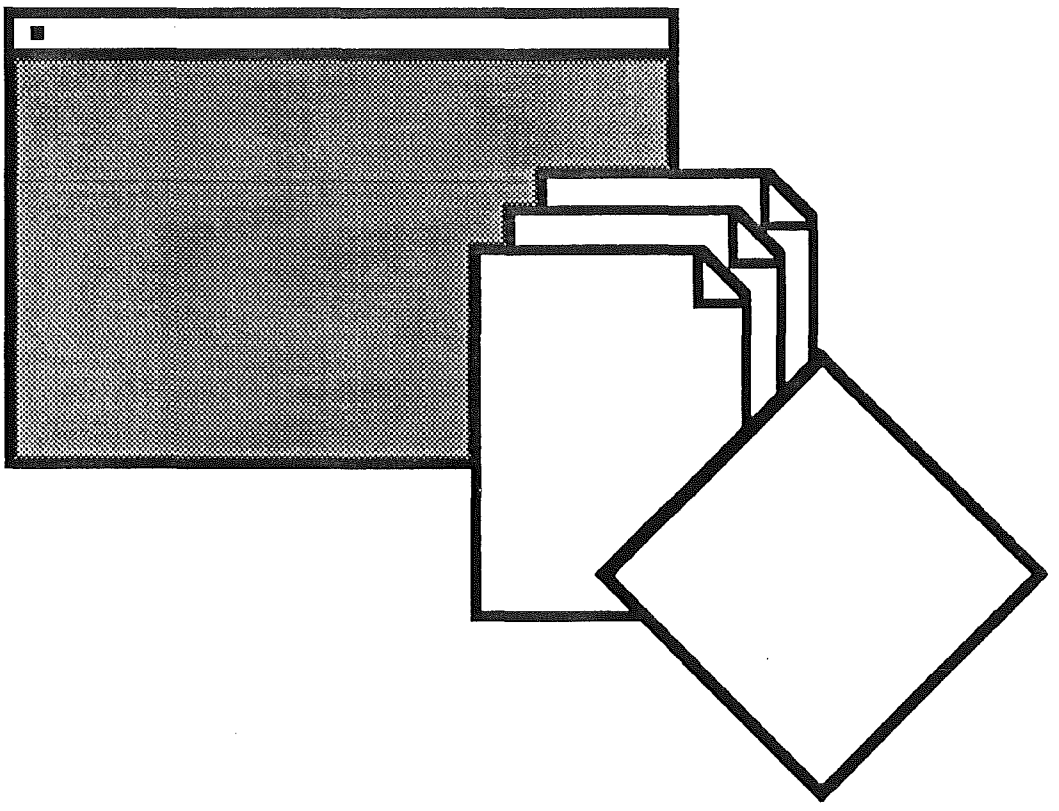
# Genesys™

**The Resource Editing, Creation and Maintenance Tool for the Apple IIGS®**



**Simple Software Systems International, Inc.**
4612 North Landing Drive
Marietta, Georgia 30066
(404) 928-4388

*Innovative Software for Innovative People*

4612 North Landing Drive NE, Marietta, GA. 30066 (404) 928-4388

# Notice

***This sheet contains last minute release information***

The version shipped in this package is version 1.2.4, which has been updated to correct some minor bugs to version 1.2.

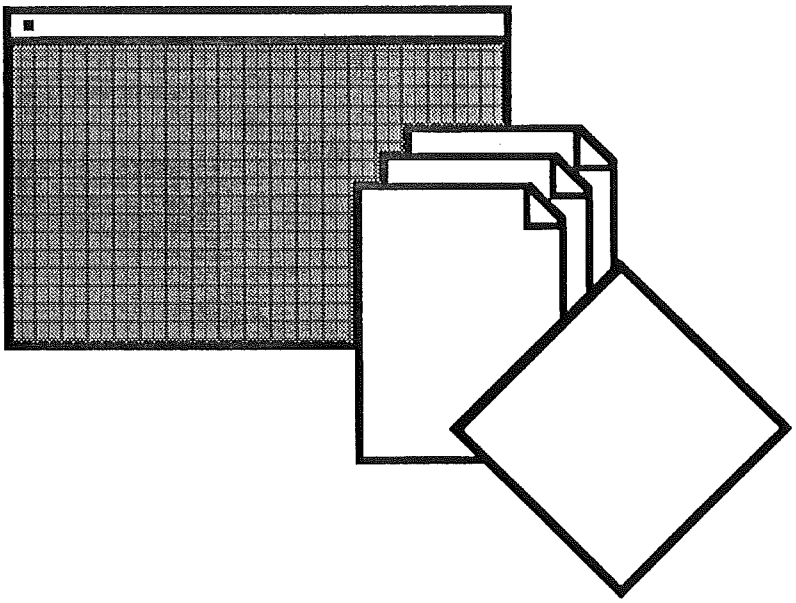
Genesys should be used with version 5.0.3 or greater System Disk. These disks are available from America Online™, GEnie™, CompuServe™ and other information networks, as well as your local Apple Dealer at no charge.

