U S E R ' S G U I D E

CAPBAK/MSW

Ver 3.2

Capture/Playback Tool for MS-Windows System

SOFTWARE RESEARCH, INC.

This document property of:

Name:_____

Company:_____

Address:_____

_____

Phone_____

SOFTWARE RESEARCH, INC.

625 Third Street

San Francisco, CA 94107-1997

Tel: (415) 957-1441

Toll Free: (800) 942-SOFT

Fax: (415) 957-0730

E-mail: support@soft.com

http://www.soft.com

(Last Update January 25, 1999)

/home/l1/wu/win-capbak/cb32/MSW-capback3.2.B

Table of Contents

CAPBAK/MSW User's Guide

List of Figures

Preface

Congratulations!

By choosing the TestWorks integrated suite of testing tools, you have taken the first step in bringing your application to the highest possible level of quality.

Software testing and quality assurance, while becoming more important in today's competitive marketplace, can dominate your resources and delay your product release. By automating the testing process, you can assure the quality of your product without needlessly depleting your resources.

Software Research, Inc. believes strongly in automated software testing. It is our goal to bring your product as close to flawlessness as possible. Our leading-edge testing techniques and coverage assurance methods are designed to give you the greatest insight into your source code.

TestWorks is the most complete solution available, with full-featured regression testing, coverage analyzers, and metric tools.

Audience

This manual is intended for software testers who are using *CAPBAK/MSW*. You should be familiar with the Microsoft Windows System and your workstation.

Typefaces Used

The following typographical conventions are used throughout this manual.

boldface Introduces or emphasizes a term that refers to **STW**'s window, its sub-menus and its options.

italics Indicates the names of files, directories, pathnames, variables, and attributes. Italics are also used for the titles of manuals, books, and chapters.

"Double Quotation Marks"

Indicates chapter titles and sections. Words with special meanings may also be set apart with double quotation marks the first time they are used.

`courier` Indicates system output such as error messages, system hints, file output, and *CAPBAK/MSW*'s keysave file language.

`Boldface Courier`

Indicates any command or data input that you are directed to type. For example, prompts and invocation commands are in this text. (**`stw`**, for instance, invokes *TestWorks*.)

CHAPTER 1

Introduction to CAPBAK/MSW

This chapter explains the *CAPBAK/MSW* basics. You will learn the basic functions of *CAPBAK/MSW*, how it can help you, and its role in Quality Assurance.

1.1 Automated Testing

In the past, application and operating environments were relatively simple. Manual testing or a few written test scripts stored in batch files were usually sufficient to fully exercise the product. Today's applications, however, are much more complex, as are the environments in which they run.

A single release of a software product realistically involves multiple versions. Therefore, over a single production cycle, a product will have to be tested several times. Additionally, as the product undergoes several releases over the course of its lifetime, the total number of times that the software must be tested can easily be over a dozen. In this respect, the traditional methods of testing can be painstaking — testing activities must be repeated again and again throughout the application's lifetime.

For testing to be most effective, it must be automated.

1.2 The STW/Regression Solution

Software Research, Inc. offers a solution, *STW/Regression*™, that automates software testing. *STW /Regression* is designed to overcome the laborious and error-prone process of manual testing. Test outcomes are recorded and compared automatically with baselines. Any discrepancies are recorded and stored for further analysis. Extraneous or irrelevant discrepancies, however, can be discarded in the comparison process. Finally, test execution, reports and statistics are available for viewing. With *STW/Regression*, you can reduce the time it would take to do the work manually by 70 to 90 percent.

STW /Regression includes the following products:

- *CAPBAK /MSW* captures and plays back keystrokes, mouse movements and user-selected images.
- *CAPBAK /MSW*'s *CBDIFF* utility compares screen fragments — with optional masking.
- *CBDIFF*™ compares text files.
- *SMARTS/MSW*™ organizes and executes a collection of tests which may use recorded scripts that are executed by *CAPBAK /MSW*.

CAPBAK /MSW and *CBDIFF* are the focus of this manual. You should consult the user manuals for each of the other *STW /Regression* products for complete details on how to use them.

Below is a *STW /Regression* flow chart. Boxes with darkened backgrounds represent the main components of *STW /Regression*.

SMARTS/MSW is in the center because it controls batch executions that produce results. *SMARTS/MSW* can run tests of all kinds, including playback from a range of capture and replay systems (depending on the

environment you are in). The other components support *SMARTS/MSW* in various ways.

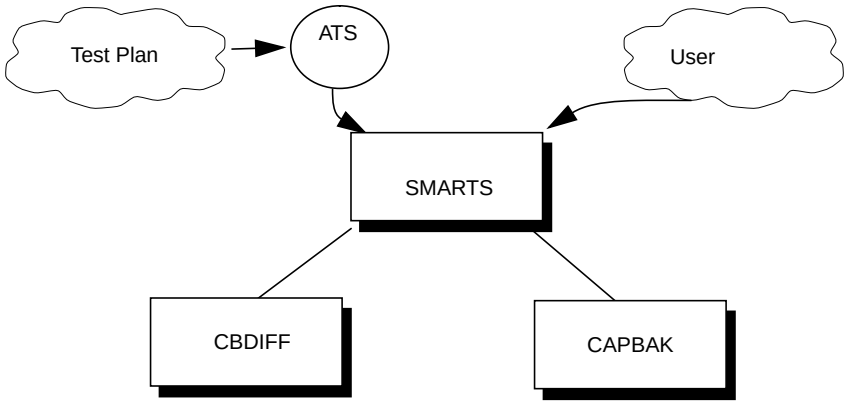


FIGURE 1 STW/Regression Flow Chart

1.3 CAPBAK/MSW's Role

Effective test automation begins with *CAPBAK /MSW*. You create a test session in which you input mouse and keyboard events to the application-under-test (AUT). During this time, you capture images off the screen. The images serve as baseline files which will be compared with response images that are captured during playback. Future tests then entail playing back the test session.

CAPBAK /MSW records all your MS-Window activities (including all keyboard and mouse events). It also captures screen images, including areas of the screen, windows, or the entire screen upon your request. During the recording session, *CAPBAK /MSW* automatically generates input statements that represent user input directed to the application-under-test (AUT). These statements are stored in a test script file called a **keysave** file.

When a test is played back, the same input statements are regenerated and sent to your program, the application-under-test. The AUT executes the previously recorded statements, exactly as before. Therefore, all keystrokes and mouse movements corresponding to the recorded input are played back. *CAPBAK /MSW* ensures reliable playback because it has a built-in synchronization feature, and it permits for user-defined synchronization points.

For comparing the behavior of two AUT versions, *CAPBAK /MSW* recaptures the same images during playback as you captured during the recording session. You can look at these corresponding images and automatically or manually compare them to determine if there are any discrepancies between the two sessions.

Optical Character Recognition (OCR) capabilities are also available for use in text comparisons.

After recording, you have the option to edit the keysave file script. By altering the recorded keysave file, you can change the behavior of playback without having to re-record the test session.

The diagram below illustrates the entire *CAPBAK /MSW* process.

Circles represent the four main modes of *CAPBAK /MSW*, which include planning a test, recording / playing back that planned test, viewing any images that may have been captured during the recording and playback sessions, and comparing those images for possible differences.

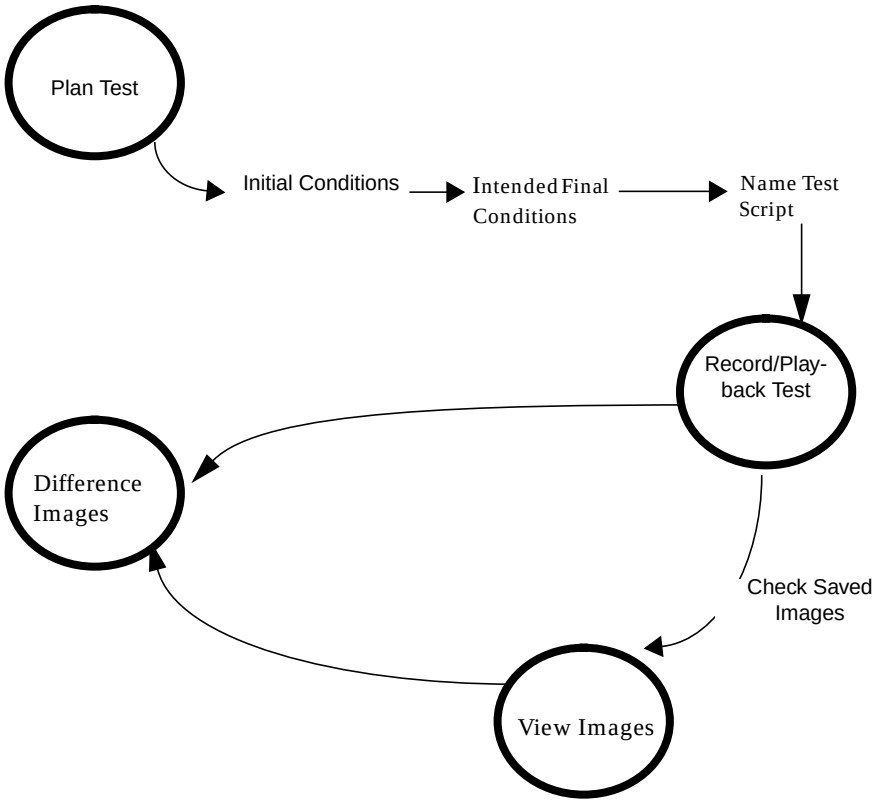


FIGURE 2 CAPBAK/MSW's Flow Chart

1.4 Main Features

You should keep these features in mind when planning your use of the *CAPBAK /MSW* product, when organizing your tests, and when integrating them into your automated test system.

- Faithful time recording of all keystroke, mouse and image captures.
- Object Mod and TrueTime Mode Capture / Playback
- Character Recognition
- C-Interpreted Script
- Host-Based Testing
- Client-Server Testing
- Multiple Synchronization Mode: automatic event synchronization every time a new window pops up; playback with a variety of synchronization methods
- Recording that permits keysave file appending
- Full or partial window capture of image data
- Automatic comparison of captured windows and partial images
- Masking of images is available
- Conditional execution during script playback (playback programming)
- Inclusion of comments in the keysave file
- Functionality accessed through a MS-Windows graphical user interface (GUI)
- Function key access for entering commands during a recording session

CHAPTER 2

Installation

This chapter shows you how to install *CAPBAK/MSW*.

2.1 System Requirements

Your computer system must have the following hardware configuration to install and run **CAPBAK/MSW**.

- Windows 95, 98 and NT
- 486 microprocessor or better
- 20 MB free disk space
- 16+ MB RAM recommended

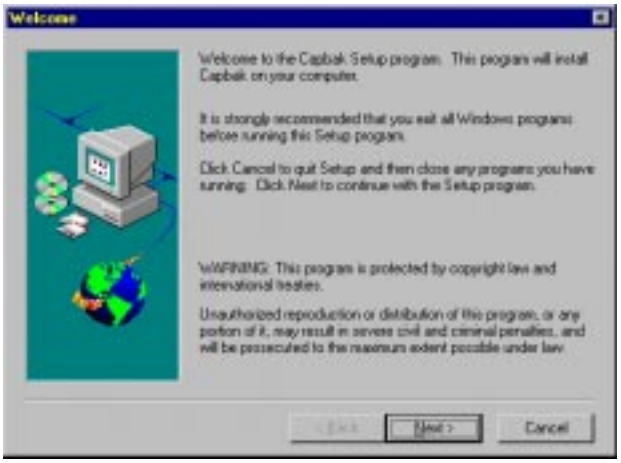
2.2 Installation Procedure

These are instructions for installing **CAPBAK/MSW**.

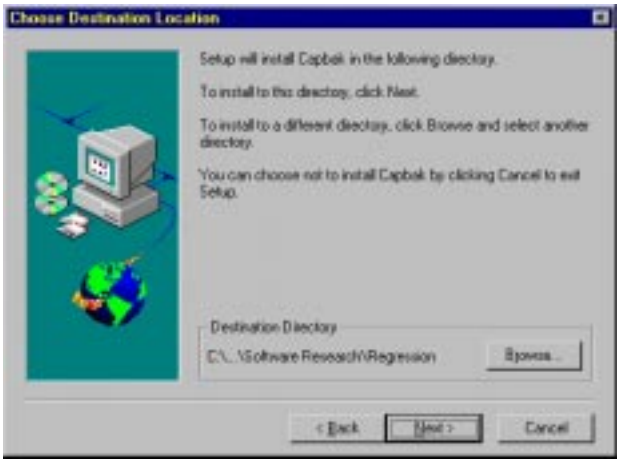
Administrator privileges are required to properly install CAPBAK in Windows NT.

1. Insert the **CD Disk** in your CDROM drive (these instructions assume D:).
2. **Activate setup.exe.**

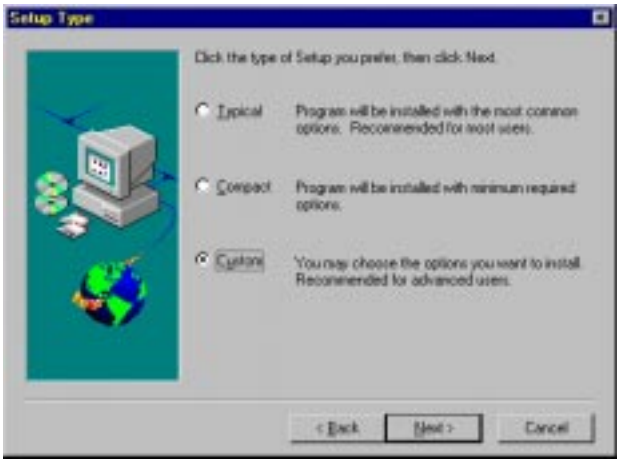
- **In Windows 95 or NT 4.0:** Display the contents of the D: drive, using either the **My Computer** icon (on the desktop) or **Windows Explorer** (on the **Start** menu, **Programs** submenu). Double-click **setup.exe**.



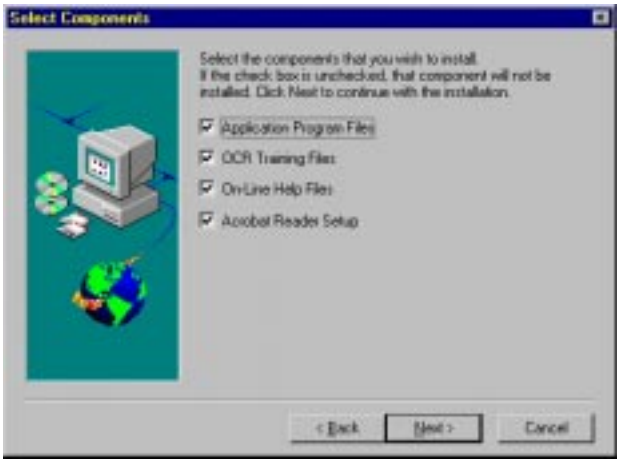
3. **setup.exe** presents you with a series of dialog boxes, beginning with the **Welcome** box shown above. Each box is a step in the installation process, and when you are satisfied with the options offered in a box you should click **Next** to go on to the next step.
4. If you click **Next** in the **Welcome** box, a second box asks you where you would like store the executables and the supporting files for **CAPBAK/MSW**.



- Click on **Next** if you want to use the **Path** indicated and to continue the installation.
- Edit the default path to your own path, then click **Next** to continue the installation.
- Click **Cancel** to end the installation.
- If you choose **Next**, a dialog box to pops up and asks you what kind of installation you prefer. It is highly recommended that you select **Custom** installation, which allows you to install the **Acrobat Reader** software that will allow you to read the on-line help that accompanies **CAPBAK/MSW**. The **Acrobat Reader** software will occupy approximately 4 MB of your computer's memory.



- Click **Next** if the Setup Type is the one you prefer.
- Click a different Setup Type, then click **Next** to continue the installation.
- Click **Back** to review or change previous dialog box queries.
- Click **Cancel** to end installation.



5. In Windows 95 or NT 4.0, after you choose **Next**, a dialog box will not pops up to ask you to choose the program group name where you would like the program icons to appear.

6. During copying, a bar gauge names the files being copied.

7. A *C:\Program Files\Software Research\Regression* directory (or the path you indicated) should have been created. **CAPBAK/MSW** will automatically store your files to this directory unless you selected otherwise.
8. The installation script also creates a program group where **CAPBAK/MSW** and its utilities are installed:

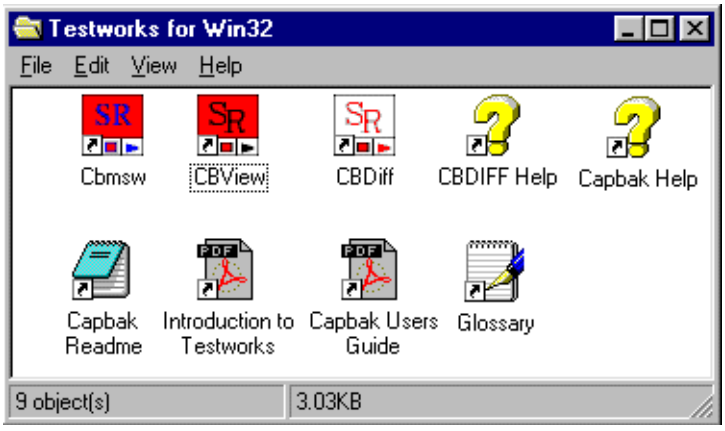


FIGURE 3 Program Group for CAPBAK/MSW

9. When the installation is complete, you should include the *STW* pathname in your system environment variable.
10. To uninstall, use the following:

- In Windows 95 and NT 4.0, double click the **Add/Remove Programs** icon in the Control Panel, highlight **CAPBAK/MSW**, and click the **Remove** button.

2.3 File List

The following files are written to your computer during the installation. The locations for these files are given for installation to a directory called *C:\Program Files\Software Research\Regression\Capbak.*

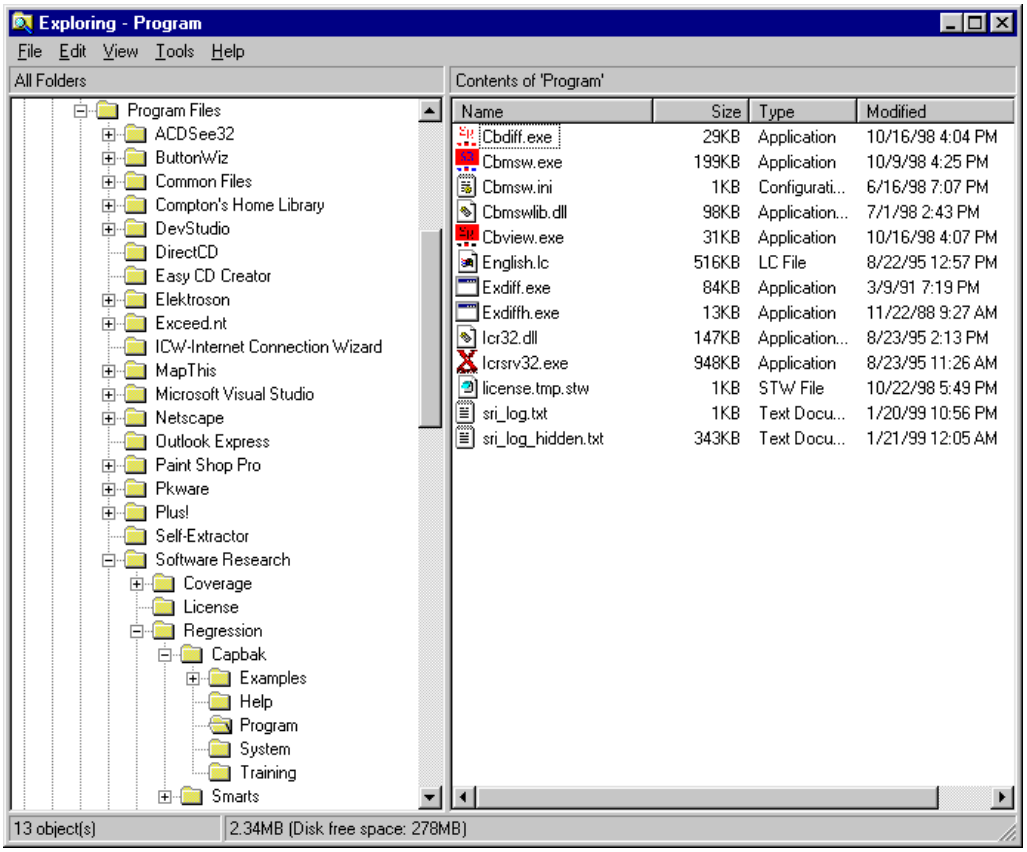


FIGURE 4 Files for CAPBAK in Windows 95 and NT

CHAPTER 3

Quick Start

This chapter shows you how to use the basic components of CAPBAK/MSW.

3.1 Getting Acquainted with CAPBAK/MSW

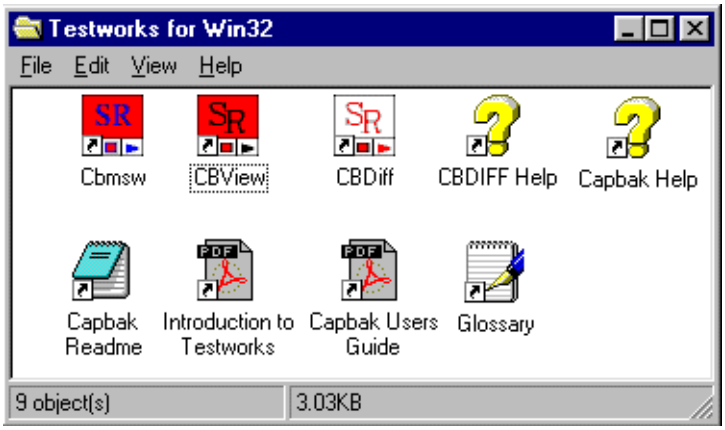
It is recommended that you complete the example in this chapter before continuing to other sections. This chapter gives you a feel for how the system is organized and permits you to create more efficient and effective tests.

When you have completed this chapter, you should be familiar with the main activities involved in using *CAPBAK/MSW*, including setting up a test, recording and playing back a test, viewing captured images, and comparing captured images. When you record a test, you establish a baseline of expected behavior. Each time you want to retest an application's operations, you play back the test. The playback lets you determine if your application performs as it did when the test was recorded. Playback creates a response of actual behavior. You can verify a recording and playback session by examining the captured images and comparing expected and actual images.

This chapter is best used if you also make reference to Chapters 4 to 11 on user manual for detailed explanations on functionality.

3.1.1 STEP 1: Starting CAPBAK/MSW

1. From **Start** click on the **TestWorks for Win32** icon in **Programs** menu.This will display the **TestWorks Group** window.



2. This window contains icons for the programs that come with CAPBAK :
   - Cbmsw (**CAPBAK MS-Windows)** invokes *CAPBAK/MSW*'s record / playback utility
   - **CBVIEW** invokes *CAPBAK/MSW*'s utility for displaying captured images
   - **CBDIFF** invokes *CAPBAK/MSW*'s utility for comparing captured images
   - Acrobat Reader setup

This tutorial demonstrates the following utilities.

3. For the first part of this tutorial you will be using the record / play-back utility of *CAPBAK/MSW*. To start this utility double click on the **Cbmsw** (CAPBAK MS-Windows**)** icon.
4. The **Record/Play** window pops up. Its VCR-like control panel records and plays back sessions.
5. The status window prompts you: **Ready** mode after initializing the OCR.

NOTE: If you want to start over, you can terminate *CAPBAK /MSW* by clicking on the **File** menu and selecting the **Exit** option.

After you click on the **Cbmsw** icon, *CAPBAK/MSW* loads. Depending on which version of Windows you are using, one of the following screens will appear:

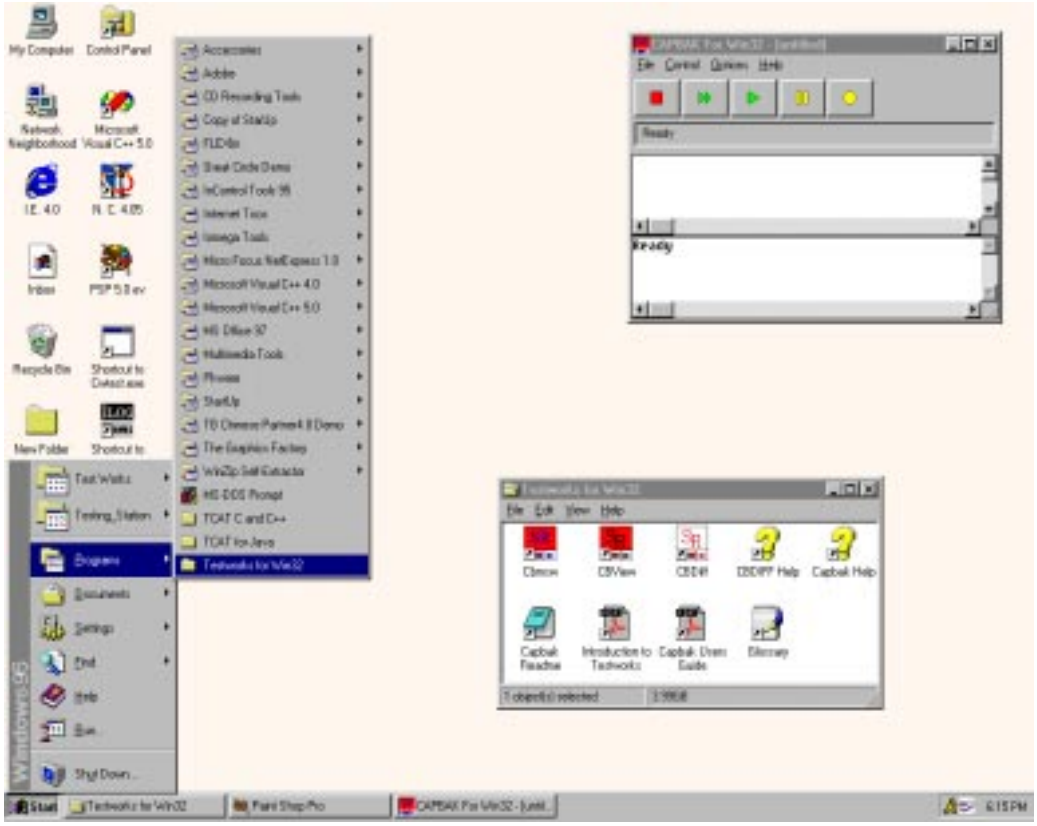


FIGURE 6 Invoking CAPBAK/MSW (Windows 95 and NT 4.0)

3.1.2 STEP 2: Starting a Recording

To start a recording session. Here's how:

1. Click on the control panel **Record** button labeled (red rectangle).
2. The **Record /Play**'s status window indicates that recording has started with the following message: **"Recording in Object Mode"**
3. You are now in recording mode. Any keyboard and mouse input you enter is recorded.
4. For this session, we will use the commonly available Microsoft Windows calculator (as described in **STEP 3**).

NOTE: You can stop recording anytime, then overwrite the old keysave file when you try recording a new session again.

When initiating a recording test session, your display should look like this.

