

Chapter O: The Macro Module

	<u>Page</u>
O-1. Introduction	O-1
O-2. Creating or Editing a Macro: Initial Considerations	O-2
O-3. Basic Procedures	O-3
O-3.1 Creating a New Macro	O-3
O-3.2. Opening an Existing Macro.....	O-3
O-3.3. The Macro Editor Window.....	O-3
O-3.4. The Macro Item Window	O-4
O-3.5. Adding an Item: An Example	O-5
O-3.6. Adding Items in the Middle of an Existing Macro.....	O-8
O-3.7. Deleting an Item.....	O-8
O-3.8. Completing a Macro.....	O-8
O-4. Error Checking	O-9
O-5. Playing a Macro	O-11
Fatal Errors	O-12
Text Messages.....	O-12
Suspend Commands	O-13
Playing to Completion	O-13
O-6. Designing a Macro to Acquire Data	O-14
O-7. Designing Macros for Repeated Playback.....	O-15
O-8. Designing Macros to Acquire Many Data Files.....	O-17
O-9. Designing Macros to Analyze Many Data Files	O-19
O-10. Using Macros to Produce Reports.....	O-20
O-10.1. Designing a Macro to Export Tables and Figures from Datapac 2K2	O-21
O-10.2. Creating the Template Document for a Report.....	O-23
Creating a New Template Document.....	O-24
Creating a Link to a Figure (Graphics File).....	O-25
Creating a Link to a Table (Text File)	O-26
Formatting Linked Text Files.....	O-27
Saving the Document Template.....	O-28
O-10.3. Producing Reports	O-28

Datapac 2K2 User's Manual, Ver 3

Chapter O: The Macro Module

Document Release Date: 12/19/2001

O-1. Introduction


The Datapac 2K2 Macro Module provides the opportunity to create, edit, and run macros that automate Datapac 2K2 operation to any desired degree. You can use a macro to completely automate the entire sequence of data acquisition, processing, and analysis tasks, including exporting the results. Or you can suspend operation at particular locations to allow manual user intervention when necessary. A macro can also be repeated automatically any number of times, thus making it possible to acquire and/or analyze many data files in one operation. Macros that are intended for repetitive use can be constructed so that different groups of activities are performed on the first repetition only, on subsequent repetitions only, or on all repetitions.

Thanks to the module's context-specific database engine, creating macros is extremely easy -- even for beginners. There are no cryptic commands to learn and no complex scripts to write. In fact, you hardly have to type anything at all! All you do is pick each activity you want the macro to perform out of a list provided for that purpose. The database engine generates the list in a context specific manner, meaning that it always contains only those options that are consistent with the activity immediately preceding it in the macro sequence. This context specificity not only helps you keep track of where you are as you create your macro, it also makes it nearly impossible to commit a fatal error.

Editing a macro is just as easy. To insert a new activity, just highlight the item at the location in the macro where you want the new activity to go, click a button (or double-click the item itself) and pick the desired item from the list. Likewise, to delete an activity, just highlight it and click another button. You can add and delete items anywhere in the macro at any time. However, doing so sometimes leads to logical inconsistencies in the sequence of activities. But even they are easy to identify with the built-in error

checking feature. The error checking feature will not only tell you whether or not a logical error exists, it will also identify its location for you.



To play a macro just click on the  (Load Macro) button in the Datapac 2K2 main window's tool bar, select the macro you wish to play and the number of times you want to repeat it, and you are off. The macro will play automatically from beginning to end unless you have either instructed the macro to produce a message window or to suspend its operation to allow the user to perform a particular task manually. Programming message windows into a macro is a great way to keep the user informed, especially when he or she is expected to do something before the macro resumes.

Macros can be used to export graphics and table displays as well. Exported text and graphics displays can be easily linked to template documents prepared in MS Word or other popular word processing applications. The result is a report that is easy to generate, yet formatted exactly the way you want it, containing exactly the information you want in it, each and every time.

O-2. Creating or Editing a Macro: Initial Considerations

Creating a macro is the process of producing a list of the activities you want the macro to perform. But unlike the normal, manual mode of operation within DATAPAC, creating macros requires you to think in more structured ways. First of all, when you are creating a macro you can only begin it from a clean desktop. In other words, you must keep in mind that *when a macro is played it will always begin under the assumption that the Datapac 2K2 main window, which is also called the desktop – and only the main window – is open*. One cannot, for example, create a macro to merely print a series of displays without also including the activities required to access each of those displays from the main window.

Second, keep in mind that *a macro can perform only one activity at a time, and it must perform each activity in a logical, step-wise sequence*. When operating Datapac 2K2 manually you are permitted to skip around from one module to another, and one display to another, with a great deal of freedom. For example, you can move from an attribute table display within the Scientific Spreadsheet module to a results display within the Power Spectrum Analysis module in one mouse click. You are not permitted that kind of freedom when you are writing a macro. Rather, you need to follow a fairly rigid, step-wise sequence of related activities. For example, after opening an attribute table display, you must first exit that display before you can exit the Scientific Spreadsheet module. And you need to exit the Scientific Spreadsheet module before you can enter the Power Spectrum Analysis module, and so on.

Third, it is important to recognize that *it is rarely possible to get down to the level of individual keystrokes or individual parameter values within a macro*. For example, one cannot change the Y-Scale gain value of a display within a macro. One can, however, use a macro to load a display parameter file with the desired gain value contained in it. Parameter files are therefore very important when using macros because they represent the only way to change most parameter values automatically. Parameter files are also important for another reason – they serve to establish the initial parameters of a task with certainty. Recall that DATAPAC always defaults to the parameters that were in use the last time a particular task or activity was closed. This is also true when you are using macros. Consequently, if you or someone else used DATAPAC for some other purpose between operations of the same macro, the conditions it encounters from one operation to the next may not be the same if you rely on the default parameters. This uncertainty can be eliminated with the use of parameter files. Expect to use many parameter files when you are creating macros.

Finally, keep in mind that *it is not possible to perform any other activities within Datapac 2K2 while you are creating or editing a macro* – at least not within the same application window. You can, however, run Datapac 2K2 in more than one application window at the same time. In that way you can use one copy to write the macro and the other copy to follow along, performing the steps you are adding to the macro as you proceed. This strategy helps you to keep track of where you are as you go along, reducing the necessity of mapping out a detailed strategy ahead of time. We highly recommend it.

O-3. Basic Procedures

O-3.1 Creating a New Macro

To create a new macro, select **Macros|Create Macro** from the main window menu bar. The window shown in Figure O-2 will then appear on the screen. As far as opening and saving macros is concerned, Datapac 2K2 treats macros just like parameter files. Consequently, the window that appears is the same **Save Parameter File window** that appears in many other contexts within Datapac 2K2. Enter the name you wish to use for the macro in the **File Name** box. Don't enter an extension -- the extension, .MAC is automatically appended. After entering the file name, click the **OK** button to exit the Save Parameter window. You are then presented with the Macro Editor Window and you are ready to begin constructing the macro. Details are provided in Section O-3.3.

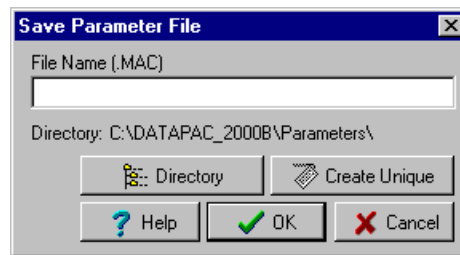



Figure O-2. The standard Save Parameter File window appears as you enter the macro creation feature. Macros are considered parameter files with the .MAC extension.

O-3.2. Opening an Existing Macro

To open an existing macro for editing, first select **Macros|Edit Macro** from the main window (desktop) menu bar. A file directory window then appears, allowing you to highlight the name of the macro file you wish to edit. Click **OK** button to proceed. The Macro Editor window then appears and you are ready to begin editing the macro. Details are provided in Section O-3.3.

O-3.3. The Macro Editor Window

An example of the the **Macro Editor** window, as it appears when a new macro is about to be created, is shown in Figure O-3. Note that the macro editor window contains a **Title** box at the top where you can enter a line of text to describe the macro you are about to create. You can then use the title as a long file name when you retrieve the macro to play it. Below the title box is the activities list – that is, the list of the activities the macro will perform on playback. *All macros must begin at the level of the Datapac 2K2 main window, which is also known as the “desktop”.* Consequently, the first line of every macro indicates that location.

To add new activities to the list, either highlight the item that you want to immediately precede the activity you are about to add and click the  (Add Item) button, or simply double-click the item in the Macro Editor window. Doing so opens another window referred to as the **Macro Item window**, described in Section O-3.4.

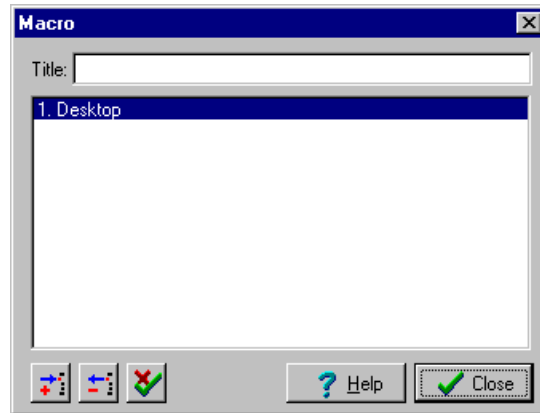


Figure O-3. The Macro Editor window as it appears when you first enter it after indicating you were about to create a macro.

