4/15/2006

Chapter 3 Preparing for Analysis: Managing Time Series Data

The previous chapter focused on data display, with very little overt attention specifically paid to MODLER's analysis or data base management capabilities. However, as noted earlier, in practice there is not necessarily a hard-and-fast separation between display and analysis, nor between these and data retrieval. For one thing, MODLER lets you combine and transform variables just by typing relevant formulae, even in the context of what is ostensibly a display command. Consequently, expressions such as:

```
100*(CONS+GOVP)/GNP
 exp(0.2*ln(WPAGR)+0.2*ln(WPEN))
```

are instantly evaluated in place and the results then displayed as plots or lines of a tabular display as easily as in the case of a single series. Moreover, these expressions can be captured from one context and pasted into another in less time than it has taken for you to read this. In the context of MODLER, they are both operative and fungible.

Thus, there are at least two types of analysis facilities built into MODLER: the implicit facilities that are embedded into MODLER's commands as well as those that are designed as explicit "analysis" features. Much the same could be said of data base management. MODLER has been designed to foster this type of coordinated processing whenever possible, with the aim of permitting you to do your work without having to worry too much about the constructive details of the process. Nevertheless, as background, it is helpful to know something about the way in which the program does its work. In particular, the provision MODLER makes for issuing free-form commands and for the automatic storage of the results are each important issues. As subjects, these operations provide a good preliminary introduction to data analysis.

Considering once more the expressions displayed above, notice in particular two of their characteristics. First, that they incorporate both operators (+, *, and /) and implicit functions ln( ) and exp(), that jointly instruct the computer to perform—in this example—addition, multiplication, and division and simultaneously transform the level values of variables, referenced by name (WPAGR and WPEN), into logarithms and exponential values. Second, that they not only refer to each of the variables (CONS, GOVP, GNP, WPAGR, and WPEN) as algebraic elements, but also simultaneously cause their values to be retrieved from an online data source, and then use these to generate the computed values of the expressions. In the process, MODLER in addition recognizes and evaluates scalar numbers (0.2 and 100) and other syntactic elements. The idea that written expressions can stand for an evaluated set of numeric values is so familar and well integrated into modern human thought processes that this particular aspect of the displayed expressions probably will not strike you as in any way new or novel, but what is actually relatively new—a result of the invention and development of the electronic computer—is the fact that these expressions are now potentially operative, not just symbolic.

Such expressions have been part of MODLER's command structure since 1968-69, now almost 40 years ago, and in those early days MODLER was one of the very first computer programs to incorporate this capability, especially in an interactive context. In order for this type of operation to be performed, it is first necessary to interpret such expressions syntactically, applying mathematical logic. Second, it is necessary both to retrieve the vector values of the variables from some source and, once the expression has been computed, to store the evaluated expression values until they can be displayed or otherwise used. MODLER retrieves variable values from what is often generically called a "data base," but more specifically is called either a "data bank" or a "Memory File," as you will know from the previous chapters of this user guide. By default, the program stores computed results in the Memory File, which is essentially a temporary storage facility that is automatically created by MODLER whenever it needs a place to store sets of numeric values for subsequent use. At the end of each session you can either dispose of any Memory File that has been created or else save its contents for use in a later session. At that time, if a Memory File has been created, MODLER will ask you what you wish to do with it. The program can also create and manage data banks, either on their own or in conjunction with the Memory File. The critical distinction between these types of storage facilities is that characteristically data banks are at least semi-permanent. They are both more archival in nature and also likely to be larger in terms of the number of series they contain.

The original concept of a macroeconomic data bank developed among econometric model builders in the middle 1960s. Even today, an economic data base generally consists of one or more data banks and this separation still appears natural both because certain conceptual groupings of time series data seem logical—flow of funds, national income and product accounts, employment data, price data, and the like—and because it is common to maintain such data banks by adding or revising observations over time, thus establishing the notion of "banking" the data. Data banks, as subdivisions of data bases, also permit collections of data to be shared easily, or else to be developed and maintained by one or more members of a research team and then used by others. One of the initial motivations, that lasted certainly until the 1990s, was the limited amount of data that could then be passed easily from one computer to the next, usually less than 2 megabytes at a time, making it important to consider file size as a criteria. In addition, computers were also slower, so that transfer times were also a consideration.

Today, in contrast, USB flash memory devices and other such finger-size transfer facilities have grown in capacity to the point that they can now hold literally as much economic data as you are likely to use in your lifetime. In addition, they can transfer this quantity from one place to another in seconds. No longer is the issue one of size as a physical constraint. Instead, it is a matter of conceptually organizing your data base so that it remains comprehensible. Working with 10 data series is quite a different matter than working with 1000 or more, and it is as you move from the first sized data base to the second that problems tend to arise.

Actually, data retrieval using MODLER could hardly be easier whatever the size of data base. In order to retrieve the values for a given time series it is necessary only to identify the data bank(s) (or Memory File) you wish to use, specify the extensive date range you wish to work with, the observational frequency of the data, and then simply ask for each of the series by name in the course of doing your work. MODLER will retrieve the data, automatically truncating the date range to fit the actual availability of the series values. Consequently, if you are fortunate enough to have someone else create your data banks, document the data series, give them easy-to-remember names and then to provide these banks to you—for example as an automatic download from the Internet—you will need to know very little about MODLER’s data base management capabilities. The temporary series you happen to create will automatically be placed in a Memory File and you can either save this file from session to session, or delete it, as your needs dictate. Since you will have created these temporary series, you will presumably know their names, so that very little other documentation, beyond name, will be required. The situation is in fact not dissimilar to that of using words in your native language. So long as you know the words and how to combine them, you are home free.

However, there is such a thing as a memory lapse, when you return to using data series created a while ago, or else the possible need to use data banks and other files created by other people that are idiocyncratic in their characteristics. To a degree, such situations are relieved by MODLER’s design. You will know from the discussion of the *Find* and *Index* facilities, found in the earlier chapters of this user guide, that MODLER provides a number of memory aids. But what it cannot provide is the actual data base documentation itself. The *Find* facility is limited by the degree to which the series are well-described. The *Index* facility is limited by the degree to which logic has been used in the naming of series, so that they are truely mnemonic in nature. The purpose of this somewhat extended chapter introduction is to make you aware of the importance of considering carefully how to name and document series, now, at the beginning, when you have only a few to work with, for as you work, like rabbits over time, they will multiply. Before you know it, instead of 10 to 50 series, you may have 500 to 1000 or more. If you have planned for this eventuality, MODLER will serve you well. If you have not, you may then be frustrated.

As a data base manager, MODLER effectively has no size limitation. It is designed to work with data banks that each can contain as many as 10,000 series and in addition it permits you to access as many as 15 data banks at a time, in addition to any Memory File. In principle, therefore, you will at any time potentially be able to choose from among approximately 150,000 series simultaneously. However, the issue is not really how many thousands of series you can "bank." Even if you do not ever wish to deal on this size level, it is still possible to organize your data banks so that they serve effectively as separate data "folders," thus providing a way to organize and classify types of data series. You can also copy from bank to bank, so that even at a later stage you will be able to combine into a single bank, or into a group of banks, sets of data series that have some logical connection, such as all being used with a given econometric model. Furthermore, you have several organizing principles as alternative choices: for example, series can be dealt with in terms of their alphabetical organization by name, but it is also possible, later, to create conceptual series groupings, on any basis, that once established will then permit you to copy series in sets from place to place. In addition, should you ever wish to organize your data in the form of a relational data base, there is a companion MODLER facility, *EISMaker*, that permits a large-scale relational data base to be created that can be used not only to organize a wide-area (even world-wide) Internet-based Economic Information System, but even in the case of a single machine can be used to create and manage MODLER data banks more or less automatically. At any point, you can become more sophisticated and comprehensive in your approach, should you wish, but fundamentally at whatever level you choose to operate at each point in time, it always remains important to have considered carefully how you document and name your series as a fundamental building block of your work. If you have done this, any transitions you later make should be quite smooth.

The purpose of the present chapter is to introduce certain of the implicit analysis features that MODLER offers, but its main focus, as the foregoing discussion implies, is the program's time series data base facilities. Simply as a user of data, these facilities are really quite easy to use and, if this is your role, you will possibly not then need to know even what has been discussed so far. Variables in an economic data base have four essential characteristics: they each have a name, an observation frequency, a date range of availability, and a storage location. Once you have specified the name of the data bank(s) you wish to use, if you then set the date range you wish to consider and specify what the observation frequency should be, all you need to do is to use the names of the series you wish to work with. You can have as many as 15 data banks open at a time, and in the thick of work you generally do not even need to know which bank contains which series.

The Nature of Free-form Commands

As we have seen in the first two chapters of this user guide, explicit commands are not the only way to operate the program, since both menus and icons are also provided, but given that econometric models necessarily involve equations and other mathematical expressions, such commands are inevitably involved when using the program. Many operations are visually oriented, but MODLER also requires that you understand at least certain aspects of its command language, sufficient to allow you, at minimum, to form equations and define transformations. This background will also be useful when considering data base management and MODLER's analytic facilities.

If you are new to the program, there are two aspects of the operations just referred to that you might feel are not yet obvious. The first is the actual process of issuing commands. One place to enter commands is the large yellow or pastel blotter space in the middle of MODLER's opening, or Central Control, screen—shown in Figure 3-1 below. This blotter is sufficiently like that of a word processing program, as well as those of other programs, that the idea of entering text into this space should not be too startling. Furthermore, as you might expect, if you click your mouse anywhere on this blotter, the immediate effect is likely to be that a cursor starts blinking slowly, at first in the upper left hand corner. Just as with a word processor, this blink signals to you the location of the text entry point—in this case the entry point of a potential command stream.

However, as an alternative, you can instead tell MODLER explicitly that you wish to enter a command, by choosing first the **File** item of the main menu and then clicking on the element **Issue Command**, as highlighted in Figure 3-1.