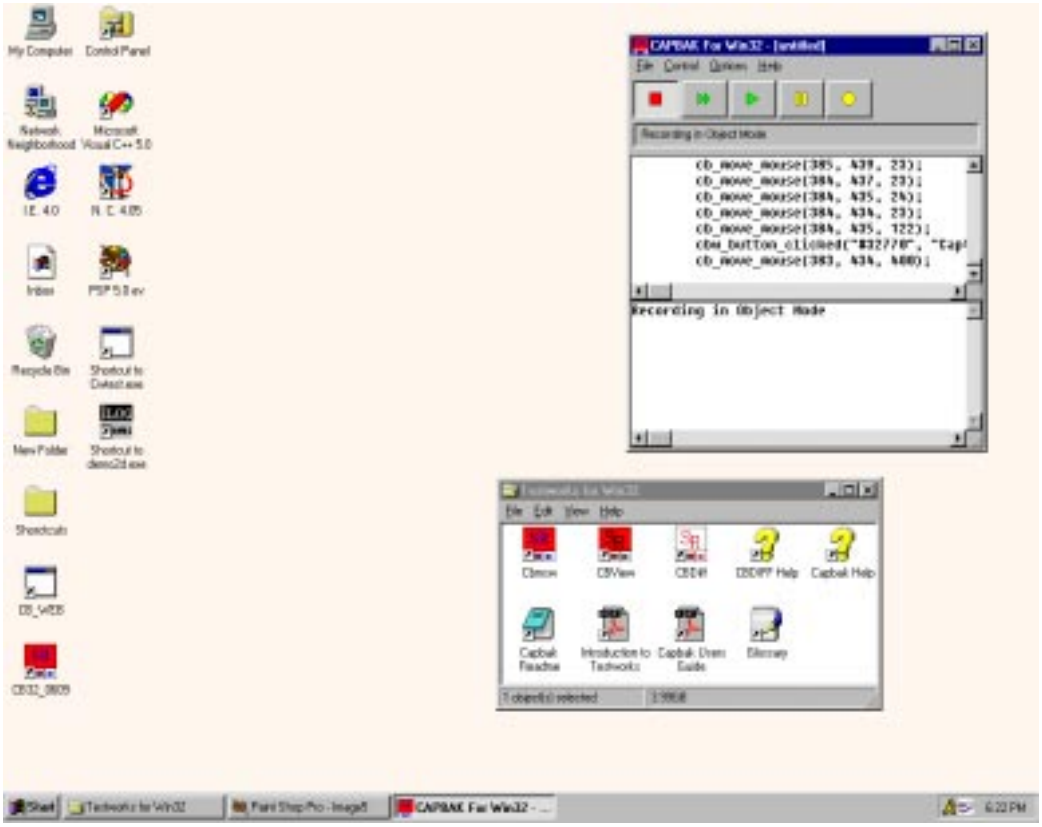


FIGURE 7 Beginning a Recording Session

3.1.3 STEP 3: Creating a Test

For this test session, try to make your recording last 5 minutes or less. Follow these steps:

1. Move the mouse around.
2. The calculator is normally located in the **Accessories Folder** window.
3. Click on the **Programs** from **Start** menu. Then click on **Calculator** from **Accessories Folder**.
4. The **Calculator** window pops up.
5. Use the calculator in a simple example, such as adding or multiplying two numbers.

The screen display of the calculator looks like this:

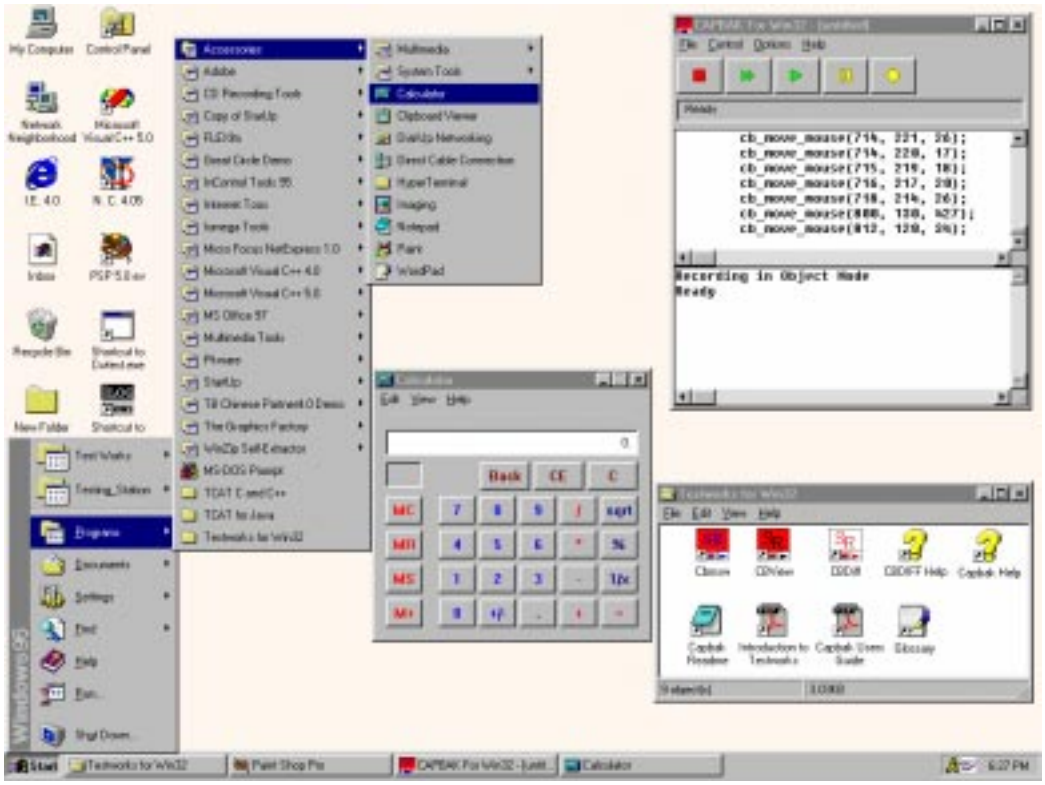


FIGURE 8 Creating a Test

3.1.4 STEP 4: Capturing a Window Image

During a test session, you have the option to command *CAPBAK /MSW* to place comments in the keysave file, define windows or images for synchronization during playback, or capture windows, images or the entire screen. These activities can be performed by either using options from the **Record/Play**'s **Hotkey** window or by using your keyboard's function keys.

During this test session, you are going to capture the **Calculator**'s window with the mouse. These instructions show you how to capture a window using either the **Hotkey** window or the function keys.

Using the **Hotkey** window:

1. Press and hold the **Ctrl**, **Alt** and **F1** function keys all at the same time when you are ready to capture the calculator window, then lift them up. Recording pauses and the **Hotkey** window pops up. The **Record/Play** window indicates recording has paused with the message: `Pausing. Recording Suspended.`
2. Make sure no part of the **Calculator** window is hidden by another window.
3. Click on the **Hotkey** window's **Save Window** button. The mouse pointer turns into a cross-hair symbol.
4. Move the cross-hair to the **Calculator** window, then click and release the mouse button. Be sure not to do this on top of any of the calculator's buttons.
5. The image is captured as <**name**>*.b01* in your working directory.

Using **Function** Keys:

1. Make sure no part of the **Calculator** window is hidden by another window, before attempting to capture it.
2. Move the mouse pointer over the **Calculator** window.
3. Press the **Ctrl**, **Alt** and **F6** function key all at the same time, then lift them up. Recording pauses as the image is being captured and the status window signals the window capture with the following message: `Saving window image in DIB file`
4. The image is captured as <**name**>*.b01* in your working directory.

   test.b01 serves as your baseline (image captured during the recording session) file.

When saving a window with the **Hotkey** window, your display should look like this:

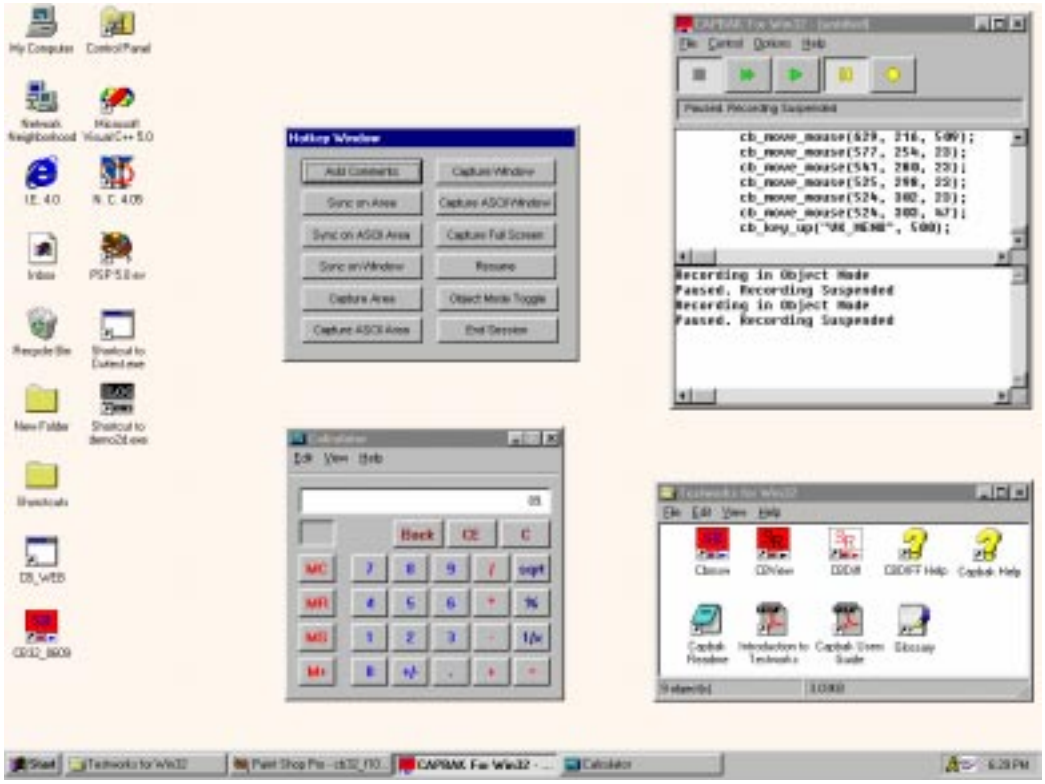


FIGURE 9 Capturing a Window

3.1.5 STEP 5: Completing a Recording Session

When you end a recording session, you should close all the applications that you opened, so it's easier for playback to begin from the same initial state that the recording session did. Depending on how you saved the calculator window, you can end your session one of two ways:

Using the **Hotkey** window:

1. If you captured the calculator using the **Hotkey** window, the calculator should still be displayed on your screen.
2. Click on the **Resume** button. The **Record/Play** window indicates recording is resumed with a `Recording` message in the status window.
3. Click on the **Calculator** window's control menu bar, then select **Close**.
4. Leave the **Record/Play** window up, as you will need it to later play back this test session.
5. Click on the **STOP** button. The **Record /Play**'s status window signals the end of a recording session with the message: `Ready`

Using the **Functions** Keys:

1. When you are ready to stop recording, press the **Ctrl**, **Alt** and **F10** function keys all at the same time and then lift them up, or you can press the **STOP** button on the **Record/Play** window.
2. Either way, the **Record /Play**'s status window signals the end of a recording session with the message: `Ready`

For Both Methods:

This completes the recording part of this tutorial. What you just did was establish a baseline of expected behavior of the **Calculator**. In terms of the **Calculato**r, this isn't much of a test. In real-life software testing, however, this type of test can be useful in finding any differences between two versions of an application program. This test can serve as one of many tests of the **Calculator**.

If a change is made is to the **Calculator**, you can play back the recording to verify the **Calculator** still works as expected.

NOTE: In most application testing situations, you will create more than one test. You can use *SMARTS* to organize your tests. Please refer to the *SMARTS User's Guide* for further information.

At the end of a recording session, your display should look like this:

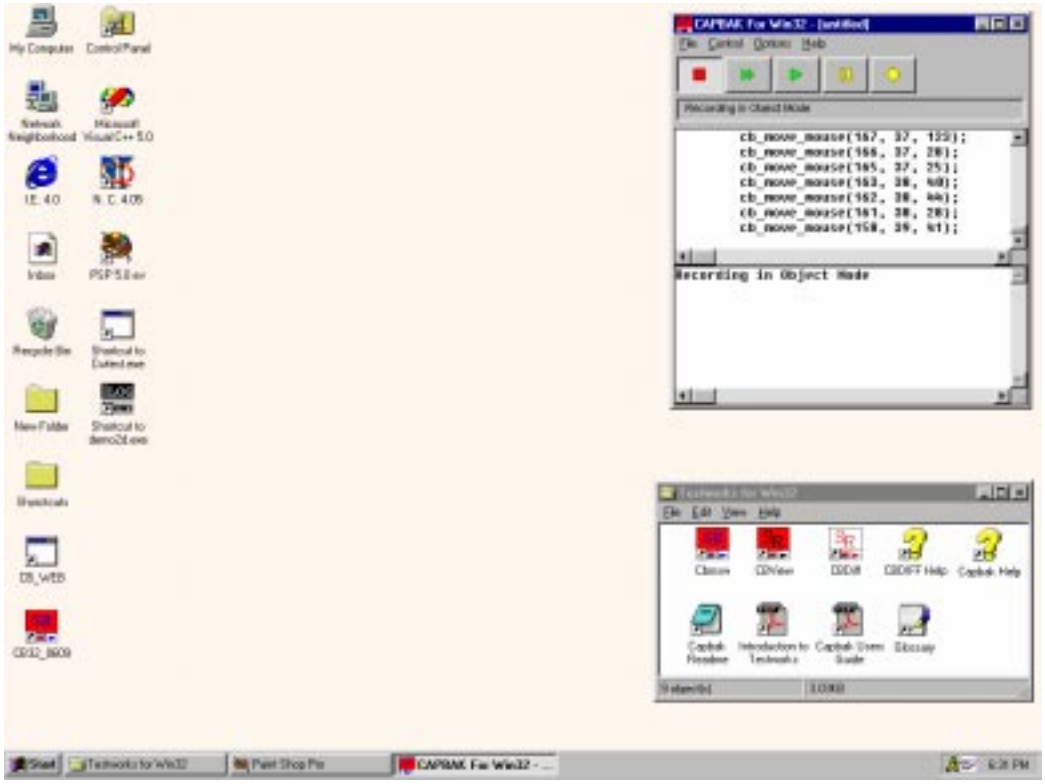


FIGURE 10 Completing a Recording Session

3.1.6 STEP 6: Saving a Keysave File

After recording a test session, use **Save As** to specify a keysave file name in which to save the test session inputs. These include all mouse, keystrokes, and any screen capture input statements. In other words, it represents your test session. Please refer to Appendix A "UNDERSTANDING THE KEYSAVE FILE LANGUAGE" for further information on keysave files.

Here's how to save a keysave file:

1. Click on the **File** menu.
2. Select the **Save/Save As** submenu. A file selection dialog box pops up.
3. Type in `test.ksv` in the **File Name** entry box. The ***.ksv** extension (as listed in the **List Files of Type** box) distinguishes keysave files from other types of files.

NOTE: When saving a keysave file, the first character must be alphabetic and *not* numeric.

---

4. Click on **OK** or press the **Enter** key, after you type in the file name.

When selecting a keysave file, your display should look like this:

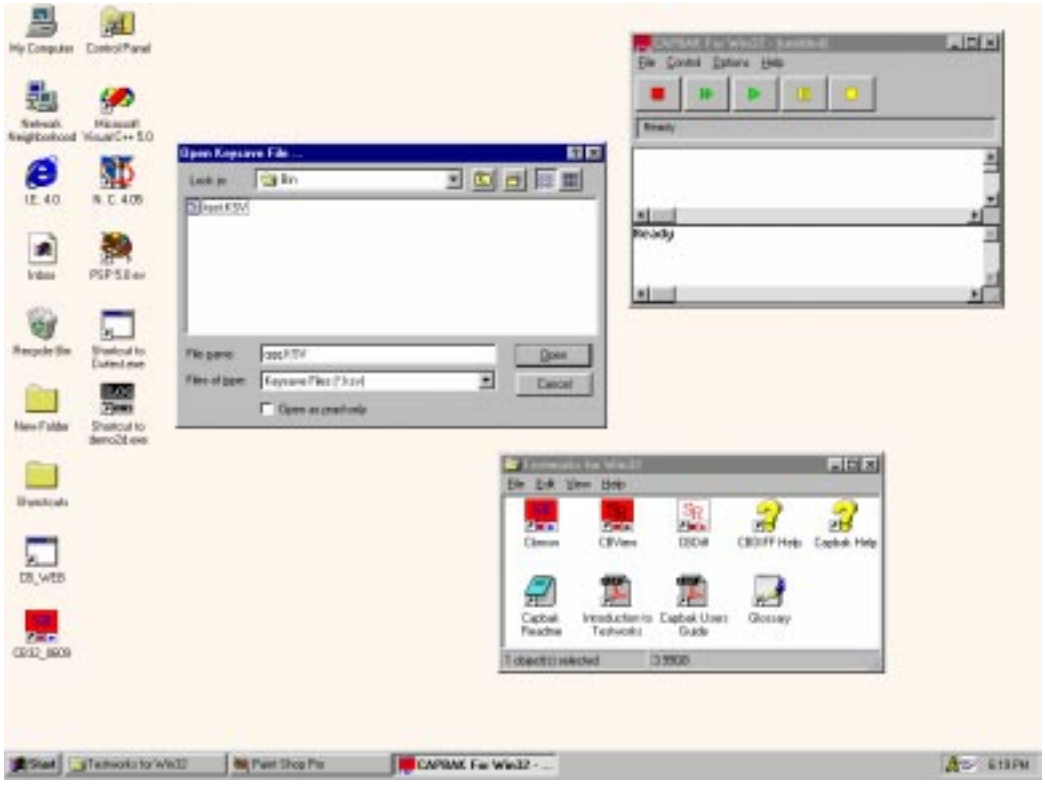


FIGURE 11 Saving a Keysave File

3.1.7 STEP 7: Initializing the CBVIEW Window

You can verify your recording test session's success by looking at the **Calculator** window image you saved during the recording session with the **CBVIEW** window. To display it:

1. Click on the **CBVIEW** icon in the **TestWorks Group** window.
2. The **CBVIEW** window pops up.

When the **CBVIEW** window opens, your display should look like this:

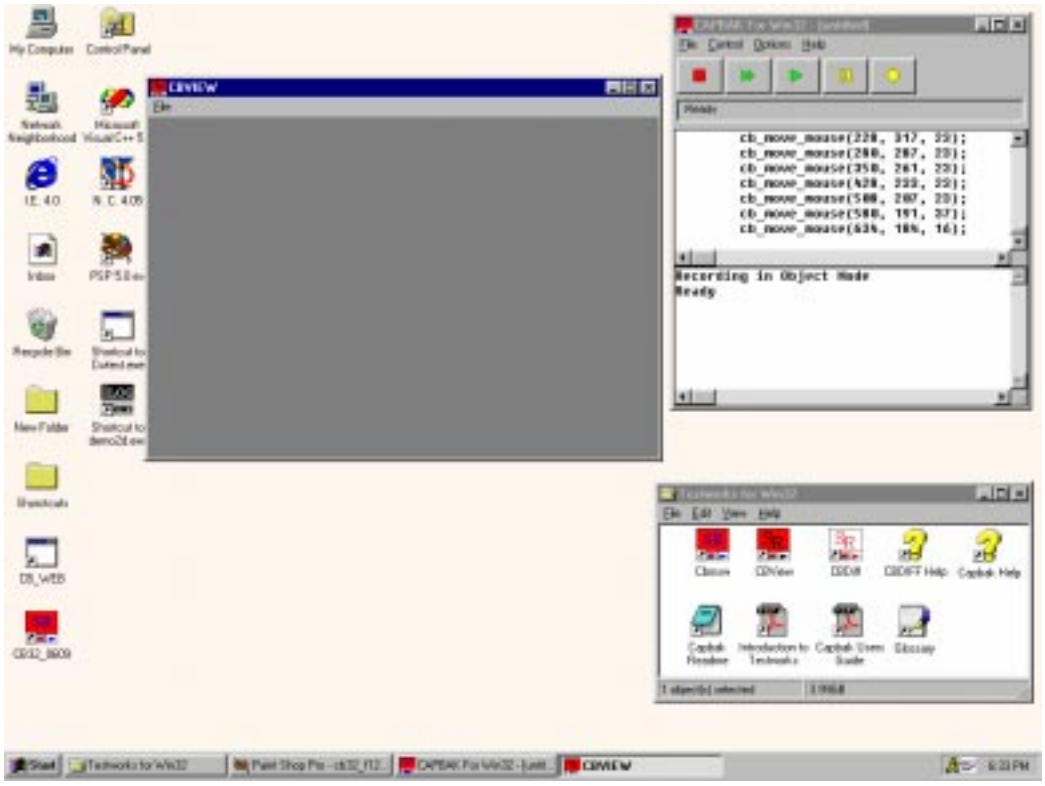


FIGURE 12 Initializing the CBVIEW Window

3.1.8 STEP 8: Displaying a Captured Image

To look at the captured image:

1. Click on the **File** menu and select the **Open** submenu. A file selection dialog box pops up.
2. Double click on *test.b01* in the file list box. You can also highlight or type in the file name and then click on **OK** or press the **Enter** key.
3. The saved **Calculator** window automatically appears in the **CBVIEW** window.
4. When you are finished viewing the image, click on the **File** menu and choose the **Exit** option.

When you display the saved **Calculator** window, your display should look like this:

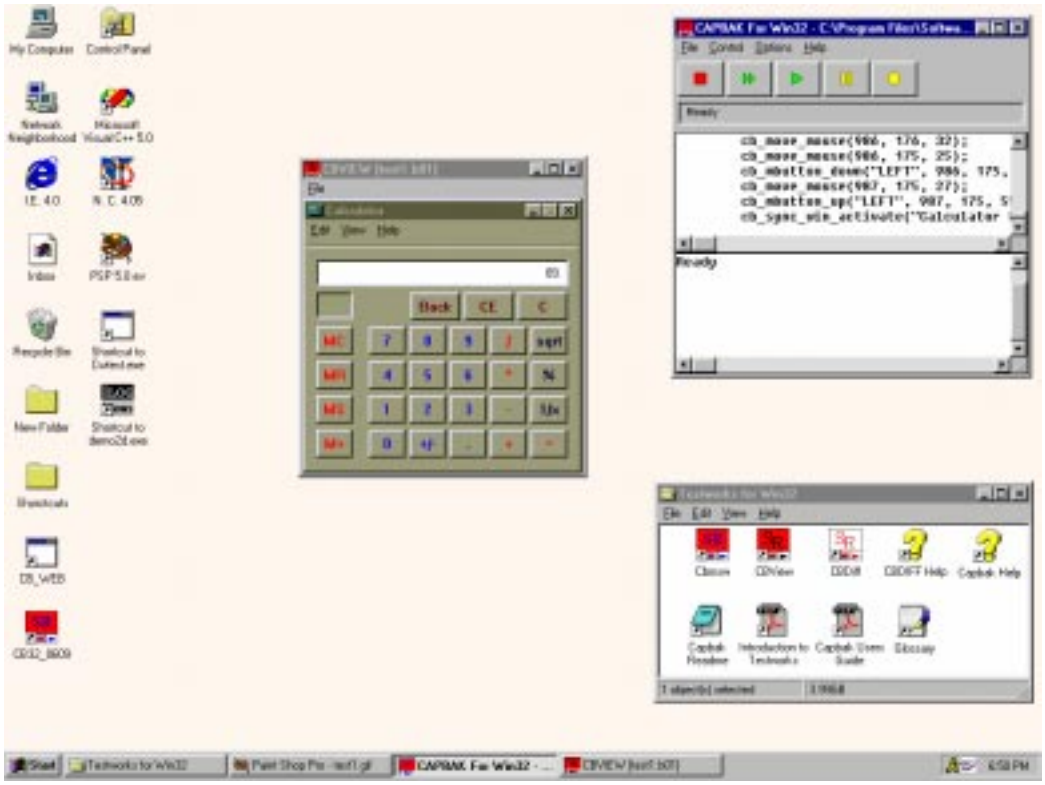


FIGURE 13 Viewing a Saved Image

3.1.9 STEP 9: Playing Back the Keysave File

Let's pretend we get an updated version of the software application, which we have to test. We can play back the recorded test session to verify the operation of the application.

Follow these steps to play back a keysave file:

1. Click on the **Record/Play** window's **PLAY** button.
2. Playback begins and the status window reads: `Playing back`
3. Mouse movements play back exactly as you entered them during the recording test session.
4. As the keysave file is playing back, do not try to move the mouse or enter keystrokes, as this may interfere with the application-under-test.
5. Watch as the **Calculator** window appears. You might see it jump to the location it popped up at during the recording session. This is *CAPBAK/MSW*'s automatic event synchronization.
6. Watch the **Calculator** window as it is captured. The image you captured during your recording test session is automatically recaptured as a response file during playback.
7. Playback ends when the status window reads: `Ready`
8. Click on the **Record/Play** window's **File** menu, then select **Close**.

At the end of a playback test session, your display should look like this:

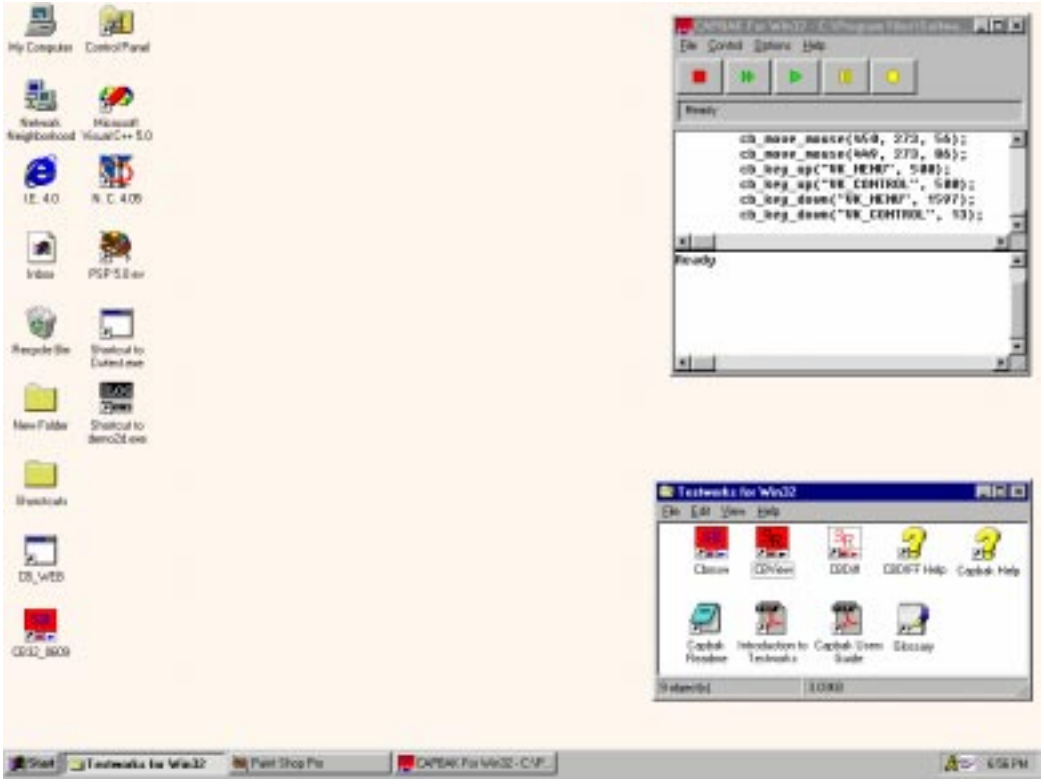


FIGURE 14 Ending a Playback Session

3.1.10 STEP 10: Reviewing Test Results

When you have completed the preceding nine steps, you should have three files in the working directory:

- A keysave file named *test.ksv*.
- A baseline image file named *test.b01*. This is the baseline image you captured during the recording session.
- A response image file named *test.r01*. This is the response image captured during the playback session.

test.b01 and *test.r01* image files should be identical to one another. You can either view these images with the **CBVIEW** window or you can compare them for differences (see **STEP 11**). To view them:

1. Click on the **CBVIEW** icon in the **STW Group** window to invoke the **CBVIEW** window.
2. First view the baseline image file *test.b01* just as you did in **STEP 8** and then the response image file *test.r01*.
3. If they look the same, they are most likely identical, which indicates a successful playback. You would need to use the **CBDIFF** utility to be completely sure that no differences exist.
4. Click on the **File** menu and select **Exit** to close the **CBVIEW** window.