The manual refers to launching and installing the System and Genesys from the enclosed system disks (see above note). The Installer is located on the `Genesys` program disk.

The manual refers to the `Genesys.2` disk. This is the volume name of the disk labeled `SCG Templates`.

# Genesys™

## The Resource Editing, Creation and Maintenance Tool for the Apple IIGS®




Simple Software Systems International, Inc.
4612 North Landing Drive
Marietta, Georgia 30066
(404) 928-4388

Manual version 1.2

1st printing April 1990
2nd printing May 1990
3rd printing October 1990

# Table of Contents

# Preface

## For the Explorer

If you enjoy the adventure of trying out a new program without reading its reference manual, then feel free to run Genesys and explore. However, be forewarned to make backup copies of all programs (including Genesys) BEFORE you modify them. Genesys is very powerful and will not prevent you from recklessly altering ANY program. Some programs may not tolerate your experimentation and may not work properly - SO BE CAREFUL!

If you have questions while using Genesys, use the context-sensitive Help feature. Choose **Help** from the Apple menu or press ⌘-**?** from the keyboard. Be sure to read the **Release Notes** in **Help** for any last minute changes. After you have explored Genesys, we highly recommend a complete reading of the entire manual.

## About This Manual

The modular nature of Genesys is reflected in this manual. Each Chapter starts over with page 1. This numbering sequence allows quick location of information and minor updating of the manual without addenda or errata sheets.

### Manual Structure

This *Table of Contents* is used to find the Chapters and sections in this manual.

This *Preface* provides information on system requirements, manual conventions, and additional references you may need and how to obtain them.

Chapter 1, *Welcome to Genesys*, will explain to you what Genesys is, how it works, and how you can use it. It is vital you read this chapter to familiarize yourself with what Genesys actually does.

Chapter 2, *Installing Genesys*, shows how to make backup copies and install Genesys and the System Software provided.

Chapter 3, *Editing a Program*, will show you how to edit an existing program and change its menu items and window positions. The program you will edit is, in fact, a copy of Genesys that you will change and run without writing a single line of programming code.

Chapter 4, *Creating a Program*, will show you how to create a program using Genesys. The program that will be created shows how Genesys can help you design and get a program up and running in no time.

Chapters 5, *The Genesys Shell*, gives precise details on how Genesys and source code generation works. Although intended as a reference, you should read this Chapter to fully familarize yourself with Genesys.

Chapter 6, *The Genesys Editors*, describes each Editor in detail and how they are used. New sections will be added or replaced to this chapter as Editors are created and updated. Again intended as a reference, you should read this Chapter to fully familarize yourself with each of the Genesys Editors you will be using.

Appendix A, *Source Code Generation Language*, provides detailed information on how you can modify Genesys to create source code in any style or language.

Appendix B, *Writing Your Own Genesys Editors*, provides information on how you can write your own Genesys Editor.

Appendix C, *Updates and Support*, provides information on obtaining support and gives the Genesys program credits and acknowledgements.

The *Glossary* has the definitions for words used in this manual. Make sure you understand the meaning of the words used in this manual for complete comprehension.

## Manual Conventions

The following designations are used in this manual:

| | |
|---|---|
| ***Note***: | An interesting or incidental remark. |
| ***Tip***: | A short cut or recommendation. |
| ***Warning***: | Important information to prevent an accident or trouble. |
| *Term:* | The first time a new term is introduced it will appear in italics. These terms are defined in the Glossary. |
| **Menu...** | A menu, window or control title. |
| | Apple menu in the menu bar. |
| ⌘ | Apple key on the keyboard. Usually used in conjunction with another key, such as "press ⌘ - **?** for **Help**." |
| `filename` | The name of a file or folder. |

- *Note:* The Control Panel *New Desk Accessory* (NDA) supplied with the System Disk has a setting for *Keyboard Translation.* In order for the **option** key to work with controls, the translation must be set to **None**. Genesys will automatically save your old setting and set it to **None** for you. When you quit Genesys, the original setting will be restored. This is the Apple preferred way of handling key equivalents. If you restart your computer without exiting Genesys using the **Quit** menu item, the Control Panel setting will remain set to **None**.

## System Requirements

In order to use Genesys, you must have the minimum hardware and software specified. Please note the recommended hardware.

### Hardware Required

- Apple IIGS or Apple //e with Apple IIGS upgrade installed (ROM 01 or greater).
- 768k of RAM.
- One 3 1/2" disk drive.

The following hardware is highly recommended, especially if you intend to do work on large programs:

- Additional memory. You can obtain up to 4 meg of memory from SSSi.
- A second disk drive. Hard disks are highly recommended.
- A printer.