O-3.4. The Macro Item Window

The Macro Item window is accessed from the Macro Editor window described in Section O-3.4. Its purpose is to allow you to select the specific activity you wish to add to your macro. An example of the **Macro Item window** is shown in Figure O-4.

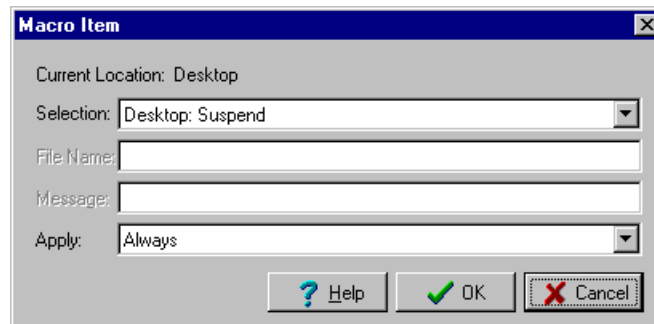


Figure O-4. An example of the Macro Item window – the window used to add a new activity to a macro.

The Macro Item window contains four boxes labeled **Selection**, **File Name**, **Message**, and **Apply**. Only one of them – the Selection box – is active all the time. The **Selection** box is the one you use to select the activity you wish to add. Specifically, it is a drop-down menu listing all of the “acceptable” activities – i.e., the activities that can be directly accessed from the program level you are currently at. For example, you can open a histogram display from the Histogram Analysis window. But you cannot access a signal averaging display from the same place. Consequently, when you are located on the histogram analysis window, the list of acceptable activities will not include a Display Average option. This context specificity prevents you from producing logically inconsistencies into your macro.

The **File Name** box is used to enter the name of a file whenever the activity you select in the Selection box calls for one. See the topic below entitled **Naming Files – The (Auto) and (Manual) Options** for details.


Similarly, the **Message** box is used to enter the text of a message whenever the selected activity is “Message”. The File Name and Message boxes are grayed out whenever the currently selected activity

does not call for them. Details can be found in the topic entitled **The Suspend and Message Options** later in this section.

The **Apply** box is important only when you intend to play back your macro in automatically repeating fashion. Specifically, it is used to inform the program on which playback repetition(s) the activities to follow will be performed. There are three options: **Always** (perform the activities on all repetitions), **First Repetition** (perform the activities on the first repetition only), and **Added Repetitions** (perform the activities on all repetitions except the first). The Always option is the default. Consequently, unless you wish to perform different activities on the first versus subsequent playback repetitions of a macro, you can ignore this box. See Sections O-7 through O-9 for information on how to design macros intended to be repeated more than once.

O-3.5. Adding an Item: An Example

As an example, let's add an activity to the macro shown in Figure O-3. Since we are just beginning to create the macro in question, the only existing item in it is "Desktop", meaning that the macro is at the level of DATAPAC's desktop, or main window. Our task is to decide where we want to go from there. A large number of activities can be accessed from the Datapac 2K2 desktop – more than any other single location in Datapac 2K2. Consequently, the list of alternatives available to you at the level of the desktop is extensive. But, for purposes of example, let's assume we first want to open a data file. The sequence of procedures is illustrated in Figure O-5. The first thing we need to do is to highlight the Desktop item in

the Macro Editor window, then click on the  button (or double-click on the item itself) to access the Add Item window. Next we click on the arrow button on the right edge of the Selection box to view the list of available alternatives and highlight the **Open Data File** option. Then we click the Macro Item window's **OK** button to return to the macro editor window. When we return the window will contain a list of two items instead of one, as shown in Figure O-6. The newly added item is always the one that is highlighted when you return to the Macro Editor window.

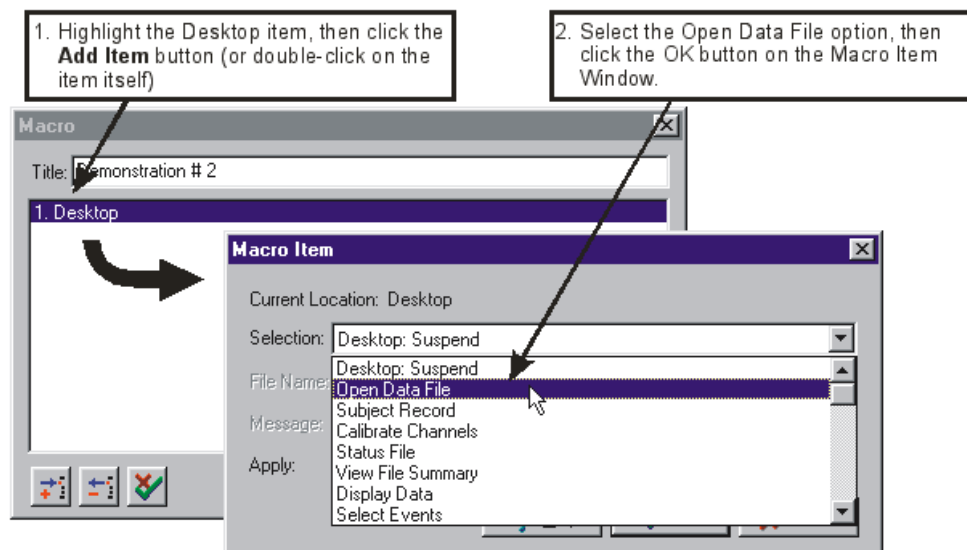


Figure O-5. Follow these steps to add the next item in the macro.

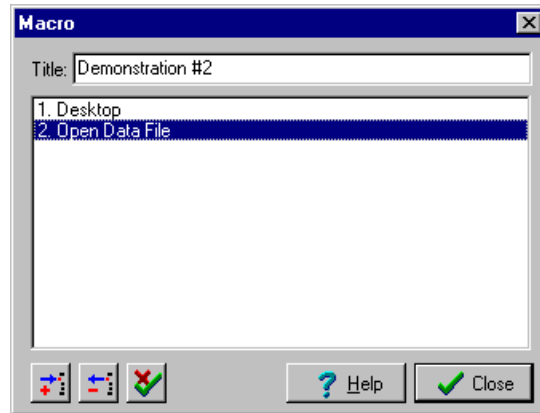



Figure O-6. The Macro Editor window showing the addition of a second item (Open Data File) to the macro.

To add the next item we click on the  (Add Item) button again. The Macro Item window again appears, but this time the Selection menu contains only those activities that can be performed from the level of the Open Data File window. As Figure O-7 (partially) shows, there are a total of ten alternatives in that context: **OK**, **Auto**, **Cancel**, **Suspend**, **Refresh**, **Highlight Next**, **Highlight Previous**, **Highlight Last**, **Highlight First**, and **Message**. If you are familiar with Datapac 2K2 you may have already realized that many of the alternatives in the list are named after options (usually buttons) that appear in the Open Data File window. For example, there is an **OK** button in the Open Data File window, as well as **Cancel**, **Refresh**, **Highlight Previous**, **Highlight Last**, and **Highlight First** buttons. *When an alternative is named for an option that exists in the program, its function is the same as that option.* For example, if you select the **OK** item the macro will perform the same function as if you clicked on the OK button yourself – that is, it will open the highlighted file and return to the desktop.

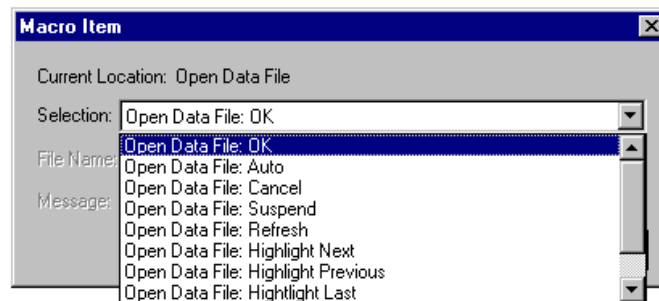


Figure O-7. At the level of the Open Data File window, there are a total of five different activities that you can perform next. Note that the current location is always reported above the selection box.

Different windows offer different options, of course. And there are a large number of windows contained in Datapac 2K2. But the same general rule applies to all: *When an alternative in the list is named for an option that exists in the window, its function is the same as that of the window option.*

The **Suspend**, **Message**, **Auto** and **Manual** options, on the other hand, never appear within the program itself. They are unique to the macro editor and their purpose is to serve useful functions when a macro is played back. Their purpose is described in the following paragraphs.

The Suspend and Message Options:

No matter what the context is, the list of available items always has both a **Suspend** and a **Message** option. The function of the **Suspend** option is to pause the macro during playback, releasing the program to manual control by the user. The user is then allowed to perform any activity they desire until macro operation is resumed (see Section O-5 for additional information about playing macros).

Suspending playback is useful when you arrive at a task that is difficult to automate for one reason or another. Naming a file is a good example – you may wish to allow the user to select a data file of his or her choosing rather than having the macro select one for them (which is the function of the **Auto** option, which is described in a later subsection). But before you suspend a macro it is usually a good idea to let the user know what they are supposed to do while the macro is suspended. That is the purpose of the **Message** option.

The **Message** option causes the macro to pause during playback and present a message window to the user. The text that appears in the message window will depend upon what you insert in the **Message** box of the Macro Item window. For example, Figure O-8 shows a situation where the Message option is selected in the **Selection** box and the Message box contains the phrase, “Open the data file you wish to analyze, then resume the macro.” That is the phrase that will appear in the message window when the macro is played.