When you view the response image file, your display should look like this:

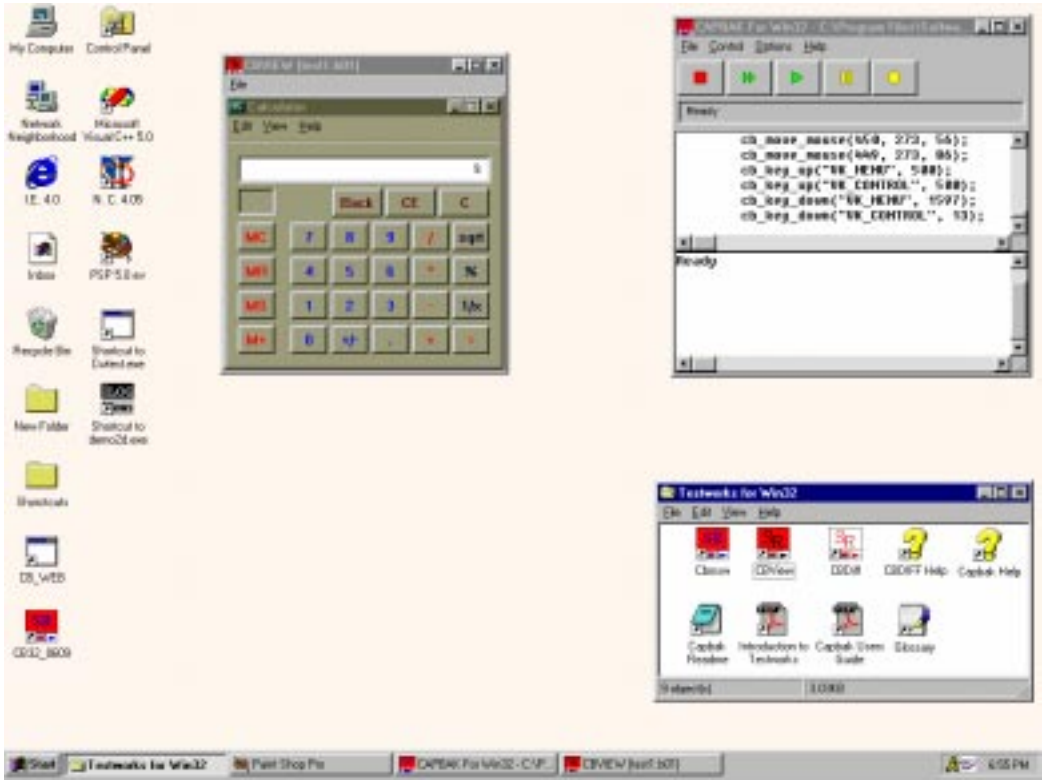


FIGURE 15 Verifying Playback

3.1.11 STEP 11: Comparing Saved Images

CAPBAK/MSW can compare differences between two saved images with its **CBDIFF** utility.

1. Click on the **CBDIFF** icon in the **TestWorks Group** window to invoke the **CBDIFF** window.
2. First, select the baseline image file *test.b01*. Click on the **File** menu and select the **Open Baseline** submenu.
3. When the file selection dialog box pops up, choose *test.b01*. The image is automatically displayed in the **CBDIFF** window.
4. Next, choose the response image file *test.r01*. Click on the **File** menu and select the **Open Response** submenu.
5. When the file selection dialog box pops up, choose *test.r01*. The image is automatically displayed in the **CBDIFF** window.
6. To compare the images: Click on the **Difference** menu and select the **Do Diff** option.
7. If both images match, then the **CBDIFF** window should be white; if the images do not match, then the **CBDIFF** window will display those bitmap differences.
8. When you are finished, click on the **File** menu and select the **Exit** option. The window closes.
9. Iconize the **TestWorks Group** window by clicking on its control menu box and selecting **Close**.

NOTE: *CAPBAK/MSW* can also verify if expected and actual images match on-the-fly during playback with its *Quick Check* mode. Please see Section 7.3 and Section 7.4 on user manual for complete information.

When you compare baseline and response image files, your display should look like this:

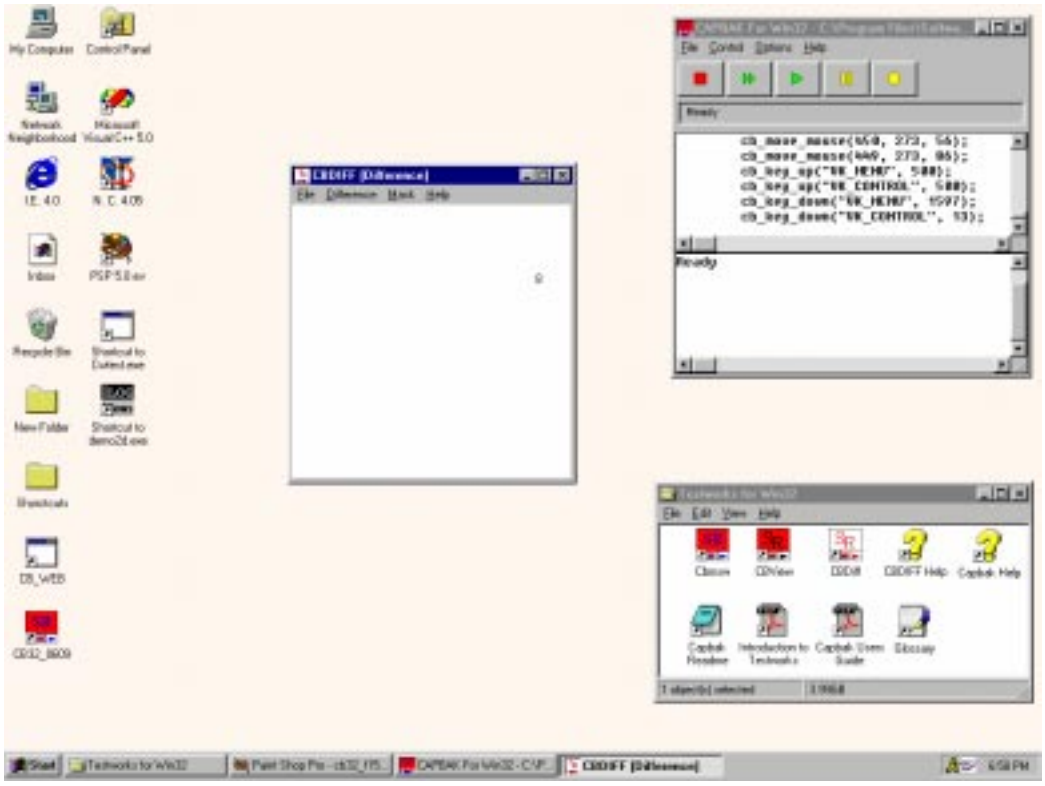


FIGURE 16 Comparing Images

3.2 Summary

If you successfully completed the preceding eleven steps, you've seen and practiced the basic skills you need to use *CAPBAK /MSW* productively. In this chapter you should have learned how to invoke *CAPBAK/MSW*, open a keysave file, record a test, capture a window, view the captured window, and play back the test, and, finally, verify a test.

- At this point, you could:
- Repeat **STEPS 1 - 11** without the manual.
- Repeat **STEPS 1 - 11** with your application.
- Turn to the chapters 4-9 on user manual to learn about additional features.
- Begin testing if you feel comfortable with *CAPBAK /MSW*.

CHAPTER 4

Understanding the User Interface

This chapter summarizes *CAPBAK*'s windows, menus and commands. Detailed description of individual commands is found in the relevant chapters of this guide.

4.1 Basic MS-Windows User Interface

This section demonstrates using file selection dialog boxes, help menus, message dialog boxes, option menus, and pull-down menus. If you are familiar with the basic MS-Windows graphical user interface (GUI) style, you can go on to Section 4.2.

4.1.1 File Selection Windows

CAPBAK/MSW's file selection windows allow you to select or specify test file names or select saved image files.

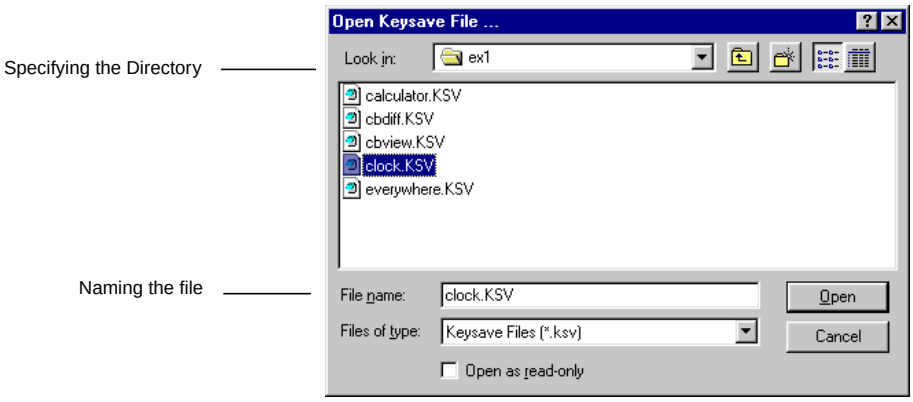


FIGURE 17 File Selection Window

File Name entry box

Selects and enters a file name.

Files of Type Specifies which files are listed in the **File Name** area. The current type of file and its extension are displayed.

Look in list box Lists directories in path defined in the **Filter** entry box. Use it to locate the desired directory.

Use the two push buttons at the right of the dialog box to issue commands:

Open Accepts the directory and file in the **File Name** entry box as the new file or the file to be opened and then exits the dialog box.

Cancel Cancels any selections made and then exits the dialog box. No file is selected as a result.

To use a file selection dialog box:

1. Click the directory name where an existing file is located or where you want a new file to be placed.
2. Select an already existing file name in the **File Name** list box or type in the name of the new file name in the **File Name** entry box, with no limit on character length.

Note: The convention for naming test files, or keysave files, is *basename.ksv*, where *ksv* represents a keysave file. Captured images take the form of *basename.bxx*, *basename.sxx*, *basename.rxx*, where *b* represents a baseline image, *s* identifies a image captured for synchronization, *r* represents a response file, and *xx* represents the original sequence in which the image was captured.

---

3. To select a keysave file name, do one of these three things:
   - Double click on the file in the **File Name** list box.
   - Highlight the file in the **File Name** list box or type in the file name in the **File Name** entry box and click on **OK**.
   - Highlight or type in the file name and press the **Enter** key.

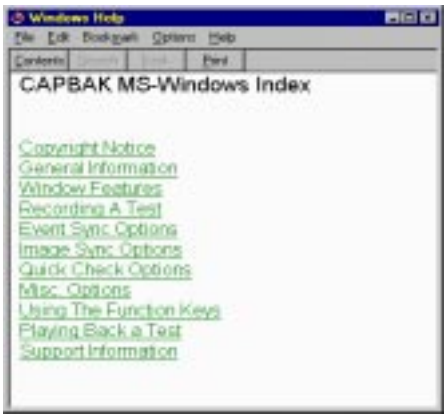
4.1.2 Help Windows

CAPBAK/MSW 's help is available for the **Record/Play** window and the **CBDIFF** windows. Its on-line help automatically brings up the pertinent text corresponding to the topic you choose.

Here's how to use the help.

1. Click on the **Help** menu.
2. Select the **CBMSW Help** submenu if you need help for the **Record/Play** window or the **CBDIFF Help** submenu for the **CBDIFF** window.
3. The **Help** window pops up with the contents of the help information.
4. Simply click on the topic you want information for and the **Help** window automatically displays it.

Help: If this is the first time you've used on-line help, you might want to choose **How To Use Help** from the **Help** menu. You can also refer to your *Microsoft Windows User's Guide* for complete information on using **Help** menus.



Click on the desired topic

FIGURE 18 Help Window

4.1.3 Pull-Down Menus

Pull-down menus are located within the menu bar of *CAPBAK/MSW*'s windows. They often contain several options. To use pull-down menus and their options, follow these steps:

1. Move the mouse pointer to the menu bar and over the menu containing the item.
2. Hold the left mouse button down. This displays the items on the menu.
3. While holding down the left mouse button, slide the mouse pointer to the menu item you want to select. The menu item is highlighted in reverse shadow.

   An ellipse (...) following an option indicates that selecting the item will bring up a pop-up window, such as a file selection window.

   A dimmed (not visible) option indicates that you may not be able to use the option with your application at the current time. For example, you may need to select another item before using this command.
4. To choose an item from a selected menu, click the item, or type the letter that is underlined in the item name (like the "E" in "Event Sync Options..." below), or use the arrow keys until you reach the item you want to select, and then press the **Enter** key.

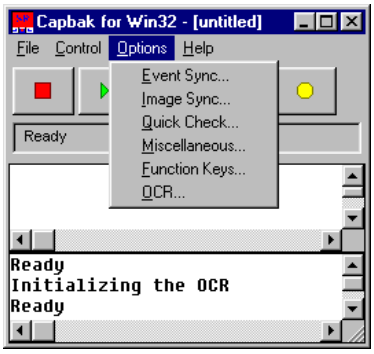


FIGURE 19 Pull-Down Menu

4.2 The Record/Play Window

The **Record/Play** window captures and replays test sessions.

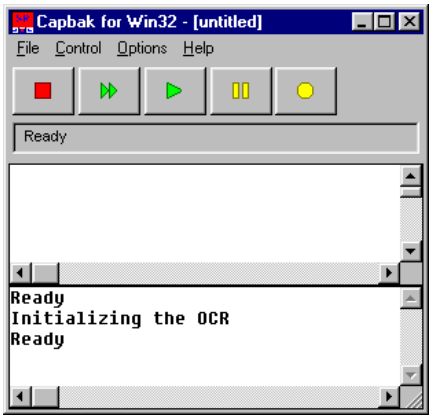


FIGURE 20 Record/Play Window

Each of the pull-down menus and commands are briefly described next.

4.2.1 File Menu

Open File: Opens up a file selection dialog box. There you can name a new keysave file or open up an existing one for the purposes of capture and playback. You should name all of your keysave files with a *.ksv* extension.

CAPBAK/MSW automatically adds a *.ksv* extension to keysave file names that have no extension. It replaces any extension other than *.ksv* with a *.ksv*. This means that all keysave file names have the *.ksv* extension.

Exit: Closes *CAPBAK/MSW*.

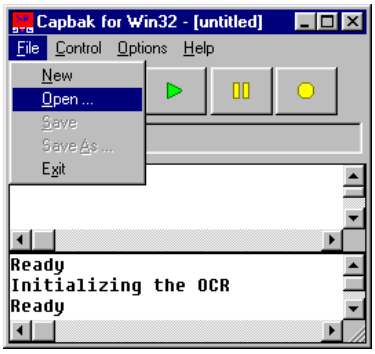


FIGURE 21 File Menu

4.2.2 Options Menu

The **Options** menu sets the record and playback options.

Event Sync Options: Invokes the **Event Sync Options** window. Please refer to Section 4.2.2.1 on page 49 for further information.

Image Sync Options: Invokes the **Image Sync Options** window. Please refer to Section 4.2.2.2 on page 50 for further information.

Quick Check Options: Invokes the **Quick Check Options** window. Please refer to Section 4.2.2.3 on page 51 for further information.

Misc. Options: Invokes the **Misc. Capbak Options** window. Please refer to Section 4.2.2.4 on page 52 for further information.

Function Keys: Invokes the **Function Keys Settings** window. Please refer to section 4.2.2.5 on page 54 for further information.

OCR Options: Invokes the **OCR Options** window. Please refer to Section 4.2.2.6 on page 55 for further information.

NOTE: Changes made to any of *CAPBAK/MSW*'s options are not saved between invocations of *CAPBAK/MSW*. To make permanent changes to options, please refer to Appendix B, "CUSTOMIZING CAPBAK/MSW", for instructions on editing the *cbmsw.ini* file.

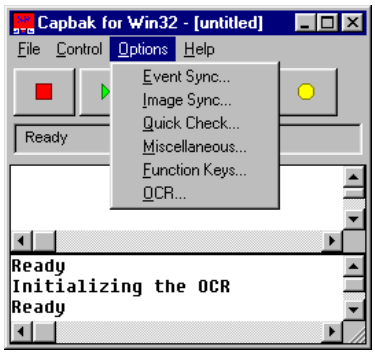


FIGURE 22 Options Menu

4.2.2.1 Event Sync Options Window

In the MS-Windows environment, windows are likely to come up at varying locations on the screen during successive playbacks. The **Event Sync Options** window's options are for synchronizing on window pop-ups:

- **Event Sync:** Defines the maximum time-out period that event synchronization, which controls synchronization on window pop-ups, will "wait" before continuing. The default is 10 seconds. If playback fails to synchronize in that time, playback continues unless the **Quit on Event Sync Error** option is turned on.

  During a playback session, this option modifies the automatically recorded `cb_sync_win_activate` statements in the keysave file.
- **Quit on Event Sync Error:** Terminates the playback session, if output synchronization fails. **The Event Sync option must accompany this option.**
- **Move Window On Event Sync:** If a window pops up in a different place during playback than it did during the recording session, this option moves the window to the right location and resizes it to the original size. **The Event Sync option must accompany this option.**

NOTE: These event synchronization options only apply to playback. Each time a window pops up during a recording session, a `cb_sync_win_activate` statement is added to the keysave file.

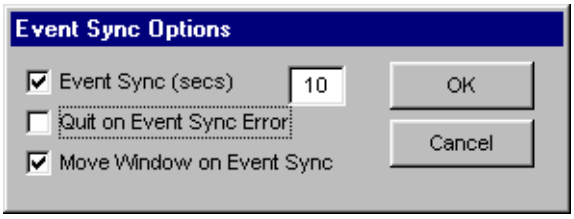


FIGURE 23 Event Sync Options Window

4.2.2.2 Image Sync Options Window

The image synchronization options apply to the playback of a test session that used the **Hotkey** window's **Sync Window** or **Wait on Area** buttons or the **F3** or **F4** function keys to define screen areas or entire windows for synchronization. *CAPBAK/MSW* pauses playback test execution until the image or window is fully displayed.

- **Image Sync:** Defines the time-out period that playback waits for a specified image or window to match a recording's drawn image. The default is 25 seconds. Unless the **Quit on Image Sync** is turned on, playback proceeds even if the image is not found. **This option must be turned on for playback image synchronization to take place.**
- **Quit on Image Sync Error:** Terminates the playback session, if image synchronization fails. **The Image Sync option must accompany this option.**
- **Move Window on Image Sync:** If an image matches but is in a different place during playback than it was during the recording session, this option moves the image to the right location. **The Image Sync option must accompany this option.**

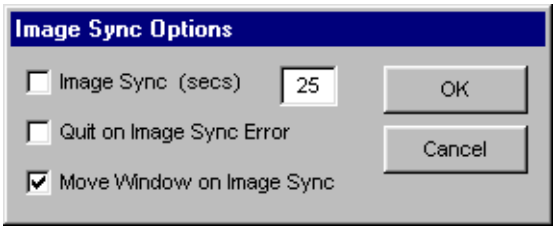


FIGURE 24 Image Sync Options Window

4.2.2.3 Quick Check Options Window

The **Quick Check Option** window's options are for the automatic comparison of baseline and response images during playback, that is playback's actual results are compared to recording's expected results. Please refer to Section 7.4 on page 96 for complete details.

- **Difference Images with Baseline:** Automatically compares response images (images captured during playback) to their corresponding baseline images (images captured during the initial recording) during playback.

  This option must be turned on in order for the other *Quick Check* options to work.
- **Quit on Number of Diff Errors:** Terminates playback if the number of times a response image fails to match a baseline image reaches the number specified. The default is 5.
- **Difference Report:** Creates a pass/fail results report of expected image and actual image comparisons. The default is *report.txt*.

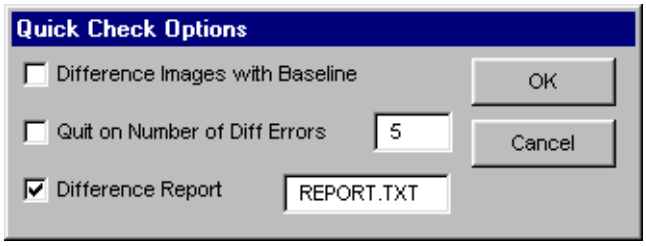


FIGURE 25 Quick Check Options Window

4.2.2.4 Misc. Capbak Options Window

This window sets various record / playback features.

- **Hotkey Win Position:** Sets the X,Y coordinates of where the **Hotkey** window will pop up during a recording session. The default is set to 0,0. The **Hotkey** window allows you to do such things as place comments in the keysave file, save images, or terminate a test. You must turn this switch on if you want a different pop-up location to apply other than 0.0.
- **Delay Multiplier:** Quickens or slows down the pace of playback. The default is set to 100 which represents real time. The higher the number, the slower playback is; the lower the number, the faster playback is. If the delay multiplier is set to a very fast speed, synchronization may be sacrificed. Every system varies so you may have to experiment with speeds. You must turn this option on if you want a different speed to apply other than 100.
- **Include Window Frame:** Includes a window's frame when a window is captured.
- **Regenerate Baseline Images:** Saves images captured during playback to baseline image files instead of response image files. This feature can be very effective if the outputs of your application change, and you want to use the new outputs as baseline images.
- **Compress Images:** Uses internal Window procedures to automatically compress images by more than 70 percent when saving them to a DIB (Device Independent Bitmap) file.

NOTE: Unfortunately, some graphic cards and device drivers have incompatabilities in them that prevent the use of image compression. If you have saved an image, but it is blank when you look at it using the **CBVIEW** or **CBDIFF** utilities, then your card and device driver probably have one of these limitations. Turn this option to off and recapture the image. The image should now appear normal.

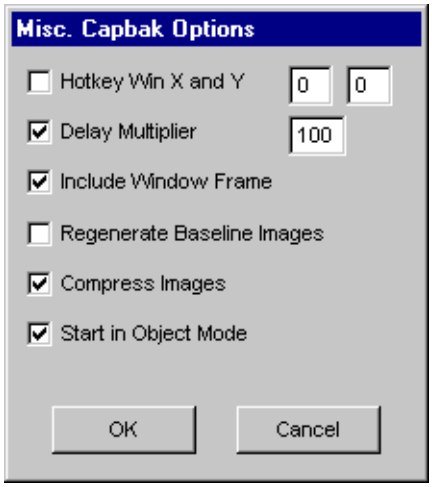


FIGURE 26 Misc. Capbak Options Window

4.2.2.5 Function Keys Options Settings

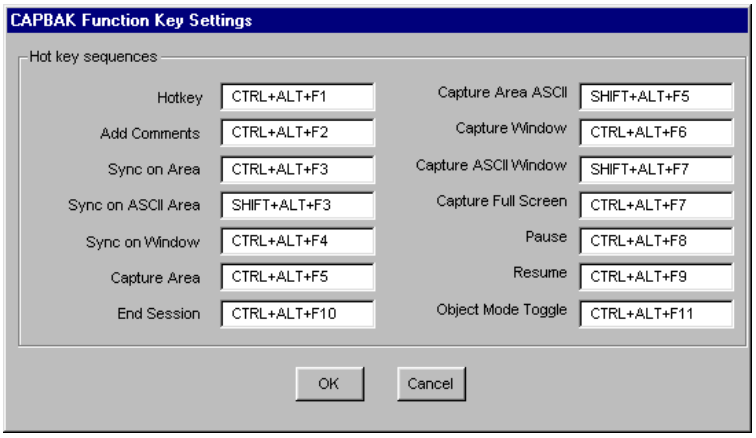


FIGURE 27 Function Keys Settings Pop-Up Window

This window allows you to set the key sequence for using *CAPBAK/MSW* function keys. A string describing a key sequence is made up of three key names separated by plus signs. They first two key names must either be **SHIFT**, **CTRL** or **ALT**. The last key name must be one of the 12 function keys -- **F1** through **F12**. The way the function keys are used in recording a test is described in Sections 6.4.2 on page 77 thru 6.4.12 on page 90.

4.2.2.6 OCR Options Settings

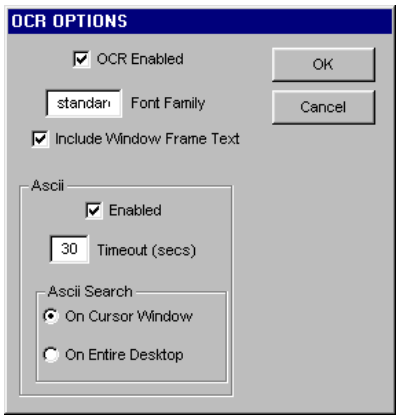


FIGURE 28 OCR Options Settings Window

This window allows you to set the options for the *CAPBAK/MSW* OCR utility. Please refer to Section 8.4.2 on page 123 for information on these settings.

4.2.3 Help Button

CBMSW Help: Brings up the **Help** window for the **Record/Play** window.

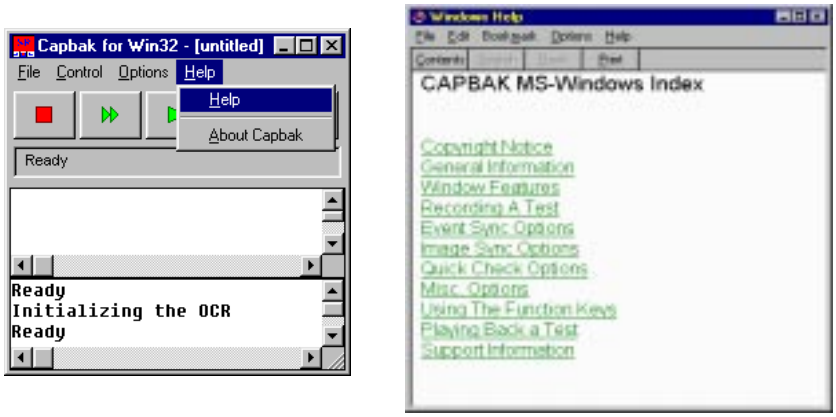


FIGURE 29 Help Window

4.2.4 Control Panel Features

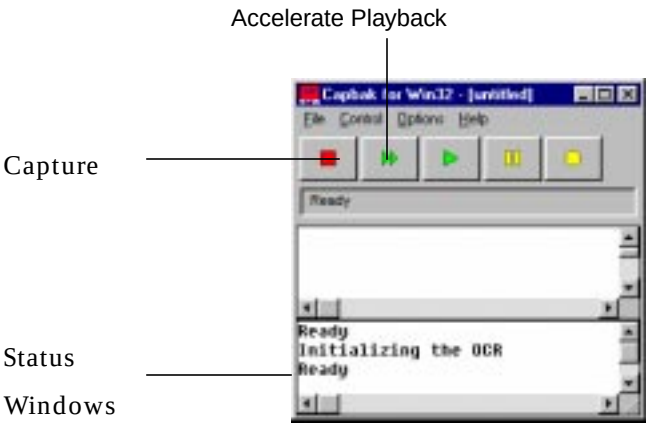


FIGURE 30 Control Panel Features

The window has the following control panel features:

- **REC:** Starts a recording session.
- **APP:** Appends to the end of an existing keysave file with new keysave records.
- **FF:** Increases a keysave file's playback speed by 10 percent (Delay = 90) for each time it is pressed *prior* to playback. Playback speed is determined by the **Delay Multiplier**'s setting in the **Misc. Capbak Options** window. (See Section 4.2.2.4 on page 52.) The default is 100 so playback occurs in real time. If this button is used once, the speed automatically adjusts to 90.