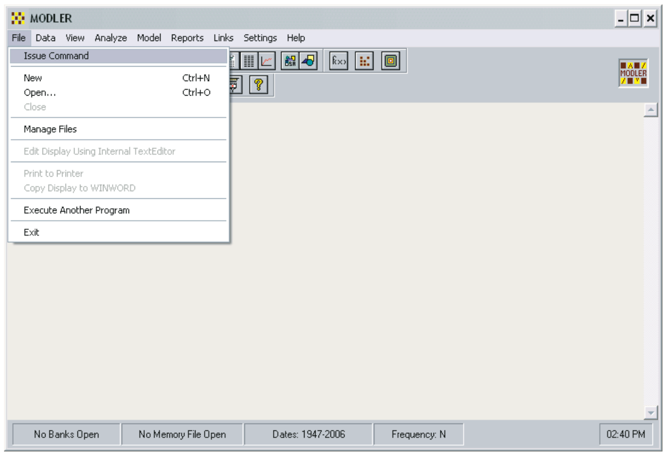


Figure 3-1. Explicitly Issuing a Command

The effect of clicking on this element will be to cause the form shown in Figure

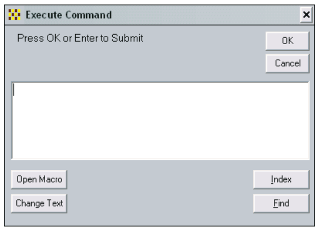


Figure 3-2. Command Entry Form

3-2 to be displayed. An obvious question at this point is, what's the difference? On one level, the only evident difference is that between an implicit action and an explicit one.

However, in one particular case, the difference does matter. You may recall from earlier chapters that whenever text appears on the blotter of MODLER's opening screen, you can capture it, simply by double-clicking on this blotter. In response, you will immediately be asked if you wish to copy this text to the program's internal text editor. Alternatively, it is also possible to swipe your mouse, using its left click button, in order to highlight a portion of the opening screen blotter's text and in the process capture at least some of it. In this instance, the captured text will immediately be transferred to the white blotter of the Execute Command form, shown in Figure 3-2, which form will then automatically appear on your screen, although in this case it will instead be entitled *Recall/Edit/Execute Command*; obviously, the purpose of this title change is to let you know that editing may be involved. But whatever the title, this form's blotter is effectively a small text-editing space, giving you not only the capability to type in a command, but also to paste text, or to cut or copy it. Right clicking your mouse on this white blotter exposes these text editor features in the form of a pop-up menu. However, in any case, you need to be aware that, in the end, when you click **OK**, MODLER expects a single command, although it can be a multiline command.

In contrast to the blotter of the form shown in Figure 3.2, the blotter of the opening screen (shown of course in Figure 3-1) can be used by you *only* for direct text entry, although MODLER additionally uses it to display commands that have been issued either via menus or when icon buttons are pressed. Even though MODLER can, you cannot *directly* use this yellow blotter to paste commands that are cut or copied from other places. An important reason for this convention is that, after the fact, this blotter also serves to provide an historical record of what has been done; it is not just a current command entry point. Therefore, to permit a new command to be pasted into the middle of other, already executed commands could be confusing—for example, would this action imply that these other commands should be re-executed? Or ignored?

The Execute Command form, shown in Figure 3-2, thus provides a solution: from one perspective, it can be viewed as providing immediately-available, rather powerful, self-contained onscreen editing facility, including the ability to edit and re-use prior commands, but without at the same time forcing you to answer questions about what to do with the results—in contrast, a necessity when using MODLER's text editor or, for that matter, any word processing program or other third party text editor. In order to avoid the need for you to intervene, this form is "trained" to place all the commands it executes directly into the program's command stream, simultaneously displaying these—but now as historical information—at the bottom of the opening screen's yellow blotter.

The Memory File

What we have actually been considering so far in this discussion are certain book keeping operations that MODLER performs, as well as particular associated conventions. The appropriate placement of commands, as well as the capability to retrieve, edit and re-use earlier commands in an immediate way, are in effect automatic book-keeping operations—or at least involve keeping track of these operations. Another important tracking function is numeric data management, for in the case of an analytical program, data management is not simply a matter of data retrieval. When transformations and other operations are performed, not only will series be retrieved and used, but they will also often be created and possibly later even modified. As you will know from previous discussion, whenever series are newly created, either explicitly or as a by-product of some operation, if they need to be kept longer than a few seconds MODLER will ordinarily put them into a temporary work file known as a "Memory File." If this file already contains other series, new series are simply added. However, if the file does not exist, it will be created. Once a Memory File is created and contains data series, you can then later retrieve and use these simply by giving their names in an appropriate context..

Consider a simple example: in order to perform a calculation, MODLER usually needs to know something about the data to be used, generally the observation frequency and the date range. Thus it is good practice to begin by setting the frequency and dates, either by clicking on the status bar or using the **Settings** menu and choosing **Boundaries**. Alternatively, from the Central Control screen you can issue the commands:

```
SET FREQ=QUARTERLY
SET DATES=198001-199204
```

Next, to cause the creation of a Memory File, from the Central Control screen try typing the simple equality:

```
A=1
```

In response to this command, MODLER will create a Memory File with suitably wide date limits, its choice of these depending on either the dates you have set or the default date limits, whichever is greater. This Memory File will contain a single series, named “A,” with all its observation values equal to 1 over the date range you set. The status bar will now display “Memory File Open” to indicate that the Memory File has been created. Click on this status bar element if you wish to view the Memory File index. Or you can display the series itself, either by typing:

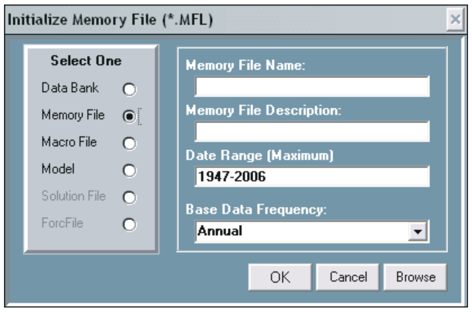
View A

or else by clicking on the **View** menu item and then choosing **Single Series or Transformation**.

But what if you would like to make changes, even to the point of deleting the new Memory File? Once a Memory File has been created, if for instance you do not like the date limits chosen by MODLER, or if you want to change the base data frequency, you can type:

CLEAR MEMORY FILE

and start again by clicking on **New** from the **File** menu category, which will display the form shown in Figure 3-3. In this case, you can make sure that the Memory File is opened with the particular base frequency and date limits you want, since MODLER permits you to supply directly a description and date limits; the latter do *not* have to conform with the date range set earlier for transformations. Notice that you will, however, need to click the radio button labelled Memory File, in order to specify that it is a Memory File that you wish to initialize.



Select One: [✓ Memory File]

Memory File Name:

Memory File Description:

Date Range (Maximum): 1947-2006

Base Data Frequency: Annual

Figure 3-3. Explicit Memory File Initialization

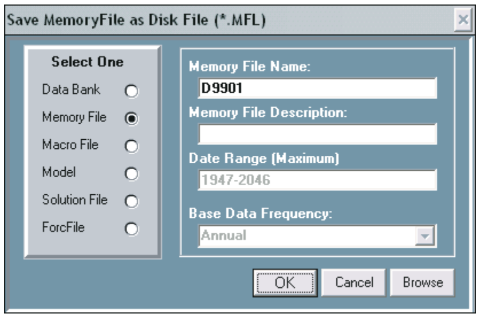
The date limits and frequency you select will jointly determine the maximum number of observations per series for this Memory File. This choice of frequency does *not* limit you to using series only of that frequency. A named series, once stored, can only contain observations of one frequency, but if they have different names, series of different frequencies can co-reside in the same Memory File. But to reserve the appropriate amount of storage space, you should initialize a Memory File with its base frequency set to the *highest* frequency of *any* series you intend to store in it.

At this point, it might be useful to pause for a moment's reflection. We have just considered two ways of creating a Memory File, the first as a by-product of creating observations on a series and the second as a conscious action. The fundamental difference between the methods is that in the first case, the date limits and base frequency are set by the operation of creating a series whereas in the second we set these limits independently of any particular transformation or other data generation operation.

In the same sense that you may want to control the file creation process, you may also want to save the Memory File from session to session. This file may be created initially in order to provide for temporary data storage. However, if you create a Memory File containing data that you want to keep, you can tell MODLER to save a copy by issuing the SAVEMF (SAVE MemoryFile) command:

SAVEMF D9901

Alternatively, you can click on **File | Save Memory File As**. In either case, MODLER will respond by displaying the form shown in Figure 3-4. If saved, the file in this example will appear in your Home directory as D9901.MFL.


Save MemoryFile as Disk File (*.MFL)

Select One
- Data Bank
- [✓ Memory File]
- Macro File
- Model
- Solution File
- ForcFile

Memory File Name: D9901
Memory File Description:
Date Range (Maximum): 1947-2046
Base Data Frequency: Annual

Figure 3-4. Saving a Memory File

To reload the Memory File data later, in the same or a subsequent MODLER session, the command is:

LOADMF D9901

The alternative to using these commands is to click on the relevant, rather obvious items under **File**, as noted. However, to use the menu options here is rather more long-winded than issuing the commands.

Combining Data in the Memory File

Once it exists, you can put data into the Memory File in several different ways. To copy a series called GNP from an open Data Bank, use:

FETCH GNP

or

FETCH GNP=GNP

or simply:

GNP=GNP

where, in the second and third cases, the name to the left of the equals sign can be different from that of the series in the Data Bank.

What is the difference between these commands? The answer is: if, previously, the Memory File series did *not* exist, they are exactly equivalent. However, if it exists, then the FETCH command *must* be used whenever the names are the same in both data bank and Memory File; otherwise, the series will instead be retrieved from the Memory File, resulting in a do-nothing operation. The reason is that, except when the FETCH command is used, the Memory File is *always searched first, before any bank*, whenever data series are to be retrieved. Conceptually, the Memory File is your immediate workspace, so that for retrieval it is primary and any open data banks are secondary; only when formally *updating* a bank is this not true.

You can also copy a group of series. For instance, the command:

COPY SERIES=CON* FROM DEMOBANK TO MEMFILE

will fetch all series from DEMOBANK the names of which begin with the letters "CON". The MODLER syntax obeys the standard wildcard conventions, with ? serving as a placeholder and * indicating that the remaining characters (including blanks) are to be accepted whatever they are. Or you can click on **Data | Copy Series** to perform the same operation.