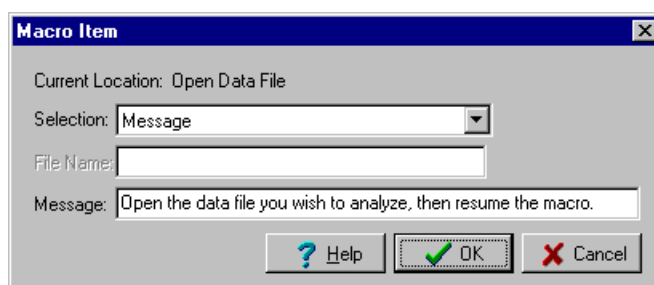


Figure O-8. Example illustrating the way to set up a message window.

The text of a message can contain up to 255 characters, including spaces and punctuation. When a macro encounters a message item during playback it will pause its operation and present the text of the message in a window. The macro will then resume operation when the message window is closed. *The user cannot perform any other activities within Datapac 2K2 in the interim.* Message windows can therefore be used to provide information to the user without disrupting the flow of the macro.

Note that both the Suspend and the Message options can be used to suspend playback of a macro. But the Suspend option suspends playback indefinitely. It is up to the user to decide when to resume playback. The user is also free to perform any activity they want manually while the macro is suspended. The message option, on the other hand, suspends playback only to display the message window. The program is not released to manual control at that time. The only thing a user can do is close the window – at which point playback resumes.

Naming Files – The Auto and Manual Options:

There are many situations where it is necessary or desirable to name files within a macro. You will probably open (load) or save many files within your macros, particularly parameter files. Likewise, every time you export a graphic or table display, you will need to name the file where it is to be saved. File loading and saving options always appear in the list of activities with a description followed by the word “Auto” or “Manual” (most of the time the word is enclosed in parentheses). For example, the list of activities at the level of a data display contains the options, “Load Parameters (Auto)” and “Load Parameters (Manual)”. It also contains the options, “Export to ASCII (Auto)” and “Export (Manual)”. As you can see, the description changes but the word “Auto” or “Manual” always appears at the end.

The word **Auto** indicates that the name of the the desired file is supplied within the macro itself. Consequently, as shown in the example illustrated in Figure O-9, when you select an **Auto** option in the Macro Item window, the **File Name** box becomes active and you are required to enter the name of a file. The word **Manual**, on the other hand, indicates that the macro will pause during playback to allow the user to enter or highlight the name of the desired file. No file name is entered at the time of macro creation.

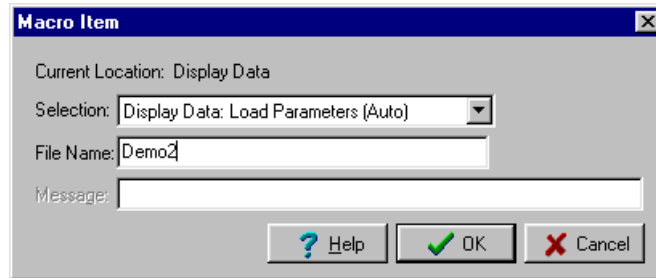



Figure O-9. The (Auto) activity requires you to name a file in the File Name box.

O-3.6. Adding Items in the Middle of an Existing Macro

Thanks to the module's context specific database engine, if you are careful to add items to a macro in sequential order from first to last, you never have to worry about whether a macro's list of activities is ordered correctly. On the other hand, if you add an item somewhere in the middle of an existing the list, the resulting sequence may no longer be logical. Use the **Error Checking** feature, described in Section O-4, to identify errors and to ensure that the resulting sequence is logical.


O-3.7. Deleting an Item

To delete an item, simply highlight it in the Macro Editor window, then click the  (Delete Item) button. When you delete an item be aware that the resulting sequence may no longer be logical. Use the **Error Checking** feature, described in Section O-4, to identify errors and to ensure that the resulting sequence is logical.

O-3.8. Completing a Macro

To terminate creation of a macro, click the Close button in the Macro Editor window. The macro is then automatically saved and you are returned to Datapac 2K2 manual operation mode. Although macros must always begin at the Datapac 2K2 desktop, or main window, they can end anywhere. Consequently, the only consideration as to where to terminate a macro is the intended utility and function of the macro itself. It should be noted, however, that if you intend to automatically repeat playback of a macro, that macro must end at the same place it begins – which is to say the desktop. Also keep in mind that macros can be easily edited at any time.

O-4. Error Checking

The error checking feature identifies logical errors in the flow of activities within a macro. Anytime you finish creating or editing a macro it is a good idea to run the error checker to make sure your macro is error-free. To run the error checker, click the  (Check for Errors) button. If no errors are found the message window shown in Figure O-10 is displayed. Then you know your macro is logical.


 Keep in mind that the error checker will only catch errors in the logical flow of activities. It will not catch other types of errors, such as misspellings or the specification of files that do not exist.



Figure O-10. After running an error check, this window appears if no errors were found.

When an error does exist the message window shown in Figure O-11 appears instead. When you click on the window's **OK** button to continue, the line where the *first* error occurred is highlighted in the Macro Editor window. There may be more, but only one line is highlighted at a time.



Figure O-11. After running an error check, this window appears if one or more errors were found.

For example, assume that the macro shown in Figure O-12 was just checked for errors. An error was found and line 11 was highlighted. What this indicates is that there is an error in the transition from line 10 (Create Many) to line 11 (Status File). If you know Datapac 2K2 well you that something must be done at the level of the Create Many window before one returns to the Status File window. Consequently, there must be at least one command or activity intervening between the Create Many line and the Status File line. There isn't one in the way the macro is currently written and that's what the problem is.

But what if you didn't know you that something must be done at the level of the Create Many window before one returns to the Status File window? What do you do then? And of course, this is just one of a very large number of possible examples. What about those too? Surely you can't be expected to know Datapac 2K2 *that* well!

Actually no, that is not the expectation. In fact, the solution is rather simple regardless of how much you know about Datapac 2K2. Moreover, the same procedure can be applied in every situation. The procedure is this: select the line *immediately before* the one the error checker highlights. Doing so will open the **Macro Item window**. Then open up the drop-down list in the **Selection** line to view the list of possible activities you can perform next at that particular level of the program. This will tell you, each and every time, what the logical choices are to maintain a logical flow. If the flow was logical, then the activity indicated in the next line of the macro would be in the list of choices. But of course it won't be – that's why there was an error, in the first place! So right away you know why the error existed. Your task now

is to figure out a way to bridge the logical gap. Even if you know only a little bit about Datapac 2K2 the solution should be apparent enough. After all, most logical errors occur because an additional line was mistakenly deleted or mistakenly added. In other words, the gap is not usually a yawning chasm. Seeing the list of possible activities should help to remind you as to what you intended to do at that point. And it's usually a matter of adding – or possibly deleting – one or two more activities. But the fact is, all you *can* do is pick one of the options that is presented to you. Then, after you return to the Macro Editor window, run the error checker again. Repeat the procedure as needed until a logical flow is achieved.

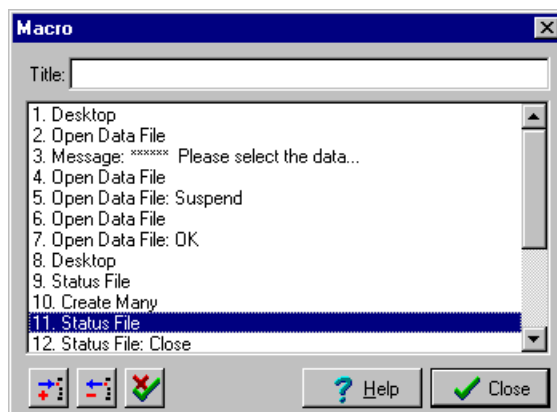


Figure O-12. The Macro Editing window that appears upon completion of the error checking operation discussed in the text.

To illustrate, let's continue our example. As indicated before, the error checker highlighted line 11 (Status File), indicating that an error exists in the transition between line 10 (Create Many) and line 11. Consequently, we select line 10. Then, when the Macro Item window appears, we examine the list of activities we can perform next. As can be seen in Figure O-13, there are a total of five possible alternatives: OK (i.e., select the default parameters), Suspend, Load Parameters (Auto), Load Parameters (Manual), and Message. Let's select the **OK** option.

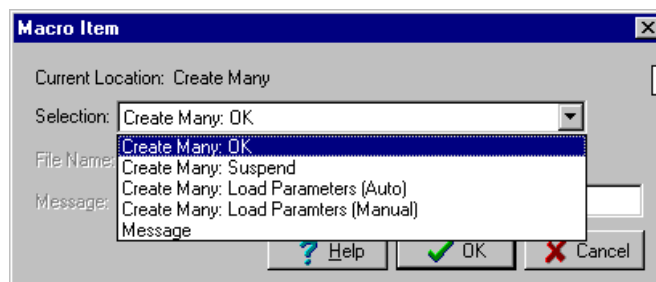


Figure O-13. The list of possible activities available at the level of the Create Many window.

When we return to the Macro Editor window it will appear as shown in Figure O-14. Note that when an activity involves the transition from one window to another, the destination window is added as an additional activity. In this case, selecting the OK option closed the Create Many window and returned us to the window from which it was called – the Status File window. We have thus closed the gap. In fact, we did it a little too well! When we run the error checker again it will highlight line 13. Why? Because *two identical activities cannot immediately follow one another*. Thus, we must delete one of them. At which point our error disappears. The procedure for eliminating virtually any error is essentially the same – just follow the flow until you arrive at the location that closes the logical gap in the macro.