  The **Delay Multiplier** option must be *on* in order for this button to work.
- **PLAY:** Starts a playback session of the selected keysave file. If you use the **FF** button prior to a playback session, then the keysave file's speed will increase 10 percent each time the **FF** button was used.
- **PAUSE:** This button pauses a recording session. Because you do not have control over the mouse during playback, you cannot use this button during a playback session.
- **STOP:** This button terminates a recording session. Because you do not have control over the mouse during playback, you cannot use this button during a playback session.
- **Status Window:** Displays all messages during recording and playback.

4.3 CBVIEW Window

All functions for viewing captured images are accessible from this window. During a recording session full screen (root), window, and partial screen captures are automatically saved as *basename.bxx* and during a playback session they are saved as *basename.rxx*. Images saved for synchronization during playback are saved as *basename.sxx* during the recording session only and are meant to be matched during playback.

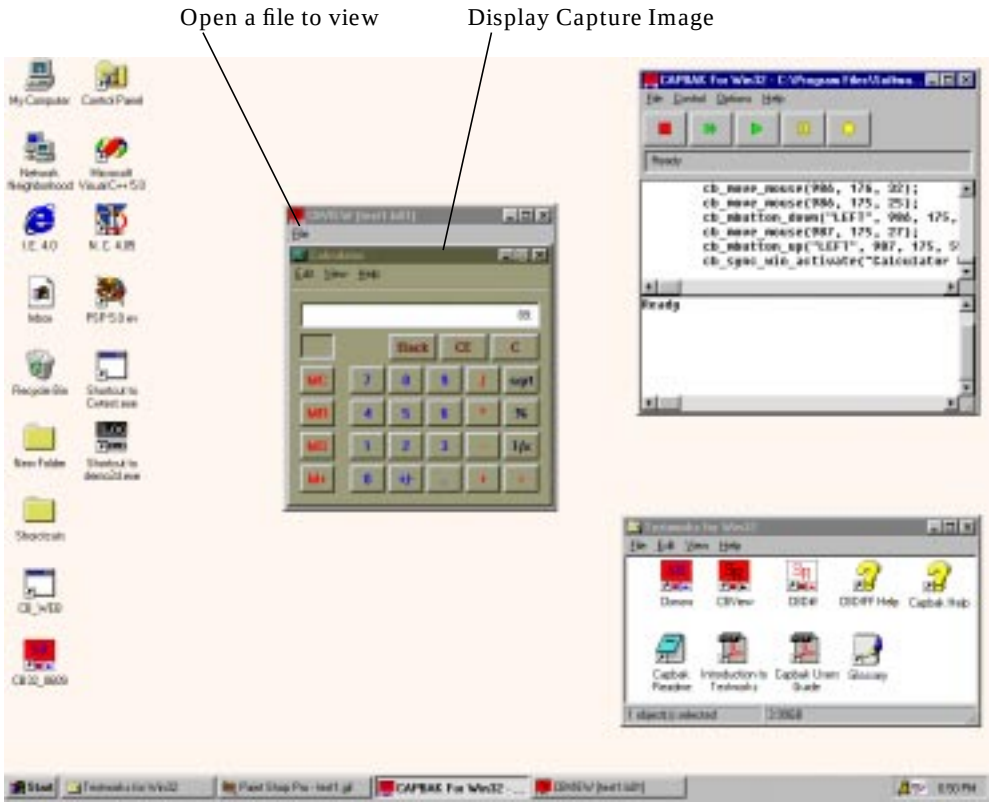


FIGURE 31 CBVIEW Window

4.3.1 File Menu

Open: Opens up a file selection dialog box. There you can choose an existing captured image file to view.

Exit: Closes the **CBVIEW** window.

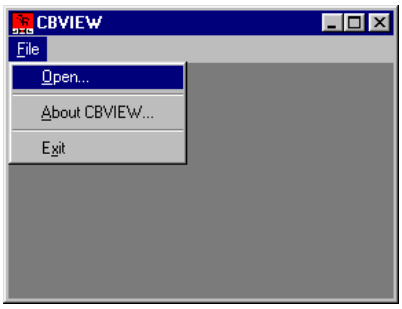


FIGURE 32 CBVIEW File Options Menu

4.4 CBDIFF Window

The **CBDIFF** window compares *CAPBAK/MSW*'s captured baseline and response files and permits masking of inconsequential areas.

CBDIFF is a quick way to determine if your test passed or failed, by determining if images captured during a recording session and images re-captured during a playback session are identical.

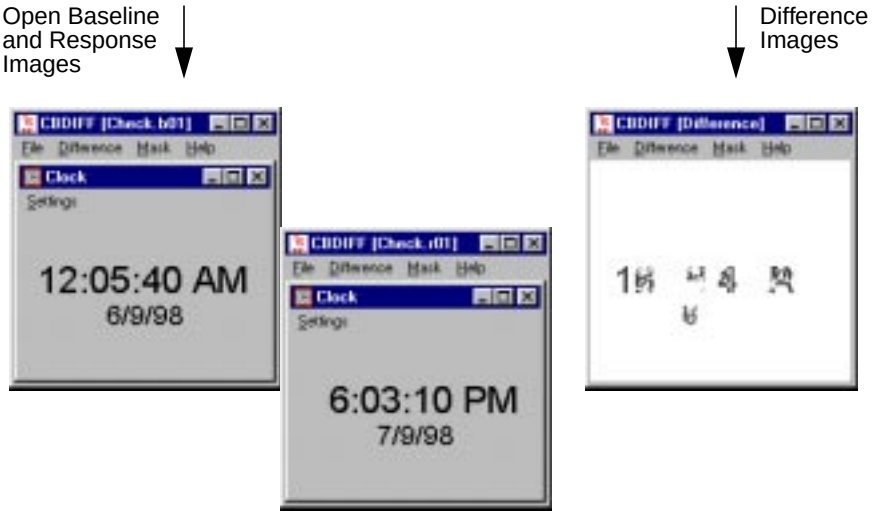


FIGURE 33 CBDIFF Window

4.4.1 File Menu

Open Baseline: Opens up a file selection dialog box where you can select a baseline file to open *(basename.b01, basename.b02* and so on) for comparison.

Open Response: Opens up a file selection dialog box where you can select a response file to open *(basename.r01, basename.r02* and so on) to compare with the baseline file you selected.

Exit: Closes the **CBDIFF** window.

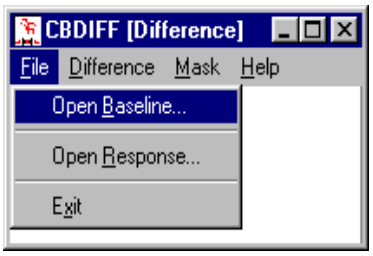


FIGURE 34 CBDIFF File Menu Window

4.4.2 Difference Menu

View Baseline: Displays the baseline file currently selected.

View Response: Displays the response file currently selected.

Do Diff: Does a comparison of the baseline and response files selected and shows the difference (if any) in **CBDIFF**'s display area.

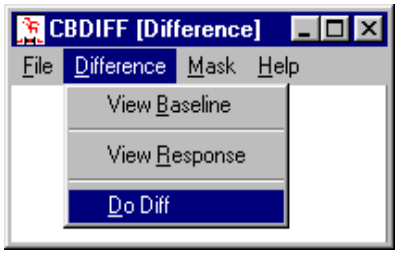


FIGURE 35 CBDIFF Difference Options Window

4.4.3 Mask Menu

Do Masking: Permits masking of a displayed image and usage of the **Read Mask File** and **Write Mask File** submenus.

Read Mask File: Opens up a file selection dialog box where you can select a saved mask file (*basename.mxx*) for an image prior to differencing.

Write Mask File: After masks are created, allows you to save mask coordinate information to a file by opening a file selection dialog box where a file can be named (*basename.mxx*).

FIGURE 36 CBDIFF Mask Options Window

4.4.4 Help Button

CBDIFF Help: Brings up the **Help** window for the **CBDIFF** window.

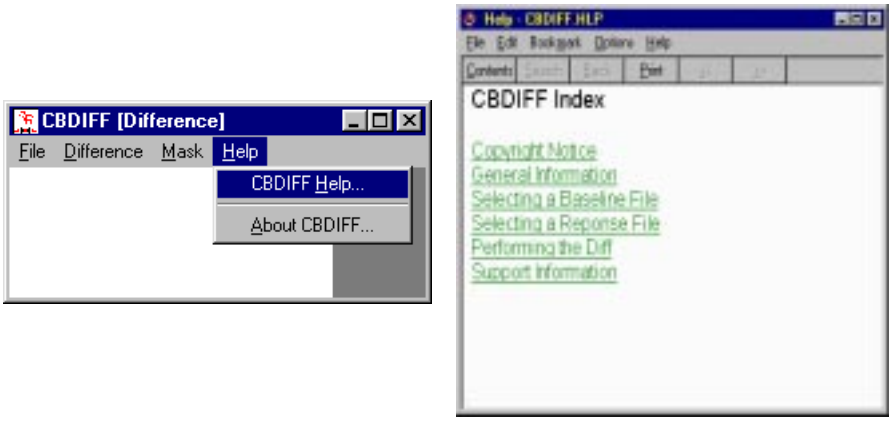


FIGURE 37 CBDIFF Help Options Window

CHAPTER 5

Invoking CAPBAK/MSW

5.1 Working Environment

Start *CAPBAK/MSW* only from Windows.

5.2 Invocation Procedure

The installation process should have added a TestWorks icon to the **Program Manager** window.

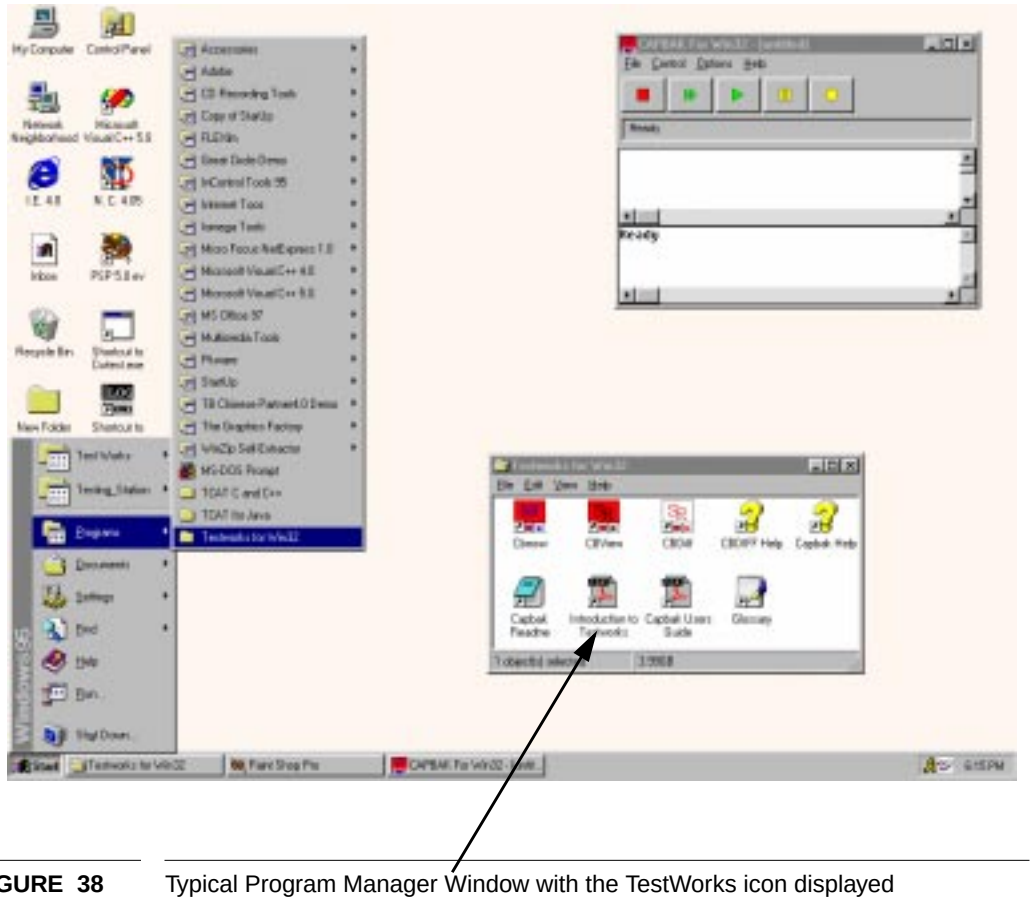


FIGURE 38 Typical Program Manager Window with the TestWorks icon displayed

Double-click on the **TestWorks** icon to open the **TestWorks Group** window.

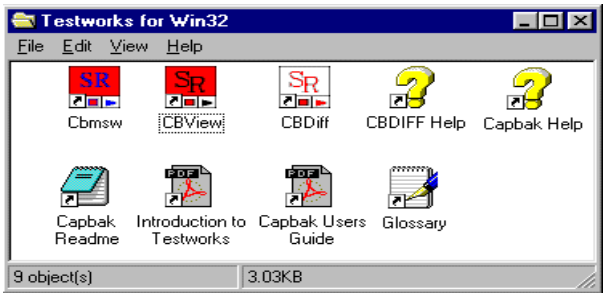


FIGURE 39 TestWorks Group Window

Each icon initiates a different *STW/Regression/MSW* utility.

Double-click on the **Cbmsw** (CAPBAK MS-Windows**)** icon to launch the record / playback utility. Please refer to Chapter 5 on page 65 and Chapter 6 on page 91 for complete information on its use.

Double-click on the **CBVIEW** icon to launch the image-displaying utility. Please refer to Chapter 8 on page 141 for complete information on its use.

Double-click on the **CBDIFF** icon to launch the image comparison utility. Please refer to Chapter 9 on page 129 for complete information on its use.

Double-click on the **SMARTS** icon to launch the test management and execution utility. Please refer to the *SMARTS/MSW* User's Manual for complete information on its use.

Double-click on the **Glossary** icon to call up a list of software testing terminology. Please refer to the *TestWorks for Windows* User's manual for more information.

CHAPTER 6

Recording a Test

This chapter explains how to plan, set up and record a test.

6.1 Introduction to Recording

Recording a test allows you to create automated regression tests. Regression tests are used to verify that product behavior continues to work even after changes have been made. In the process of recording a user's interactions with the application-under-test (AUT), *CAPBAK/MSW* creates a keysave file, or a test script file. This script can be played back at any time to determine whether the product still behaves as it did when the script was recorded.

6.2 Planning a Test Session

Prior to recording, it is important to decide what it is that you want to record. You should consider such things as the windows or images you may capture so you can compare the behavior of two versions of an application-under-test and to ensure synchronization of playback. The more you plan a test in advance, the greater the chance you will not have to re-record the test.

For the purposes of playback, the environment must be identical to the conditions in which the test was created. In other words, always make sure that the initial setups for both recording and playback are identical.

To ensure consistency between the recording and playback sessions, consider the following:

- If the application you are testing is menu-driven, then start recording each test from a well-defined point. In other words, close all applications not required to perform this test.
- If a test is completely closed, a new test should begin from the same starting point as the prior test.

6.3 Initiating the Record/Play Window

Initiate the **Record/Play** window by double-clicking on the **Capbak MS-Windows** icon in the **TestWorks Group** window. The **Record/Play** window pops up.

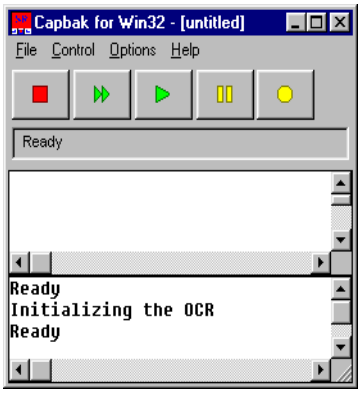


FIGURE 40 Record/Play Window

Before recording a test, you may want to familiarize yourself with the **Record/Play** window's control panel buttons that can be used during a recording session:

- **REC:** Starts a recording session. When a keysave file name is selected, click on this button to begin a session.
- **PAUSE:** Pauses a recording session. During recording, simply click on this button whenever you want to pause a session. You can easily resume the same session by pressing the **Ctl**, **Alt** and **F9** function keys simultaneously.
- **STOP:** Terminates a recording session. Click on this button whenever you want to stop recording.

6.3.1 Selecting Options for Recording

After naming a keysave file, you may want to turn on some options that can effect your recording session:

1. Click on the **Options** menu.
2. Select the **Misc. Options** submenu. (The **Event Sync Options**, **Image Sync Options**, and the **Quick Check Options** submenus supply playback options only.)
3. The **Misc. Capbak Options** window pops up. Please refer to Section 4.2.2.4 on page 52 for definitions of the options.
4. To select an option, simply click on the corresponding check button. To edit options with specification regions, position the mouse pointer so it is in the specification region and then click the mouse button. A cursor appears indicating you can then edit.

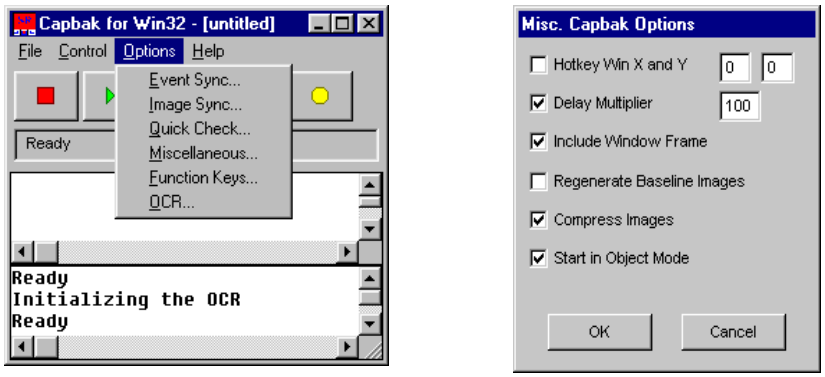


FIGURE 41 Selecting Options

6.4 Recording Steps

When you are ready to start recording a test session, keep these suggestions in mind:

- If you close the Window Explorer while recording, you may cause a GPF (General Protection Fault).
- When the automatic **Event Sync** option is on (which is the default), be careful when a window pops up.

Warning: Do not move the mouse or type in any keystrokes until 2 seconds after the window pops up and is completely redrawn. If you do, synchronization may not succeed during playback.

- When data lists are likely to change, use keystrokes to enter input instead of the mouse. For example, when opening a file, type in the file name rather than selecting the file from a list with the mouse.
- Avoid holding down the mouse in a repeated key action (for example, moving a window display using the scroll bar). Doing so can initiate a time-sensitive operation that cannot precisely be recreated.
- Avoid double mouse clicks when possible. When opening an icon, for instance, click and then use a carriage return. If you must use double clicks, be sure to decrease the speed between double clicks when you are playing back. See Section 7.1 on page 91 for more details.
- Look to the **Record/Play** window's status window for messages regarding prompts, warnings and system failures.
- When you capture images or windows, make sure they are not hidden or covered up by any other windows. Otherwise the image or window will not be captured properly.

To record a test session:

1. Click on the **RECORD** botton. Click on **APPEND** to append keysave records to the keysave file selected.
2. The status window signals the start of a recording session with: `Recording`
3. You are now in recording mode. Perform the test you have planned by directing input to your application using the keyboard and mouse.
4. Use *CAPBAK/MSW*'s function keys to:
   - activate the **Hotkey** pop-up window where you can use any of its options;
   - place comments in the keysave file; define an image for synchronization during playback
   - synchronize on a window
   - capture a partial image, a window or the full screen
   - pause the test session
   - resume the recording
   - end a recording session

   Use of these function keys is described next in Section 6.4.2 on page 77 through 6.4.12 on page 90.

5. If you don't want to use the function keys and are finished recording, you can terminate recording by pressing the button labeled **STOP** in the **Record/Play** window. The status window signals the end of the recording with the following message: `Ready`

6.4.1 Saving a Keysave File

After a recording session, use **Save As** to save a new test keysave file. This file serves as a script composed of the event statements generated automatically during recording in response to your input to the application-under-test. It contains all mouse, keystrokes, and screen capture events. If you would like to see what the keysave syntax is, please refer to Appendix A, "UNDERSTANDING THE KEYSAVE FILE LANGUAGE".

To save the keysave file, follow these steps:

1. Click on the **File** menu.
2. Select the **Save As** submenu. The file Save As dialog box pops up.
3. Type a name to save a new keysave file in the **File Name** entry box. The convention for naming keysave files is *basename.ksv,* where ***.ksv*** represents a keysave file. *CAPBAK/MSW* automatically adds a *.ksv* extension to keysave file names that have no extension. It replaces any extension other than *.ksv* with *.ksv.* This means that all keysave files have the *.ksv* extension.

   For further information on using the file selection dialog box, please refer to Section 4.1.1 on page 42, "File Selection Windows".

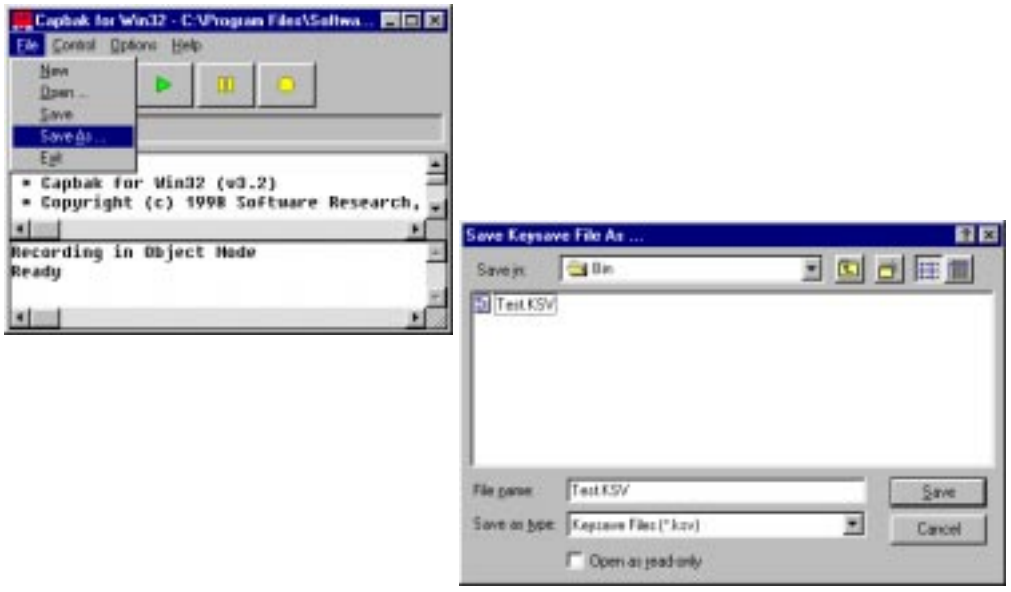


FIGURE 42 Saving a New Test

Warning: If you select an already existing keysave file, a confirmation dialog box will pop up to warn you that an existing keysave file will be overwritten.

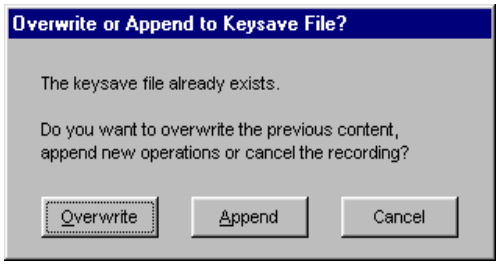


FIGURE 43 Confirmation Dialog Box

6.4.2 Using the Function Keys

When you are at the point in your recording session where you want to issue commands to *CAPBAK/MSW*, you can use the twelve function keys available. Each one is described below.

In the *READ THIS FIRST* envelope, you should have received function key templates printed on a clear plastic sheet. Just cut out one of the templates and place it above your keyboard's function keys as a handy reference.

6.4.3 F1 Function Key

This function key is also referred to as the “hotkey” because it invokes the **Hotkey** window. The **Hotkey** window’s buttons allow you to perform commands identical to the function keys. If you are a first time user, it is a good idea to go through the hotkey’s protocol structure of issuing commands until you get familiar with the function keys.

To use this function key:

1. Press the **Ctrl**, **Alt** and **F1** keys at the same time, then release. The recording session pauses and the **Hotkey** window pops up.

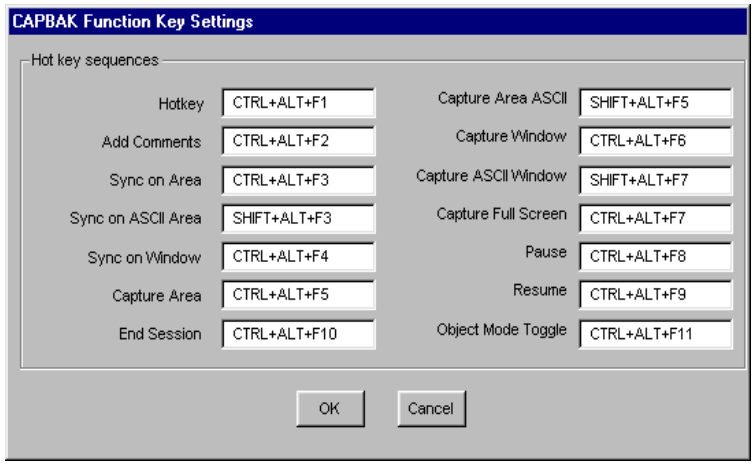


FIGURE 44 Selecting the Hotkey Window

2. The invocation window states:

```
Pausing. Recording Suspended
```

3. Select one of the following options by clicking on the appropriate button:
   - **Add Comments** to place comments in the keysave file.
   - **Wait on Area** to save a partial screen area for synchronization.
   - **Sync on ASCII** to save text for synchronization.
   - **Sync Window** to save a window for synchronization.
   - **Save Area** to save an area of the screen.
   - **Save Area ASCII** to save an area's text.
   - **Save Window** to save an entire window.
   - **Save Window ASCII** to save an entire window's text.
   - **Save Screen** to save the full screen
   - **Resume** to return to the recording session.
   - **End Session** to terminate the recording session.
4. Operation of each button is described next.

6.4.3.1 Add Comments Button

During a long recording session, you may want to use the **Add Comments** button to place comments in the keysave file. You can also use the **F2** function key (see Section 6.4.4 on page 87, “F2 Function Key”).

For instance, as you record a test, you may capture several kinds of images (such as synchronized images, partial images, window images and root images) which can create a large keysave file. By placing a comment identifying the action you took in the keysave file, you can easily find the location of these image captures or actions.

When this button is used, it inserts verbatim what you type in. You do not have to type in `/*comment*/`. “`/*`” and “`*/`” are automatically added before and after each *`comment`* you type in.

If you insert a comment, your recording and playback sessions are not altered. Although the comments are present in the keysave file, recording and playback sessions are not interrupted.

To add comments to the keysave file, follow these steps:

1. Click on the **Add Comments** button.
2. A dialog window pops up.

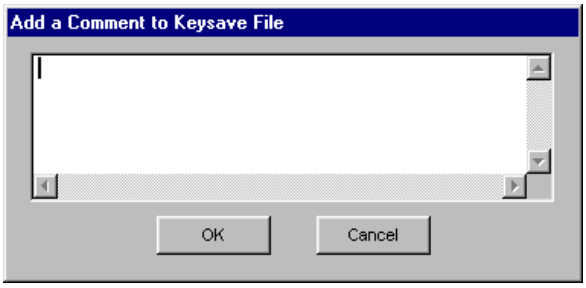


FIGURE 45 Adding Comments to the Keysave File

3. At the cursor point, type in your input.
4. Choose **OK** to insert the input into the keysave file.
5. Or, choose **Cancel** and no action is taken.

6.4.3.2 Sync Window Button

In the MS-Windows environment, applications don't always respond at the same speed. The time that elapses until a window is redrawn may vary from one test execution to another. The speed may be influenced by the current load on your machine, or by your busy network.