### Software Required

All software required has been furnished on the disks provided. Make sure to follow the instructions for installation to be sure you have the latest System software.

- ***Warning:*** Genesys requires a minimum of System Software 5.0.3 to work correctly! Earlier versions of the 5.0 operating system have a bug that can crash Genesys and corrupt your files. Genesys is always shipped with the latest version of the System Software available.

- ***Note:*** Make sure you open **Help** from the  menu and read the release notes.

## Additional References

You should have access to the following references if you intend to do any serious development work with Genesys. These references are available from:

A.P.D.A
Apple Computer, Inc.
20525 Mariani Ave.
Cupertino, CA 95014
(800) 282-2732

### Toolbox Reference Manuals

*Apple IIGS Toolbox Reference Volumes 1, 2 and 3*

Provide information on tool calls and are essential for program development.

### GS/OS Reference Manuals

*GS/OS Reference Volumes 1 and 2*

Provide information on operating system calls. Essential for program development.

### Other

*Programmers introduction to the Apple IIGS*

Examples in Pascal, C and Assembly help you learn to program the Apple IIGS.

*Human Interface Guidelines*

Detailed guidelines covering acceptable interface design criteria. These guidelines should be in every Apple IIGS users library.

# Chapter 1: Welcome to Genesys

## What Is Genesys?

### The Way It Was

If you understand anything at all about programs, you know by now that a program consists of computer instructions that tell the computer what to do. This is half true, as a computer program consists of 2 parts: the computer instructions, and the *data* that the computer instructions use. For example, the program may contain computer instructions that tell the computer to draw an *icon* on the screen, but the program must also contain the *data* for the icon that defines its appearance, color and size.

In the course of creating a program the typical programmer decides what the program will do and how it should look and present information to the users of the program. The programmer uses the Apple IIGS *tool calls* to help save time. Creating a *desktop* interface, with *pull-down menus*, *windows*, icons, and the like would take a VERY long time without these important tool calls.

After creating and verifying the program code and data structures, the programmer then *compiles*, *links*, and *executes* the program. Assuming the data structures are correct the first time (unlikely) and the program code is correct the first time (unlikely), the programmer sees the beginnings of the program. But the programmer notices the icons aren't quite right, the positions of windows aren't quite right, and some menus need to be added.

At this point the programmer stops the program from executing (if it hasn't stopped by itself), re-edits the program code, changes the position of the icons and windows, re-compiles, re-links, and re-executes the program. This time the icons are too far over because of a calculation error, and the programmer forgot to add those needed menus that didn't get added previously. The programmer also realizes that a few more windows are needed.

This process continues until a.) the programmer finishes the program in spite of it all, usually months after the program was needed, b.) the programmer decides the nice interface that was designed isn't worth the effort, or c.) gives up. It gets worse when dealing with large programs. A large program can contain hundreds of icons, windows, menus and different types of data structures. This leads to poor interface design due to time constraints, a high resistance to program code changes for existing and future enhancements, and lack of enthusiasm to learn Apple IIGS programming.

---

**A Better Way**

Genesys was created to make it faster and easier to create, edit, and maintain programs. Genesys lets you create the results you want before you even begin to type, compile, or link a single line of code. Program design and development takes weeks instead of months. Program maintenance takes hours instead of weeks. Communicating and experimenting with program ideas is as easy as pointing and clicking. Changing existing programs to look and feel the way YOU want them to becomes a reality. Using Genesys, the learning curve for programming the Apple IIGS has been cut down to months instead of years.

Professional programmers can use Genesys to create, edit and maintain existing programs. The days of tedious positioning of user interface elements is over. And since Genesys allows creation of source code in ANY language, you don't have to learn a new way of doing things. Genesys works the way you do.

Beginning programmers can develop programs that look and feel like professional programs while learning more about Apple IIGS programming. You can study the source code generated by Genesys and learn the way the Apple IIGS works and see the techniques used by professionals to write high-quality programs.

Non-programmers can use Genesys to edit programs and change menus, key equivalents, windows, and generally custom tailor programs to their needs. New program designs can quickly be generated and given to the professional to complete.

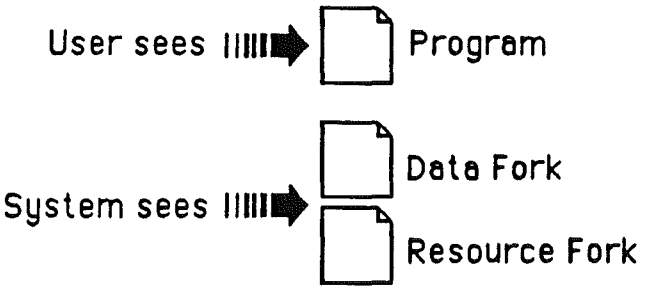
Educators can use Genesys and teach the latest techniques of computer programming showing real examples of running programs with fully commented source code.

How does Genesys do all this? Using *Resources*.....