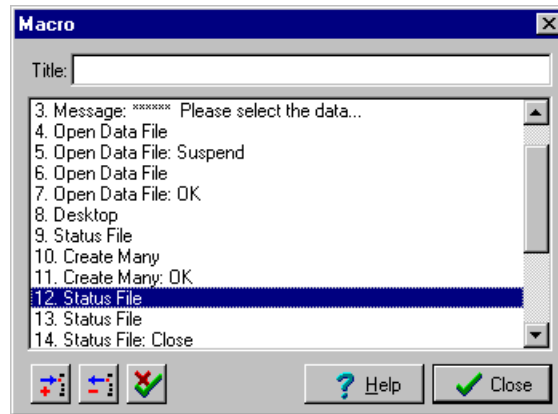



Figure O-14. The Macro Editing window as it appears after selecting the Create Many: OK option, as discussed in the text.

O-5. Playing a Macro

Before you play a macro, first close all windows so that you are viewing only the Datapac 2K2 desktop, or main window. Also, be sure that the initial conditions are consistent with what the macro expects to see. For example, if the macro does not load a data file, a data file should already be open. Likewise, if the macro does not load a set of processing parameters, it may be assuming that some processing parameters are already available. When writing a macro it is a good idea to inform the user of the necessary initial conditions through text messages at the beginning of the macro.

To begin a macro, select **Macros|Load Macro** from the main window menu bar, or click on the  (Load Macro) button in the tool bar. A directory window then appears, allowing you to select the macro you wish to play. An example is shown in Figure O-15. Highlight the desired macro and click the **OK** button.

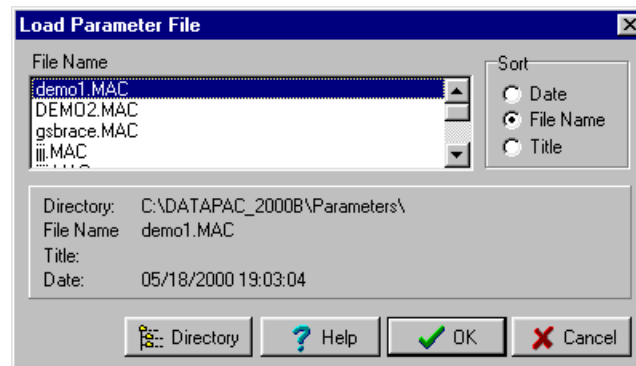


Figure O-15. An example of the directory window for selecting a macro to play.

Finally, you are presented with the window shown in Figure O-16. Indicate the number of times you wish to automatically repeat the macro in the **Number of Repetitions** box. Repeating a macro more than once is useful when you wish to automatically acquire, process and/or analyze more than one data file in the same manner. See Sections O-7 through O-9 for additional details on designing macros for repeated playback.

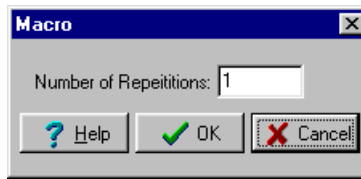


Figure O-16. Indicate the number of times you want the macro to repeat automatically.

After specifying the number of times you want the macro to repeat, click the **OK** button. The selected macro then begins executing immediately. It will stop on four conditions: (1) it encounters a fatal error; (2) it encounters a text message; (3) it encounters a suspend command, or; (4) it plays to completion. Each of these conditions are described below.

Fatal Errors

A fatal error occurs when a defined activity cannot be completed. Fatal errors occur for a variety of reasons, including the existence of a logical error in the macro, inappropriate initial conditions, or when a specified file cannot be found. When a fatal error is encountered the macro automatically aborts its operation and produces a message window similar to the one shown in Figure O-17. Note that the message window identifies the step in the macro that caused the fatal error. An aborted macro cannot be restarted from the point it was aborted. It must be restarted from the beginning.



Figure O-17. An example of the message window that appears when a macro encounters a fatal error during playback.

Text Messages

When the macro encounters a text message command it suspends its operation and produces a message window containing the text the creator of the macro provided for it. For example, the message in the example shown in Figure O-18 reads "***** **Please select the data file you wish to analyze** *****", because that is what the creator of the macro programmed into it.

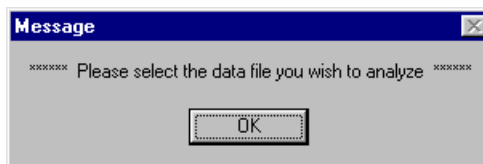


Figure O-18. An example of a text message window that appears during macro playback.

It is important to note that the presence of a message window does not release Datapac 2K2 to manual control. Message windows always remain in the foreground, and no other operations can be performed

while one is present. The only thing the user can do is click the window's **OK** button to resume operation. Message windows are therefore purely informational.

Suspend Commands

When a suspend command is encountered, macro operation is suspended indefinitely, and program control is released to manual mode. To resume a macro, click your right mouse button anywhere on the Datapac 2K2 desktop to produce the macro menu shown in Figure O-19, then select the **Macro Resume** option.

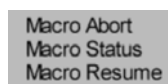


Figure O-19. To resume a suspended macro, right click on the Datapac 2K2 desktop, then select Macro Resume from the macro menu that appears.

Likewise, if you want to know where you are in the macro while it is suspended, select the **Macro Status** button from the same menu. When you do so a window similar to the example shown in Figure O-20 appears, reporting the number and description of the step you are on.

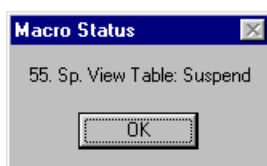


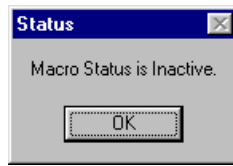
Figure O-20. An example of the Macro Status window.

While a macro is suspended the user has complete freedom to do whatever he or she wants. As a result a careless user can change the conditions enough to cause the macro to abort when its operation is resumed. Fortunately, the macro feature is smart enough to allow a macro to find its way back to the place where it was suspended in most cases. But one situation that is difficult for a macro to overcome, however, is a situation where the user has closed windows that the macro opened. For example, if the macro is suspended on a table or display window – particularly a results table or display window in one of the analysis modules – and the user closes that display window while the macro is suspended, then resumes the macro without re-opening the window, the macro may abort. Generally, therefore, a good rule of thumb is: **do not close windows that the macro opened itself**. It is okay to minimize them if you want. Minimized windows are still considered open. It is also okay to open other windows, do other analyses, and so on, while a macro is suspended. But the moment the macro is resumed, it will close all of the windows it did not open itself and try to locate the window that was active when it was suspended.

Playing to Completion

When a macro plays to completion, control is automatically returned to manual mode. No message appears at the end of a macro unless a message window was inserted when the macro was created. Consequently, it isn't always apparent when a macro completes its operation. You can, however, check to see whether a macro is running or not at any time by clicking your right mouse button on any Datapac 2K2 display window or almost any parameter window. Doing so produces a the macro menu shown in Figure O-19. Select the **Macro Status** option to see whether a macro is running. If no macro is running the macro status window shown in Figure O-21, reporting that the "Macro Status is Inactive". On the other hand, if a macro is running the macro status window will report the item number and description of

the activity that was in effect when the macro's status was requested. An example was shown previously in Figure O-20.



O-21. When no macro is playing, the macro status window reports, "Macro Status is Inactive".

O-6. Designing a Macro to Acquire Data

For the most part, creating a macro to automate data acquisition tasks is no different than creating a macro to automate any other task. But there is one little quirk about this situation that should be kept in mind.

The quirk exists because of the way the manual operating mode Datapac 2K2 is designed. In the manual mode you can select the Acquire option when you either do or do not have a data file already open. What happens next differs between those two conditions. If you have a data file already open you will be transported directly to the Scope window. If you *don't* already have a data file open then the program requires you to create one first (and also allows you to enter any desired subject data) before presenting the Scope window. A macro cannot distinguish between the two conditions, so it always acts as if a data file is already open. If a macro selects the Acquire option when a data file is *not* already open the poor thing gets confused. Consequently, ***always be sure you open a data file before selecting the Acquire option*** (which enters the Data Acquisition module, by the way). This is true even if you intend to create a new data file before acquiring data. Once you are at the level of the Scope window you can always create a new data file, add or edit subject information, load file and scope parameters, and so on.

Figure O-22 shows one appropriate procedure to follow prior to selecting the Acquire option. Note that the macro enters the Open Data File window, selects the OK option, and returns to the desktop before selecting the Acquire option. This specific procedure will open the currently designated default data file, whatever that may be. There are other variations, of course. For example, you may find it more advantageous to open a particular data file (using the Auto option described in Section O-3.4)¹. Or you may prefer to present a message window instructing the user to open a file before proceeding with the macro. But however you do it, the important thing is to ensure that a data file is open when the macro enters the Acquire option.

¹ Data files that were acquired with DATAPAC still have the file and scope parameters that were used to acquire the file attached in the form of companion files. The primary rationale for doing so is to allow you to return to the Data Acquisition module at any time and add more data to the file. But this organization has another benefit – you can use an existing data file to supply the file and scope parameters for new data files that are subsequently created. Of course, you can also load file and scope parameter files to achieve the same effect.

