 'EEEE'. All these were originally on the old file.

Also on the new file are arrays 'gamm' and 'prod', and the new version of 'AAAA' from extra Header Array files.

The new file also contains a second copy of array 'cdef' from the old file now renamed 'fghi'.

APPENDIX A – TEXT FILES

MODHAR allows the use of prepared text files to input a variety of data, the data values of an array, the scaling factors to be applied to values in an array, or the replacement values for arrays or subarrays.

A text file is a file which can be viewed or edited with a text editor.

Full details of preparing arrays for text files are given in Appendix C of Document GPD-1.

In *MODHAR* when using option choice **at**, the text file can contain more than one array of data. In other parts of *MODHAR*, only one array should be given on each text file.

Differences from earlier versions of MODHAR

Although the text file format for Release 5 is different from that for earlier releases, it is most important to realise that **Header Array files** created using Release 4.2.02 or earlier releases (including these prepared using text files) **do not need any changes for use with Release 5**, However, if you wish to change a Release 4.2.02 text file and use it with Release 5 GEMPACK software to create a new Header Array file, you will need to change the text file. This is easy to do, as explained below.

The initial "how much data" information on text files no longer contains the number of dimensions before the dimension sizes. This should be removed from the start of the line,

In the previews version of *MODHAR*, the arrays had to be given in column order. To convert text files used with Release 4.2.02 of *GEMPACK* for use with Release 5 of *GEMPACK*, the simplest way is to specify 'col_order' as part of the initial line and also to add a final semicolon ';'.

Example

For a three dimensional matrix 2 ∞ 3 ∞ 4, for real values, the initial "how much data" information could be

```
2   3   4   col_order ;
```

followed by the data values.

A more complete description of the simple procedure for converting Release 4.2.02 text files for use with Release 5 *GEMPACK* programs is given in section C.3 of Appendix C of GPD-1.

References

GEMPACK DOCUMENTS*

GPD-1, *An Introduction to GEMPACK,* Second edition, April 1994, pp. 252+15.

---

* The numbering of GEMPACK Documents has been re-started with Release 5 of GEMPACK, when the abbreviation "GPD" was first used. Previous editions of these documents did not have the same numbers as the current editions. Pre-Release-5 documents are numbered "GED-xx".