In addition, you can create a new series in the Memory File (or overwrite an existing one) using a formula or function: for example

SAVRAT=100*(1-CONS$/YPERS$)
COSTINDEX=exp(0.2*ln(WPAGR)+0.2*ln(WPEN))

Notice, by the way, that these commands each contain an equals sign and, to the left of it, a series name. For the evaluated results of an expression to be stored in a

Memory File, this convention must be observed. In the introduction to this chapter, we discussed expressions used in display commands and you may have got the impression there that somehow these were magically stored even without being explicitly named. Unnamed expressions and transformations are never stored.

Managing The Memory File

The Memory File has an index very much like that of a Data Bank. You can search this index using the INDEX command; for example:

INDEX
INDEX (CON-CP)

Note that the word INDEX is used on its own, rather than in combination with any sort of file name, and in this form is understood by MODLER to refer to the Memory File. Both these commands produce brief indexes, but the second displays only series the names of which fall into the name range from CON* to CP*. Alternatively, if you wish to find the names of series that have particular characteristics, and display a standard index, you can use the expanded form of this command:

INDEX (CON-CP, STD)

The same commands that are used to build and update series documentation in a Data Bank can also be used for Memory File series. However, the Data Bank (rather than the Memory File) is intended to be used for archival storage; if you go to the trouble of documenting your series it is usually best to set up your own Data Bank, as described later in this chapter, although you can document in the Memory File and then move the documented series, *as a new series*, to a data bank..

A useful Memory File facility is the RENAME command:

RENAME CPI CPIX

Here it is used to rename CPI to CPIX. You may want to rename a series before displaying it in a hard-copy plot, in order to group series alphabetically, or to distinguish variants from different sources (as, for example, in the case of results from different simulations of a model).

Sometimes you may also wish to delete one or more Memory File series. The command is:

DELETE seriesname

Similarly to the case when you delete a DOS or Windows file, the delete command does not actually remove the series from the Memory File; instead, the series is simply "turned off." Consequently a series can be restored as well:

RESTORE seriesname (k)

where k is the location number ("series number") displayed when a series is deleted. This number is automatically displayed during the deletion process and if you think that you will ever need to restore the series you should make a note of it, inasmuch as it will not appear again. The principal reason this number appears is so that you can then and there restore a series that you have unintentionally deleted, rather than because you are expected to want to restore series as a general operation.

The menu analogues of all these commands can be found under the **Data** category of the Central Control screen menu or, in the case of the index command, by clicking on the status bar item "Memory File Open."

Operators and Functions

Several mathematical expressions have been displayed so far, more or less without comment concerning their style and form. In general, expressions are written using the normal mathematical conventions, although in a few cases the particular conventions reflect the present computer context, as opposed to using pencil and paper. The essential difference, other than the lack of Greek symbols (which would be tedious to use on a standard computer keyboard) reflects both the operative nature of computer commands and that time series variables are vectors, not scalars. In particular, lagged observations are indicated by putting (-k) after the series name, where k is some integer number: for example

DGNP1=GNP-GNP(-1)
DGNP2=GNP-0.5*GNP(-1)-0.2*GNP(-3)-0.2*GNP(-4)

This particular convention reflects that in a vector representational context, it is obviously only necessary to express the offset. However, the minus sign is critical; a positive value in the parentheses means something quite different.

Most of the implicit functions available are listed in Figure 3.5 below. They include the TIME function and dummies, for example.

TIME(1,198001)
DUM(Q3=1)
DUM(197401=1.0)

Figure 3.5

MODLER Functions

LN(expr)	natural logarithm
EXP(expr)	exponential
LOG10(expr)	logarithm to base 10
SIN/COS/TAN(expr)	trigonometric functions
ABS(expr)	absolute value
ROUND(expr)	round to whole number
TRUNC(expr)	truncate to whole number
MOD(expr1,expr2)	remainder of expr1 on division by expr2
PDIFF(expr1,expr2)	absolute difference expr1 minus expr2
SIGN(expr1,expr2)	absolute value of expr1 with sign of expr2
APCT(expr)	annualized one period percentage change
APCT(j,expr)	annualized percent change in series compared with n periods before (j=1,2,4,12 or 52)
CUMS(C,expr)	Cumulative Sum: x(t=x(t-1)+expr(t); x(0)=C
DIF(expr)	first difference: expr(t) - expr(t-1)
DIF(n,expr)	expr(t) - expr(t-n)
GROW(C,expr)	Grow Function: x(t)=x(t-1){1+expr(t)*0.01};x(0)=C
LAG(n,expr)	expr(t-n)
LEAD(n,expr)	expr(t+n)
LGT(expr)	logit: ln[expr/(1-expr)]
LN(n,expr)	ln(expr(t-n))
MAVG(n,expr)	Sum[expr(t),...,expr(t-n+1)]/n
TRND(expr)	Time trend regression values of series
MAX(expr)	maximum value in current date range
MIN(expr)	minimum value in current date range
MEAN(expr)	mean value over current date range
SDEV(expr)	standard deviation over current date range
VAR(expr)	variance of current date range
SELECT(i/j,series)	ith observation out of every j
FTRAN(CODE,series)	higher to lower frequency conversion
DIS(CODE,series)	lower to higher frequency conversion
TIME(value,date)	time variable taking set value at given date
DUM(date=value)	all observations zero except at given date
DUM(Fk=value)	seasonal dummy: F=S,Q,M, or W; k=1,2,...

The arithmetic operators recognized by MODLER are:

******	Power
*****	Multiplication
/	Division
+	Summation
-	Subtraction

These operators are given in hierarchical order. Power first, Multiplication and Division (co-equal) second, and Summation and Subtraction (co-equal) lowest. As usual, parentheses can be used to override this ordering. Otherwise, as is normal, processing proceeds from left to right whenever terms of comparable order are involved.

The logical operators are:

.AND.
.OR.

and the relational:

.LT.
.LE.
.EQ.
.GE.
.GT.

Mathematical, logical, and relational operators may be mixed in expressions. Mathmatical operators are highest order, Relational are second, and Logical lowest. In this case also, parentheses may be used to override the implicit ordering conventions. As before, whenever comparable terms are involved, processing proceeds from left to right in the normal manner.

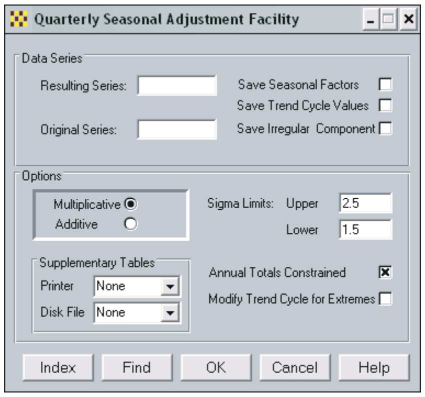
Seasonal Adjustment

In addition to permitting you to make arithmetic and functional tranformations, MODLER allows you to perform both monthly and quarterly seasonal adjustment, using the Census X-11 method; as an option, weekly seasonal adjustment is also supported. Separate user guides are available that describe the seasonal adjustment facilities in detail. However, at the most basic level, to seasonally adjust a series, simply execute the MODLER command:

newseries=**SEAS**(oldseries)

where “new series” is the name under which you wish the seasonally adjusted results to be stored and “oldseries” is the name of the original unadjusted series. Depending upon the frequency set, MODLER will automatically execute the appropriate version of the seasonal adjustment procedure. The oldseries observations will be retrieved from an open Data Bank or Memory File. The results will be automatically stored in the Memory File or designated Data Bank, depending upon whether or not an UPDATE command is in force; this command is described later in this chapter.

The alternative to issuing such commands is to click on the **Data** category of the Central Control Screen menu, then choose **Seasonal Adjustment**. If the observation frequency is set to Quarterly, the form shown as Figure 3.6 will be displayed. This form indicates the full range of options available. It also provides immediate access to context sensitive help.


Quarterly Seasonal Adjustment Facility

Data Series
Resulting Series:
Original Series:
Save Seasonal Factors ☐
Save Trend Cycle Values ☐
Save Irregular Component ☐

Options
Multiplicative ●
Additive ○
Sigma Limits: Upper 2.5
Lower 1.5
Supplementary Tables
Printer None
Disk File None
Annual Totals Constrained ☒
Modify Trend Cycle for Extremes ☐

Figure 3-6. Quarterly Seasonal Adjustment

Seasonal adjustment is interesting here for two reasons. The first is that the procedure allows you to adjust for seasonal variation, which in itself is often desirable. The second is that this procedure is performed using a powerful MODLER facility: the capability to execute other programs concurrently as subtasks, including sharing data. To perform seasonal adjustment, MODLER will actually use a separate program, which one depending upon the frequency of the data: MODX11Q (quarterly), MODX11M (monthly), and MODSAWK (weekly). But while you need to be aware that these programs exist—since they must be in the PATH or in the same directory as the MODLER executable—as they execute you will be only vaguely, if at all, conscious that you are not using just another MODLER internal command, so transparent is their operation.

Typing In New Data

Analysis can also involve updating series. You can use ordinary in-line commands to enter series observations, either to create new series or to revise or extend existing ones. The in-line command method can be demonstrated by example:

CONS$,198402:2330.1,2334.5,2339.6,2348.1

If the frequency is quarterly, this particular command enters values for the variable CONS$ for the second quarter of 1984 through the first quarter of 1985, overwriting any data already in the Memory File for the same series and dates; however, outside these dates any existing values are left undisturbed. Notice that the series name is followed by a comma, a date and a colon, and that the observations follow after that, one after another, with single commas in between.

Commands providing observations on a particular series may run over several lines provided each line-to-be-continued ends with a comma to tell MODLER that there is more to come. As indicated, the observations should generally be separated by single comas. Comas used as line continuation characters must be placed at the end of an observation value; individual numbers should never be split between two lines.

MODLER also recognizes the * sign in a data entry command as indicating multiple occurances of the same value. For example:

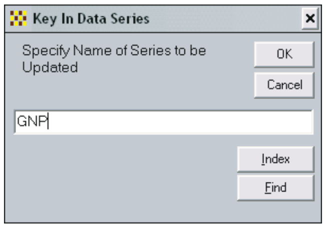
DUMMY01,194701:0*40,1*4

directs that 40 zeros should be entered, beginning in the first period of 1947, followed by 4 ones. As will be discussed later, these in-line commands can be placed within textfiles, known as command macros, where their use generally makes the most sense. Such macro files can be preserved over time, yielding an audit trail for past updates.