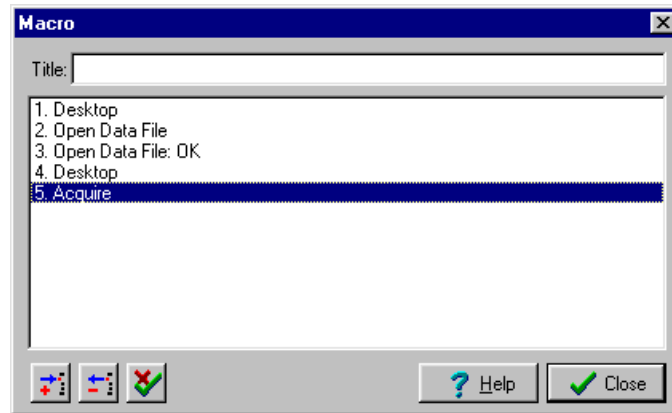


Figure O-22. It is important to open a data file before selecting the Acquire option to enter the Data Acquisition module. This example shows one way to do that.

O-7. Designing Macros for Repeated Playback

There are a few special considerations involved when designing a macro for repeated playback. First, and perhaps most importantly, you must be sure to begin and end the macro at the same program level. Of course, since macros must begin at the level of the Datapac 2K2 desktop, or main window, that is also where the macro must end if you intend to use it in a repeating sequence.

The second consideration involves making sure that the proper preconditions are met before a macro is run. That's always a good idea, of course. But when you are designing a macro for repeated playback it is especially important. To cite a simple example, let's say you intend to create a macro that will acquire a series of data files for a given subject each time it is played. Of course, every time you play the macro you want to acquire a series of data files for a different subject. Consequently, you can't program the subject name into the macro. If you did that, you would have to edit the macro prior to running each time. Thus it would be best to design the macro to suspend its operation to allow the user to insert the subject's name. But since the subject's name doesn't change, you only want the macro to stop on the first repetition, not on subsequent ones. How do you do that?

The scenario just described is an example of a class of situations where it is necessary to establish proper preconditions before a sequence of operations can be effectively automated. But once you have those conditions established, it is often counterproductive to be required to do it again, and again, and again...

Fortunately, the Macro module was designed with such scenarios in mind. To borrow a little programming terminology, a macro can be considered as a collection of subroutines where each subroutine consists of the activities that are performed between returns to the desktop. For example, a macro may include a series of activities to open or acquire a data file, then return to the desktop. Then it may load a set of processing parameters and return to the desktop. Then it may select events and return to the desktop, and so on. Each excursion away from the desktop, and proceeding until you return to it, can be considered a separate subroutine. The macro module lets you decide whether a given subroutine is performed on the first repetition only, on subsequent repetitions only, or on all repetitions. That is the purpose of the **Apply** box in the Macro Item window.

The Apply box is located in the Macro Item window, as illustrated in Figure O-23. It contains three options: **Always** (perform the activities on all repetitions), **First Repetition** (perform the activities on the first repetition only), and **Added Repetitions** (perform the activities on all repetitions except the first). The Apply box is accessible only when you access the Macro Item window from the Macro Editor window by selecting a "Desktop" item, also as illustrated in Figure O-23. It is greyed out at all other times. The

reason is that, as discussed in the previous paragraph, a “subroutine” starts when you leave the desktop and proceeds until you return to the desktop. And once you select an Apply option it is applied to all of the activities in the subroutine.

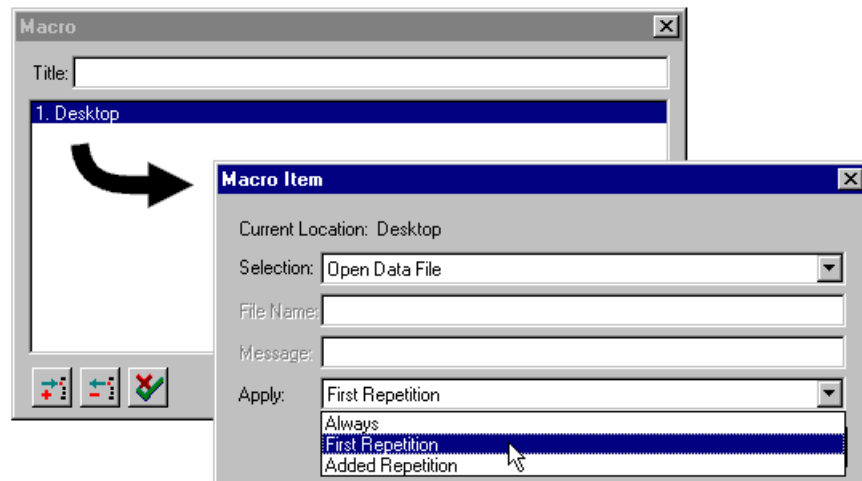


Figure O-23. The Apply box is used to determine whether the activities in the subroutine are performed on the first repetition only, on subsequent repetitions only, or on all repetitions. It is available only when you select the “Desktop” item in the Macro Editor window.

When you return to the Macro Editor window, you can identify the Apply option that you selected for each subroutine by looking at the item immediately following each “Desktop” item. If that item ends with the initials (FR) in parentheses, the corresponding subroutine is to be performed on the first repetition only. The items in line 2 and line 6 of the Macro Editor window shown in Figure O-24 illustrate the point. Similarly, if the item following a “Desktop” item ends with the initials (AR) in parentheses, the corresponding subroutine is to be performed on additional repetitions (i.e., all repetitions but the first). Finally, if the item following a “Desktop” item does not end in either (FR) or (AR), the corresponding subroutine is to be performed on all repetitions.

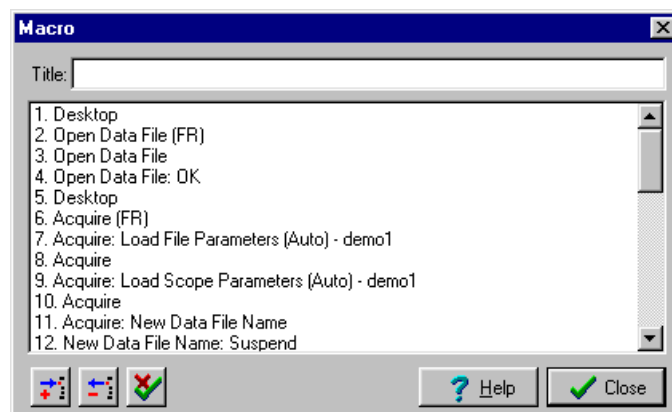


Figure O-24. Lines 2 and 6 end with the initials (FR), meaning that the activity on that line, as well as the activities on all subsequent lines until the next Desktop item, will be performed only on the first playback repetition.

O-8. Designing Macros to Acquire Many Data Files

Since this section concerns itself with acquiring data files, the issues discussed in Section O-6 also apply here. Likewise, since macros that are designed to acquire multiple data files are, almost by definition, also designed to be played in repeated fashion², the the issues discussed in Section O-7 also pertain here. In fact, one major purpose of this section is to apply the concepts described in those previous sections to a relatively simple, real life example. A second purpose is to introduce a feature which was recently added to Datapac 2K2 to simplify the acquisition of multiple data files in a sequential fashion. That feature is the auto-incrementing filename feature. And although it is not a feature that is unique to macro operation (one can conceivably use it in regular manual mode as well), it was certainly intended to facilitate the process within a macro.

In Section O-6 we mentioned that it is important to ensure that a data file is open before you use a macro to select the Acquire option to enter the Data Acquisition module from the desktop. Obviously that consideration applies when you are acquiring multiple data files as well. Likewise, we mentioned in Section O-7 that it is often necessary to establish proper preconditions before a sequence of operations can be effectively automated. That consideration applies to the present case as well.

Consider the macro shown in Figure O-25. First of all, notice that after starting off on the Desktop (of course), the macro returns to the Desktop again a total of three times – on lines 4, 22, and 31. There are therefore a total of three subroutines in this macro. The first subroutine, spanning lines 2 to 3, opens the default data file (just as illustrated earlier in Figure O-24). Note that the first line of this subroutine (line 2) ends with the initials (FR) to indicate that it will be performed only on the first playback cycle. Likewise, the first line of the second subroutine (line 5) also ends in the initials (FR), thus also indicating that this subroutine will be performed only on the first playback cycle as well.

² Of course it is possible to create a long macro that performs essentially the same acquisition subroutines over and over again, but that is not a very efficient way to go about it.