## What are Resources?

### A Resource Primer

A *resource*, as defined by Apple Computer, is "a formatted collection of data, such as a menu, a font, or a program itself." Resources were introduced to the Apple IIGS with the release of Apple's System Disk 5.0 in May of 1989. Resources allow a program to keep its program code in one place in a file (referred to as the *data fork* or more appropriately, the *program fork*), and its data structures in another place in the file (referred to as the *resource fork*). For example, the programmer can now place all the menu bar definitions in the resource fork for the program, and make one simple tool call to allocate and draw the menu bar. In the past, the programmer had to define all the declarations for the menu bar in the program itself.




As the above illustrates, the user or programmer sees the entire program as a single file or icon in the Finder. The Apple IIGS operating system sees this file as it really is, a file composed of two parts - the data fork containing the program AND the resource fork containing the resource definitions that the program will use for its menu bars, windows, icons and the like.

- ***Warning:*** Pre-System 5.0 file copying utilities only copy the file's data fork and not its resource fork. Use the Finder on the System Disk provided to copy files.

### The Resource Compiler

Apple also introduced the *Resource Compiler* with System 5.0. The resource compiler takes the data for the program, written by the programmer in a new language called *REZ*, and creates the resource fork. This process is just as tedious as placing the data for the program in the program code itself, but does give you access to resources for your program, although you have to learn ANOTHER language.

You will also need to use *Duplicate*, another utility from Apple. Duplicate will attach the resource fork you've created and compiled to the finished program that uses resources.

### The Resource Editor

The best method to use in creating a program with resources is to use a resource editor. A resource editor should, in a fashion, allow you to directly edit a program's resources, and change them. It should also allow you to create resources and attach the resource fork to your program.

Although we use the term loosely, Genesys is classified as a resource editor. Genesys allows you to create AND modify program resources directly (goodbye REZ source code and compiler) and attach your newly created resource to your program (goodbye Duplicate utility). The similarity stops there. Genesys immediately exceeds the "resource editor" definition by adding a host of features not found anywhere, such as full program source generation for ANY language (including REZ), full editor and language extensibility, and a context sensitive help system, just to name just a few.

- *Note:* If you need further information on resources or resource forks, please refer to your Apple IIGS Toolbox and GS/OS Reference manuals.

Now that you understand Genesys and its use of resources, continue on with Chapter 2, *Installing Genesys*, and Chapter 3, *Editing a Program*.

# Chapter 2: Installation

## What to Install and How

Genesys comes with four diskettes: the Genesys *Program Disk* containing Genesys and its files and folders, the *Templates Disk* containing the Genesys *Source Code Generation Templates*, a *System Disk* with the most recent version of the Apple IIGS system software, and a *System Tools* disk containing the *Installer* and utilities.

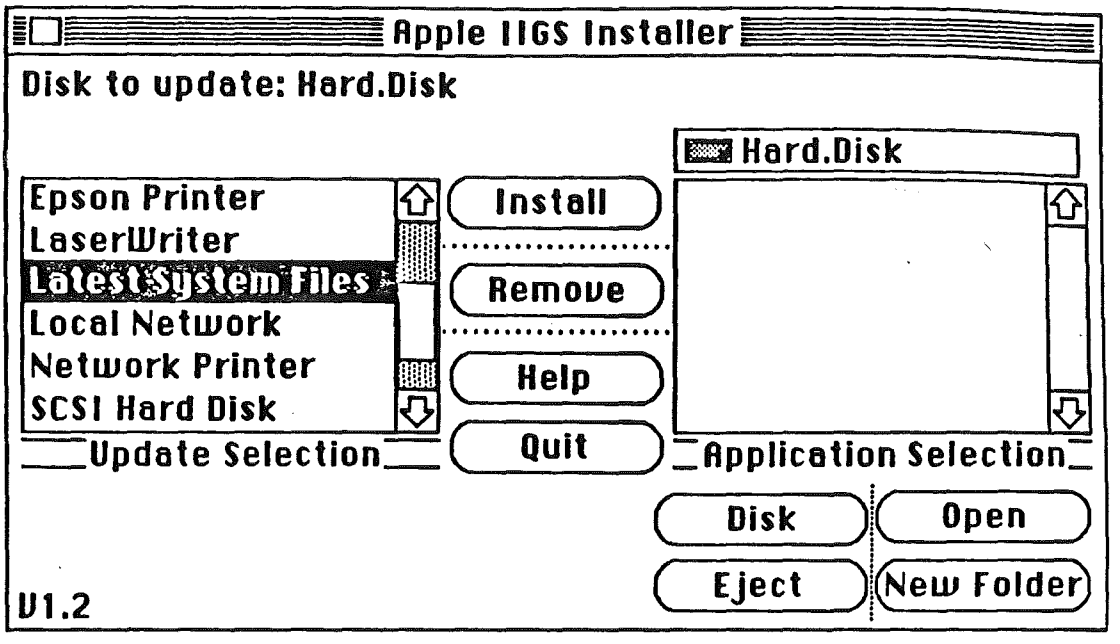
- *Tip:* If you are not familiar with using the Finder, refer to the appropriate Apple IIGS manual that came with your computer.