Series Update Command

The free-form data input command just considered, once mastered, provides a quick and efficient way to enter a few observations on a series. You can obvioiusly use it to pinpoint a particular set of observations. However, this type of command is less useful when you wish to change a number of observations for a series at quite different date points. It is generally not useful if instead or also you wish to change the documentation for a series.

Another, more set-piece way to enter or edit series observations is to click on **Data** followed by **Key In Series**, which causes the form shown in Figure 3-7, at the top of the next page, to be displayed. The Memory File series specified using this form may be either an existing or a new one.



Specify Name of Series to be Updated

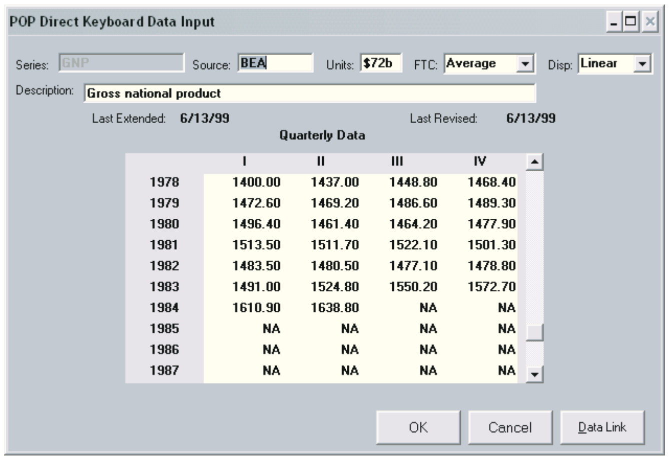
GNP

Figure 3.7 Series Update Selection Form

Once you have provided the series mnemonic and have clicked **OK** or pressed the <Enter> key a form similar to that shown in Figure 3-8 should immediately appear.

This general data entry form permits you to edit the series documentation interactively as well as to add and revise observations. An aspect of this form that you should note particularly is that you can use the text editing options (cut, copy, and paste) to capture and transfer either documentration or observations from an external machine-readable source. If, for instance, you happen to be using your Web browser in conjunction with MODLER and wish to transfer an observation from a Web page to a field on this form, all that you need do is to place the point of your mouse cursor on the value you wish to capture, doubleclick, then right-click to cause a floating edit menu to appear, and next choose **Copy**. Then, place your mouse point at the location you wish to put this value, right-click your mouse once again, and then select **Paste**.

Alternatively, this form can instead be used to copy observations on a specific series from a spreadsheet to MODLER; in this case, you will not need to type in the numbers. Click the button labelled **Data Link** and when the data link form appears, click on its **Help** button for full details.



Series: GNP Source: BEA Units: $72b FTC: Average Disp: Linear

Description: Gross national product

Last Extended: 6/13/99 Last Revised: 6/13/99

Quarterly Data

	I	II	III	IV
1978	1400.00	1437.00	1448.80	1468.40
1979	1472.60	1469.20	1486.60	1489.30
1980	1496.40	1461.40	1464.20	1477.90
1981	1513.50	1511.70	1522.10	1501.30
1982	1483.50	1480.50	1477.10	1478.80
1983	1491.00	1524.80	1550.20	1572.70
1984	1610.90	1638.80	NA	NA
1985	NA	NA	NA	NA
1986	NA	NA	NA	NA
1987	NA	NA	NA	NA

Figure 3-8. Series Update Form

Data Entry Methods Compared

Which method of data input it is best to use in a particular case depends upon a number of considerations, including the number of observations and the number of series. As indicated earlier, for a single series or if you would like to change series documentation, the onscreen editor shown as Figure 3-8 offers a good choice. Notice that, for a given series, all its relevant documentation, as well as the values of any observations, can be changed using this screen.

Alternatively, if you wish to enter a few observations to each of several series, a brute force choice is to use the command-oriented free-form data entry format described earlier, having created an ASCII text file to contain the set of commands. In this case, one option is to use MODLER's internal text editor to enter the values and then use the **Run** command to execute this macro. The text editor screen can be displayed from the Central Control screen by choosing **View | Macro.** This editing screen is shown in Figure 3-9.

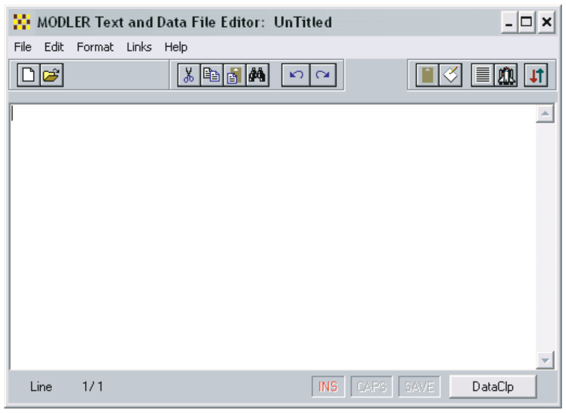


Figure 3-9. Internal Text and Data File Editor screen

When you have created and saved the macro and then closed the text editor, you will be looking directly at the Central Control screen. Click on its command space, and then type the Run command there. For example:

RUN NEWDATA

where NEWDATA.MAC contains the data *and is located in the macro files directory*. Among its virtues, this approach allows later recovery from any data file creation errors, since the data will be in a macro saved on your hard disk. The macro can be edited and re-run any number of times. Note, however, that in all cases the line length of the commands in your macro must be a *maximum of 80 characters*; a macro file simply simulates keyboard input.

In addition, if subsequently you will be revising data series, or adding observations at either end of a series, it is possible to text edit your file, and then simply re-execute the original RUN command. Alternatively, you can create a new text file, containing only the revised and new observations. The aim at the moment is simply to introduce possibilities; over time you will discover which methods work best for you in each context.

MODLER Import Facilities

But what if the data are already in machine-readable form? You may have obtained a data file that is not in the MODLER data entry format or you may wish to download data from the Internet or some other online source. Instead of a single series or a handfull of series, many, even hundreds of data series, may need to be imported.

There are a variety of import facilities supported by MODLER, and these are best distinguished by file format. One possibility is that the observations are contained in some type of standard numeric data file. Historically, such files tended to take the form of (ASCII) text files in one of several formats, TSD, CSV, and PRN being the most widely used. Alternatively, it has become increasingly common today for data to be distributed in the form of a spreadsheet file, a so-called “worksheet” file coded in the proprietary (binary) format of Excel or other spreadsheet package. Alternatively, the data may not be contained in a recognizable data file format *per se*, but instead take the form of numbers embedded in a machine-readable news release or other essentially textual report. These numbers may appear in the body of the release or report, or they may appear in separate, integrated or appended tables.

The several possibilities just mentioned generally correspond to the options displayed in the lower third of the **Data** drop down menu list shown in Figure 1-3 on page 4 of this user guide: **Convert Local Data File**, **Import Spreadsheet Data**, **Open Internet Connection, Update from Textfile using DataClp**. The option **Open Internet Connection** also permits the downloading of a standard numeric data file and its conversion once the file is resident on your hard disk.

One aspect of the data importation process that is common to all these options is the requirement that any set of observations imported into MODLER must *always* be linked to one or more specific mnemonic series names. In addition, MODLER data series must each be given (or have already) a particular observation frequency. If series names, obeying the MODLER naming conventions, are not included with the data, you must supply them. Similarly, you will need to specify the observation frequency if this is not otherwise specified.

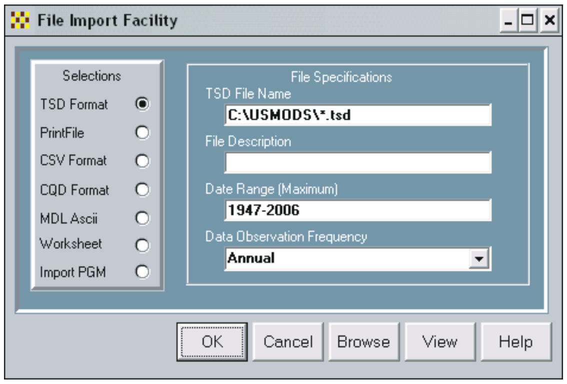
MODLER data series also have a date range of available observations. For existing MODLER series, the availability of observations is automatically adjusted as you load new observations or revise old ones. However, in order for MODLER to be able to keep track, either the imported data must have dates attached, or you must specify the dates corresponding to these observations. When you are importing observations, you must keep in mind the information that must be supplied in order for MODLER to make sense of the process.

Certain types of import files, most notably TSD formatted files, contain fully documented data, specifying series names, observation frequencies, dates of availability, and other pertinent information. Economic data vendors, such as Global Insight, Economy.Com and Haver Analytics will supply data in the TSD format. However, in many cases, data files that you obtain will consist simply of the observations, without any supporting documentation.

Convert Local Data File

If you click on **Data** and then **Convert Local Data File**, you should immediately see the form shown below in Figure 3-10 . If alternatively, you have chosen to **Open Internet Connection**, once the connection has been made, you should also see, next to your browser screen, this same form. In either case, click the **Help** button for specific details, inasmuch as the procedure to follow depends for its specifics on the type of file to be converted. The context-specific help facilities provided by MODLER for data imports are both focused on the specific task and intentionally quite detailed. In contrast, the present user guide aims only to introduce the topic.