As you record a test, you may want to specify a window for playback synchronization. Although playback offers automatic output synchronization (for new parent windows and pop-ups), it is recommended that you use this button or the corresponding **F4** function key, so you can verify the exact image of a window you captured is at the exact same location during playback.

CAPBAK/MSW pauses playback execution for a specified amount of time until an image is found on the screen that matches the saved window. Unless you turn on the **Quit on Image Sync Error** option in the **Image Sync Options** window, playback will always proceed regardless ofailure to match the window. When a window is selected, *CAPBAK/MSW* simultaneously generates a

```
cb_sync_win_image("path\\basename.sxx", X, Y, width, height);
```

event statement in the keysave file.

To use the **Sync Window** button:

1. The window that you plan on capturing for synchronization must be *completely* visible. If it is partially hidden, then playback image synchronization will not be reliable.
2. Click on the **Sync Window** button. The mouse pointer symbol turns into a cross-hair symbol.
3. Move the cross-hairs to a window that you want to synchronize on and click the mouse, making sure the window is *completely* visible.
4. The image is saved as *basename.sxx*.

When you want to view the synchronized window later on, you can use the **CBVIEW** utility. The image will be saved as *basename.sxx*, where *s* indicates that it is a synchronized image and *xx* shows the original sequence in which the image was captured starting at 01. Please refer to Chapter 8 on page 141, "Displaying Captured Images", for information on displaying captured images.

Warning: If the window frame is included in the capture, then the window probably won't be captured accurately. This occurs because MS-Windows can only focus on one window at a time. When you are capturing a window with the **Hotkey** window, the input focus is on the **Hotkey** window, the window to be captured. This will change the color of the border area of the window that had the input focus. During playback the **Hotkey** window does not pop up, so the border area color of the window that had the input focus doesn't change.

To resolve this problem, turn off the **Include Window Frame** option in the **Misc. Capbak Options** window if it is on (its defaulted to off) or use the **F4** function key, or mask off the top border.

6.4.3.3 Wait on Area Button

If this button is used during a recording session, playback will wait until an image of the area matches the area saved. You can also use the **F3** function key. It takes a "snapshot" portion of a window or the screen and uses it as an image that must be matched during playback before proceeding. Keep in mind, however, only the portion you specify is captured.

When an image is selected, *CAPBAK/MSW* simultaneously generates a

```
cb_wait_area("path\\basename.sxx", X, Y,
width, height);
```

event statement in the keysave file.

To sync on an image:

1. Click on the **Wait on Area** button. The mouse pointer symbol turns into a cross-hair symbol.
2. Move the cross-hairs to the area containing the image you want to synchronize.
3. Click and hold down the mouse button, then drag the mouse until it includes the whole image you want to save, and let go of the mouse button.
4. The image is saved as *basename.sxx*.

When you want to view the synchronized image later on, you can use the **CBVIEW** window. The image will be saved as *basename.sxx*, where *s* indicates that it is a synchronized image and *xx* shows the original sequence in which the image was captured starting at 01. Please refer to Chapter 8 on page 69, "Recording a Test" for information on displaying captured images.

Warning: If the window frame is included in the screen image capture, then the image probably won't be captured accurately. This occurs because MS-Windows can only focus on one window at a time. When you are capturing an image with the **Hotkey** window, the focus in on the **Hotkey** window, not the image to be captured. This results in different border colors. To resolve this problem, capture screen areas that *do not* include any window frames or use the **F3** function key.

6.4.3.4 Save Area Button

Save Area captures an area of the screen. You can also use the corresponding **F5** function key. Captured images are used for comparing the behavior of recording and playback sessions of an application-under-test. By later comparing the two images, the exact nature of a discrepancy can be determined.

When an image is captured, *CAPBAK/MSW* simultaneously generates a

```
cb_save_area("path\\basename.sxx", X, Y, width, height);
```

event statement in the keysave file.

To capture a partial window, follow these steps:

1. Click on the **Save Area** button.
2. Move the cross-hairs to the area containing the image(s) you want to capture.
3. Click and hold down the mouse button.
4. Drag the mouse until it includes the whole image you want to save.
5. Let go of the mouse button. The image is captured.
6. The image is saved as *basename.bxx*.
7. During playback a corresponding response file named *basename.rxx* will be created. You can use the **CBDIFF** utility to compare the two images.

When you want to view the saved baseline image later on, you can use the **CBVIEW** window. The image will be saved as *basename.bx*x, where *b* indicates that it is a baseline image and *xx* shows the original sequence in which the image was captured starting at 01. Please refer to Chapter 8 on page 141, "Displaying Captured Images", for information on using the **CBVIEW** window.

Warning: If the window frame is included in the image capture, then the image probably won't be captured accurately. This occurs because MS-Windows can only focus on one window at a time. When you are capturing an image that includes a border with the **Hotkey** window, the input focus is on the **Hotkey** window, not the image that includes the border of a window that is being captured. This will change the color of the border area of the window that had the input focus. During playback the **Hotkey** window does not pop up, so the color of the border area doesn't change of the window that had the input focus. To resolve this problem, capture screen areas that *do not* include any window frames or use the **F5** function key.

6.4.3.5 Save Window Button

The **Save Window** button captures an entire window. You can also use the corresponding **F6** function key. By later comparing a saved window image with the playback response image file, you can further verify the accuracy of a playback session.

When an image is captured, *CAPBAK/MSW* simultaneously generates a

```
cb_save_window("path\\basename.bxx", X, Y);
```

event statement in the keysave file test script.

To save a window, follow these steps:

1. The window that you plan on capturing must be *completely* visible. If it is partially hidden, then the window will not be properly captured.
2. Click on the **Save Window** button.
3. Move the cross-hairs to the window to be saved.
4. The image is saved as *basename.bxx*.
5. During playback a corresponding response file named *basename.rxx* will be created. You can use the **CBDIFF** utility to compare the two images.

When you want to view the saved window later on, you can use the **CBVIEW** window. The image will be saved as *basename.bxx*, where *b* indicates that it is a baseline image and *xx* shows the original sequence in which the image was captured starting at 01. Please refer to Chapter 8 on page 141, "Displaying Captured Images" for information on using the **CBVEW** window.

Warning: If the window frame is included in the window capture, then the window probably won't be captured accurately. This occurs because MS-Windows can only focus on one window at a time. When you are capturing a window with the **Hotkey** window, the focus in on the **Hotkey** window, not the window to be captured. This results in different border colors.To resolve this problem, turn off the **Include Window Frame** option in the **Misc. Capbak Options** window or use the **F6** function key.

6.4.3.6 Save Screen Button

If your application-under-test occupies the full screen, then you should capture a full screen image.

The **Save Screen** button captures the whole screen (the desktop window). You can also use the corresponding **F7** function key.

When screen is captured, *CAPBAK/MSW* simultaneously generates a

```
cb_save_screen("path\\basename.bxx");
```

event statement in the keysave file test script.

To save the entire screen, follow these steps:

1. Click on the **Save Screen** button.
2. The **Record/Play** and the **Hotkey** windows disappear from the screen, then *CAPBAK/MSW* automatically captures the screen.
3. When the **Record/Play** and the **Hotkey** windows reappear, the full screen is completely captured as *basename.bxx*.

When you want to view the saved full screen later on, you can use the **CBVIEW** window. The image will be saved as *basename.bxx*, where *b* indicates that it is a baseline image and *xx* shows the original sequence in which the image was captured. Please refer to Chapter 8 on page 141, "Displaying Captured Images" for information on using the **CBVIEW** window.

6.4.3.7 Resume Button

The **Resume** button continues recording a test, after issuing any of the **Hotkey** window's options. To continue a recording session:

1. Click on the **Resume** button.
2. The **Hotkey** window disappears and you continue recording.
3. The status window then reads: `Recording`. You begin again issuing keystrokes, mouse movements, and more commands.

6.4.3.8 End Session Button

The **End Session** button terminates the recording session. You can also use the corresponding **F10** function key. To end your recording session, follow these steps:

1. Click on the **End Session** button.
2. The **Hotkey** window disappears and the recording session is stopped.
3. The status window then reads: `Ready.`

6.4.4 F2 Function Key

Equivalent to the **Hotkey**'s **Add Comments** button. To use:

1. Press the **Ctrl**, **Alt** and **F2** keys simultaneously, then release. The recording pauses.
2. The **Add Comments to Keysave File** dialog box pops up.
3. At the cursor's point, type in your input.
4. When you are done, click on the **OK** button.
5. Or, select **Cancel** and no action is taken.

6.4.5 F3 Function Key

Equivalent to the **Hotkey**'s **Wait on Area** button.

Defines synchronization areas. What you're doing is capturing a portion of the screen and using it as an image that playback must verify before proceeding.

To use:

1. Press the **Ctrl**, **Alt** and **F3** keys simultaneously, then release. Recording pauses.
2. When the cross-hair symbol appears, move it to the area containing the image you want to synchronize.
3. Click and hold down the mouse button.
4. Drag the mouse until it includes the whole image you want to save. Then let go of the mouse button. The status window signals capture of the synchronized image the following message:

   ```
   Saving area image in DIB file
   ```
5. The image is saved as *basename.sxx*.

6.4.6 F4 Function Key

Equivalent to the **Hotkey**'s **Sync Window** button to capture entire windows for playback image synchronization.

To use:

1. The window that you plan on capturing for synchronization must be *completely* visible. If it is partially hidden, then playback image synchronization will not be reliable.
2. Move your mouse over to the window you want to save.
3. Press the **Ctrl**, **Alt** and **F4** keys simultaneously, then release. The recording pauses.
4. The status window signals capture of the synchronized window the following message:

   ```
   Saving window image in DIB file
   ```
5. The image is saved as *basename.sxx*.

6.4.7 F5 Function Key

Equivalent to the **Hotkey**'s **Save Area** button.

Captures an area image of the screen.

1. Press the **Ctrl**, **Alt** and **F5** keys simultaneously, then release. The recording pauses.
2. Move the cross-hairs to the area containing the area you want to save.
3. Click and hold down the mouse button.
4. Drag the mouse until it includes the whole image you want to save. Then let go of the mouse button. The image is captured.
5. The status window signals capture with the following message:

   ```
   Saving area image in DIB file
   ```
6. The baseline image is saved as *basename.bxx*.

6.4.8 F6 Function Key

Equivalent to the **Hotkey**'s **Save Window** button.

Captures whatever window is located directly underneath the mouse pointer. By later comparing this image with the playback response file image, you can further verify the accuracy of a playback session. To save the mouse window, follow these steps:

1. The window that you plan on capturing must be *completely* visible. If it is partially hidden, then the window will not be properly captured.
2. Move your mouse over to the window you want to save.
3. Press the **Ctrl**, **Alt** and **F6** keys simultaneously, then release. The recording pauses.
4. The status window signals capture of the window with the following message:

   ```
   Saving window image in DIB file
   ```

5. The window is saved as *basename.bxx*.

6.4.9 F7 Function Key

Equivalent to the **Hotkey**'s **Save Screen** button.

Captures the full screen. To save the root window, follow these steps:

1. Press the **Ctrl**, **Alt** and **F7** keys simultaneously, then release. The recording pauses.
2. The **Record/Play** window disappears from the screen.
3. When the **Record/Play** window reappears, the full screen is completely captured as *basename.bxx*.

6.4.10 F8 Function Key

Pauses a recording session.

1. Press the **Ctrl**, **Alt** and **F8** keys simultaneously, then release.
2. Recording stops and the status window reads:

```
Pausing. Recording Suspended
```

6.4.11 F9 Function Key

Equivalent to the **Hotkey**'s **Resume** button.

Resumes a paused recording session.

1. Press the **Ctrl**, **Alt** and **F9** keys simultaneously, then release.
2. The status window signals recording continuance with the following message:

```
Recording
```

6.4.12 F10 Function Key

Equivalent to the **Hotkey**'s **End Session** button.

Terminates the recording session. To end your recording session:

1. Press the **Ctrl**, **Alt** and **F10** keys simultaneously, then release.
2. Recording stops and the status window reads:

```
Ready
```

CHAPTER 7

Playing Back a Test

This chapter discusses the two modes of playback: *Playback* and *Quick Check*.

7.1 Replaying Tests

A test can be played back in two modes: *CAPBAK/MSW*'s regular *Playback* mode, or the *Quick Check* mode. The *Playback* mode is used to validate execution and to generate or update a test's expected results. The *Quick Check* mode is used to compare the behavior of the AUT to its behavior during a previous execution.

The *Playback* mode is described in Section 6.3 and the *Quick Check* mode is described in Section 6.4.

Prior to playing back a test, you should be aware of a known problem with double clicking with the mouse. How long the interval between the clicks is — as perceived by the window management software — can be subject to some variability. In most cases this is due to the processing load on your computer.

This problem arises most commonly when you are selecting an icon or a file in a file selection dialog window where you have the option to double-click on a file selection to which you've pointed with the cursor. In these situations, you should decrease the speed between double clicks. Here's how:

1. Click from **Start --> Settings --> Control Panel**.

2. There should be a **Control Panel** window pops up.

FIGURE 46 Control Panel Window/Control /Panel Icon/Main Group window

3. There should be a **Mouse** icon in the Control Panel window. Double click on it. One of the following windows pops up.

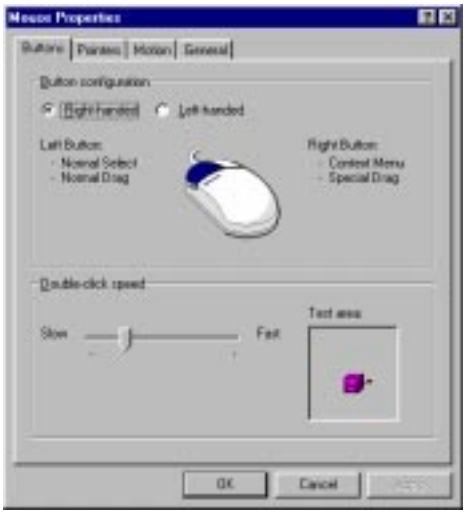


FIGURE 47 Mouse Window 95 and NT4.0

4. To decrease the speed between double clicks, move the **Double Click Speed** slide bar to slow, then choose **OK**.

7.2 Using the Record/Play Window's Buttons for Playback

Before playing back a test, you may want to familiarize yourself with using the following **Record/Play** window's control panel buttons for playback.

- **FF:** Accelerates playback's speed by 10 percent for each time it is pressed prior to a playback session. Playback speed is determined by the **Delay Multiplier** option in the **Misc. Capbak Options** window. The default is 100 to represent true time playback. Each time this button is used, the **Delay Multiplier**'s number decreases by 10 percent, speeding up playback.

Note: The **Delay Multiplier** option must be turned *on* in order for this button to work.

- **PLAY:** Begins a playback session. If you use the **FF** button during a recording session, then playback will increase its speed 10 percent for each time the **FF** button was used.
- **STOP:** Terminates a playback session.

7.3 Playback Mode

1. To play back a recorded keysave file, make sure the **Record/Play** window is up.
2. If not already selected, open the test to be replayed. Each time you replay a test, you overwrite the existing set of expected results.
3. Select any options you may want for playback from the **Options** menu, except any of the options from the **Quick Check Options** window. These options are only necessary for playback's *Quick Check* mode.

   If you used the **Hotkey** window's **Sync Window** or **Wait on Area** buttons or the **F3** or **F4** function keys during your recording session, you must also make sure the necessary image synchronization options are turned on in the **Image Sync Options** window. Please see Section 4.2.2.2 on page 50 for further information.

   If your application popped up any window during the recording, make sure the automatic event synchronizations options in the **Event Sync Options** window are turned on in the so that the `cb_sync_win_activate` keysave file statements can be interpreted. Please see Section 4.2.2.1 on page 49 for further information.
4. Click on the **PLAY** button.
5. The status window signals the start of a playback session with the prompt: `Playing back`
6. The keysave file should start playing back.
7. **Do not move the mouse or try to input from the keyboard.** Once play back begins, you cannot interrupt with the keyboard or mouse. If you do, playback may be unreliable.

   For this reason, *CAPBAK/MSW* has supplied function keys which allow you to perform several activities, such as speeding up and slowing down the rate of playback, inserting or appending new keysave file records into the keysave file, pausing, resuming or terminating playback. The available functions are described in Section 7.5 on page 98.
8. You can use the **Record/Play** window's **STOP** button to terminate the session, but you must *first* pause with the **F8** function key, because the mouse is not free to select these buttons during playback.
9. If you used the **Record/Play** window's **FF** button *prior* to playback, playback's speed increases by 10 percent for each time it was used.
10. Images are automatically captured as response files (*basename.r01*, *basename.r02*,...) during playback, where *r* represents a response, or expected, image file. You can look at these files with the **CBVIEW**

utility or you can compare them with their corresponding baseline image for differences with the **CBDIFF** utility.

The sequence of numbers for the saved images during playback follow that of the recording session. Hence, *basename.b01* corresponds to *basename.r01* and so on.

11. Images or windows that you captured for synchronization (using the **Hotkey** window's **Sync Window** or **Wait on Area** buttons or the **F3** and **F4** function keys), *basename.sxx*, do not generate response files during playback.

Instead, if the **Image Sync Options** window's **Image Sync** option is on, playback pauses test execution for the number of seconds specified, or until the window or area image is displayed and verified. For windows, playback finds the window of the screen whose image matches that saved in *basename.sxx*, then moves the window so that is at the proper location. For areas, playback waits until an image of the saved area's X, Y coordinates and width and height matches the saved image in *basename.sxx*. On average, image synchronization takes less than 5 seconds per captured image or window.

If synchronization is successful, playback issues the following message in the status window:

```
Syncing on Window Image
```

or

```
Syncing on Area Image
```

If the image cannot be found, you will receive one of two messages:

```
Image sync failed, continuing.
```

If you did not turn on the **Image Sync Options** windows's **Quit on Image Sync Error** option.

```
Image sync failed, quitting.
```

If you did turn on the **Quit on Image Sync Error** option.

12. When play back is complete, the status window reads:

```
Ready
```

7.4 Quick Check Mode

The *Quick Check* mode involves replaying a test in order to generate a new set of AUT responses. The new responses, the actual results, are compared with earlier results, that is, the expected results of the test.

Quick Check verifies an application's behavior by automatically comparing any currently captured actual images or windows with the image or window that was captured and stored as the expected result. These captured images do not include synchronized images. Any actual image or window that differs from the expected image is captured and saved. A difference report is created, which indicates if any images mismatched.

When images are different, you can use the **CBDIFF** utility to see the actual difference. Please refer to Chapter 9 on page 129, "Comparing Images" for further information.

To perform playback in the *Quick Check* mode:

1. Open the test to be replayed.
2. Select the options from the **Quick Check Options** window.
   - **Difference Images with Baseline** to automatically compare the actual results with the expected results. This option must be turned on for the *Quick Check* mode to work.
   - **Difference Report** writes the verifications results to a file that you specify.
   - **Quit on Number of Diff Errors** exits playback if the number of difference failures is equal to the number specified.

     Please refer to Section 4.2.2.3 on page 51 for further information.
3. Click on the **PLAY** button.
4. Play back the file just as would in the *Playback* mode (Section 7.3 on page 94).
5. When playback captures an actual image, *Quick Check* compares it with the expected image. Quick check takes fewer than 5 seconds for each comparison. When images are being compared, the status window states:

   ```
   Performing Quick Check Diff
   ```

6. If the **Difference Report** option is on, *CAPBAK/MSW* produces a report, as in the following sample, that lists all the verification results.

```
Report for test run on file C:\CAPBAK\WORK\TEST.KSV:
Images C:\CAPBAK\WORK\TEST.b01 and C:\CAPBAK\WORK\TEST.r01
          were equal.
Images C:\CAPBAK\WORK\TEST.b02 and C:\CAPBAK\WORK\TEST.r02
          were NOT equal.
Images C:\CAPBAK\WORK\TEST.b03 and C:\CAPBAK\WORK\TEST.r03
          were equal.
Images C:\CAPBAK\WORK\TEST.b04 and C:\CAPBAK\WORK\TEST.r04
          were NOT equal.
```

7.5 Using the Function Keys During Playback

By using function keys, you can vary the speed of playback, pause playback to insert or append a new recording into the keysave file, and terminate the session altogether.

In the supplied *READ THIS FIRST* envelope, you should have received function key templates printed on a clear plastic sheet. Just cut out one of the templates and place it above your keyboard's function keys. This template will provide you with a quick reference to the function keys.

7.5.1 F4 Function Key

Allows you to decrease the value set for the **Misc. Capbak Options** window's **Delay Multiplier** option by ten percent. The default delay is set to 100, which plays back a test session in real time. Decreasing the value speeds up the playback session. To use:

1. Prior to playing back a session, turn on the **Delay Multiplier** option.
2. During playback, press the **Ctrl**, **Alt** and **F4** keys all at the same time, then release them.

Playback speeds up by ten percent.

3. To continue speeding up playback, press the **Ctrl**, **Alt** and **F4** keys several more times.

Note: There is no guarantee of a synchronized playback if the **Delay Multiplier** speed becomes too high (such as 30).

7.5.2 F5 Function Key

Allows you to increase the value set for the **Misc. Capbak Options** window's **Delay Multiplier** option by ten percent. The default delay is set to 100, which plays back a test session in real time. Increasing the value slows down the playback session. To use:

1. Prior to playing back a session, turn on the **Delay Multiplier** option.
2. During playback, press the **Ctrl**, **Alt** and **F5** keys all at the same time, then lift them up.
3. Playback slows down by ten percent.
4. To continue slowing down playback, press the **Ctrl**, **Alt** and **F5** keys several more times.

7.5.3 F8 Function Key

Allows you to pause a playback session. To use:

1. Press the **Ctrl**, **Alt** and **F8** keys all at the same time, then lift them up. Recording pauses.

   The status window signals a pause with the following message:

   ```
   Pausing. Playback Suspended
   ```

7.5.4 F9 Function Key

Allows you to resume a paused playback session. To use:

1. Press the **Ctrl**, **Alt** and **F9** keys all at the same time, then lift them up.

   The status window signals playback is resumed with the following message:

   ```
   Playing Back
   ```

   The session continues to playback at the point where it was paused.

7.5.5 F10 Function Key

Allows you to terminate a playback session. To use:

1. Press the **Ctrl**, **Alt** and **F10** keys all at the same time, then lift them up.

   Playback stops and the status window reads:

   ```
   Ready.
   ```

CHAPTER 8

Character Recognition

CAPBAK/MSW's character recognition capabilities extend the life of keysave files by tolerating minor application changes, extracting characters and finding the location of a character string.

8.1 System Requirements

8.1.1 Minimum Requirements

To run the OCR engine, you need at least a 486 machine at 33 MHz, with 8MB RAM, a swap file of at least 6 MB, running Windows Ver 3.1 or later. You also need to have approximately 10 MB free space on your hard drive,to install all the OCR utilities.

8.1.2 Recommended Requirements

We recommend that your machine be at least a 486 at 33MHz and have 16 MB RAM and a swap file of at least 18 MB.

8.2 Understanding the OCR Technology

Software Research has licensed OCR technology from Xerox Imaging Systems (XIS) to provide generalized character recognition capabilities. This technology is called Optical Character Recognition (OCR) and its implementation with *CAPBAK/MSW* allows the user to:

- Capture a character string for playback synchronization.
- Capture the characters from an area of the screen or an entire window to a file.

8.3 OCR Restrictions

Character recognition is subject to the following restrictions:

- OCR can extract up to 10,000 characters.
- Character synchronization capturing is limited to one line, five words.
- Software Research, Inc. has trained the OCR software to recognize the most common fonts in MS-Windows.
- Characters must be either black or white.
- Characters must be at least 8 point size.

8.4 Character-Based Synchronization

Output generated by the application-under-test (AUT) may vary at any given time. Application changes, such as button location, fonts and background colors and open-file menu list output, cause differences between recording and playback sessions that the user may want reported.

In order for these kinds of changes to be tolerated *CAPBAK/MSW* offers character synchronization. Character synchronization is accomplished with the keysave file `cb_sync_ascii ();` function.

This function reads a line of text from an area of the screen or from a window, and returns the read text as a character string that is to be located during playback.

A character string is captured and *CAPBAK/MSW* searches for it in one of the following locations:

1. The window that the mouse is in.
2. The entire screen.

Because the OCR technology has to search the specified area for characters during playback, searches on #2 are slow. The location of the search can be set with one of the sub-options in the GUI's **ASCII Options** window. Please refer to Section 8.4.2 on page 107 and Section 8.4.3 on page 109 for further information.

8.4.1 Synchronizing on a Character String

During your recording session you can synchronize on a character string by using the **Shift, Alt** and **F3** keys.

You should synchronize on *each* location you think will change in an application. If you plan on interchanging the locations of an **OK** and a **Cancel** button in a file selection window, for instance, you must capture both the **OK** and **Cancel** character strings for playback synchronization to occur.

To synchronize on a character string:

1. When you want to synchronize on a character string, press the **Shift, Alt** and **F3** keys at the same time and then lift them up. The recording session pauses and the cursor becomes a cross-hairs.
2. To bound the characters to be read, move the cross-hairs to area containing the text and move the cross-hairs to one corner of the character string.
3. Press and hold down the mouse button, drag the mouse until the rectangle traced contains the entire string to be included. A string should consist of whole words, rather than segments of a word. A word is any text that has a space before and after it.
   - It is recommended that you bind the rectangle as close as possible to the string.
   - Make sure to capture whole characters; the OCR technology cannot interpret partial characters.
   - Character strings must be in one line and no more than five words.

4. When the entire character string is in the rectangle, release the mouse button.

   The status window states: `Saving ASCII for syncing...`

   The **Confirm OCR Text** window pops up with the character string you selected shown in the display area.

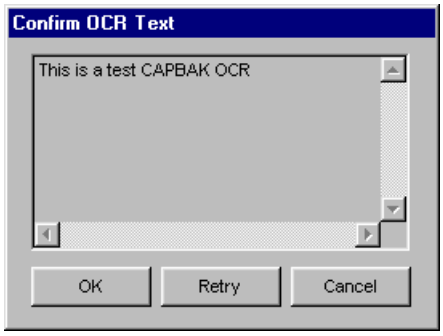


FIGURE 48 Confirm OCR Text Window

- Use the scroll bars to maneuver throughout the **Confirm OCR Text** window.
- Click on **OK** if the character string displayed is what you want to synchronize on.
- Click on **Cancel** if you do not want to use the character string displayed and you want to issue another command.
- Click on **Retry** if the character string is not the string you wanted. The status window states: `Saving ASCII for syncing...` The cursor turns into a cross-hairs so another additional character string can be captured. Please follow steps 2-4 to capture another character string.

5. If the **OK** button was selected, *CAPBAK/MSW* signals capture of the character string with a bell and the status window then reads:

   ```
   Recording...
   ```

   CAPBAK/MSW generates a `cb_sync_ascii (x,y,"string")` event statement in the keysave file, indicating the string to be synchronized on when the test is played back.

   Here, *x*, *x* are the coordinates of the corner at which you started drawing the rectangle for the string; *string* is the captured character string.

   The cursor will then automatically move to the center of the string you synchronized on.

NOTE: Until the next sequence of "button down" and "button up" is recorded (i.e. `cb_key_up` (); and `cb_key_down` (); in the keysave file, *relative motion* is recorded (i.e. `cb_move_mouse_rel (x,y delay`). Motion is relative to where mouse was before the movement. When the mouse pointer is first moved, its motion is relative to the center of the string. Each subsequent movement is then relative to the last mouse movement.

Relative motion is key to OCR synchronization. The mouse movement during playback will be based on — that is, synchronized on — the new location of the ASCII string you synchronized on during the recording session.