1. Insert the System Disk provided and turn on your Apple IIGS.
2. Using the Finder, make a *Backup Copy* of each disk in the Genesys package, and put the originals in a safe place.

- *Note:* You only need to install the System Software and the Genesys program if you are using a hard drive. If you are NOT using a hard drive, proceed to *Chapter 3* using the backup copies you've made.

### Installing System Software

- *Note:* In the unlikely event that you have a version of the System Software that is later than the one shipped with Genesys, do NOT install the System Software as directed, and proceed to the section *Installing Genesys.*




1. With the Finder running place the `System.Tools` disk in a drive.
2. Open the disk and run the program `Installer`.
3. Select **Latest System Files** from the left list.
4. Select the hard disk you will install the System Software to using the **Disk** button.
5. Select the **Install** button and wait until the Installer completes the installation.

---

### Installing Genesys

- *Tip:* If you need more information about the installations you will be doing, select the name from the left list and click on the **Help** button.

### Install Genesys

1. Select **Genesys** from the left list.
2. Select the hard disk you will install Genesys to using the **Disk** button. You can install Genesys in a folder or create a new folder to install Genesys to.
3. Select the **Install** button and wait until the Installer completes the installation.

### Install Genesys SCG Templates

1. Select **Genesys SCG Templates** and the **Install** button to install the *SCG templates* to the same disk/folder you installed Genesys. If not, you will be asked for the templates disk (`Genesys.2`) EVERYTIME you generate source code.

### Install Genesys.H

1. If you plan on using the APW C compiler to compile source code generated with Genesys, select **Genesys.H for APW C** and the **Install** button to install a header file that is referenced in the source code generated by Genesys. You should install this header file in the same place you keep your other APW C header files.

### Install Genesys.Icons

1. If you would like the icons supplied with Genesys to be utilized in the Finder, select **Genesys Icons** and the **Install** button.

### Install Genesys Font

1. Genesys has an option to view source or text files. If you would like to view these files with a non-proportional font (preferred), select **Genesys Font** and the **Install** button.

You have finished installing Genesys. Select the **Quit** button and proceed to Chapter 3, *Editing a Program.*

# Chapter 3: Editing a Program

## Editing Genesys

### Copying and Launching Genesys

The first thing we want to do BEFORE we begin our first Genesys editing session is to make a copy of the Genesys program. Why? Because that is the program we will be editing! If you are not sure how to use the Finder, refer to the appropriate manual that came with your computer. Assuming you have the Finder up and running, proceed with the following steps:

1. Locate the file Genesys and click on it to select it.
2. Select **Duplicate** from the **File** menu. Accept the new filename Genesys2.
3. Double-click on Genesys to run it.

- *Note:* If you are using a 3 1/2" drive system, you'll want to copy the file Genesys to a work disk first, as you won't have enough room on the Genesys program disk to make a copy of it. Simply format a blank disk right in the Finder and drag the file Genesys to the work disk. Then rename the new file to Genesys2 so it will be easier to follow along.

After a few moments, you'll be in Genesys, ready to begin your editing adventure.

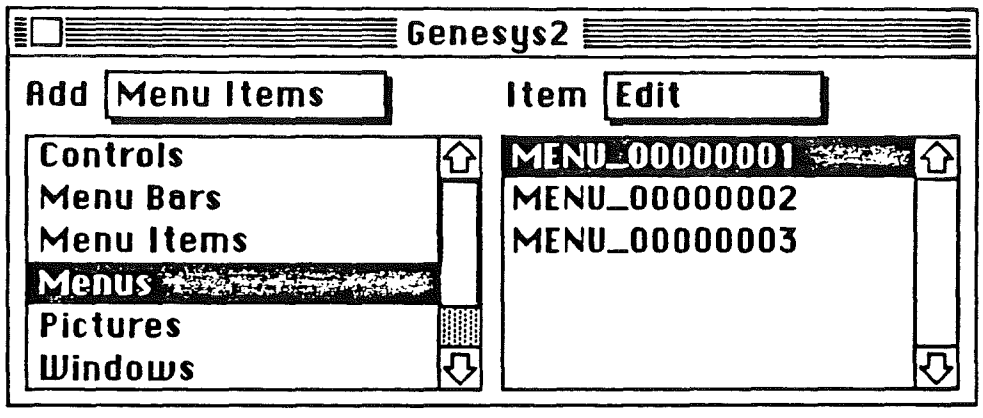
### Opening Genesys2

The first thing you'll need to do is open the newly created file Genesys2. Pull down the **File** menu and select **Open...** You'll be presented with a *Get File* dialog. Locate the file Genesys2, click on it, and select the **Open** button. You will be presented with the *Resource Chooser* window. This window shows the *types* of

resources (the left list in the window), and the items for each type (the right list in the window).