In general, the form shown in Figure 3-10 permits you to identify the type of file you wish to convert and then to select the individual data file by name. The file types include TSD, Printfile (PRN), Comma Separated (CSV), Comma and Quotes delimted (CQD), MODLER data entry format (MDL), Lotus worksheet (WKx), and output from the MODLER general file import program, the latter permitting you to specify the particular format of a wide variety of ASCII file formats. In each case, you will also obviously need to select the observation frequency and the appropriate date range. Notice in particular that these settings control the actual data import and



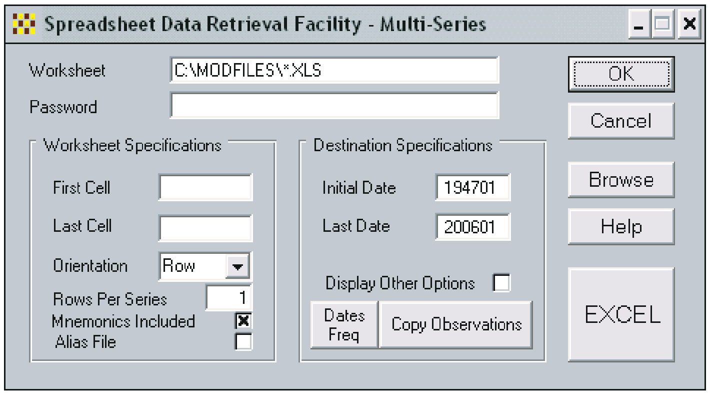
TSD File Name: C:\USMODS\*.tsd
File Description:
Date Range (Maximum): 1947-2006
Data Observation Frequency: Annual

Figure 3-10. File Conversion Facility

may restrict both the number and type of observations that are imported—this restriction feature is actually beneficial: because of it you are able to import a subset of the available observations, if you wish.

Import Spreadsheet Data

If you click on **Data** and then **Import Spreadsheet Data**, you should instead see the form shown in Figure 3-11. Click its **Help** button for specific instructions as to how to import data from a spreadsheet file. However, briefly, this form obviously is designed to manage the transfer of observations from a spreadsheet to the MODLER environment. The large button that in Figure 3-11 is labelled Excel is so labelled because, in this example, that particular package has been chosen as the default. In other cases, the name of another spreadsheet package will appear here. But, whichever the particular default, when this button is pressed the effect is to launch your spreadsheet program.



Worksheet C:\MODFILES\*.XLS
Password

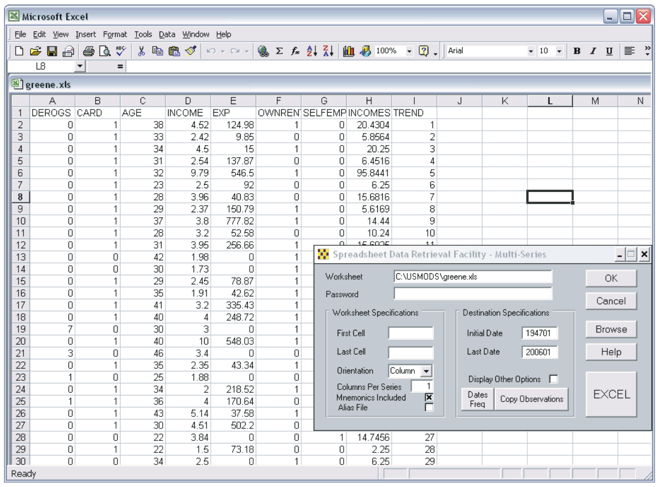
Worksheet Specifications
First Cell
Last Cell
Orientation Row
Rows Per Series 1
Mnemonics Included [✓]
Alias File [ ]

Destination Specifications
Initial Date 194701
Last Date 200601
Display Other Options [ ]

Figure 3-11. Spreadsheet Data Retrieval Facility

You will next see on your screen the spreadsheet data grid, with the MODLER form superimposed, as is illustrated by Figure 3.12 on the next page. Essentially, you are able to copy rows or columns of spreadsheet data, each row or column being interpreted as observations on a particular variable. As many as 500 rows or columns (that is, series) can be copied at a time, subject to spreadsheet limits.

The copy process itself is implicit, with no visible sign that anything has occurred, except perhaps the brief appearance of the hourglass cursor. However, a spoolfile (IMPORT.SPL) is also generated automatically, which you can then display to provide details of the observations copied for each data series; the Spoolfile icon that automatically appears on the toolbar of the Central Control screen indicates that this file has been created.



	A	B	C	D	E	F	G	H	I
1	DEROGS	CARD	AGE	INCOME	EXP	OWNRENT	SELFEMP	INCOMES	TREND
2	0	1	38	4.52	124.98	1	0	20.4304	1
3	0	1	33	2.42	9.85	0	0	5.8564	2
4	0	1	34	4.5	15	1	0	20.25	3
5	0	1	31	2.54	137.87	0	0	6.4516	4
6	0	1	32	9.79	546.5	1	0	95.8441	5
7	0	1	23	2.5	92	0	0	6.25	6
8	0	1	28	3.96	40.83	0	0	15.6816	7
9	0	1	29	2.37	150.79	1	0	5.6169	8
10	0	1	37	3.8	777.82	1	0	14.44	9
11	0	1	28	3.2	52.58	0	0	10.24	10
12	0	1	31	3.95	256.66	1	0	15.6025	11
13	0	0	42	1.98	0	1			
14	0	0	30	1.73	0	1			
15	0	1	29	2.45	78.87	1			
16	0	1	35	1.91	42.62	1			
17	0	1	41	3.2	335.43	1			
18	0	1	40	4	248.72	1			
19	7	0	30	3	0	1			
20	0	1	40	10	548.03	1			
21	3	0	46	3.4	0	0			
22	0	1	35	2.35	43.34	1			
23	1	0	25	1.88	0	0			
24	0	1	34	2	218.52	1			
25	1	1	36	4	170.64	0			
26	0	1	43	5.14	37.58	1			
27	0	1	30	4.51	502.2	0			
28	0	0	22	3.84	0	0	1	14.7456	27
29	0	1	22	1.5	73.18	0	0	2.25	28
30	0	0	34	2.5	0	1	0	6.25	29

Spreadsheet Data Retrieval Facility - Multi-Series

Worksheet: C:\USMODS\greene.xls
Password:

Worksheet Specifications
First Cell:
Last Cell:
Orientation: Column
Columns Per Series: 1
Mnemonics Included ☒
Alias File ☐

Destination Specifications
Initial Date: 194701
Last Date: 200601
Display Other Options ☐

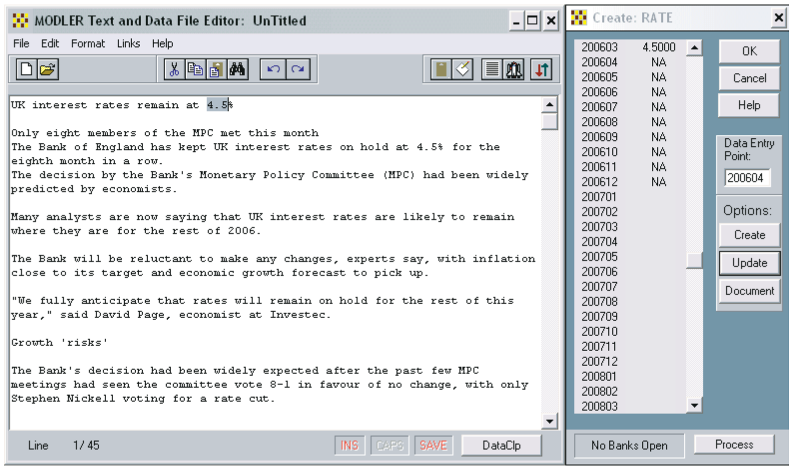
Figure 3-12. Spreadsheet Data Extraction

The purpose of this brief description is simply to make you aware of the general characteristics of the MODLER spreadsheet data import facility. Obviously, there is much more that could be said, inasmuch as there are a variety of ways in which spreadsheet data files can be organized and formatted. For this reason, an additional, specifically focused user guide, entitled *Data Import Guide - Importing Data From Excel Worksheets: Some Examples*, has been separately developed, the purpose of which is to provide a sequence of worked examples that progressively become more complex. This document can be found in the *Learning Tools* section of the www.modler.com website, as can also the Excel files that provide the examples. The document's filename is SSDNLOAD.PDF.

Capturing Data from a Textfile Using DataClp

As noted earlier, it is sometimes desirable to be able to capture observations embedded in a report or news release. If you click on **Data** and then **Update from Textfile Using DataClp**, you should see again the form shown earlier in Figure 3-9, which is MODLER's internal text editor. Notice that this screen is also described by its title as a "Data File Editor." Using cut and paste or simply by opening an ASCII file, you will be able to display either text or a textfile on the blotter of this screen. The particular significance of cut and paste (or copy and paste) is that this is often the easiest way to capture text that originally appears on an Internet browser screen.

If, once you have loaded a textfile, you then press the button labelled DataClp at the lower right of the screen a form will be appended on the right-hand-side of the text editor; this form is displayed in the context of Figure 3-13 shown on the next page.



```
UK interest rates remain at 4.5%

Only eight members of the MPC met this month
The Bank of England has kept UK interest rates on hold at 4.5% for the
eighth month in a row.
The decision by the Bank's Monetary Policy Committee (MPC) had been widely
predicted by economists.

Many analysts are now saying that UK interest rates are likely to remain
where they are for the rest of 2006.

The Bank will be reluctant to make any changes, experts say, with inflation
close to its target and economic growth forecast to pick up.

"We fully anticipate that rates will remain on hold for the rest of this
year," said David Page, economist at Investec.

Growth 'risks'

The Bank's decision had been widely expected after the past few MPC
meetings had seen the committee vote 8-1 in favour of no change, with only
Stephen Nickell voting for a rate cut.
```

200603	4.5000
200604	NA
200605	NA
200606	NA
200607	NA
200608	NA
200609	NA
200610	NA
200611	NA
200612	NA
200701	
200702	
200703	
200704	
200705	
200706	
200707	
200708	
200709	
200710	
200711	
200712	
200801	
200802	
200803	

Figure 3-13. Data Clp Form Appended to Text Editor

Press the **Help** button on the DataClp form for details concerning the process of copying numbers from the text editor screen to a MODLER data series. However, basically what is involved is that you highlight the numbers—single numbers or a sequence—that you wish to copy, using your mouse in the usual way. Then specify the date that the (first) number corresponds to in the series, by filling in the text box labelled "Data Entry Point" and then pressing the <Enter> key. The number or numbers will then be immediately included in the series' observations.

You may update existing series (note the **Update** button on the form) or create a new series (note the **Create** button). You can also change the series' documentation (note the **Document** button). When you have included all the observations you wish, press the **OK** button to store the new or modified series in the Memory File—or data bank, if you have opened one for update, as will be described later in this chapter.