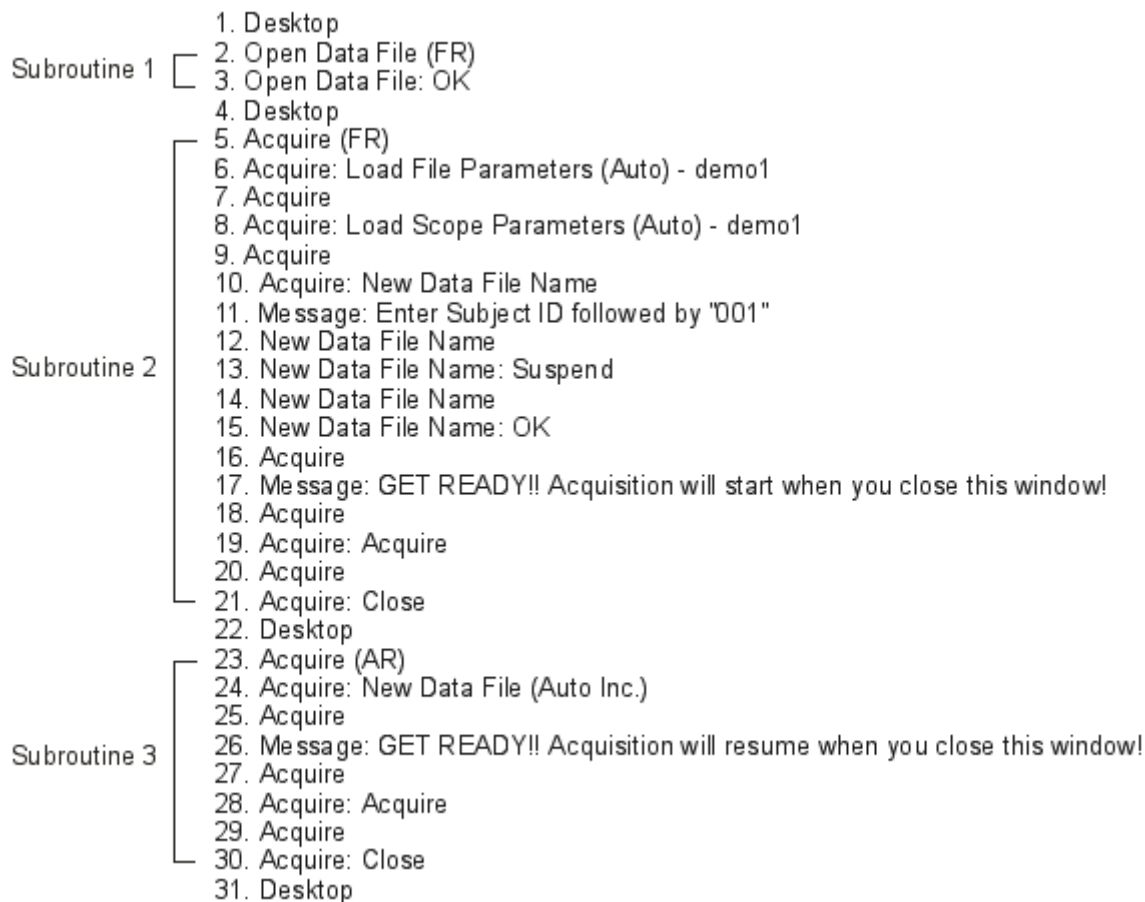


Figure O-25. A simple macro illustrating the proper way to design a macro to acquire multiple data files.

Most of the activities performed by the second subroutine (which spans lines 5 – 21) are setup procedures designed to establish the proper preconditions for data acquisition to proceed most efficiently. Specifically, it first automatically loads a set of file parameters (line 6; file name is “demo1”) followed by a set of scope parameters (line 8; file name is “demo1”). Then it accesses the New Data File window (lines 11 – 15) where the macro suspends its operation after telling the user to “Enter Subject ID followed by ‘001’”. In other words, the user is supposed to enter the subject’s ID, followed by the characters, “001”, in the File Name box of the New Data File window. For example, if the subject’s ID is “RWL”, the user should enter “RWL001”. That becomes the name of the first data file. We’ll explain the reason for the “001” extension shortly³. Finally, after performing the setup procedures just described, the second subroutine finally acquires the first data file (line 19) before returning to the desktop (line 22).

The third and final subroutine again enters the Data Acquisition module (line 23). This time through, however, it selects the module’s **Auto Increment File Name** feature (abbreviated as Auto-Inc. on line 24) to create a new data file before acquiring data to it (line 30) and returning to the desktop. The Auto Increment File Name feature is described in the next paragraph. As the (AR) initials indicate at the end of line 23, the third subroutine will be run on all playback repetitions except the first. Note that despite its brevity, this third subroutine is the one that accomplishes the main goal of the macro – to acquire multiple data files in an automated fashion. Of course, this particular macro is designed to pause on each

³ Notice how we used a message window followed by a suspend command to tell the user what was expected of them, then allow them to do it.



repetition to present a message window alerting the user that the next acquisition session is about to take place (line 26), but other than that it is capable of accomplishing its goal completely on its own.



The **Auto Increment File Name** feature automatically creates a name for a new data file by incrementing the last three digits of the currently open filename by one. The auto incrementing filename feature will not work unless the filename ends in a three-digit number, and it is for this reason that we instructed the user to add a "001" after the subject ID as a way of naming the initial data file during the second subroutine. For example, if the name of the currently open open data file is "RWL001", then the auto-incrementing filename feature will create a new data file with the name "RWL002". Likewise, if the currently open data file is named "RWL002", then the new data file will be named "RWL003", and so on⁴. The auto incrementing file name feature thus allows a macro to generate sequentially numbered data files completely automatically.

O-9. Designing Macros to Analyze Many Data Files

Macros can be used to process and analyze multiple data files in much the same way that they can acquire them. But again, you have to set things up correctly for best performance. There are just two primary considerations to keep in mind when you are creating or editing a macro for batch processing. First, the macro must begin and end at the same place. And again, since macros must begin at the level of the Datapac 2K2 desktop, or main window, that is also where the macro must end if you intend to use in a repeating sequence. Second, the macro must have a way to start in the right place and to select a different data file with each repetition.

One can, of course, suspend the macro at the level of the Open Data File window on each repetition and require the user to select the file they want to analyze. But that does not yield completely automated operation. There is a better alternative: use the **Highlight First**, **Highlight Last**, **Highlight Next** and **Highlight Previous** options that are available in the Open Data File window. Remember that in the normal manual mode of operation, you typically use your mouse to highlight the data file you want to open, then click on the OK button to open it. You can't use a mouse operation in a macro if you want to fully automate it. That's why these options are important – they provide a way to move the highlight around the file list in an automated fashion. Specifically, the **Highlight First** option (which appears in the

Open Data File window as the  button) moves the highlight to the first file in the list. Likewise, the **Highlight Last** option (which appears as the  button) moves the highlight to the last file in the list.

The **Highlight Next** and **Highlight Previous** options (which appear as the  and  buttons) move the highlight to the next file in the list in the descending or ascending direction, respectively.

Consider the macro shown in Figure O-26. Note that the first subroutine (lines 2 - 5), which is performed only on the first playback repetition, goes from the desktop to the Open Data File window where it first selects the **Highlight First** option (line 3), then the **OK** option to open the first file in the directory list (line 5) before returning to the desktop. This operation ensures that the macro always starts out at top of the filelist. That may not always be appropriate, of course. If it's not, then another strategy can be employed⁵. But the idea is to make sure the files are listed correctly and that you are starting out at the right place in the list.

The second subroutine (lines 7 – 10), again enters the Open Data File window where it selects the **Highlight Next** option (line 8), then the **OK** option to open the newly highlighted file (line 10) before

⁴ The Auto Increment File Name feature operates on the last three digits of a filename. Because of that, you can use the feature to acquire up to 1000 files in sequence.

⁵ For example, you can have the macro suspend operation on the first playback repetition to allow the user to highlight the first file manually before resuming operation. Also keep in mind that files can be sorted using a variety of means – by date, but subject name, by ID code, etc.

returning to the desktop. This subroutine is performed on subsequent playback repetitions but not the first, allowing the macro to increment through a list of files with each repetition. The third and subsequent repetitions, which are performed on all repetitions, then perform the processing and analysis operations.

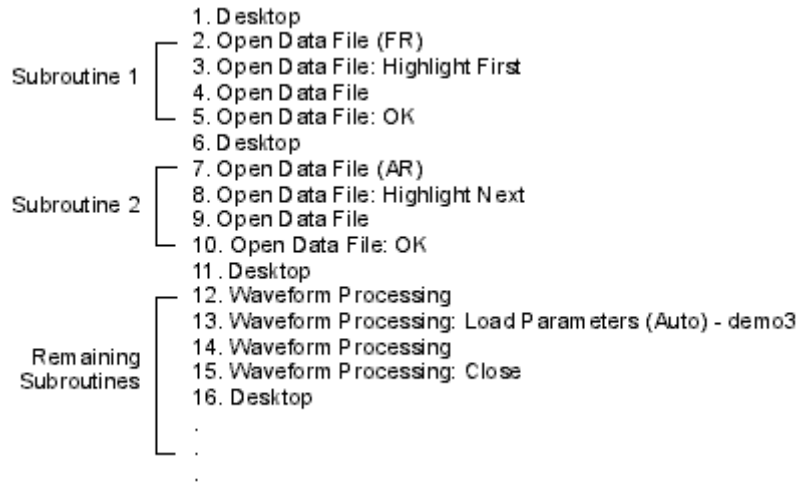


Figure O-26.

O-10. Using Macros to Produce Reports


Reports are documents that are generated from a document template that has been prepared in a word processing application (e.g., Microsoft Word). A document template is very similar to a standard document except that you can use it to produce many individual reports very easily. In fact, the only important difference between a document template and a standard document is that the template contains links to files that have been exported from Datapac 2K2 -- or any other source, for that matter. That is the beauty of using an established word processing application -- all of its power and versatility can be employed to render reports that are formatted exactly the way you want them, and which contain exactly the information you desire.

It is beyond the scope of this discussion to provide a primer on document production in a word processing application. The reader is expected to already have a good working knowledge of how to create documents. Rather, we will focus on three issues: (1) how to create a simple document template with links to text and graphics files that have been exported from Datapac 2K2; (2) how to create and use a macro to export text and graphics files from Datapac 2K2, and; (3) how to generate individual reports from the template document. Text and graphics files can be exported from Datapac 2K2 without using macros, of course. But macros do make the process much easier and less prone to error. Consequently, our discussion will focus on designing macros for that purpose.

The remaining discussion assumes the use of Microsoft Word, specifically the Word 2000 version. Earlier versions of Word, such as Word 97, are also acceptable, but the procedures we discuss may differ in detail somewhat. Other word processing applications have not been tested. We assume they will work but we cannot guarantee them, and therefore we cannot recommend any alternatives.