- *Note:* Resources are always referred to in this way. Every resource will have a type (such as a menu), and each type will have associated items (such as a File menu, Apple menu, etc).

You should have a resource chooser window somewhat like this:

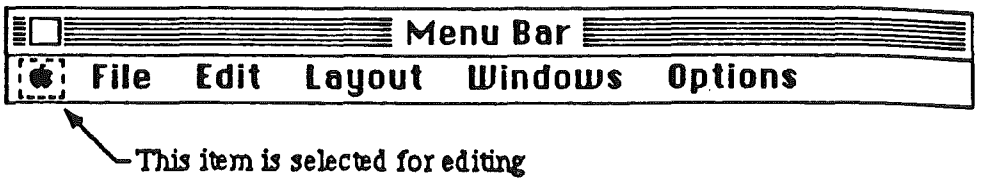



---

### Editing the Genesys2 Menus

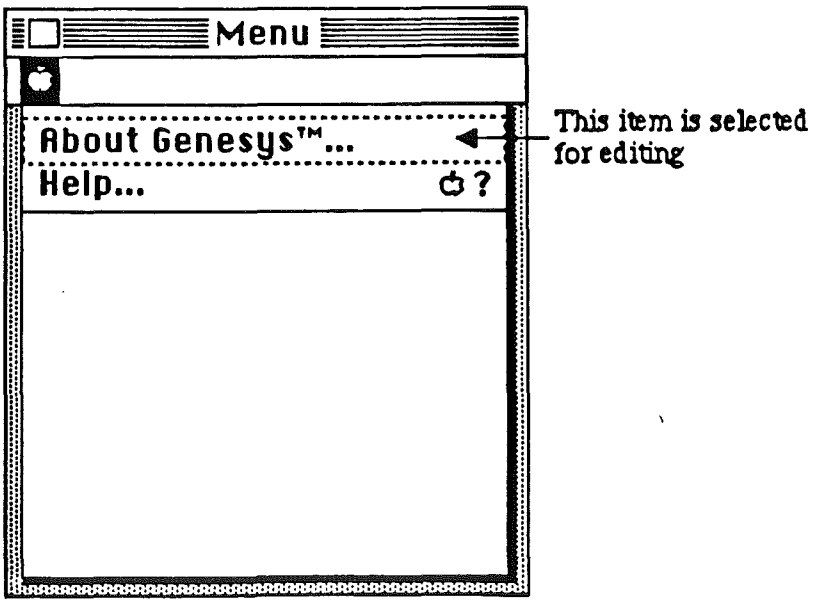
What we are going to do now is change the menu **Help** to say **HELP!!!** and add a key equivalent to the **About Genesys™...** menu item as well as change its appearance. We'll also play some tricks with the menus in the menu bar.

To edit the Genesys menu bar, select **Menu Bar** in the types list (the left list). You'll see the Genesys menu bar in the items list (the right list) which says **Genesys Menu Bar**. This is the Genesys menu bar. To edit this item, select the **Item** popup above the items list, and select **Edit**. You will be presented with the Genesys menu bar. This is the *Menu Bar Editor*.



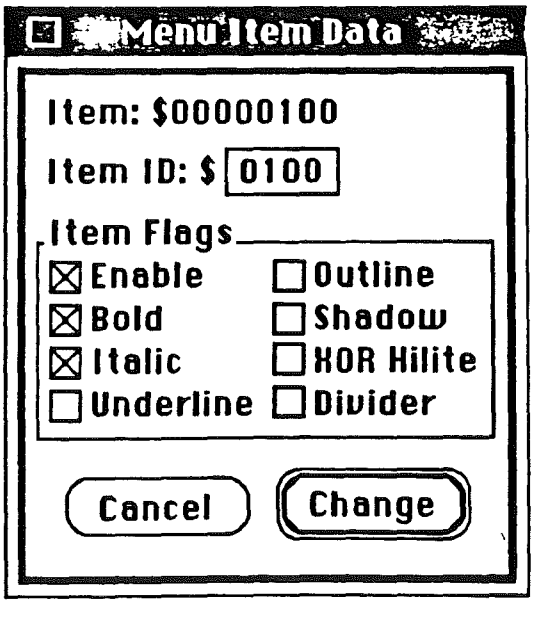

Each Editor has its own editing window, and you can usually open as many editing windows as you need at any one time. You may also notice that when you opened the Menu Bar Editor, the **Options** menu, which has been dimmed up to this point, became active. The **Options** menu is strictly for the use of an Editor. Each separate Editor (in most cases) places its options in this menu. Most of your editing is done by the use of the Editor's **Options** menu or by clicking and typing in the Editor's window.