Displaying Groups of Series

Series once created or revised can be displayed. For instance, the TAB SERIES command allows you to produce quickly a very basic table that displays series in columns. One option allows you to select the series to be displayed by declaring the initial letters of their names. For example, if the series with names beginning with the letters D or E are to be retrieved from a Data Bank, use a command of the form:

TABULATE SERIES=D-E* **FROM** DEMOBANK

where the Data Bank name (here DEMOBANK) appears in the command; the Data

Bank must already be open for access for this command to execute. If, in contrast, the series are to be retrieved from the Memory File, use a command of the form:

TABULATE SERIES=D-E*

where “**FROM** MEMORYFILE” will be implicitly presumed by MODLER.

Sometimes, you may wish to collect and display series from one or more Data Banks—or selected series from a particular bank that do not group alphabetically. In this case, one option is to use the Memory File as a collection point. The following sequence of commands illustrates the process, although in practice you may actually wish to access two or more banks, since the series in a single bank can be easily displayed using a command like that above, as we just considered.

First load the Memory File:

CLEAR MEMORY FILE
ACCESS DEMOBANK
SET FREQ=Q
SET DATES=7801-7902
FETCH POP
FETCH POPWA
FETCH LABF
FETCH EMPL

using the FETCH command described earlier. Then display the series as a column table by typing:

TAB SERIES=A-Z

As before, the absence of any bank name implies that the series will be retrieved from the Memory File.

Context-Related Groups

The TAB command just described references groups that are defined by series name or location, where the names are alphabetically contiguous and the locations follow one another. Alternatively, MODLER permits groups to be defined by a **GROUP-FILE**, which is an ASCII file containing the names of series; it must have the extent **.GRP**. In this case, the TAB command will take the form:

TAB GROUP=groupname

The characteristics of a groupfile are as follows: It must be a pure ASCII file, containing no control characters, except Carriage Return Line Feed (equivalent to pressing the Enter key on your keyboard). Each line must be 80 characters or less and contain series names, separated by blanks. It can contain a maximum of 3500 series names. It must have the extent **.GRP**.

Groupfiles can also be referenced in COPY commands, EXPORT commands, and PRINT commands:

> **COPY GROUP=**MYGROUP **FROM** memoryfile **TO** bank
> **EXPORT GROUP=**MYGROUP **FROM** bank **VIA** TSD
> **PRINT GROUP=**MYGROUP

where syntactically the principal distinction is the replacement of the word **SERIES** by the word **GROUP**.

The menu analogues of these command are all found under the **Data** category of the Central Control screen menu, where you will also find two other options: **Make Group File** and **Edit Group File**. If you click on the first of these, a small form will appear similar to that shown in Figure 3-14.

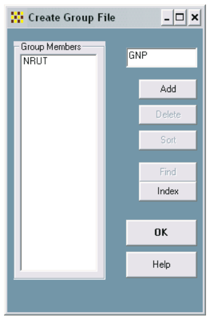


Figure 3-14. Group File Creation Form

Any series name entered in the text box in the upper right quadrant of this form can be added to the list of group members either by clicking the **Add** button or else by pressing the <Enter> key on your keyboard. Notice also the **Help** and **Index** buttons. The first of these provides information about the Groupfile creation and modification process. The second allows you to display the index of any currently accessible Memory File or data bank.

After you have finished adding all the series names you wish to the new group file, when you press the OK button on this form, you will be provided a form that will permit you to name this group file, after which the file can be used, as discussed earlier. Incidentally, the only difference between the **Make Group File** and **Edit Group File** choices is that in the second case MODLER will expect the Group File to exist previously and will therefore ask you to provide its name. Otherwise, they are functionally the same.

Building and Modifying Data Banks

Why A Data Bank

To this point, although both data banks and the Memory File have been mentioned and considered, including data retrieval operations that are either Memory File or data bank related, the focus of attention has principally been data base maintenance operations related to the Memory File. As indicated at the beginning of this chapter, a fundamental characteristic of the Memory File is that it is automatically created or made available for use as required. In a particular case it might be created intially because MODLER needs it for temporary storage. Alternatively, it might be initialized more or less explicitly by you, either in response to your Initialize command or because you issue a LoadMF command. However, once it is there, to use it for retrieval simply requires asking for a series that happens to be located in it. Furthermore, any series can be added, deleted, overwritten, renamed, or otherwise modified without notice. This easy changability is both the Memory File's strength and its weakness as a data base facility, for on the one hand it is there and available but on the other you cannot rely on its integrity: to make changes to a series, all you need to do is to provide its name to the left of the equals sign, in the context of a transformation command, and the results of that transformation will be copied to the Memory File under that series name. The changes are immediate, whether completely intentional or not. One of the very few instances of any protection for a previously created Memory File series is if you attempt to modify a series using data of a different observational frequency. In this case, a series once created must be deleted before its name can be used to store data of a different observational frequency. However, this qualification is the exception rather than the rule.

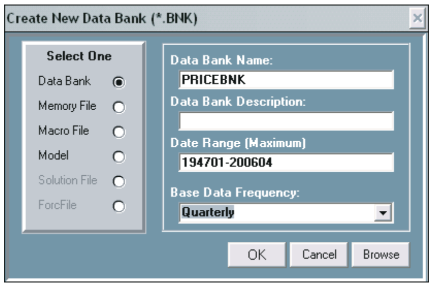
In contrast, a data bank is only opened when you specify that it should be. Even opened, when opened for *access*, its contents cannot be changed: only if you declare that it should be opened for *storage* or *update* is there any chance that any series observations might be modified. However, before considering these aspects further, the obvious first question is: Why a Data Bank? Actually, there are several advantages to storing certain data permanently in a Data Bank, rather than a Memory File. One of these is faster access to the observations, and generally a higher level of documentation (if you or someone else provides it), plus the capacity to hold a much larger number of series, a maximum of 10,000, rather than 500. However, in addition, a data bank has the characteristic that it can be interviewed even if it is not open: you can directly print the index of any bank that is located on any accessible drive; as a result, you can often find out without accessing it whether or not a given bank contains a particular data series. In contrast, a Memory File must first be loaded before its index can be displayed. A data bank's further benefit is that you will be able to update or extend the series easily, while keeping them distinct from any temporary or working data in the current Memory File. Generally, it is a good idea to put series into a bank if you want to use them repeatedly or to make them available to other people. As mentioned before, it is also possible to access simultaneously as many as 15 Data Banks, whereas only one Memory File can be loaded at a time, although it is possible to merge two or more of these. In addition, banks are designed to be used both in a single user environment and on a network, whereas a Memory File is meant to be used exclusively by one person. Of course, the defect of the Memory File—fundamentally its role as a temporary work file—is also its advantage: Simplicity. The creation and maintenance of Data Banks requires more thought and organizational care, in large part because of their archival characteristics. The presumption is that bank series are valuable and likely to be used either by more than one person or, if by only one person, over a sufficiently long period of time that it is worth going to the trouble of documenting the work.

Creating A Data Bank

But let us suppose for the sake of argument that you have given all this some thought and have decided that it is worth considering how to use data banks. In particular, suppose you are going to build a Data Bank of price indexes derived from series in DEMOBANK. Start by typing:

```
SET FREQ=Q
CREATE PRICEBNK
```

to tell MODLER to initialize the Data Bank; this bank will appear on the data bank directory as PRICEBNK.BNK. Alternatively, you can click on **File** and then **New**. In either case, you should see next a display similar to that in Figure 3-15. This display shows as filled in the entries that must be provided. The bank description, in particular, although strongly recommended inasmuch as it is prominently displayed in various contexts, is not strictly necessary and can be provided later.


Create New Data Bank (*.BNK)

Select One: [✓ Data Bank] Memory File Macro File Model Solution File ForcFile

Data Bank Name: PRICEBNK
Data Bank Description:
Date Range (Maximum): 194701-200604
Base Data Frequency: Quarterly

Figure 3-15. Data Bank Creation Form

Once you fill in the required entries and click **OK**, MODLER immediately creates the bank. From this point on, although empty of series, the bank exists, and for the time being (until you Close it or leave MODLER) it is open for updating. “Open for Updating” means that any new documentation and data will go into the bank, not the Memory File. Whenever a bank is open for updating, ordinarily it can be accessed on more or less the same terms as a Memory File, which is to say that during this time it is designated as the *primary* storage file.

Important: there are, however, certain things you must do: for one thing, before you can create a bank, you must set the observation frequency. This setting does not limit your ability to put series of different frequencies in the same bank (provided they have different names), but it will establish the maximum number of observations on any series in the bank. For example, the period from 1950 to 2049 defines a hundred observations of annual data, 400 observations if the frequency is quarterly, or 1200 observations if monthly. Thus if the base observation frequency is annual, for example, you will only be able to include just over 8 years of data on any monthly series you might also choose to include (100/12=8.33). This obviously could be a limitation if you are intending to use the bank mainly for monthly frequency data. In addition, you should also bear in mind that although MODLER Data Banks are designed to store a maximum of 1600 observations on each series, a bank that hits this limit will tend to be large, particularly if it contains more than a few series. Consequently, you should be careful when setting the date limits: ideally, you should use only the space you actually need; another new bank can be created easily, and existing series copied to it, whenever you need to store more observations.

Data Entry Options

The next few subsections of this chapter describe methods of data entry for series. However, these descriptions should be regarded as introductory, rather than exhaustive. There are other options available: these are described in the relevant sections of the online help facility, which also contain more detailed information about the methods described below. Since MODLER’s context help facilities are available at the push of a button while you work, they are worth using, rather than referring back regularly to this document. You can also print sections of the help file.

You will notice also that commands are given prominence in this section. Initially you might prefer to use the menu options, inasmuch as these do not require you to know the commands in advance. However, once you begin to work regularly with MODLER, you are likely to find that commands provide the capability to form macro files that can be executed periodically to perform data bank maintenance chores, rather than using the (interactive) menu commands that must be issued individually each time you perform the operations.