O-10.1. Designing a Macro to Export Tables and Figures from Datapac 2K2

Macros can be used to automate the export of tables and figures just like any other activity. The first step is to make sure you are at a suitable location in the macro editor window. For example, the line that is highlighted in the macro editor window shown in Figure O-27 reads “Display Data”, indicating that a data display window is open at that location in the macro. Double-click the line that the highlight is on, or select

the  (Add Item) button to access the Macro Item window.

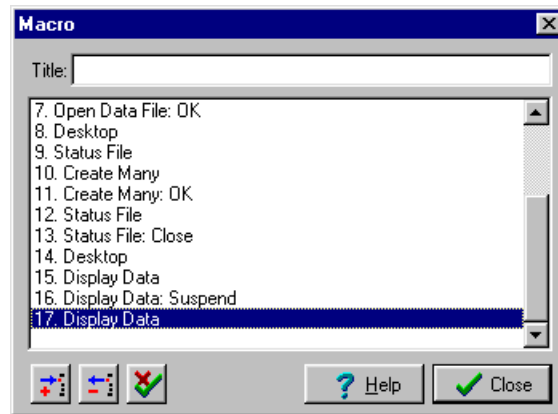


Figure O-27. Example illustrating the first step in using a macro to export a graphics file.

Since in this particular context we are exporting a figure, select the **Export to EMF (Auto)** option⁶ in the drop-down menu associated with the Macro Item window's **Selection** box. You may recall from the discussion of naming files back in Section O-2.2 that when you select the Auto option you are required to enter a filename in the Macro Item window's **File Name** box, as illustrated in Figure O-28. Enter only the filename prefix, not the extension. The proper extension is automatically applied. After entering the filename, click the OK button to return to the Macro Editor window. The macro is now programmed to export the figure to a file with a specific name.

The procedure for exporting a table to a text file is nearly identical except you select the **Export to ASCII** option from the drop-down menu associated with the Selection box.

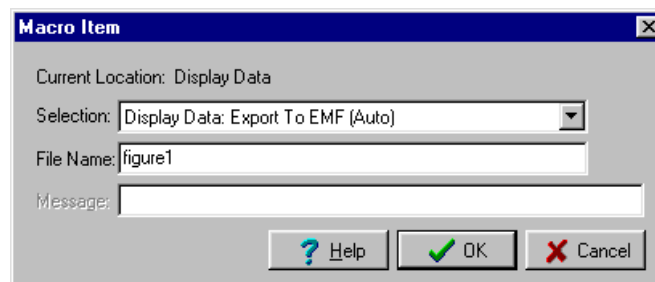


Figure O-28. When naming a file within a macro, enter only the prefix of the filename, never an extension.

⁶ The **Export to WMF (Auto)** option can also be used, but the EMF format is preferred whenever there is a choice.

There is no limit to the number of tables and figures exported by a single macro, so you can repeat the procedures described above as many times as needed. Once you have finished creating or editing the macro, play it once all the way through so that an entire set of exported files are generated. This is important to do because the files must exist before you can link them into your template document.



For whatever reason, Word protects files that are (or were) linked to any document that was opened at any time since Word itself was opened. It does not matter whether the document is currently open or closed. If you play a macro that attempts to write to a protected file, you may crash Datapac 2K2, and may cause the character font to be reset (we have no idea why it happens – it's very strange). Consequently, ***it is strongly recommended that you close Word while you play a macro.***



When setting up tables that you intend to export and link into a template document, be sure to limit the width of the table to a size that will fit into the document without wrapping. See the **Formatting Linked Text Files** topic in Section O-10.2 for more information.

O-10.2. Creating the Template Document for a Report

In its simplest form the template document for a report is merely a document that contains links to one or more text or graphics files. Figure O-29 shows an example of a report in its simplest form. The template that produced it consists of just two links, one to a graphic file and the other to a text file. Both linked files were exported from Datapac 2K2.

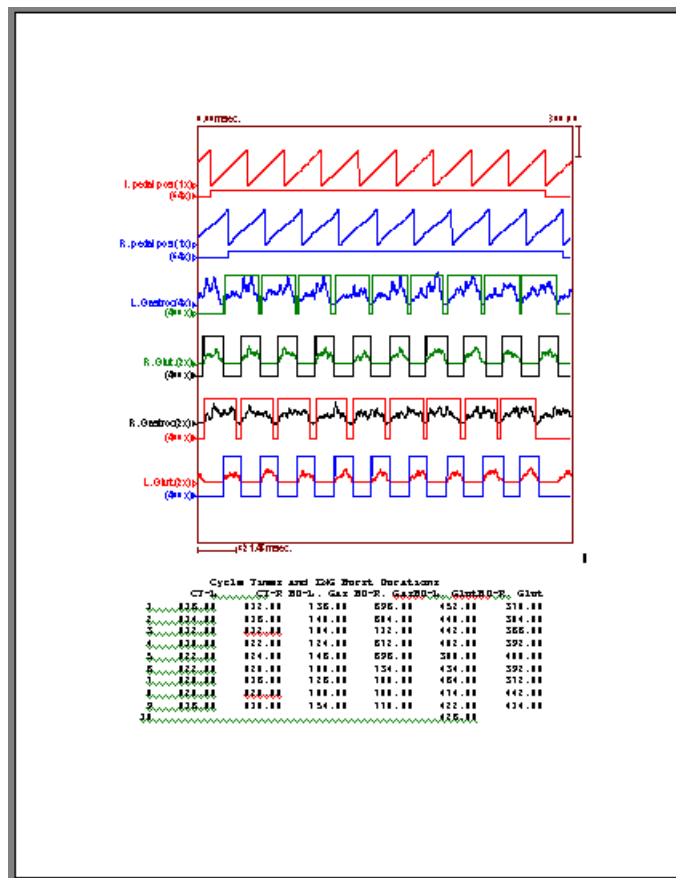


Figure O-29. An example of a simple report document.

Of course a template document can be far more robust and elaborate than the example shown in Figure O-29. It can contain titles, sections, headings, and various forms of “boilerplate” text⁷. It can also contain links to items from sources other than Datapac 2K2. That is the beauty of using an established word processing application such as Microsoft Word; all of its power and versatility is available to you. Anything you can incorporate into a regular document you can incorporate into a document template and the reports generated from it. It is beyond the scope of this discussion to review the many features and capabilities of Word and applications like it, however. Rather, we will limit our discussion to those procedures that are directly pertinent to our needs. Specifically, we will explain how to create a simple document template and how to establish links to other files within it.

⁷ “Boilerplate” text is any text that is incorporated as part of the template document itself, as opposed to text that is imported from some other source (e.g., a linked file). Boilerplate text, as well as any other elements incorporated into the template document itself, do not change from one report to another when multiple reports are produced from the same template.

Creating a New Template Document

To create a new template document, open Word, then select **File|New**. Upon doing so you are presented with a window listing the available templates. An example is shown in Figure O-30. Highlight the one that most closely suits your needs. For this discussion we will use the **Blank Document** template. Also select the **Template** option in the **Create New** section at the bottom right. Then click the **OK** button to close the window and return to Word's document editing window.

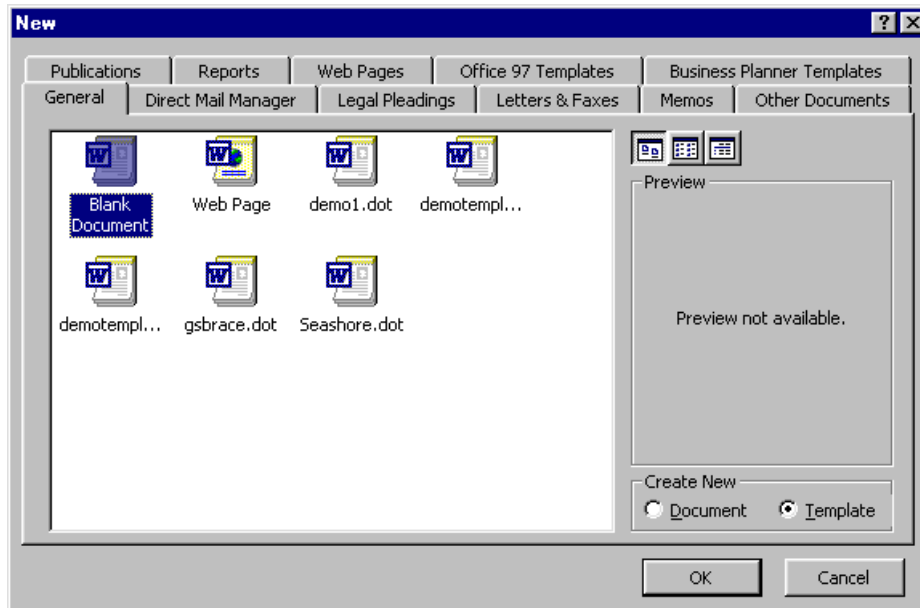


Figure O-30. Select the type of template document you want to use.

Creating a Link to a Figure (Graphics File)

To create a link to a figure, or graphics file, first position the pointer at the location in the template document where you want to insert the figure, then select **Insert|Picture|From File** from the menu bar. The process is illustrated in Figure O-31.