6. Continue your recording session as you normally would.

8.4.2 Selecting Character Synchronization Options From the GUI

Prior to playback you may want to select some of the available character synchronization options with the **OCR Options** window.

To initialize the **OCR Options** window:

1. Click on the **Options** menu.
2. Select the **OCR Options** sub-menu.
3. The **OCR Options** window pops up (Figure 47).

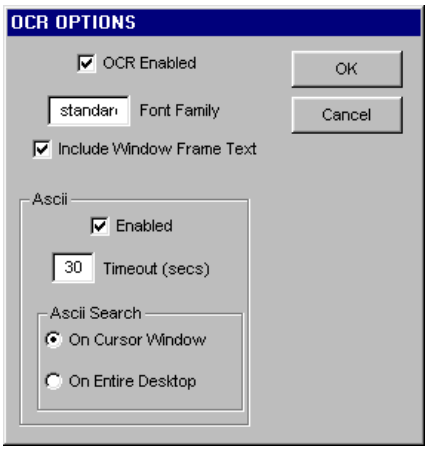


FIGURE 49 OCR Options Window

You can select any of the following options:

- **OCR Enabled:** Turns the OCR recognition utility on.
- **Font Family:** Specifies the name of the font family for character synchronization. During the installation procedure, the default font family is automatically placed in the *.ini* file. Please refer to the *Installation Instructions* for further information on appending the default to the *.ini* file.
- **Include Window Frame Text:** The window frame text, including banners and menus will be included in the string captured**.**
- **ASCII Sync:** The OCR utility will wait a specified time for a character string to be found. If check box is turned off, playback will proceed even if the character string is not found.
- **Time Out:** Defines the time-out period that playback waits for a character string to be found (**ASCII Sync**). The default is 30 seconds.
- **ASCII Search On Window:** Searches for a captured character string in the window that contains the cursor.
- **ASCII Search On Entire Screen:** Searches for a captured character string throughout the entire screen. This may take the system a long time.

You can change the defaults for any of the above options by editing the following lines in the *cbmsw.ini* file, which is located in *C:\windows*.

```
OCR_ACTIVE=YES
OCR_FONT_FAMILY=STANDARD
OCR_INCLUDE_FRAME=YES
OCR_ASCII_SYNC=YES
OCR_SYNC_TIMEOUT=30
OCR_ASCII_SYNC_WHERE=WINDOW
```

8.4.3 Playing Back with Character Synchronization

Prior to playback you should have set any necessary options with the **OCR Options** window, as described in Section 8.4.2 on page 107.

When the `cb_sync_ascii` *(x,y,"string")* is encountered during playback:

1. Playback pauses.
2. The cursor moves to the *x,y* coordinate location specified in the keysave file and searches for the string.
3. The area of the search depends on what you selected in the GUI's **ASCII Sync Options** window. Please see Section 8.4.2 on page 107.
4. If the string is found, the cursor moves to the middle of the string found.

8.5 Capturing ASCII Characters

Previously, users could only capture bitmap images. While bitmap images are useful for general playback verification, they can be inaccurate in certain cases where the output of an image remains the same but the font has changed. In other words, bitmap captures are concerned with pixel differences, not the actual values within the captured image. *CAPBAK/MSW*'s ASCII character captures, on the other hand, record only the values from an area of the screen or a window. During playback the values are captured again and compared for differences with the recording's saved values.

ASCII characters can be captured from areas of the screen or entire windows.

8.5.1 Screen Area Character Capturing —Save ASCII Partial Button

To capture the values from an area of the screen using the **Hotkey** window's **Save ASCII Partial** button:

1. When you want to save the values from a particular area of the screen during a recording session, press the **Ctrl** and **Alt** and **F1** function keys.

   The recording session pauses and the **Hotkey** window pops up.

   The status window states:

   ```
   Pausing.
   ```

2. Select the **Save ASCII Area** button.

   The cursor becomes a cross-hairs.

3. To bound the characters to be read, move the cross-hairs to area containing the text and move the cross-hairs to one corner of the character string.

4. Press and hold down the mouse button, drag the mouse until the rectangle traced contains the entire area of values to be included. Make sure you captured whole characters. Otherwise the OCR technology will be unable to interpret these characters.

5. When the character string with whole character values is in the rectangle, release the mouse button.

   The status window states:

   ```
   Saving ASCII area.
   ```

   The **Confirm OCR Text** window pops up with the selected character string shown in its display area.

   - Use the scroll bars to maneuver throughout the **Confirm OCR Text** window.
   - Click on **OK** if the character string displayed is what you want.
   - Click on **Cancel** if you do not want to use the character string displayed and you want to issue another command.
   - Click on **Retry** if the screen area's values are not what you intended to capture. The status window states: `Saving ASCII area`. The cursor turns into a cross-hairs so you can capture another screen area's values. Please follow steps 3-4 to capture another character string.

6. If the **OK** button was selected, *CAPBAK/MSW* signals capture of the characters with a bell and the status window states:

   ```
   Saving ASCII area.
   ```

   The characters are saved as *basename.axx*, where *a* indicates that it is an ASCII character baseline file and *xx* shows the original sequence in which the file was captured. You can view this file with any text editor.

   CAPBAK/MSW generates a `cb_ascii_area` *(x,y,width,height*) event statement in the keysave file. Here, *x,y* are the coordinates of the corner at which you started drawing the rectangle and *width* and *height* are the dimensions of that rectangle.

   The OCR technology also creates a temporary working file of the relative motion named *ocr.out*.

7. Use the **Hotkey** window as you normally would to resume recording or to issue other commands.

During playback the `cb_ascii_area` *(x,y,width,height*) statement is interpreted and a response file of the values is captured named *basename.zxx*, where *z* indicates that it is an ASCII character response file and *xx* shows the original sequence in which the file was captured.

Playback verifies that *basename.zxx* was captured with the following message in the status window of the **Record/Play** window:

```
Saving ASCII area.
```

8.5.2 Screen Area Character Capturing — Using the Shift+Alt + F5 Keys

To capture the values from an area of the screen using the **Shift**, **Alt** and **F5** keys:

1. When you want to synchronize on a character string, press the **Shift**, **Alt** and **F5** keys at the same time and then lift them up.

   The recording session pauses and the cursor becomes a cross-hairs.

2. To bound the characters to be read, move the cross-hairs to the area containing the text and move the cross-hairs to one corner of the character string.
3. Press and hold down the mouse button, drag the mouse until the rectangle traced contains the entire string to be included.
4. When the character string with whole character values is in the rectangle, release the mouse button.

   The status window states: `Saving ASCII area.`

   The **Confirm OCR Text** window pops up with the character values from the area of the screen you captured displayed.

   - Use the scroll bars to maneuver throughout the **Confirm OCR Text** window.
   - Click on **OK** if the character string displayed is what you want.
   - Click on **Cancel** if you do not want to use the character string displayed and you want to issue another command.
   - Click on **Retry** if the screen area's values are not what you intended to capture. The status window states: `Saving ASCII area`. The cursor turns into a cross-hairs so you can capture another screen area's values. Please follow steps 2-4 to capture another character string.

5. If the **OK** button was selected, *CAPBAK/MSW* signals capture of the character string with a bell and the status window states:

   ```
   Recording.
   ```

   The characters are saved as *basename.axx*, where *a* indicates that it is an ASCII character baseline file and *xx* shows the original sequence in which the file was captured. You can view this file with any text editor.

   CAPBAK/MSW generates a `cb_save_ascii_area` *(x,y, width, height*) event statement in the keysave file. Here, *x,y* are the coordinates of the corner at which you started drawing the rectangle and *width* and *height* are the dimensions of that rectangle.

6. Continue the recording session.

   During playback the `cb_save_ascii_area` *(x,y, width, height*) statement is interpreted and a response file of the characters is captured named *basename.zxx*, where *z* indicates that it is an ASCII character response file and *xx* shows the original sequence in which the image was captured.

   Playback verifies that *basename.zxx* was captured with the following message in the status window of the **Record/Play** window:

   ```
   Saving ASCII area.
   ```

8.5.3 Window Character Capturing — Using the Save ASCII Window Button

To synchronize on a character string using the **Hotkey** window's **Save ASCII Window** button:

1. When you want to save the characters from a window, press the **Ctrl +Alt+F1** function keys.

   The recording session pauses and the **Hotkey** window pops up.

   The status window states:

   ```
   Pausing. Recording Suspended.
   ```

2. Select the **Save ASCII Window** button.

   The cursor becomes a cross-hairs.

3. Move the cross-hairs to the window to be saved.

4. Click and release the mouse button, making sure not to click on an application's activation buttons.

   The status window states: `Saving ASCII window ...`

   The OCR technology will take some time to read all the values from the window so the system may hesitate for awhile. When the characters are interpreted, the **Confirm OCR Text** window pops up with all the window's values displayed.

   - Use the scroll bars to maneuver throughout the **Confirm OCR Text** window.
   - Click on **OK** if the character string displayed is what you want.
   - Click on **Cancel** if you do not want to use the character string displayed and you want to issue another command.
   - Click on **Retry** if the window you captured is not what you intended to capture. The status window states: `ASCII window saving...retry`. The cursor turns into a cross-hairs so you can capture another window's values.

5. If the **OK** button was selected, *CAPBAK/MSW* signals capture of the window's values with a bell and the status window states:

   Recording.

   The characters are saved as *basename.axx*, where *a* indicates that it is an ASCII character baseline file and *xx* shows the original sequence in which the file was captured. You can view this file with any text editor.

   CAPBAK/MSW generates a `cb_save_ascii_window (x,y)` event statement in the keysave file. *x*,*y* are the coordinates of a point contained by the window.

6. The status window states:

   `Recording.`

7. Use the **Hotkey** window as you normally would to resume recording or to issue other commands.

   During playback the `cb_save_ascii_window (x,y)` statement is interpreted and a response file of the characters is captured named *basename.zxx*, where *z* indicates that it is an ASCII character response file, and *xx* shows the original sequence in which the image was captured.

   Playback verifies that *basename.zxx* was captured with a bell and the following message is in the status window of the **Record/Play** window:

   `Recording.`

8.5.4 Window Character Capturing — Using the Shift+Alt + F7 Keys

To capture the values from an area of the screen using the **Shift, Alt** and **F6** keys:

1. Move the mouse pointer so it is on top of the window with the values you want to save.

   If you want to save a pull-down menu, click on the pull-down menu and drag the mouse pointer so that it is located within the menu button's borders. If the mouse pointer is located between two buttons or on a button's borders when the window is captured, recording may fail and the window may not be captured.

2. Press the **Shift**, **Alt** and **F7** keys at the same time and then lift them up.

3. A cross-hairs symbol will appear. Click this on the window from which you want to capture text.

   The OCR technology will take some time to read all the values from the window so the system may hesitate for awhile. When the characters are interpreted, the **Confirm OCR Text** window pops up with all the window's values displayed.

   - Use the scroll bars to maneuver throughout the **Confirm OCR Text** window.
   - Click on **OK** if the character string displayed is what you want.
   - Click on **Cancel** if you do not want to use the character string displayed and you want to issue another command.
   - Click on **Retry** if the window you captured is not what you intended to capture. The status window states: `ASCII window saving...retry`. The cursor turns into a cross-hairs so you can capture another window's values.

4. If the **OK** button was selected, the status window states: `Recording.`

   The characters are saved as *basename.axx*, where *a* indicates that it is an ASCII character baseline image and *xx* shows the original sequence in which the file was captured. You can view this file with any text editor.

   CAPBAK/X generates a `cb_save_ascii_window (x,y)` event statement in the keysave file. Here, *x,y* are the coordinates that are contained by the window whose text you are saving.

5. Continue the recording session.

   During playback the `cb_save_ascii_window (x,y)` statement is interpreted and a response file of the characters is captured named *basename.zxx*, where *z* indicates that it is a response file, and *xx* shows the original sequence in which the file was captured. Playback verifies that *basename.zxx* was captured with the following message in the status window of the **Record/Play** window:

   ```
   Recording.
   ```

8.6 OCR Start-Up

In addition to its use with *CAPBAK/MSW*, the OCR can be used as a stand-alone utility. To begin the OCR recognition process, double-click on the OCR icon in the STW window .

8.6.1 File Pull-Down

When you are finished with the OCR utility, choose **Exit** from the file pull-down menu. If you have not already saved the text to a file, you will get a pop-up dialog box reading **Save recognized text?** (Figure 47). If you wish to save the text to file, click **Yes**; if not, click **No**.

8.6.2 Area

When the **Area** button is clicked on, a "cross-hairs" icon will appear. You will use this cross-hairs symbol to draw a rectangle around the area of the screen you wish to capture. You do this by clicking the left mouse button on the corner of the desired area, and dragging the mouse to complete the rectangle.

NOTE: It is important that you limit your rectangle drag to areas with only one background and one foreground color. For example, if you were to select an area where one part of the screen has white as the foreground color, and another part where white is the background color, the recognition process would not work well at all.

8.6.3 Window

The **Window** recognition capability works similarly to the **Area** recognition capability. Again, it is important to limit captured window areas to windows with only one foreground and one background color.

When you double-click on **Window**, the cross-hairs icon will appear once more. In this instance, click the cross-hairs on a window whose text you wish to capture.

NOTE: The OCR engine will try to convert anything in the window, such as scroll buttons, arrows, etc. into some kind of text, so you would be better off not trying to capture windows with lots of non-ASCII character types, i.e. extraneous matter that may be construed by the OCR engine as characters. This is an example of the computer operating premise GIGO (“garbage in, garbage out”).

8.6.4 Viewing Recognized Text

After the OCR has recognized all the text it is capable of, that text will appear in the large area in the middle of the **Confirm OCR Text Window**, as in the Figure 43 example.

NOTE: This file can be edited using **Notepad** or any standard Windows ASCII editor.

8.7 Saving Text to File

After you've captured the area you want to extract text from, click OK to save the text that you have captured.

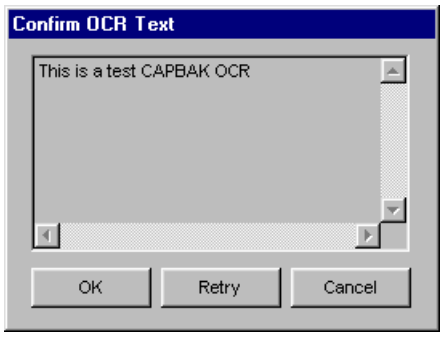


FIGURE 50 Save As Dialog Box

CHAPTER 9

Displaying Captured Images

This chapter describes how to verify tests by viewing baseline and response files.

9.1 Types of Images

You can display images that were captured during the recording (called baseline images) and playback sessions (called response images).

Here is a complete list of the images available for viewing:

- Screen areas and windows captured for playback image synchronization with the **Hotkey** window's **Sync Window** or **Wait on Area** buttons or the **F3** or **F4** function keys.

NOTE: These images are only created during a recording session. During playback, the test execution pauses until the screen area or window is redrawn.

- Screen areas captured with the **Hotkey** window's **Save Area** button or the **F5** function key.
- Full windows captured with **Hotkey** window's **Save Window** button or the **F6** function key.
- Full screen images captured with the **Hotkey** window's **Save Screen** button or the **F7** function key.

Baseline files are distinguished by a *bxx* extension and response files are distinguished by a *rxx* extension, where *xx* shows the original sequence of captures. Synchronized images are distinguished by their *sxx* extension.

NOTE: Unfortunately, some graphic cards and device drivers have inherent limitations that prevent use of image compression. If you have saved an image, but it is blank when you look at it using the **CBVIEW** utility, then your card and device driver probably have these limitations. Turn the **Compress Images** option to off in the **Misc. Capbak Options** window and recapture the image. It should now appear normal using **CBVIEW**.

9.2 Steps to Displaying a Captured Image

1. Click on the **CBVIEW** icon in the **TestWorks Group** window.
2. The **CBVIEW** window pops up.

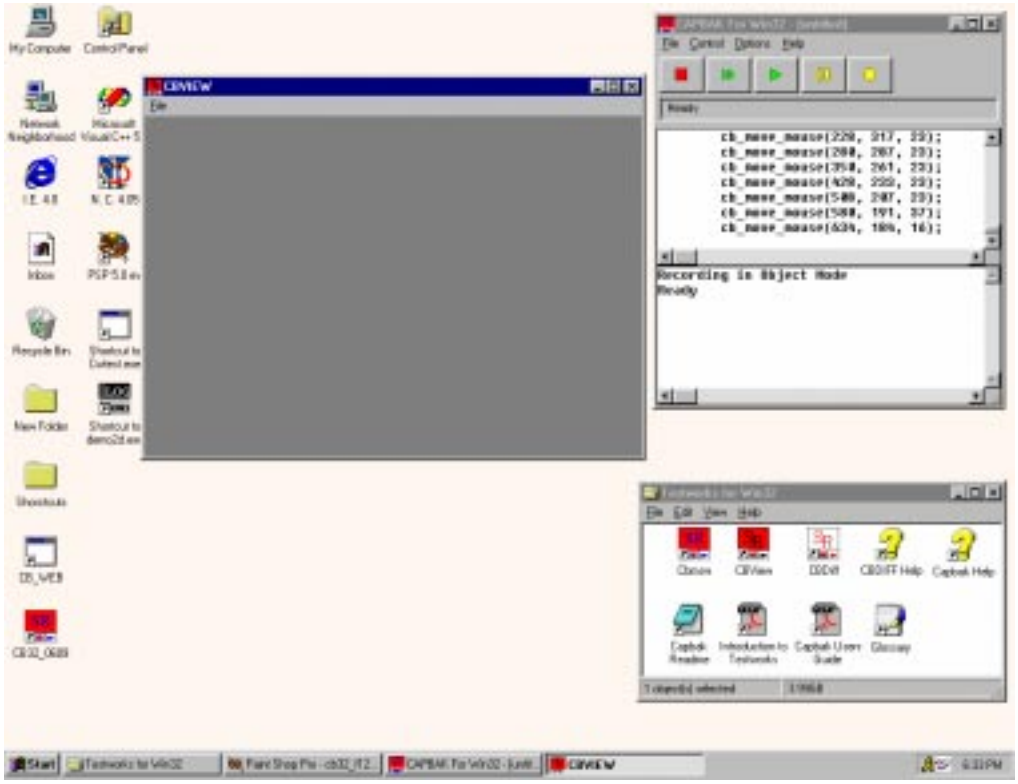


FIGURE 51 CBVIEW Window

3. Click on the **File** menu and select the **Open** submenu. A file selection dialog box pops up.

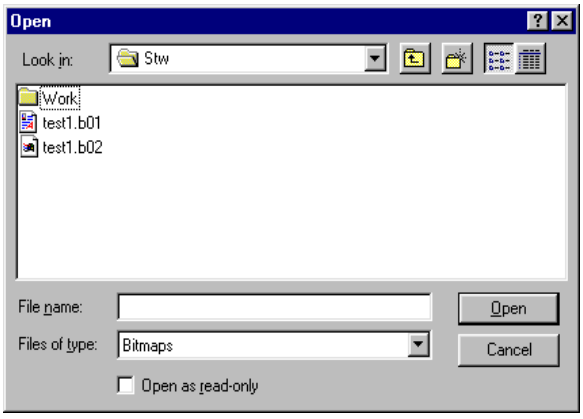


FIGURE 52 File Selection Window

4. You can view images that are saved in the Device Independent (DIB) format. *CAPBAK/MSW* file extensions are *sxx*, *bxx*, and *rxx* for DIB files.
5. To select and view an image file, do one of these three things:
   - Double click on the file in the **File Name** list box.
   - Highlight or type in the file name and click on **OK**.
   - Highlight or type in the file name and then press **Enter**.
6. The image you selected is displayed in the **CBVIEW** window.

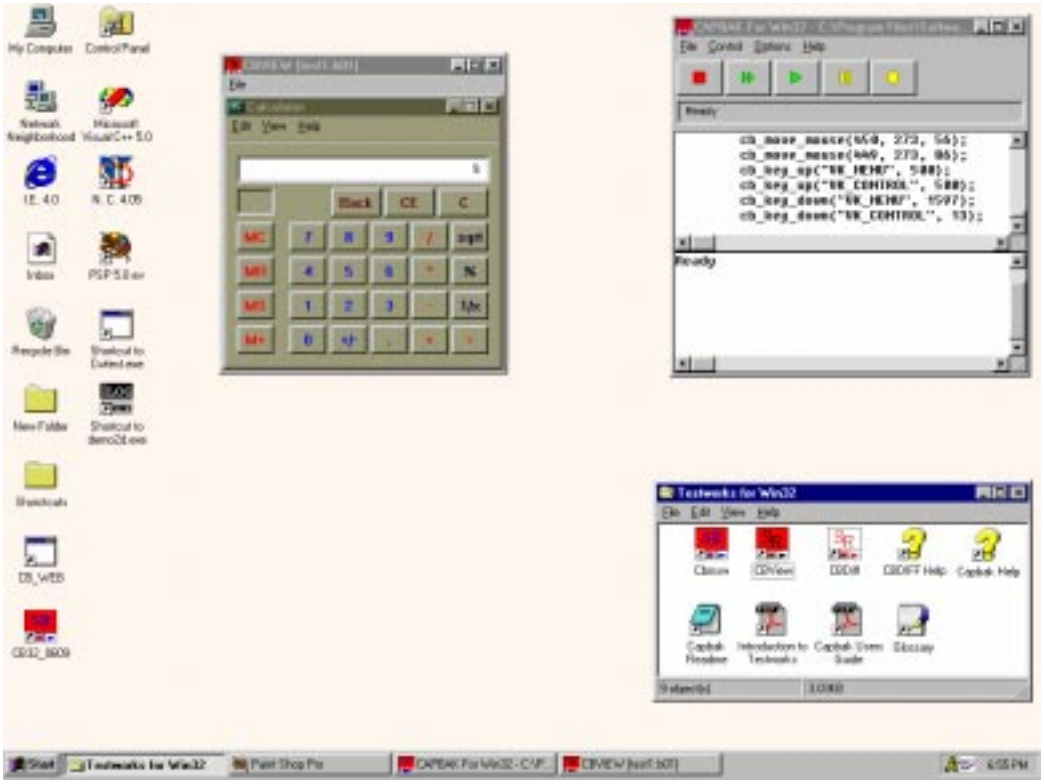


FIGURE 53 Displayed Image

7. To verify a playback, you should first view a baseline image (such as *basename.b01*) and then view the corresponding response file image (such as *basename.r0*1).
8. When you are finished viewing the image, click on the **File** menu and choose the **Exit** option.

   While the **CBVIEW** utility can give a good idea if images captured during recording and playing back match, you can only be completely sure by using the *Quick Check* mode during playback or using the **CBDIFF** comparison utility. Please refer to Section 7.4 on page 96 for information on the *Quick Check* mode and to Chapter 9 on page 129, "Comparing Images" for more information about **CBDIFF**.

CHAPTER 10

Comparing Images

This chapter describes how to difference baseline images with response images.

10.1 The CBDIFF Utility

You can display and then compare the differences between expected AUT images that were captured during a recording session to the actual images that were captured during playback with the **CBDIFF** utility. Inconsequential differences such as changed dates, times and file list differences can be masked out.

This feature can be utilized if you used the regular *Playback* mode and are unsure if some of the images matched. You should also use this utility if you use the *Quick Check* mode. The *Quick Check* mode creates a report that indicates if images matched or not. For the images that mismatched, you can use the **CBDIFF** utility to view the actual differences.

These types of captured images can be displayed and then compared for differences:

- Screen areas captured with the **Hotkey** window's **Save Area** button or the **F5** function key.
- Windows captured with **Hotkey** window's **Save Window** button or the **F6** function key.
- Full screen images captured with the **Hotkey** window's **Save Screen** button or the **F7** function key.

Expected, or baseline files, are distinguished by a *bxx* extension and response files are distinguished by a *rxx* extension, where *xx* shows the original sequence of captures.

NOTE: Unfortunately, some graphic cards and device drivers have inherent limitations that prevent use of image compression. If you have saved an image, but it is blank when you look at it using **CBDIFF**, then your card and device driver probably have such limitations. Turn the **Compress Images** option to off in the **Misc. Capbak Options** window and recapture the image. It should now appear normal using **CBVIEW**.

10.2 Comparing Captured Images

1. Click the **CBDIFF** icon in the **TestWorks Group** window.
2. The **CBDIFF** window pops up.

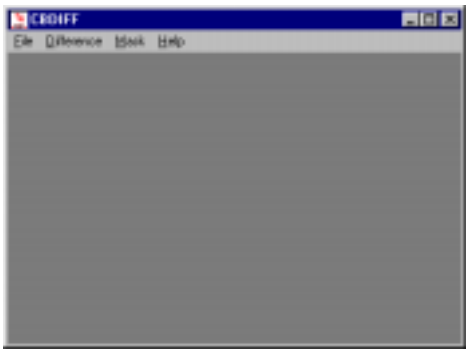


FIGURE 54 CBDIFF Window

3. Click on the **File** menu and select the **Open Baseline** submenu. A file selection dialog box pops up.

4. To select a file, do one of these three things:
   - Double click on the file in the **File Name** list box.
   - Highlight or type in the file name and click on **OK**.
   - Highlight or type in the file name and then press **Enter**.

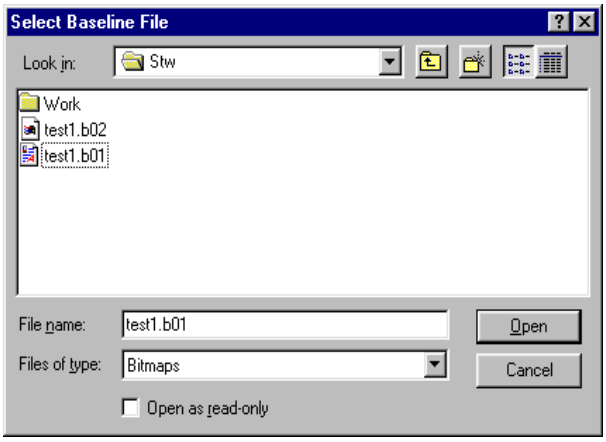


FIGURE 55 Expected Image Selection

5. The image you selected is displayed in the **CBDIFF** window.

FIGURE 56 Displayed Expected Image

6. Click on the **File** menu and select the **Open Response** submenu to select the response file.
7. When the file selection dialog box pops up, select the corresponding response file to the baseline file you already selected.

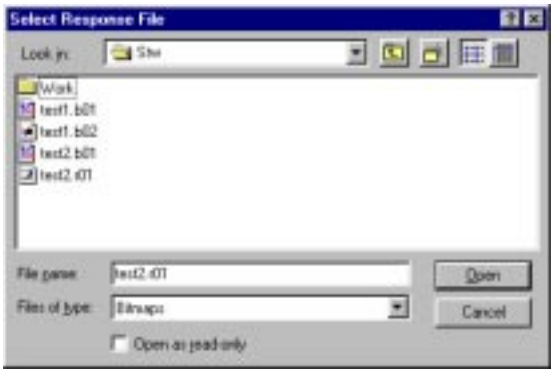


FIGURE 57 Actual Image Selection

8. The image is automatically displayed in the **CBDIFF** window.

FIGURE 58 Displayed Actual Image

9. To compare the images, click on the **Difference** menu and select the **Do Diff** option.
10. If both images match, then the **CBDIFF** window should be completely white; if the images do not match, then the **CBDIFF** window will display those bitmap differences (as shown below).

FIGURE 59 Displayed Differences

11. When you are finished, click on the **File** menu and select the **Exit** option. The window closes.

10.3 Masking Regions of Images

Baseline and response images comparison differences usually consist of changed dates, times and file list differences. These differences are insignificant, raising flags that are not necessary.