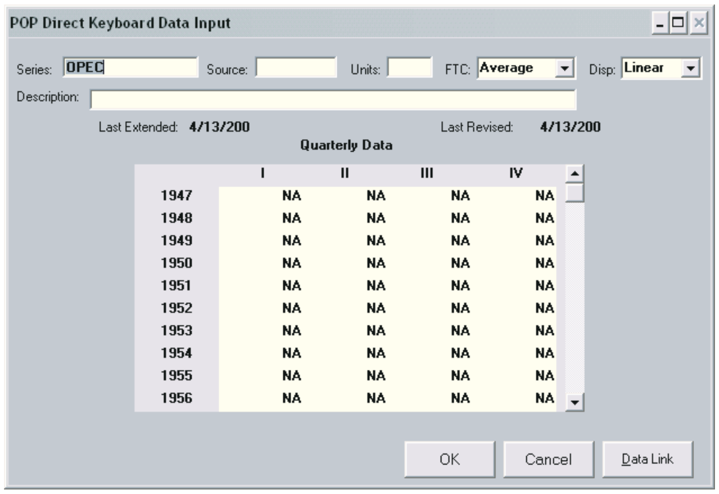
The POP Facility

The easiest way to load single series into a bank from the keyboard is to use the POP on-screen data entry facility, introduced earlier in this chapter. To create a new series, from the Central Control screen either click **Data | Key In Series** or simply type **POP** and the seriesname you wish to use:

POP seriesname

in the process providing a mnemonic name for the series. MODLER will then display the series as shown in Figure 3-16, providing space for both documentation on the series and observations. This example uses the name OPEC. If the series already exists, the available documentation and observations will be displayed, ready for you to make onscreen changes as appropriate. Otherwise the documentation fields will be blank.

The prior existence of the series needs a little consideration, for there is always a question whether this condition refers only to the bank in question, or to all data banks and any Memory File that might be open. Remember that MODLER searches the Memory File and any data banks that are open for series with the name you have given, thus even if the given bank has no series with a particular name, such a series could be found elsewhere. You can limit this search only to the bank currently open for update by going to the Central Control screen (MODLER's opening screen) and then choosing **Settings|Operating Protocols**, and finally the element **Strict Update Rule for Data Series**. If the Strict Update Rule is in force (this element is checked), only when a series exists in the current Update bank will it be treated as an existing series; otherwise, if an existing series with the same name cannot be found in this bank, MODLER will treat the series as a new series.



Series: OPEC Source: Units: FTC: Average Disp: Linear

Description:

Last Extended: 4/13/200 Last Revised: 4/13/200

Quarterly Data

	I	II	III	IV
1947	NA	NA	NA	NA
1948	NA	NA	NA	NA
1949	NA	NA	NA	NA
1950	NA	NA	NA	NA
1951	NA	NA	NA	NA
1952	NA	NA	NA	NA
1953	NA	NA	NA	NA
1954	NA	NA	NA	NA
1955	NA	NA	NA	NA
1956	NA	NA	NA	NA

Figure 3-16. Series Data Entry Form

The Macro Entry Facility

Alternatively, you can use the Macro Entry facility, also introduced earlier, so called because it allows you to organize the data entry as a macro containing observations on all the series you wish to include. For each series, the format is:

seriesname,date:obs1,obs2,....,obsn

where the seriesname is the mnemonic, the date is the date of the first observation, and the observations are typed one after another in the order in which they should be included. MODLER will determine the dates associated with observations after the first. Note that a colon separates the date from the first observation. The observations can be separated by blanks or commas, but there should be no blanks or other non-numeric characters within a series name, date, or observation.

For each series, you must start on a new command line. Each command line must be 80 characters or less and if you wish to continue the observations from one command line to another, the last observation on the line to be continued must be immediately followed by a comma. A command line can begin in the first column or thereafter. A command line example may be helpful:

```
OPEC,7001: 1.4 1.4 1.4 1.4 1.6 1.8 1.8 1.8,
1.8 1.8 1.9 2.0 2.1 2.4 2.8 3.5,
9.2 9.6 9.8 10.4 10.5 10.5 10.5 11.5,
11.5 11.5 11.5 11.5 12.1 12.1 12.7 12.7,
12.7 12.7 12.7 12.7 13.9 17.0 20.2 23.5
OPEC1,7001: 1.2 1.3 1.4 1.5 1.6 1.8 1.9 1.8,
1.8 1.8 1.9 2.0 2.1 2.4 2.7 3.2
```

Notice that here observations on two series are involved and that each of the entry commands are multi-line commands. such that commas appear at the end of each line that the command continues to the next line.

These commands can be typed in interactively, but usually it will be more convenient (and safer) to embed them in a macro file. If you wish to work interactively, you may in the end find it much more convenient to use the form shown above in Figure 3-16. Using this form, you can both add observations and the documentation for a series. As indicated before, you can access that form from the Central Control screen **Data** menu item: **Data|Key in Series**.

However, if you want to use commands, either interactively or in the context of a macro file, the command method of data entry also allows you to include series documentation, which can be done in either of two ways. If you wish to document a new series before you add the observations, the process is known as *defining a series* and takes the following form:

DEFINE SERIES: seriesname
DESC: series description
SOURCE=code,UNITS=code,FTC=code,DIS=code,FREQ=code

The series name should appear on the first line, after **DEFINE SERIES**. The description (**DESC:**) can be up to 72 characters, but codes for **SOURCE** and **UNITS** must be kept short (up to 8 characters for the source and 4 for the units). The **FTC** is the Frequency Transformation Criterion used for automatic frequency conversion and **DIS** is the conversion criterion for lower to higher frequency. **FREQ** is the series frequency, and can be established by default—as the same as that currently set. For example, consider:

DEFINE SERIES: CPI
DESC: CPI, urban households, 1967=100
SOURCE=FRB, UNITS=indx,FTC=avg,DIS=line,FREQ=quar

If, alternatively, you wish to include the observations before you document the series, then the procedure is known as *documenting a series* and takes the following form:

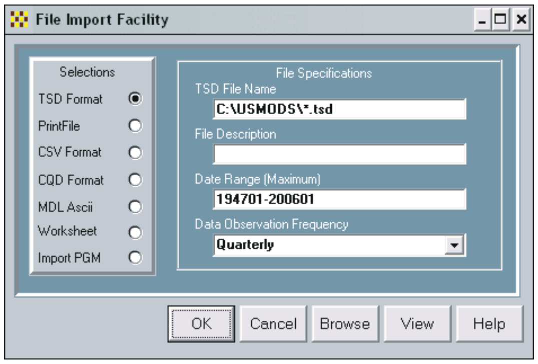
DOCUMENT: seriesname
DESC: series description
SOURCE=code, UNITS=code,FTC=code,DIS=code,FREQ=code

Procedurally, the only difference is the use of the word **DOCUMENT** instead of **DEFINE SERIES**. The **DEFINE SERIES** command must not be used two or more times for a given series; once defined a series cannot be redefined. In contrast, there is no limit on the number of times a series can be documented—or, as it happens, redocumented.

The **DEFINE SERIES** and **DOCUMENT** commands can be included in the same macro files as the data entry commands, provided that any DEFINE SERIES commands come first. Alternatively, they can be put into one or more separate macros; there is no need to mix them.

The File Import facility

One of MODLER's most useful data entry facilities takes any of several forms. One method is to click on **Data** and then **Convert Local Data File** from the Central Control screen menu, which causes the form shown in Figure 3-17 to be displayed:


File Import Facility

Selections: TSD Format [✓], PrintFile, CSV Format, CQD Format, MDL Ascii, Worksheet, Import PGM

File Specifications

TSD File Name: C:\USMODS\*.tsd

File Description:

Date Range (Maximum): 194701-200601

Data Observation Frequency: Quarterly

Figure 3-17. Local Data File File Format Conversion

Context sensitive help is available by clicking on the Help button, which then provides details on the particular conversion process for each type of file format that can be converted. Alternatively, from the Central Control screen's yellow blotter you can enter a command of the form:

IMPORT programname

or

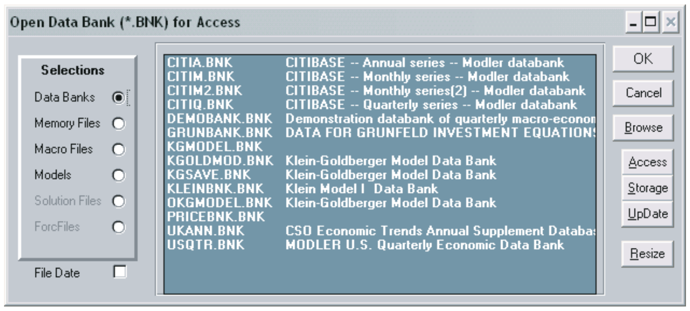
IMPORT programname filename

depending upon the data source. The alternative forms of the command cause the execution of a variety of data conversion programs, including programs to import data from LOTUS worksheet files, TSD format files, LOTUS Printfiles, mainframe downloading routines and many others. Additional programs can be written relatively easily, so that the possibilities are open-ended. Prototype conversion routines are available at the www.modler.com website. Creating such programs makes sense if you know that you will be receiving regularly files of a particular type.

The specific characteristics of the data import procedure varies from data source to data source. However, all have the characteristic that once the file format is accounted for, the IMPORT command will cause data series observations to be imported and stored in a Data Bank (or a MEMORY FILE, as appropriate). In some cases, series documentation will be imported with the data; in others it will not and you will need to document the series separately. For specific details, consult your data base vendor.

Copying Series from an Existing Data Bank or Memory file

Another way to include data series in a new Data Bank is to copy from an existing



Selections

Data Banks [✓]
Memory Files
Macro Files
Models
Solution Files
ForcFiles

File Date

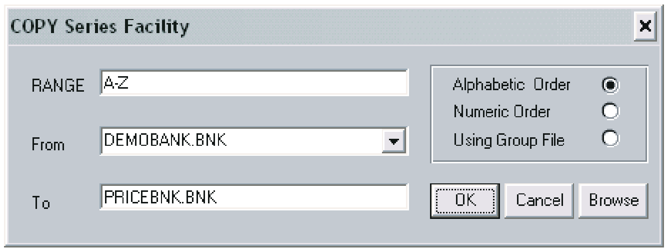
CITIA.BNK	CITIBASE -- Annual series -- Modler databank
CITIM.BNK	CITIBASE -- Monthly series -- Modler databank
CITIM2.BNK	CITIBASE -- Monthly series(2) -- Modler databank
CITIQ.BNK	CITIBASE -- Quarterly series -- Modler databank
DEMOBANK.BNK	Demonstration databank of quarterly macro-econor
GRUNBANK.BNK	DATA FOR GRUNFELD INVESTMENT EQUATION!
KGMODEL.BNK	
KGOLDMOD.BNK	Klein-Goldberger Model Data Bank
KGSAVE.BNK	Klein-Goldberger Model Data Bank
KLEINBNK.BNK	Klein Model I Data Bank
OKGMODEL.BNK	Klein-Goldberger Model Data Bank
PRICEBNK.BNK	
UKANN.BNK	CSO Economic Trends Annual Supplement Databa:
USQTR.BNK	MODLER U.S. Quarterly Economic Data Bank

Figure 3-18. Open File Form

Data Bank or Memory File. If you wish to use menu commands, begin at the Central Control screen by clicking on **File** and then **Open**. You should then see a form similar to that shown just above in Figure 3-18.