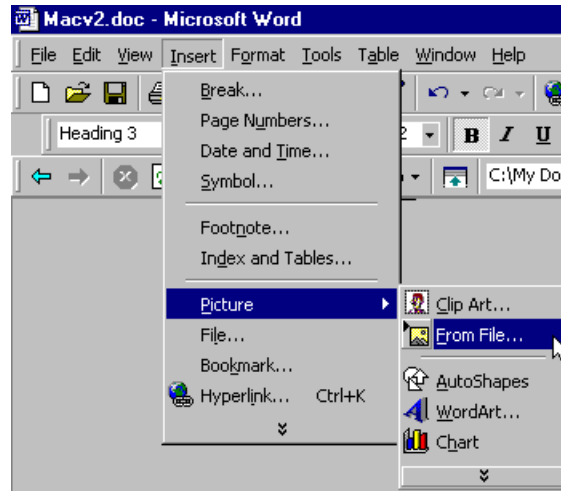


Figure O-31. To insert a link to a figure into your document, select **Insert|Picture|From File** from Word's menu bar.

Next you are presented with a file directory window, as illustrated in Figure O-32. Find the drive and directory containing the file, then highlight the filename in the filename list box on the left side of the window (or enter the information into the **File name** edit box). Next, click on the arrow button to the right of the Insert button and select **Insert and Link**. You are then returned to the document editing window. That's all there is to it. The link is now established, and the figure should now appear in your document. Once it does, size and position the figure in the document as you see fit – just as you would any other figure.

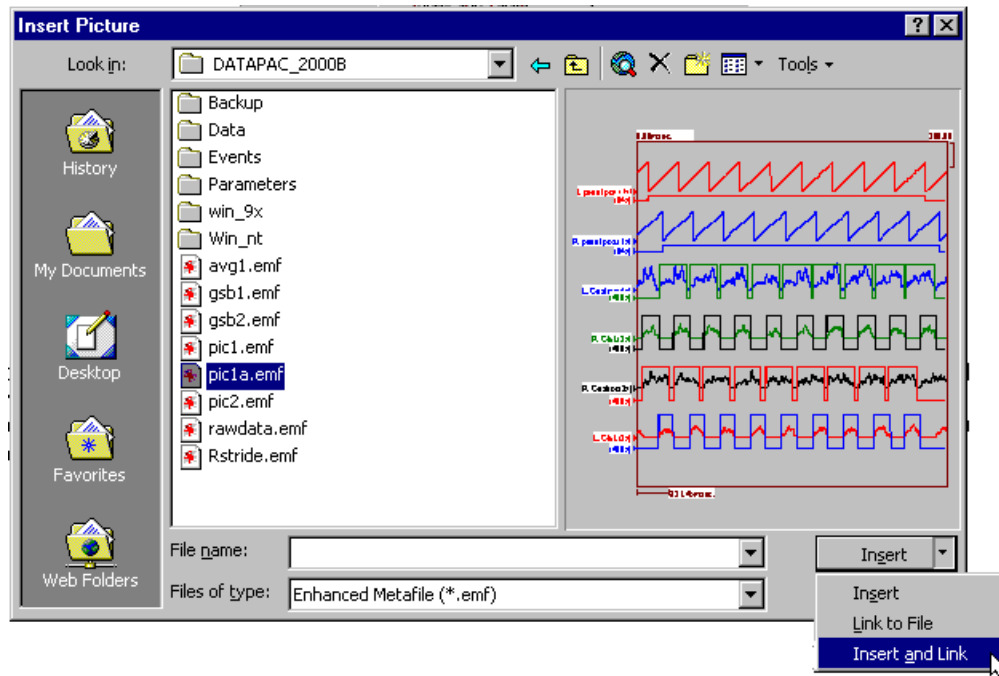


Figure O-32. An example of the directory window that appears for the purpose of inserting a figure into your template document.

Creating a Link to a Table (Text File)

To create a link to a text file, first position the pointer at the location in the template document where you want to insert the figure, then select **Insert|File** from the menu bar. The process is illustrated in Figure O-33.

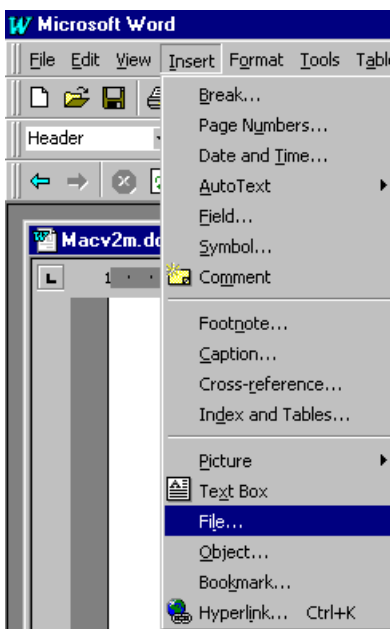


Figure O-33. To insert a link to a text file into your document, select **Insert|File** from Word's menu bar.

Next you are presented with a file directory window, as illustrated in Figure O-34. Set the **Files of type** option to either **Text files (*.txt)** or **All files (*.*)**. Find the drive and directory containing the file, then highlight the file name in the list box on the left side of the window (or enter the information into the **File name** edit box). Finally, be sure to check the **Link to file** check box before clicking on the **OK** button to close the file directory window and return to the document editing window. That's all there is to it. The link is now established, and the contents of the text file should now appear in your document.

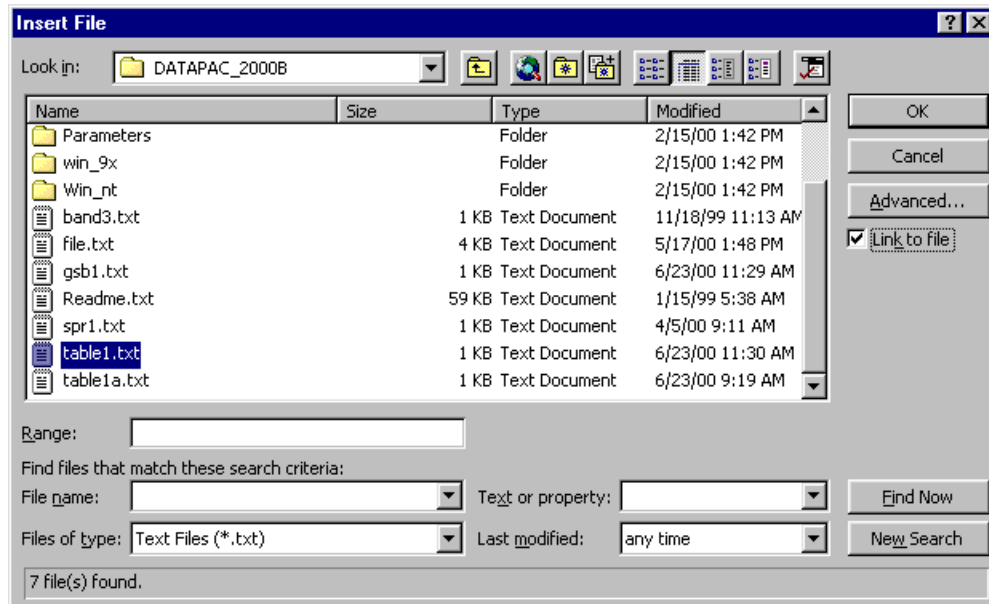


Figure O-34. The Word directory window that is presented to allow you to select a file to link to your document.

Formatting Linked Text Files

Linked text files (or any imported text file for that matter) are formatted with the “Plain Text” style. Consequently, you can change the characteristics of the imported text (the type of font, font size, etc.) by editing the Plain Text style. To edit the Plain Text style, select **Format|Style...** from Word’s menu bar. When the Style window appears, highlight the **Plain Text** style and click on the **Modify** button. That will bring you to the **Modify Style** window. Next, click on the **Format** button and select the type of modification you wish to make. For example, to change the style or size of the font, select **Format|Font**.

It is important to recognize that text files generated within Datapac 2K2 – and in particular tables that are composed of two or more columns -- do not behave well when they are viewed in a proportional, or variable width font. Most Windows fonts are variable width. Consequently, you only have a few choices available to you. “Courier New” is a good choice, and should be the default selection. But there are a few others as well. Unfortunately they are hard to find.

Font size is also an important consideration. You should select a font size that will allow the entire width of the table to fit on one line. If you let the table wrap around, it will become difficult to read. If you find that you must employ a font size that is too small, then compose a narrower table, either by eliminating columns or reducing their width. Adjusting the paper orientation to landscape is also a viable option.

Saving the Document Template

Once you have completed inserting the links to all of the files you wish to include in your template document (and have inserted any additional embellishments as well), you are ready to save your template file. To do so, select the **File|Save As** option from Word's menu bar, and enter a name for your document template file. Finally, close the template file.

O-10.3. Producing Reports

Once your template document is created you are ready to produce reports. To produce a report, first select **File|New** from Word's menu bar. When the New window appears, find the document template file you created, highlight it, and click the **OK** button.

Next, select **Edit|Links....** from Word's menu bar. A window listing the files linked to the document then appears. Highlight all of the file names, then click on the **Update Now** button. Finally, click on the **Close** button to exit the Links window and return to the document editing window. Examine the report document to make sure all the links have been updated correctly. Finally, select **Edit|Links....** from Word's menu bar again. Again highlight all of the file names, then click on the **Break Links** button. That will break the links to the original source files and preserve the contents of the figures and tables as they presently exist so there is no chance that they will be inadvertently updated in the future.

To save the report, select **File|Save** (or **File|Save As**), enter a name for the file, and click on the **Save** button. That's all you need to do! You are now ready to re-run your macro on new data in preparation for your next report.



For whatever reason, Word protects files that are (or were) linked to any document that was opened at any time since Word itself was opened. It does not matter whether the document is currently open or closed. If you play a macro that attempts to write to a protected file, you may crash Datapac 2K2, and may cause the character font to be reset. Consequently, ***it is strongly recommended that you close Word (not just the template document) when you are ready to replay the macro in Datapac 2K2.***