These kinds of differences can be ignored with masks, which ignore user-specified areas of images. To determine if mask are necessary:

1. Select the baseline and response file as you normally would during a comparison.
2. Do a difference of the two images. If the differences appears to be what you would consider unimportant, then creating masks can resolve this. Below is an example where using a mask can avoid an unimportant difference.

FIGURE 60 Unnecessary Differences

10.3.1 Creating a Mask

To mask a region:

1. The baseline, response or diff image you intend to mask must be displayed. You can display an image with the **Difference** menu's **View Baseline** (for baseline images), **View Response** (for response images) and **Do Diff** (for the baseline / response difference image) options.
2. Click on the **Mask** menu.
3. Select the **Do Masking** toggle. A check mark should appear before the **Do Masking** toggle and the **Read Mask File** and **Write Mask File** options should no longer be grayed out, indicating their availability.
4. Move the mouse pointer to an area you want to mask.
5. Press and hold down the left mouse button, drag the mouse until the rectangle traced contains the entire region to be masked.
6. When the entire are you want to mask is in the rectangle, release the left mouse button. The mask is created. In our example of the different times, we created a mask in the baseline file.

FIGURE 61 Creating a Mask

7. If you create a mask that is not needed, place the mouse pointer inside the mask and click the right mouse button. The mask disappears.

10.3.2 Difference Using a Mask

1. After creating masks, click on the Difference menu and select Do Diff. The regions that were masked should have been ignored in the difference.

FIGURE 62 Difference Using a Mask

2. If you want the mask displayed during a difference, click on the **Mask** menu and select **Do Masking** after the difference has been displayed. This will toggle on **Do Masking** and the mask will be displayed.
3. Create as many masks as necessary.

10.3.3 Saving Mask Information

After masks are made, the mask coordinate information is *not* automatically saved. After you create masks for an image, you *must* save it to a file:

1. The image where the mask(s) was created must be displayed.
2. Check the **Do Masking** toggle to on.
3. Select the **Write Mask File** option. A file selection dialog box pops up.

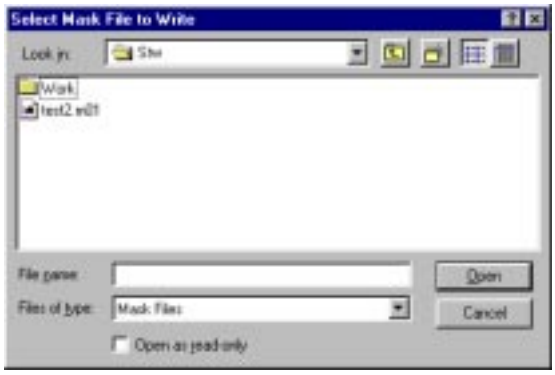


FIGURE 63 Saving Mask Information

4. Because the directory mask is ***.m***, type in a name that follows this format so mask files can be easily identified and then select **OK**. We recommend naming mask files *basename.mxx*, where *basename* is the name of the keysave file and *xx* corresponds to the baseline and the response file's capture extension number.

5. The mask information for the displayed image is written to the file.

If you want to compare images and you have already masked one or both of those images (and saved the mask), then open the corresponding mask file for those images prior to comparison:

1. The image that was masked should be displayed.
2. The **Do Masking** toggle must be checked to on.
3. Select the **Read Mask File** option. A file selection dialog box pops up.

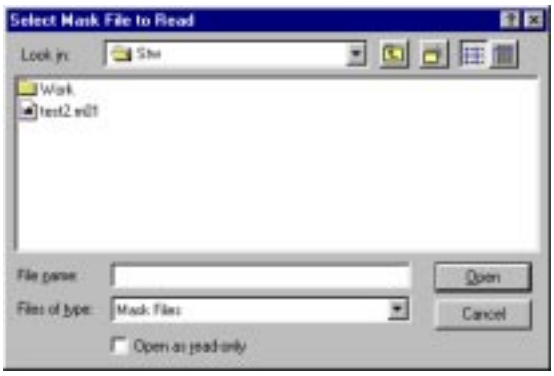


FIGURE 64 Reading In Mask Information

4. Select the mask file for the image displayed.
5. The mask coordinate information is read to the displayed image and the original masks your created should be displayed on the image.

CHAPTER 11

Synchronization Issues

In the MS-Windows System environment, an application's windows are likely to come up at varying locations and times between capture and playback sessions. This chapter addresses the cause of these synchronization issues and tells you how to combat them with *CAPBAK/MSW*.

11.1 Problem Description

The interaction of automated test playback of previously recorded user inputs to the application or client can sometimes cause severe de-synchronization problems.

For any of a variety of reasons, the playback mechanism that is in use can operate in such a way that the coherence between the recording and the response is lost. Briefly what happens is this: The playing back process runs too fast and the playback "runs wild." The resulting loss of synchronization is very often either chaotic behavior of the application, possible loss of essential test output information, or both.

Whatever the cause, loss of synchronization is serious: you risk losing valuable test time and possibly risk losing coherent test processing. How de-synchronization can happen — and what you can do about it — may be important factors in implementing your automated test regression systems.

11.1.1 Applications and Synchronization

De-synchronization problems arise because the application itself is not behaving reproducibly.

An example would be when your application opens a pop-up window and, for variety, chooses to position it at location that is computed by your own program to be somewhere near the middle but never at the same spot. To handle this problem, you can use *CAPBAK/MSW*'s automatic event synchronization options as described below:

1. Click on the **Options** menu.
2. Select the **Event Sync Options** submenu.
3. The **Event Sync Options** window pops up.
4. Make sure the event synchronization switches you need are turned on. Please refer to Section 4.2.2.1 on page 49 for a list of the options.
5. If you want playback to exit if it fails, turn on the **Quit on Event Sync Error** option.
6. When a window pops up during a recording session, *CAPBAK/MSW* automatically generates a

   `cb_sync_win_activate("`*window_name*`",x,y,`*width*`,`*height*`);`

   statement in the keysave file. If you invoked the calculator window during a test session, you may get a statement similar to the one below:

   ```
   cb_sync_win_activate("Calculator",
         154,110,482,337);
   ```

   When the test is played back, `cb_sync_win_activate` is interpreted, and *CAPBAK/MSW*:

   - Waits for a window that has a "Calculator" title.
   - If the **Move Window On Event Sync** option is on, repositions the window if the coordinates are different and changes the width and height if they are different.

11.1.2 Timing a Synchronization

In addition to application pop-up synchronization problems, applications do not always respond at the same speed during the capture and play-back phases. In other words the time that elapsed until a window is redrawn may vary from one test run to another. To handle this:

- Use the *CAPBAK/MSW* image synchronization features to pause execution until a window or partial image is redrawn. This approach assumes that the user records a set of images during the recording session that will act as the synchronization method during playback.
- *CAPBAK/MSW* pauses test execution for a number of seconds (defaulted to 25 seconds) to wait for the exact same image to occur at the exact same place on the screen. During the time-out period, *CAPBAK/MSW* searches for the image until it is found or the delay has timed out.
- If the image is not found, playback will either exit or continue depending on what you specify.

11.1.3 Activating Image Synchronization

To activate image synchronization from the *CAPBAK/MSW* **Record/Play** window:

1. Click on the **Options** menu.
2. Select the **Image Sync Options** submenu.
3. The **Image Sync Options** window pops up.
4. Make sure the image synchronization switches you need are turned on. Please refer to Section 4.2.2.2 on page 50 for a list of the options.
5. If you want playback to exit if it fails, turn the **Quit on Image Sync Error** option on.
6. During a recording session, you can specify an image or window to match for image synchronization during playback by using the **Hotkey** window's **Wait on Area** button or the **F3** function key for screen area images or the **Hotkey** window's **Sync Window** or **F4** function key for windows.

   Please see Section 6.4.2 on page 77 through 6.4.12 on page 90 for complete instructions on specifying images or windows for synchronization.
7. If you save an area for playback image synchronization a

   `cb_wait_area("`*path*`\\`*basename.sx*`",x,y,`*width*`,`*height*`)`

   is generated in the keysave file, which playback understands to mean to wait until an image of the area that begins at *x*, *y* and has a width of *width* and height of *height* matches the image saved in *basename.sxx* before continuing execution of the keysave file script.
8. If you save a window for playback image synchronization a

   `cb_sync_win_image("`*path*`\\`*basename.sxx*`",x,y,`*width*`,`*height*`)`

   is generated in the keysave file, which playback understands to mean to find the window of the screen whose image matches that saved in file *basename.sxx*. Playback then moves the window so its upper left-hand corner is at coordinate *x*, *y* and its size is *width* and *height*.

11.1.4 Delay Multiplier

Adjust timing during playback with the **Delay Multiplier** option in the **Misc. Capbak Options** window or manually add a {DELAY} [*number*] line to the keysave file, where *number* is the delay multiplier number. The delay multiplier default is 100, which represents real time. Decreasing the value speeds up playback; increasing the value slows down playback.

You should experiment with various speeds until you find one that is not too fast to de-synchronize a session and too slow to test the patience of the tester and stay with that speed. Typically, a delay multiplier of < 10 will overdrive the application and a value of > 300 will probably assure synchronization, but will be too slow for even the most patient tester.

It is best to stay with a constant speed. If you use the **Record/Play** window's **FF** button too much prior to playback, you may jeopardize synchronization for playback.

11.2 Summary

Achieving playback synchronization usually involves using a combination of output and image synchronization features along with a consistent playback speed.

APPENDIX A

Understanding the Keysave File Language

This appendix describes the keysave file's syntax. It defines all of the keysave input statements you will encounter.

A.1 Keysave File Statements

The process of recording captures information and saves it in a keysave file. Playback, then, interprets this information. The keysave file is recorded as a single C function. You can easily program these test scripts with the enhanced C interpreter programming language. Please refer to Section A.2 for further details on this language.

Depending on the kinds of activities during a recording session, *CAP-BAK/MSW* will add the following kinds of statements to the keysave file:

cb_move_mouse(*x*,*y*,*delay*)

After a delay of *n* milliseconds, mouse cursor moves by an offset of *x*, *y* relative to the last mouse position.

cb_mbutton_down("*button*",*x*,*y*,*delay)*

Identifies a mouse button press at position *x*, *y* after a delay of *n* milliseconds. "*button*" corresponds to your mouse's three buttons: "LEFT", "MIDDLE" or "RIGHT".

cb_mbutton_up("*button*",*x*,*y*,*delay*)

Identifies a mouse button release at position *x*, *y* after a delay of *n* milliseconds. "*button*" corresponds to your mouse's three buttons: "LEFT", "MIDDLE" or "RIGHT".

cb_key_down("*key*",*delay*)

Identifies a designated key press after a delay of *n* milliseconds.

cb_key_up("*key*",*delay*)

Identifies a designated key release after a delay of *n* milliseconds.

cb_sync_win_activate("*window_name*",*x*,*y*,*width*,*height*)

Looks for a window named "*window_name*" that has just been activated and moves it so that its upper left-hand corner is at coordinate *x*, *y* and resizes it so that it has a width of *width* and a height of *height*. This is *CAPBAK/MSW*'s automatic event synchronization on a pop-up window.

NOTE: If you want this statement to be interpreted during playback, make sure the options are turned on in the **Event Sync Options** window. Whenever a window pops up are varying locations, these options check the position of the window and relocate it if necessary. See Section 4.2.2.1 on page 49 for complete information on the event synchronization options.

The statements below are generated when you specify a wait period while a window or area is redrawn during playback for the purposes of image synchronization. You can use the **Hotkey** window's **Sync Window** or the **F4** function key to specify a window for synchronization and the **Hotkey** window's **Wait on Area** button or the **F3** function key to specify a screen area image for synchronization.

Make sure the options in the **Image Sync Options** window are turned on for your playback session. See Section 4.2.2.2 on page 50 for information on these options.

cb_wait_area("*filename*",*x*,*y*,*width*,*height*)

Waits until an image appears in the area that begins at x, y and has a width of `width` and a height of `height` that matches the image saved in file *basename.sxx*, then continues execution of the keysave file script.

cb_sync_win_image("*filename*",*x*,*y*,*width*,*height*)

Finds the window of the screen whose images matches that saved in *basename.sxx*, then moves the window so that its upper left corner is at coordinate *x*, *y*. The following statements are generated when you capture images and windows:

cb_save_area("*filename*",*x*,*y*,*width*,*height*)

Saves an image of the screen that begins at position *x*, *y* and has width of *width* and a height of *height* into *basename.bxx* when the **Hotkey** window's **Save Area** button or the **F5** function key are used. Please refer to Section 6.4.3 on page 78 and 6.4.7 on page 88 for further information.

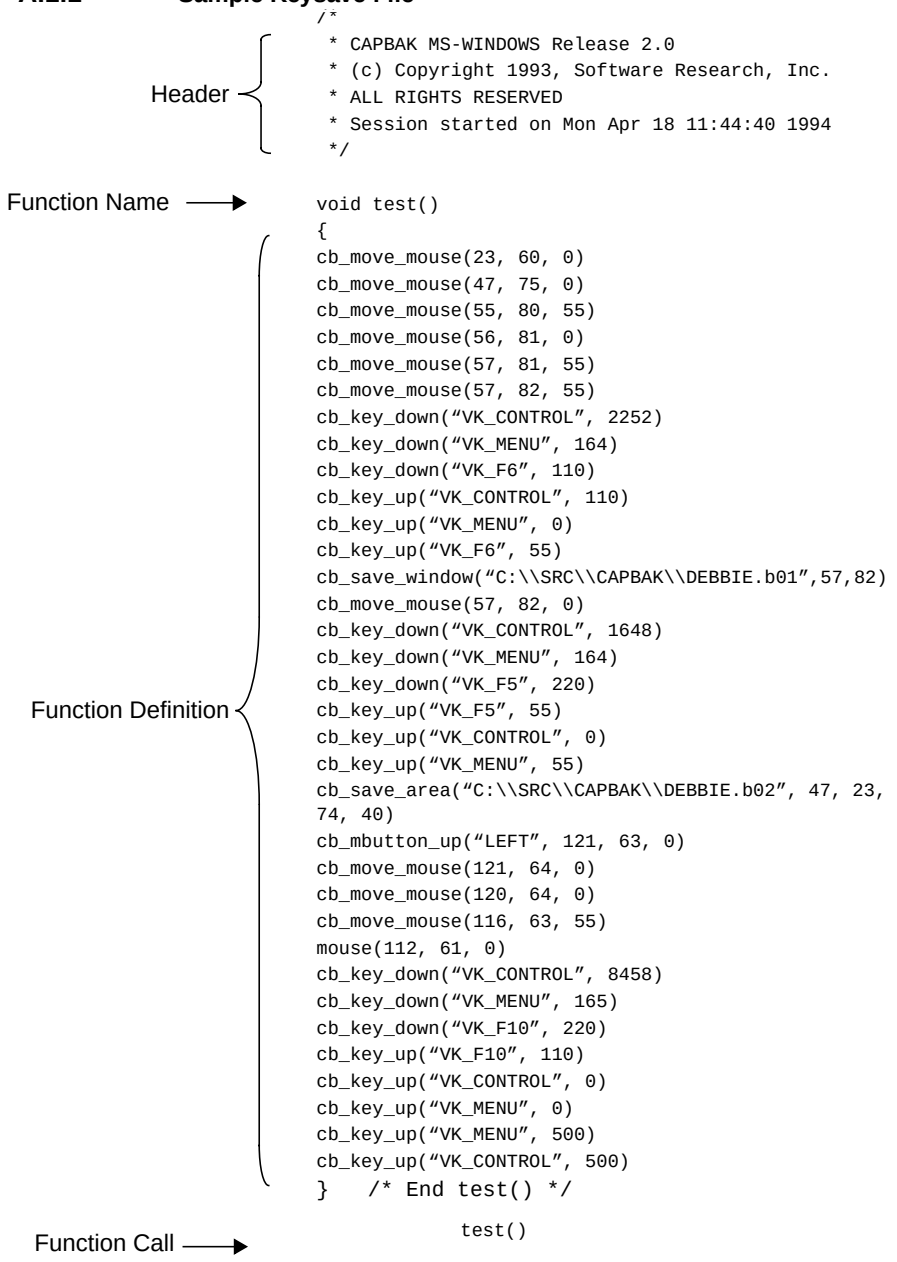
cb_save_screen("*filename*")

Saves the full screen into *basename.bxx* when the **Hotkey** window's **Save Screen** button or the **F7** function key are used. Please refer to Section 6.4.3 on page 78 and 6.4.9 on page 89 for further information.

cb_save_window("*filename*",*x*,*y*)

Saves a window that contains the point *x*, *y* into *basename.bxx* when the **Hotkey** window's **Save Window** button or the **F6** function key are used. Please refer to Section 6.4.3 on page 78 and 6.4.8 on page 89 for further information.

A.1.1 Sample Keysave File




```
/*
 * CAPBAK MS-WINDOWS Release 2.0
 * (c) Copyright 1993, Software Research, Inc.
 * ALL RIGHTS RESERVED
 * Session started on Mon Apr 18 11:44:40 1994
 */

void test()
{
cb_move_mouse(23, 60, 0)
cb_move_mouse(47, 75, 0)
cb_move_mouse(55, 80, 55)
cb_move_mouse(56, 81, 0)
cb_move_mouse(57, 81, 55)
cb_move_mouse(57, 82, 55)
cb_key_down("VK_CONTROL", 2252)
cb_key_down("VK_MENU", 164)
cb_key_down("VK_F6", 110)
cb_key_up("VK_CONTROL", 110)
cb_key_up("VK_MENU", 0)
cb_key_up("VK_F6", 55)
cb_save_window("C:\\SRC\\CAPBAK\\DEBBIE.b01",57,82)
cb_move_mouse(57, 82, 0)
cb_key_down("VK_CONTROL", 1648)
cb_key_down("VK_MENU", 164)
cb_key_down("VK_F5", 220)
cb_key_up("VK_F5", 55)
cb_key_up("VK_CONTROL", 0)
cb_key_up("VK_MENU", 55)
cb_save_area("C:\\SRC\\CAPBAK\\DEBBIE.b02", 47, 23,
74, 40)
cb_mbutton_up("LEFT", 121, 63, 0)
cb_move_mouse(121, 64, 0)
cb_move_mouse(120, 64, 0)
cb_move_mouse(116, 63, 55)
mouse(112, 61, 0)
cb_key_down("VK_CONTROL", 8458)
cb_key_down("VK_MENU", 165)
cb_key_down("VK_F10", 220)
cb_key_up("VK_F10", 110)
cb_key_up("VK_CONTROL", 0)
cb_key_up("VK_MENU", 0)
cb_key_up("VK_MENU", 500)
cb_key_up("VK_CONTROL", 500)
}   /* End test() */

test()
```

FIGURE 65 Sample Keysave File

A.2 Programming the Keysave Files for Playback

Once the keysave file is created, you can add such things as conditional statements and loops by using normal C syntax with *CAPBAK/MSW*'s C-interpretive language.

The C-interpreter programming language allows you to modify recorded keysave files. It is a subset of the C language. This language supports C scalar data types, most C expressions and control-flow constructs.

If an error is made during modification and the keysave file is played back, playback will display a message box with the type of error and the line number of the keysave where the error is located.

The input file format is a subset of the C language as specified by the ANSI standard. The following subsections describe the supported subset.

A.2.1 Data Types

The intepreter supports the following scalar types: `char`, `short`, `int`, `long`, `float`, `double`, `void`, and pointers. Arrays of the scalar types are also supported.

The following are *not* supported: `typedef`s, structures, unions, and `enum`s

A.2.2 Expressions

Precedence rules and conversions rules are as described by the ANSI standard.

A.2.3 Constants

Fixed and floating-point constants are allowed as specified by the ANSI standard. Double-quoted strings and single-quoted characters are allowed.

`Long` and `unsigned` are *not* supported.

A.2.4 Variables

Variables of up to 31 characters are supported with standard C naming conventions.

A.2.5 Operators

For the type `double`, the following C expression operators are supported: `sizeof`, `=`, `+`, `-` (unary), `-` (binary), `*`, `/`, `<`,`>`, `<=`, `>=`, `==`, `!=`, `!`, function call, and array reference.

For the type `int`, the following C expression operators are supported: `sizeof`, `+`, `-` (unary), `-` (binary), `*`, `/`, `%`, `|`, `&` (binary), `^`, `<`, `>`, `<=`, `>=`, `==`, `!=`, `!`, `++`, `—`, `>>`, `<<`, `~`, function call, and array reference.

The following operators are not supported: `?:`, casts, `->`, `&&`. `||`, and `..`

A.2.6 Statements

The following statement constructs are supported: `for`, `while`, `if`, `break`, `return`, and compound statements. Here are two examples:

`if` (expr) {statement}

`if` (expr) {statement} **`else`** {statement}

`for` (expr; expr; expr)

{statement}

The following statement constructs are not supported: `switch`, `continue`, `goto`, `do....while`, and statement labels.

A.2.7 include Directives

The C-interpretive language follows the same rules of C when including files.

```
#include "filename"
```

`"filename`" must follow the C language restrictions.

A.2.8 Standard C Library Functions

These functions are system-dependent; please refer to your system "man pages" to determine if they exist on your machine.

`sprintf()`
`rand()`
`sscanf()`
`strcat()`
`strcpy()`
`open()`
`read()`
`write()`
`time()`

```
system()
exit()
free()
malloc()
strcmp()
ctime()
```

Note: `exit()` does not accept parameters in CAPBAK/MSW.

A.2.9 Preprocessor Directives

No preprocessor directives are supported except for the `include` directive.Library Functions

The following runtime library functions are available:

`cb_change_cwd`(*"path"*)

Changes the current working directory of the process to the one specified in "*path*".

`cb_check_area`(*"filename",x,y,width,height*)

Compare an image of the screen area that begins at *x,y* and has a width of *width* and a height of *height* with the image saved in file "*filename*".
Return 1 if images are the same; otherwise, return 0.

`cb_close`("*file_handle*")

Closes the file identified by the "*file handle.*"

Example:

```
int i;
i=cb_file_open ("c:\\foo.text")
cb_close (i)
```

`cb_file_exists`(*"filename"*)

Returns 1 if file "*filename*" exists, 0 if it does not.

`cb_key_down`(*key,delay*)

Press the designated *key* after *delay* milliseconds.

`cb_key_up`(*key,delay*)

Lift the designated *key* after *delay* milliseconds.

`cb_mbutton_down`(*"which_button",x,y,delay*)

Press either "`RIGHT`", "`LEFT`", or "`MIDDLE`" mouse button at position *x,y* after *delay* milliseconds.

cb_mbutton_down_rel("*button*",*x*,*y*,*delay*))

After a delay of *delay* milliseconds, identifies a mouse button press by an offset of *x*,*y* relative to the last mouse position. "button" corresponds to your mouse's three buttons: "RIGHT", "MIDDLE" or "LEFT".

cb_mbutton_up("*button*",*x*,*y*,*delay*)

Lift up either "RIGHT", "MIDDLE" or "LEFT" mouse button at position *x*,*y* after *delay* milliseconds.

cb_mbutton_up_rel("*button*",*x*,*y*,*delay*)

After *delay* milliseconds, identifies a mouse button release by an offset of *x*,*y* relative to the last mouse position. "button" corresponds to your mouse's three buttons: "RIGHT", "MIDDLE" or "LEFT".

cb_move_mouse_rel(*x*,*y*,*delay*)

After a delay of *n* milliseconds, mouse cursor moves by an offset of *x*,*y* relative to the last mouse position.

cb_open("*filename*")

Opens the file for reading. Returns an integer file handle that is used by other *CAPBAK* routines; returns zero if it can't open the file. *CAPBAK* can only have five files open at a time, so close files after use.

Example:

```
int i;
i=cb_file_open ("c:\\foo.text")
```

cb_random(*upper_bound*)

Returns a randomly selected integer value within the bounds specified by integer value *upper_bound*.

cb_read_string ("*char_array*","*delimiter*","*file_handle*")

Reads a string from the opened file ("*file_handle*") and fills ("*char_array*") with it. It continues filling the string "*char_array*" until it reaches the character identified by "*delimiter*". It then puts a NULL character at the end of the array and returns the number of characters it read. It does NOT put the delimiter in the array:

Example:

```
int i;
char text [80];
i=cb_file_open ("c:\\foo.text")
cb_read_string (text,",",i)
cb_close(i)
```

cb_save_window(*"filename"*,x,y)

Save an image of the top window containing the location *x,y* into the file "filename".

cb_save_area(*"filename",x,y,width,height*)

Save an image of the screen area that begins at position *x,y* and has a width of *width* and a height of *height* into the file "*filename*".

cb_save_screen(*"filename"*)

Save an image of the entire screen into file "*filename*".

cb_start_app(*"application_filename"*)

Starts up the Windows application whose filename is *"application_filename"*. Either *"application_filename"* should be a full pathname, or the *"application_filename"*'s directory should be in your *DOS $PATH*. This function acts similarly to the **Run** command in the Program Manager. A user can edit a keysave file and replace a series of mouse movements and clicks that start an application with this single function. Starting an application will then be independent of the location of the application's icon.

cb_sync_ascii(*"string",x,y*)

Looks for the string "*string*" in the window containing the location *x*, y and plays the "System Asterisk" sound if the text is found. If the text is not found, it either continues executing or stops based on the state of the ASCII checkbox in the OCR Options dialog box.

cb_sync_win_activate(*window_name,x,y,width,height*)

Look for a window named *window_name* that has just activated. Move it so that its upper left-hand corner is at coordinate *x,y* and size it so it has *width* of width and a height of *height*.

cb_sync_win_image(*"filename",x,y,width,height*)

Wait for the window on screen at coded rates *x,y* with size of *width* and *height* whose image matches that saved in file "*filename*".

cb_type("*char_string*",*delay_between_characters*)

Reads the string "*char_string*" which can be either a character array or a quoted string, and types the string's characters into the active window.

The string can contain only ASCII characters available on the keyboard. Function keys and special keys such as "Ctrl", "Enter", "Shift" and "Alt" cannot be in the string. These special keys must be typed using the **cb_key_up()** and **cb_key_down()** functions.

delay_between_characters is the interval, in milliseconds, between the typing of each character.

Example:

```
cb_type("This is a test: &*@#",250)
```

cb_wait_area(*"filename",area_x,area_y,area_width,area_height*)

Wait until an image of the area that begins at *x,y* and has a width of *width* a height of *height* matches the image saved in the file "*filename*" before continuing execution of the keysave file script.

cb_wait(*wait_time*)

Wait or "sleep" for *wait_time* seconds before continuing execution of the keysave file.

cb_save_window_ascii(*"filename",x,y,*)

Saves the test recognized from the window containing the location *x,y* into the file "filename".

cb_save_area_ascii(*"filename",x,y,width,height*)

Saves the text recognized from the search area that begins at position x,y and has a width of *width* and a height of *height* in the file "*filename*".

APPENDIX B

Customizing CAPBAK/MSW

This appendix explains where the setup information is stored and gives you instructions on how to change it.

B.1 CAPBAK/MSW Setup Information

You customize the *CAPBAK/MSW* GUI defaults by changing the initialization file. This chapter explains where the setup information is stored and gives instructions on changing it.