The Menu Bar Editor's first item in the **Options** menu is **Edit menu...(⌘E)**. This menu becomes active when a menu in the menu bar is selected for editing. To select a menu for editing, click on a menu in the Menu Bar Editor's window, or choose **Next item (⌘>)** from the **Options** menu. Click on the  menu. The red selection rectangle should be marching around the  menu (referred to as "*marching ants*"). Select **Edit menu...** from the **Options** menu to edit the  menu. Another window will appear that represents the menu items in the  pull down menu. This is the *Menu Editor.*



- *Note:* Notice that this window has a red-lined title bar. This is your indication that you are using a Genesys *Sub-Editor*. A Sub-Editor is functionally equivalent to any other Editor, but this type of Editor expects you to complete the action in the window before proceeding.

Select the **About Genesys™...** menu item by clicking on it. The marching ants should be on this item at this time showing that the item clicked on is the one that editing actions will apply to. Pull down the **Options** menu and select **Edit item data (⌘E)**. You will get another window titled "Menu Item Data". This window allows you to change the menu item's ID number (DO NOT DO THIS), and its appearance. Let's change the appearance of this menu item. Click on the **Bold** and **Italic** check boxes. Notice that these changes take an instant effect. If you did not wish to keep the changes you made, you would click on the **Cancel** button to cancel all changes. Accept the changes made by clicking on the **Change** button. The window closes, and you are back to the Menu Editor.



- *Note:* You have just used another type of window in Genesys known as a *Movable Modal Dialog*. This type of window was designed to have the function of a regular dialog window, but allows you to move the window around the desktop to see the effects of the changes you are making to the window underneath. Movable Modal Dialogs have solid red title bars.

Let's add a *key equivalent* to the **About Genesys™...** menu item. This will allow you to press the ⌘ key in conjunction with another key so you do not have to pull down the menu. Since this menu item is still selected, we can go right into editing its text. To do this, double-click on the item (or press the **return** key). You will now have three text entry fields (*line edits*) in place of the menu item. The first line edit is for a character that will be used in front of the menu item. The second line edit is for the text of the menu item, and the third line edit is for a key equivalent. You can click on a field or press the **tab** key to move between these fields. Press the **tab** key now to place the cursor in the key equivalent field. Enter an upper and lower case **G** on this field, and click outside of the line edits to complete your text editing.

- *Note:* Menu items always have two characters associated with them that are the upper and lower case equivalents of the character equivalent (if used). It shouldn't matter if the user has the **Shift** key held down when using a menu equivalent. If you only enter a single character in this field, the Menu Editor will duplicate the character entered. In our example, if we had only entered the character **G**, it would be used twice, and the user would be REQUIRED to press the **Shift** key for the menu item!

Now try to edit the **Help** menu item to read **Help!!!**. You can also experiment with changing the data for this menu item, changing its keyboard equivalent, or adding another menu item of your own. When you are finished, click in the Menu Editor's close box to return you to the Menu Bar Editor, and click in the Menu Bar Editor's close box to return you to the Resource Chooser Window.

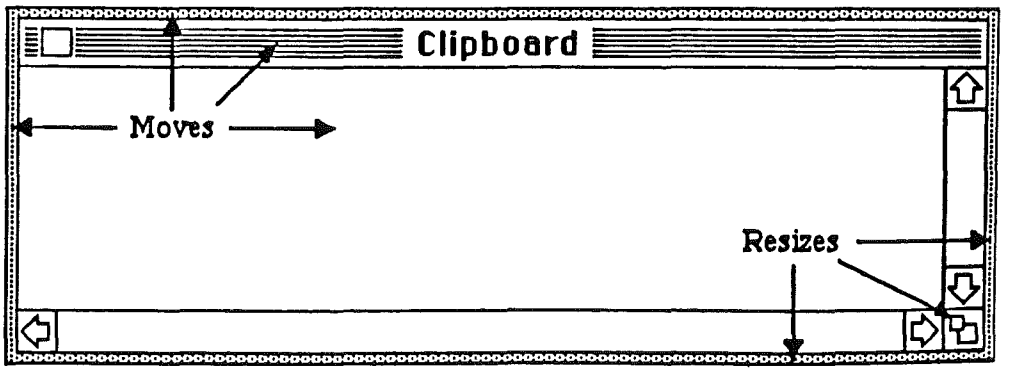
- *Tip* If you do something you didn't want to do, and want to restore the file to the way it was, close all the windows and select **Revert...** from the **File** menu. And don't forget to select **Help** from the menu if you need it!

### Editing the Genesys2 Clipboard Window

Genesys has a window it uses to display the contents of the clipboard known as the Clipboard Window. Just to get a feel for editing windows, let's change the Clipboard Window's size and position on the desktop.

Click on **Window** in the Resource Chooser Window's type list. You should see a window in the items list labeled **Clipboard Window**. Edit this item by selecting **Edit** from the **Item** popup menu. The window will appear with a yellow border surrounding the actual window. This is the *Window Editor*.