Notice that the radio button next to *Data Banks* is clicked. Notice also the three buttons together on the right of the form that are labeled respectively **Access**, **Storage** and **Update**. When you open the existing bank, you should choose **Access** (the default), since this is the bank *from which* the series might be copied. The bank *to which* the coping will be done is the **Storage** bank. In fact, you will need to execute the File and Open commands twice, the first to open the access bank and the second the storage bank, if you will be copying from one bank to another. If you are copying from an existing (open) Memory File, then you will only open the Storage bank. It is also a very good idea, before you open the bank or banks directly affected to close all other banks in advance, so as not to accidentally copy from or to the wrong bank.

The menu form of the copy command itself is found under the **Data** menu item **Copy Series**. When you have clicked on these in turn, you should see the form shown in Figure 3-19. Notice that you can choose to copy by the alphabetic ordering of the names of series, by numeric order, or using a group file, that can be created using the procedures described earlier on pages 90-91. The available *From* banks or Memory File will be listed in the drop down text box on the form. The *To* bank can only be a bank opened as a **Storage** bank. Once you have made your choices, simply press the OK button and the series will be copied.



RANGE A-Z

From DEMOBANK.BNK

To PRICEBNK.BNK

[✓ Alphabetic Order] Numeric Order Using Group File

Figure 3-19. Copy Series Form

Alternatively you can use typed commands. The basic COPY command takes the form:

COPY SERIES=x-y **FROM** file1 **TO** databank

where the variations parallel those shown in Figure 3-19. Here x-y denotes a range of series; for example A-ZZZ, where this choice indicates that series with names beginning with the letter A through ZZZ should be copied. The range x-y can be defined in terms of either series names or series numbers (you may know there are 233 series to be copied: x-y might be 1-233). The designation file1 denotes either the name of the Data Bank from which the series should be copied or the word MEMORYFILE, if this is the source. The destination (copy-to) data bank should be the name of the new Data Bank or, more generally, the name of the destination data storage file, data bank or Memory File. For example, consider:

COPY SERIES=A-ZZZ FROM DEMOBANK TO PRICEBNK

The first time a series is copied, all the existing documentation on that series will automatically be copied also. Subsequently, only the observations will be copied. In either case, which observations are copied depends on the last frequency and data range commands executed; be sure to set the frequency and date range just before you copy series to make sure the right observations are copied, either using the **SET DATES** and **SET FREQUENCY** commands or by clicking on the status bar items for frequency and dates.

If you have been following closely, you might have noticed that in this discussion of copying to a bank, the copy to bank has been referred to as the *Storage* bank in contrast to the use of the term *Update* bank when we considered creating new series individually. This is actually a very important point.

The copy command—in both in-line command and menu form—permits series to be copied to a bank from some external source. If a series already exists in the bank, it will be overwritten for the date range set at the time the copy command is executed, and it does not much matter whether observations previously existed for that time period. In contrast, when single series are created (or later, when they might be updated), any existing observations will be displayed, if you use the entry method exemplified in Figure 3-16 above. Consequently, in this case, you will need to change only those observations that have changed. Notice that in this case, what will have occurred is that existing observations are displayed, then changed, so that a bank in a very real sense is being *updated*. Conceptually, this occurs even if previously the series did not exist, or if the particular observations on an existing series did not exist. The specification of the bank being modified as being an *Update* bank tells MODLER to modify what is there, if anything. In contrast, the copy command is a multi-series command, in general, and is uni-directional: we are storing observations in the *copy to* bank irrespective of what is there before the copy command is executed.

Actually, a bank that is open for Update can also be used as a storage bank, in the sense of the copy command. However, in this case, MODLER does more work behind the scenes inasmuch as before each copy, MODLER will go and look at the Update bank to see if the series copied already exists. Nothing really happens otherwise, but this particular action is wasted, thus slowing down the copy process if the copy-to bank is opened as an Update bank rather than a Storage bank.

There is, however, one circumstance that such fine distinctions do matter. If a bank has had its index corrupted, but still contains some or all its data series, it is possible, using a copy command of the form:

COPY SERIES=1-k **FROM** thatbank **TO** NewBank

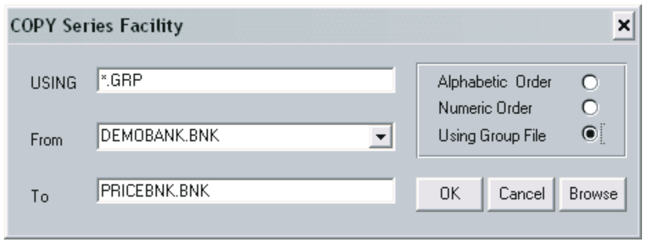
to at least partially recrete the corrupted bank. In this case, the value k should be set at least equal to the maximum number of series in the bank—you can set it to 10000, if you wish. The name *thatbank* should be the name of the corrupted bank which MUST be opened as an *Access* bank. The name *NewBank* should be the name of a new, replacement bank and this bank MUST be opened as a *Storage* bank.

When this copy command is executed, MODLER will begin copying from the first record of the corrupted bank and to the degree possible will copy each series to the replacement bank, in the process creating a new index for the replacement bank.

In this case, we emphatically do not want the corrupted bank's index to be used at all, and for this reason we want to have as much separation as possible between the copy-from and copy-to banks. Because the copy series command is numeric in style, in this case MODLER does not need to reference the index of the copy-from bank, and (originally in the interest of speed) will not. If, instead, you were to open the NewBank as an Update bank, MODLER will attempt to use both banks indices in the copying process, and might therefore end up corrupting the index of the copy-to bank as well, since MODLER would then be using the corrupted bank's index in a (fruitless) effort to determine if each copied series already exists.

Copying Groups of Series to a Bank

You can also copy groups, having selected *Using Group File* on the form shown in Figure 3-20, in which case the form will look similar to that shown in Figure 3-19. Notice that you must provide the name of the group file that you will be using. This command of course presumes the existence of at least one group file; such files can be created using the facilities described earlier on pages 90-91.



COPY Series Facility

USING *.GRP

From DEMOBANK.BNK

To PRICEBNK.BNK

Alphabetic Order ○
Numeric Order ○
Using Group File ●

Figure 3-20. Copy Series by Group Form

Alternatively, if you prefer to type the command, then this is done using the syntax:

COPY GROUP=groupname **FROM** File1 **TO** PRICEBNK

where File1 is either the word MEMORY FILE or the name of an accessed bank. PRICEBNK used here reflects the example above; more generally you obviously should use the name of the bank to which you wish to copy the series. Groups can of course contain series that have some affinity (as defined by you) apart from the alphabetic ordering of their names.

Updating the Data Bank

The description of the COPY SERIES command just above specifically refers to a new Data Bank, since the discussion began with the idea of a new data bank. However, copying series is a common procedure when using existing Data Banks as well. The general rules are easily stated: If you want to fetch data from a Data Bank (read-only), use **ACCESS** to open the bank. If you want to move data into a Data Bank (write-only), open the bank with the **STORAGE** command. To perform these operations, you can use the forms shown earlier in Figures 3-18 through 3-20.

Alternatively, you can use commands. For example, to copy all series beginning withWPfrom DEMOBANK into the PRICEBNK bank you could type:

```
ACC DEMOBANK
STORAGE PRICEBNK
COPY SER=WP* FROM DEMOBANK TO PRICEBNK
```

Notice the use of the wildcard * operator in WP*. It is also possible to copy only certain series. For example, to copy individual series from the Memory File, type:

```
CLOSE ALL BANKS
STORAGE PRICEBNK
STORE: CPI
CLOSE PRICEBNK
```

Of these commands, the **STORE:** command both causes CPI to be retrieved from the Memory File and then to be stored in PRICEBNK, the *Storage* bank. In this case, you can be sure that the data values come from the Memory File since the STORAGE and STORE commands are preceded by a CLOSE ALL BANKS command, thus insuring that no *Access* bank will be open. Otherwise, MODLER will copy series from wherever it first finds them, be it another Data Bank or the Memory File, storing them in whatever bank is designated as the *Storage* bank.

To open a Data Bank for retrieval and storage simultaneously (read-write), you use the UPDATE command. You will need this command if you want to transform data already contained in the Data Bank and if you do not want to use the STORE command; which after all involves two steps. For example, to rebase a price index already in the PRICEBNK, you could use the commands:

```
CLEAR MEMORYFILE
UPDATE PRICEBNK
CPI=CPI/100
CLOSE PRICEBNK
```

The essence of the concept of *updating* a Data Bank is that data must be retrieved, revised or transformed, and then re-stored. Only the update command allows selected observations on a particular series to be changed without changing others within the update time interval.

Some Comments on Series Descriptions

The manner in which series are described is important both in terms of the use of the FIND command and for printouts of Data Bank indices. You should experiment with the FIND command as you build the bank to make sure that series are easy to find by concept. Similarly, it may be useful from time to time to produce a hard copy of the fully documented Data Bank index, just to see how the descriptions scan.

Supplementary Programs for Database Management

At the website, www.modler.com, on the Learning Tools page, you will find a useful document *Supplementary Programs for Data Base Management and Related Uses*. The pdf filename is DATAMAN.PDF. This document describes several programs that you can use to validate data banks, as well as to assist you in their development. You should immediately download and read this document, and you should certainly read it if you encounter any problems creating or using MODLER data banks. The corresponding data management programs can be downloaded from this website in the context of the self-extracting file DATAMAN.EXE.