Resource files setup files are text files. You can edit them with any standard text editor. *CAPBAK/MSW* graphical user interface defaults are set in the *cbmsw.ini* file, which is located in *C:\windows*. The following is the *cbmsw.ini* file. You can change the set defaults by manually editing it to avoid re-setting user interface parameters every time.

```
[CBMSW]
CAPBAK_LOCATION=NO
CAPBAK_X=0
CAPBAK_Y=O
HOTKEY_LOCATION=NO
HOTKEY_X=0
HOTKEY_Y=0
USE_DELAY_MULTIPLIER=YES
DELAY_MULTIPLIER=100
REGENERATE_BASE_LINE_IMAGES=NO
INCLUDE_WINDOW_FRAME=YES
COMPRESS_IMAGES=YES
QUIT_ON_EVENT_SYNC_ERROR=NO
WAIT_ON_EVENT=YES
EVENT_TIMEOUT=10
MOVE_ON_EVENT_SYNC=YES
QUIT_ON_IMAGE_SYNC_ERROR=NO
WAIT_ON_IMAGE=NO
IMAGE_TIMEOUT=25
MOVE_ON_IMAGE_SYNC=YES
```

```
D0_DIFFERENCING=NO
MAKE_DIFF_REPORT=YES
DIFF_REPORT_NAME=REPORT.TXT
QUIT_ON_DIFF_ERROR=NO
DIFF_ERROR_LIMIT=5
OCR_ACTIVE=YES
OCR_FONT_FAMILY=STANDARD
OCR_INCLUDE_FRAME=YES
OCR_ASCII_SYNC=YES
OCR_SYNC_TIMEOUT=30
OCR_ASCII_SYNC_WHERE=WINDOW
HOTKEY_KEY=CTRL+ALT+F1
COMMENT_KEY=CTRL+ALT+F2
SYNC_AREA_KEY=CTRL+ALT+F3
SYNC_ASCII_KEY=SHIFT+ALT+F3
SYNC_WINDOW_KEY=CTRL+ALT+F4
SAVE_AREA_KEY=CTRL+ALT+F5
SAVE_AREA_ASCII_KEY=CTRL+ALT+F5
SAVE_WINDOW_KEY=CTRL+ALT+F6
SAVE_WINDOW_ASCII_KEY=SHIFT+ALT+F7
SAVE_SCREEN_KEY=CTRL+ALT+F7
PAUSE_KEY=CTRL+ALT+F8
RESUME_KEY=CTRL+ALT+F9
END_KEY=CTRL+ALT+F10
```

FIGURE 66 CBMSW.INI File

cbmsw.ini's entries are described next.

These entries affect the default settings for the options in the **Misc. Capbak Options** window:

`CAPBAK_LOCATION=NO`

Determines whether or not the **CAPBAK Win Position** option is turned on. It is defaulted to off.

`CAPBAK_X=0`

`CAPBAK_Y=O` Determines the X,Y coordinates of where the **CAPBAK** window will pop-up. They are defaulted to 0,0.

`HOTKEY_LOCATION=NO`

Determines whether or not the **Hotkey Win Position** option is turned on. It is defaulted to off.

HOTKEY_X=0

HOTKEY_Y=0 Determines the X,Y coordinates of where the **Hotkey** window will pop-up during a recording session. They are defaulted to 0,0.

USE_DELAY_MULTIPLIER=YES

Determines whether or not the **Delay Multiplier** option is turned on. It is defaulted to on.

DELAY_MULTIPLIER=100

Determines the default speed for the **Delay Multiplier** option. 100 represents real time.

REGENERATE_BASE_LINE_IMAGES=NO

Determines whether or not the **Regenerate Baseline Images** option is turned on. It is defaulted to off.

INCLUDE_WINDOW_FRAME=YES

Determines whether or not the **Include Window Frame** option is turned on. It is defaulted to on.

COMPRESS_IMAGES=YES

Determines whether or not the **Compress Images** option is turned on. It is defaulted to on.

Please refer to Section 4.2.2.4 on page 52 for further details on the **Misc. Capbak Options** window.

These entries affect the default settings for the options in the **Event Sync Options** window:

QUIT_ON_EVENT_SYNC_ERROR=NO

Determines whether or not the **Quit on Event Sync Error** option is turned on. It is defaulted to off.

WAIT_ON_EVENT=YES

Determines whether or not the **Event Sync** option is on. It is defaulted to on.

EVENT_TIMEOUT=10

Determines the maximum time-out period for the **Event Sync** option. The default is 10 seconds.

MOVE_ON_EVENT_SYNC=YES

Determines whether or not the **Move Window on Event Sync** option is on. It is defaulted on.

Please refer to Section 4.2.2.1 on page 49 for further details on the **Event Sync Options** window.

These entries affect the default settings for the options in the **Image Sync Options** window:

`QUIT_ON_IMAGE_SYNC_ERROR=NO`

Determines whether or not the **Quit on Image Sync Error** option is turned on. It is defaulted to off.

`WAIT_ON_IMAGE=NO`

Determines whether or not the **Image Sync** option is on. It is defaulted to off.

`IMAGE_TIMEOUT=25`

Determines the maximum time-out period for the **Image Sync** option. The default is 25 seconds.

`MOVE_ON_IMAGE_SYNC=YES`

Determines whether or not the **Move Window on Image Sync** option is on. It is defaulted to on.

Please refer to Section 4.2.2.2 on page 50 for further details on the **Image Sync Options** window.

These entries effect the default settings for the options in the **Quick Check Options** window:

`DO_DIFFERENCING=NO`

Determines whether or not the **Difference Images with Baseline** option is on. It is defaulted to off.

`MAKE_DIFF_REPORT=YES`

Determines whether of not the **Difference Report** option is on. It is defaulted to on.

`DIFF_REPORT_NAME=REPORT.TXT`

Determines the name of the **Difference Report**'s report name. The default is set to *report.txt*.

`QUIT_ON_DIFF_ERROR=NO`

Determines whether or not the **Quit on Number of Diff Errors** option is on. It is defaulted to off.

`DIFF_ERROR_LIMIT=5`

Determines the error number for the **Quit on Number of Diff Errors** option. The default is set to 5.

These entries affect the default settings for the options in the **OCR Options** window:

`OCR_ACTIVE=YES` Turns the OCR recognition utility on.

`OCR_FONT_FAMILY=STANDARD`

Specifies the name of the font family for character synchronization.

`OCR_INCLUDE_FRAME=YES`

The window frame text, including banners and menus will be included in the string captured**.**

`OCR_ASCII_SYNC=YES`

Default is YES. If this option is turned off, playback will proceed even if the character string is not found.

`OCR_SYNC_TIMEOUT=30`

Defines the time-out period that playback waits for a character string to be found. The default is 30 seconds.

`OCR_ASCII_SYNC_WHERE=WINDOW`

Searches for a captured character string in the window that contains the cursor. If this is not turned on, the OCR utility searches for a captured character string throughout the entire screen. This may take the system a long time.

These settings affect the default settings for the options in the **Function Key Settings** window. These settings describe the key combinations needed to perform various actions during recording. Please see Section 6.4.2 on page 77 through 6.4.12 on page 90 for specific information on actions performed.

```
HOTKEY_KEY=CTRL+ALT+F1
COMMENT_KEY=CTRL+ALT+F2
SYNC_AREA_KEY=CTRL+ALT+F3
SYNC_ASCII_KEY=SHIFT+ALT+F3
SYNC_WINDOW_KEY=CTRL+ALT+F4
SAVE_AREA_KEY=CTRL+ALT+F5
SAVE_AREA_ASCII_KEY=CTRL+ALT+F5
SAVE_WINDOW_KEY=CTRL+ALT+F6
SAVE_WINDOW_ASCII_KEY=SHIFT+ALT+F6
SAVE_SCREEN_KEY=CTRL+ALT+F7
PAUSE_KEY=CTRL+ALT+F8
RESUME_KEY=CTRL+ALT+F9
```

```
END_KEY=CTRL+ALT+F10
```

APPENDIX C

EXDIFF Utility

This chapter explains the *EXDIFF* differencing utility which works in conjunction with the other tools in the STW / Regression product set.

C.1 EXDIFF's Role

Most systems are supplied with a standard file differencing utility, which reports the compared files as either the same or different. It is left to the user to determine whether the registered differences are of any significant value. For regression testing, such a restrictive and manual evaluation process can become quite cumbersome.

As a stand-alone tool or within the *STW/Regression* tool set, *EXDIFF* is a test evaluation facility that extends commonly-available file comparison utilities. *EXDIFF* compares files of various logical structures:

- Byte-oriented files
- Line-oriented files
- *CAPBAK* keysave files

A typical differencing utility will register extraneous differences, such as changed dates and processing information, and flag the files as different. *EXDIFF* provides masking options which allow the user to specify areas within ASCII files to be ignored during the differencing process.

C.1.1 Byte-Oriented and Line-Oriented Files

The *EXDIFF* default mode **exdiff** differences two ASCII files as byte-oriented or line-oriented files.

Byte-oriented files are typically binary or character-based files. When processing files in byte mode, the **exdiff** utility compares two ASCII or binary files for byte equivalence.

Line-oriented files are ASCII text files such as manuals, letters and source programs. *EXDIFF* incorporates many features of the standard UNIX **diff** utility and has a compatible syntax and subset of options. Comparison

output can also be formatted as a more comprehensively annotated listing, including pathnames and execution date for the compared files.

Prior to executing the **exdiff** utility, an ASCII file can be reformatted and / or divided (based on user-specified byte numbers, line numbers or character strings) using the **EX** command line utilities. Alternatively, extensive masking commands can be established in an **exdiff** resource configuration (RC) file to allow extraneous differences to be ignored during the comparison process.

C.1.2 Bitmap Image Files

CBDIFF, the image comparison version of *EXDIFF*, performs a pixel-by-pixel comparison of two saved image files and displays compared or differenced images to the screen. Please see Chapter 9, “COMPARING IMAGES” for more details on the *CBDIFF* utility.

C.1.3 CAPBAK Keysave Files

During a recording session, the *CAPBAK* capture / playback tool automatically generates input statements that represent user input directed to the application under test (AUT). The test input statements are then stored, in binary format, as a keysave file.

When processing files in keysave mode, the **exdiff** utility lists the command line number and key number where difference occurs.

For further information on *CAPBAK* keysave files, please refer to Appendix A, “UNDERSTANDING THE KEYSAVE FILE LANGUAGE”.

C.2 Main Features

The following list indicates the more significant features of the *EXDIFF* application:

- Allows comparison and masking procedures to be executed from either a window-based graphical user interface or the command line.
- Performs differencing on-the-fly during *CAPBAK*'s playback.
- Enables differencing process to be executed either manually or as a batch process within a *SMARTS* testing script.
- Provides line and byte comparisons of two saved ASCII files with the **exdiff** utility. Trailing blanks (spaces and tabs) can be ignored and strings of blanks can be compared as equal.
- Allows ASCII comparison output to either resemble output for the UNIX **diff** utility or to be formatted as a comprehensively annotated listing.
- Reformats and/or divides an ASCII file (based on user-specified byte numbers, line numbers or characters strings) with the **EX** utility, preparing the files for more thorough differencing with the **exdiff** utility.
- Allows extensive masking restrictions for similar ASCII files to be established in an **exdiff** resource configuration (RC) file.

When the **exdiff** utility is executed, either from the GUI or the command line, a configuration file which includes runtime parameters may also be invoked. In *EXDIFF*, this file is referred to as the resource configuration (RC) file and the default is *exdiff.rc*. If **exdiff** is invoked without the RC file, then the embedded default values for the *EXDIFF* application are used.

NOTE: RC file runtime commands will take precedence over runtime options specified from the command line.

Runtime parameters for the **exdiff** utility can be set or modified using any of the following methods:

1. Specified in an RC file prior to invoking *EXDIFF*.
2. Indicated during the command-line invocation.
3. Specified from the GUI.

Information on the Resource Configuration File follows.

C.3 exdiff Resource Configuration (RC) File

The **exdiff** RC file is comprised of a series of runtime parameters, which are set one per line and listed in any order. Many of the available **exdiff** RC commands are equivalent to functions that can otherwise be performed via the *EXDIFF* GUI or the command line. In addition, various **exdiff** RC options extend the **exdiff** utility's comparison capabilities by allowing extensive masking restrictions to be specified for:

- Byte (individual characters)
- Line
- Pattern (character string)
- Character(s) preceding and / or following a pattern

If an invoked RC file contains no syntax errors, then the **exdiff** utility compares two specified ASCII or *CAPBAK/UNIX* keysave files based on the RC file's run-time parameters.

NOTE: RC file runtime options will take precedence over runtime options specified from the command line.

The **exdiff** RC file can be created or modified with a text editor. While any name may be used for an **exdiff** RC file, the convention is to use *.rc* as the filename extension. Using the indicated syntax, the following runtime commands can be specified within an **exdiff** RC file:

*comment*

options= *command_line_option(s)*
byteoff= *value1* | *value1*-*value2* **[,** *value2* | *value3*-*value4* ...**]**
lineoff= *value1* | *value1*-*value2* **[,** *value2* | *value3*-*value4* ...**]**
pattern *N1* **"***pattern***"** *N2*

A line beginning with the pound character, **#**, is considered a comment and is ignored by the configuration file processor. The **options** command is examined in Section C.4 on page 167; **byteoff**, **lineoff** and **pattern** masking commands are examined in Section C.5 on page 168.

A sample **exdiff** RC file is provided in Section C.6 on page 175.

C.4 Specifying Command Line Options

The syntax for specifying **exdiff** command line options an **exdiff** RC file is:

options= *command_line_option(s)*

The *command_line_option(s)* indicate **exdiff**'s runtime options.

Rules for specifying command line options are as follows:

1. Parameters cannot be set for command line options; only the default values are available.

   For example, when executed from the command line, the *N* parameter of the **-c** *N* option controls the number of lines printed before and after each discovered difference. When indicated as a runtime option from within an **exdiff** RC file, the *N* parameter of the **-c** option cannot be set and the default value of 3 lines is used.

2. The **exdiff** help switch (**-h**) cannot be specified as an **options** variable. The following is a sample **options** command in an **exdiff** RC file:

```
options = -b -c
```

Switches can specified in a compressed format. For example, the previous sample **options** command could be rewritten as follows:

```
options = -bc
```

C.5 Specifying Masking Commands

Complex regression analysis most often requires the comparison of similar baseline and response files which were generated in two separate executions of a product test. In many cases, however, inconsequential differences throughout similar ASCII files (such as dates, headers, footers, or names) prevent **exdiff** from reliably differentiating the files.

Extensive masking commands can be established in the **exdiff** RC file, allowing extraneous differences throughout similar ASCII files to be ignored *during* the comparison process.

NOTE: *Prior* to executing the **exdiff** utility, an ASCII file can be reformatted and / or divided (based on user-specified byte numbers, line numbers or character strings) using the **EX** command line utilities.

The **exdiff** RC masking commands are as follows:

- The **byteoff** command indicates byte (individual characters) to be masked **.**
- The **lineoff** command specifies line numbers to be masked.
- The **pattern** command indicates a characters string (and a specified number of characters preceding and / or following the string) to be masked.

The **byteoff** and **lineoff** commands are examined in Section C.5.1 on page 169; the **pattern** command is examined in Section C.5.2 on page 172.

C.5.1 Byte and Line Masking Commands

The **exdiff** RC **byteoff** and **lineoff** commands specify byte masking and line masking, respectively.

Arguments for the **byteoff** and **lineoff** commands are specified using either of the following syntax formats (or a combination thereof):

byteoff | **lineoff=** *value1* **[**, *value2*...**]**

or:

byteoff | **lineoff=** *value1*-*value2* **[**, *value3*-*value4*...**]**

The first syntax format is used to mask an individual byte or line number. The second syntax format is used to mask a range of bytes or line numbers. Descriptions of command parameters are as follows:

value1 Individual byte or line number.

value1-value2 Range of bytes or line numbers.

[, *value2*...**]**

[, *value3-value4*...**]** Subsequent values or value ranges are to be separated by commas

---

NOTE: The notation **[...]** represents a instance which can occur zero or more times.

C.5.1.1 Byteoff and Lineoff Rules

Rules for specifying the **exdiff** RC file **byteoff** and **lineoff** commands are as follows:

1. Indicated values must be positive integers and can be specified in any order.
2. The **byteoff** and **lineoff** commands can be listed in any order within an **exdiff** RC file. However, a **lineoff** command takes precedence over a **byteoff** command.

   Example: The contents of an input file named *insight1* is:

   ```
   This is a GREAT show.
   But he is glad to leave.
   ```

   The contents of an input file named *insight2* is:

   ```
   this is a great show.
   But he is glad to be in the lobby.
   ```

   An **exdiff** RC file named *insight.rc* contains the following **byteoff** and **line off** commands:

   ```
   byteoff= 1, 11-15
   lineoff= 2
   ```

   Although the **byteoff** command does not mask all differences on the second line of both files, the entire line is masked by the **lineoff** command. Therefore, if the *insight.rc* file is invoked when comparing *insight1* and *insight2* with the **exdiff** utility, the files will register as equal.
3. When calculating offsets for the **byteoff** command, be certain to include spaces, tabs and carriage returns.

C.5.1.2 Byteoff and Lineoff Examples

The following is a sample **byteoff** command in an **exdiff** RC file:

byteoff= 305, 2, 607-941, 83

In the above sample, the 2nd, 83rd and 305th byte, as well as bytes 607 through 941, will be masked in the compared ASCII files.

Byte values can also be specified on individual lines. For example, the previous sample **byteoff** statement could be rewritten as follows:

byteoff = 2

byteoff = 83

byteoff = 607-941

byteoff = 305

The following is a sample **lineoff** command in an **exdiff** RC file:

lineoff= 39-57, 11, 43

In the above sample, the 11th and 43rd line, as well as lines 39 through 57, will be masked in the compared ASCII files.

NOTE: While any value can be combined or separated, values may *not* be duplicated. The following masking statements are *incorrect*:

---

lineoff = 305, 2, 305

or

lineoff = 305, 2
lineoff = 305

In both instances, byte **305** has been specified twice and will result in an error message.

C.5.2 Pattern Masking Command

The **exdiff** RC **pattern** command searches the comparison files for an indicated character string. If located, the pattern and an indicated number of characters preceding or following the pattern are masked. If the pattern is not located in *both* comparison files, the entire command is ignored by the **exdiff** utility.

Arguments for the **pattern** command are specified using the following syntax:

pattern *N1* "*pattern_string*" *N2*

N1	Number of characters to be masked *before* the matched pattern string.
"*pattern_string*"	Character string to be located and masked.
N2	Number of characters to be masked *after* the matched pattern string.

Rules for specifying the **exdiff** RC file **pattern** command is as follows:

1. Negative numbers and numbers which exceed file line length cannot be specified. That is:
   - The number of characters masked before a pattern string cannot go beyond the prior newline character ("\n") and
   - The number of characters masked after a pattern string cannot go beyond the next newline character ("\n").
2. The **pattern** command *must* have three arguments; if not, an error message is issued. To mask *only* a **pattern** string, the command is indicated as follows:

   pattern 0 "*pattern_string*" **0**
3. The **pattern** command can be listed in any order with **byteoff** and **lineoff** commands within an **exdiff** RC file. However, a **lineoff** command takes precedence over a **pattern** command. For an example, refer to the second rule for specifying the **byteoff** and **lineoff** commands (Section C.4 on page 167).

   The **exdiff** RC file can maintain an unlimited number of pattern commands, subject to general memory constraints.

C.5.2.1 Wild Card and Escape Characters

Wild card and escape characters are specified within a **pattern** string as follows:

?	Mask an instance of *ANY* character.
\?	Mask an instance of the ? character.
****	Mask an instance of the \ character.

Example Usage: The **pattern** string **?AND?** matches the FORTRAN logical operator .AND. as well as the characters SANDY. However, the **pattern** string **\?AND\?** matches **only** on the string ?AND?. Special Characters Allowed

Using the indicated escape notations, the following "white space characters" are accepted within a **pattern** string:

\t	Horizontal tab character
\v	Vertical tab character
\f	Form feed character
\a	Bell character
\b	Backspace character

C.5.2.2 Disallowed Characters

The following "white space characters" (and their corresponding escape notations) *cannot* be used in a pattern:

\n	Newline
\r	Carriage Return
\0	Null Character

NOTE: The null character is used exclusively to terminate a file.

C.5.2.3 Octal Character Specification

A character may be represented within the **pattern** string, using the octal format \OOO where OOO is an octal number.

NOTE: Only octal character values < 177 are allowed; an error message is issued if a larger number is specified.

Example Usage: Specifying nonprintable characters.

C.5.2.4 pattern Examples

Example 1: **`pattern 0 "date" 12`**

Masks the **pattern** string `date` and the 12 characters *after* the string.

Example 2: **`pattern 20 "Smith" 0`**

Masks the **pattern** string `Smith` and the 12 characters *preceding* the string.

Example 3: **`pattern 0 "05?25?94" 0`**

Masks the **pattern** string **05?25?94**, where **?** represents any character. Thus, various formats for the date March 25, 1994, such as **05/25/94** and **05-25-94**, will be ignored by the **exdiff** utility.

NOTE: If **?** is to be the actual comparison character, then the command *must* be written as follows:

```
pattern 12 "05\?25\?94" 0
```

*Example 4:*pattern 3 "\t" 2

Masks a horizontal tab character, as well as the three characters preceding and the two characters following the horizontal tab.

C.6 Sample exdiff RC File

```
#
# Invoke this exdiff RC file with the following
ASCII # files:
#    resume93
#    resume94
#
byteoff= 6-10, 65
#
lineoff= 1, 23-35, 50
#
pattern 0 "December" 4
#
options= -c
#
```

APPENDIX D

LAN Control of CAPBAK/MSW

This appendix describes a simple process that provides for control of *CAPBAK/MSW* playback sessions from a central ‘‘control computer’’.

D.1 Problem Description

Assume that there are several machines in a multiple machine PC-based LAN that is also accessed by a UNIX workstation. Such an architecture is fairly common when a UNIX machine is used as a file server and the PCs are used as workstations.

The goal is to provide for playback of recordings, made previously with *CAPBAK/MSW*, but which must be controlled from within *SMARTS*’ executions on the UNIX workstation. If this can be accomplished then it will be possible to run many (possibly, many hundreds) of ‘‘simulated sessions’’ from a single control point.

Because it isn’t possible to launch a process on a PC running MS Windows, we assume here that *CAPBAK/MSW* will have been manually started from each PC in the LAN. The goal, thus, is to fire off a particular script on a particular PC, knowing that *CAPBAK/MSW* is already running there.

SMARTS will provide the PC’s with information on which script to run, and will create or control a semaphore file whose existence causes the PC to start execution of the script. It will then compare resulting **.r** or**.x** files with baselines to determine success or failure of the tests.

Lastly, obviously, the UNIX control machine and all of the PC’s share a file system. This means that the UNIX machine can write to files that the PC’s can read, and vice versa.

D.2 CAPBAK/MSW Script Structure

The central mechanism for control, i.e. the semaphore that is used to assure coherence of the communciation between the UNIX machine and the PC's running *CAPBAK/MSW*, is the presence / absence of a single "semaphore file".

Here is the structure of the control sequence as seen from the *CAPBAK/MSW* side:

```
while (semaphore file created by SMARTS doesn't exist)
         {
         wait;
         }
 Finish and create semaphore file for SMARTS
```

In effect, the *CAPBAK/MSW* process is kept in a wait loop until the file that is preventing it from continuing actually appears. Once it appears, the *CAPBAK/MSW* activity proceeds normally, and ends by creating a corresponding semaphore file for use by *SMARTS*. (Presumably, then, *SMARTS* on the UNIX machine will erase the original file.) .

D.3 SMARTS ATS Script Structure

Here is the control script that is launched by *SMARTS*, running on the UNIX box, that stays in a wait loop until the answer is received by the *CAPBAK/MSW* script:

```
Clean up (remove) prior semaphore files.

Create semaphore file for CAPBAK

while (semaphore file created by CAPBAK doesn't exist)
        {
wait;
        }
signal SMARTS is complete
```

D.4 Example Scripts

Following are actual scripts that show these principles in operation.

D.4.1 sync.ats

This is the script that would be processed by *SMARTS* on the UNIX machine.

```
/* CAPBAK/MSW Application Note Scripts.


(c) Copyright 1994 by Software Research Inc.
 ALL RIGHTS RESERVED.


This script is part of a pair that show how to operate
CAPBAK/MSW from SMARTS running on UNIX assuming that
the machines share a common file system. This file is
an “ats” file that is read by SMARTS. *//* The group
sync allows SMARTS on Unix and Capbak MSW on a PC to
synchronize using semaphore files. */


define group sync


/* CAPBAK on a PC is waiting for a file to be created
before it continues a playback session. SMARTS creates
that file, signalling CBMSW to begin. SMARTS then
waits for CAPBAK to finish before it completes. It
knows CAPBAK has finished because just before CAPBAK
ends, CAPBAK creates the file that SMARTS is waiting
for. */
{
         define case setup


         /* This case cleans up for the beginning of
         this test */
         {
         source “Cleaning up for synchronization”;
         activation
         “rm sync.txt”,
         “rm sync.r01”,
         “rm sync.out”;
         noevaluation;
         }
         define case create


         /* This case creates the file that CAPBAK is
         waiting for so it can continue its playback.
         */
```

```
            {
            activation
            "sleep 5",
            "touch sync.txt";
noevaluation;}

            /* This section waits for the file sync.r01
            to be created before SMARTS continues. When
            CAPBAK MSW finishes, it createsthis file.
            SMARTS then knows it can finish. */

            while ("test_it sync.r01")
            {

            /* While the file sync.r01 doesn't exist,
            sleep */

            define case sleep{source";
            activation
            "sleep 5";
            noevaluation;
            }}
            define case finish

            /* Signal the user that SMARTS has fin-
            ished.*/
            {
            source" ";
            activation"cp sync.rep sync.out";
            evaluation with baseline
            "sync.out" vs. "sync.bsl";
            }
}
```

D.4.2 sync.ksv

Here is the script that is read by *CAPBAK/MSW* on the PC running MSWindows.

```
/*
 * CAPBAK MS-WINDOWS Release 2.0
 * (c) Copyright 1993, Software Research, Inc.
 * ALL RIGHTS RESERVED
 * Session started on Fri Oct 28 10:44:41 1994
 */

/* CAPBAK/MSW Application Note Scripts.

(c) Copyright 1994 by Software Research Inc. ALL
RIGHTS RESERVED.

This script is a playback script that interacts with
files that are controlled by SMARTS running on a UNIX
machine but sharing the same file system. */

void sync()
{
         int i;

         i = 0;
         cb_move_mouse(54, 148, 0);
         while ( i == 0 )
         {
         i = cb_file_exists("l:\\tmp\\sync.txt");
         if (i==1)
         break;
         cb_wait(5);
         }
         cb_move_mouse(52, 147, 275);
         cb_move_mouse(50, 145, 55);

         /* In the next section the keysave file tells
         CAPBAK MSW to do various things ...Then just
         before it finishes, CAPBAK saves a section of
         the screen to a file. SMARTS is waiting for
         this file to exist.When it sees the file,
         SMARTS ends, too. */
```

```
            cb_key_down("VK_CONTROL", 2252);
            cb_key_down("VK_MENU", 164);
            cb_key_down("VK_F6", 165);
            cb_key_up("VK_F6", 110);
            cb_key_up("VK_MENU", 55);
            cb_key_up("VK_CONTROL", 0);
            cb_save_window("L:\\TMP\\SYNC.b01", 148, 51);
            cb_move_mouse(148, 51, 0);
            cb_move_mouse(204, 58, 2087);
            cb_move_mouse(206, 58, 55);
            cb_move_mouse(207, 58, 220);
            cb_move_mouse(214, 58, 55);
            cb_move_mouse(220, 58, 55);
            cb_move_mouse(221, 58, 0);
            cb_move_mouse(223, 58, 165);
            cb_move_mouse(224, 58, 109);
            cb_move_mouse(230, 58, 55);
            cb_move_mouse(231, 58, 110);
            cb_move_mouse(231, 59, 714);
            cb_mbutton_down("LEFT", 231, 59, 55);
            cb_mbutton_up("LEFT", 231, 59, 275);
} /* End sync() */

sync();
```

Index

Numerics

A

B

C

D

E

F

H

I

K

L

M

N

O

P

V

W