Resize the window by clicking and resizing the window using it's grow box. Move the window to the position you desire by dragging it by it's title bar. You can also add/remove the zoom box at this time using the Window Editor's **Options** menu.



- *Note:* There may be cases in which you will create or edit a window that has no title bar or grow box (such as a dialog window), but still need a method to resize and reposition the window. The Window Editor uses the yellow border surrounding the window for this function. Any window can be positioned by dragging the window by its top or left sides. Dragging a window by its right or bottom sides will resize the window (this works for Controls too). And since there is no close box, any window can be closed by selecting **Close (⌘W)** from the **File** menu.

When you are finished positioning and sizing the window, click in its close box to close it, or select **Close** from the **File** menu.

---

### Saving and Running Genesys2

Select **Save (⌘S)** from the **File** menu to save the changes you've made to this file. If you decided at this time to save the file under a different name, you could do so using the **Save as...** menu item.

Select **Quit (⌘Q)** from the **File** menu to return to the Finder. When you are back in the Finder, run the `Genesys2` program that you've been editing.

- *Warning:* Make sure the file `Genesys2` is in the same folder as the program `Genesys`. Genesys needs access to the other files and folders found here. If you were editing `Genesys2` from a 3 1/2" disk, you should now rename `Genesys2` to `Genesys` and copy it over the existing `Genesys` on your program disk. DO THIS ONLY ON YOUR BACKUP COPY!

After you get into the modified Genesys program, pull down the  menu to see the changes you've made to this menu. Pull down the **Edit** menu and choose the **Show clipboard** item to see the results of your window editing. Keep in mind you made these changes to the program without typing, compiling or linking a single line of code!

## Where to Go From Here

In this chapter we did some simple editing of a program. Although it seems quite a bit was covered, you've only scratched the surface exploring the capabilities of Genesys. Now that you have the general idea of how Genesys operates, you should have no trouble at all editing other programs that have resources.

Make sure you take a look at the sections for each of the Genesys Editors to get a thorough understanding of what they can do. This chapter only covered a few points on a few of the editors. There are many more editors.

If you purchased Genesys as an aid to programming, make sure you read and follow along with the next chapter, *Creating a Program*. This chapter will guide you on the process of creating a program interface, generating source code, compiling, linking and running the program you will be creating with Genesys.

The appendix on source code generation is a must for any person that wishes to tailor the source code output that Genesys creates to meet individual needs or tastes. Also included are appendices on creating your own Genesys Editors, and how to obtain support and product updates for Genesys.

Of course, if you just want to go off and experiment with Genesys for awhile, please do so, but make sure you read the note earlier in the manual, *For the Explorer*. After you're done exploring, come back and read the rest of the manual to pick up anything you may have missed.

- *Note:* You may want to make a copy of the `GEN.EDIT` folder, as a few of the Genesys Editors have resource forks you can edit. You can change the Icon Editor's default icon, the Picture Editor's default picture, the Menu Bar Editor's default menu items, and so on. You can make a copy of your `CDEVs` folder, and edit those too. The Control Panel uses these CDEVs to display the icons that appear in the Control Panel's window. You may want to change these icons to your own. DO NOT change the width of these icons. Many are expected to be 28 pixels wide.

# Chapter 4: Creating a Program

In this chapter, you'll be shown just how easy it is to create an Apple IIGS program using Genesys. You will create menu bars, windows, and other elements of the program's interface, generate source code for the program, then compile, link and execute the program you've created.

The first step in creating any program is developing a preliminary design of what the program will do. This "top down" approach to writing a program can usually take weeks, even months, depending on the type of program being created. The interface is a large portion of this definition, but since you are using Genesys, the definition process shouldn't take long at all.

The source code generated in this Chapter is "C". We will be using the ORCA/C compiler from The ByteWorks, Inc. Since Genesys will be creating the source code for you, minimal time will be spent compiling, linking and running the program!

- *Note:* Genesys currently supports full program generation for Orca C, Orca Pascal, TML Pascal and MPW Pascal. Other languages generate the data structures. If you wish to generate full program source code, refer to the Appendix on writing your own Genesys Source Code Generation Templates. A later version of Genesys will allow you to generate source code for any language in any format.

After you become familar with Genesys, you should be able to run through this chapter in about 15 minutes or less. If you've programmed on the Apple IIGS before using Genesys, you'll notice that your development time is cut down dramatically.

## Program Overview

1. The program shall conform to Apple's *Human Interface Guidelines.*
2. The program will have a standard *Menu Bar*, containing standard *Menu Items.*
3. The program will have an *Alert Window*, showing the program's name and credits.
4. The program will support *New Desk Accessories.*

5. The program will have a *Tool Table* to load and start the necessary tools it needs.
6. The program will have a *Window* with various *controls* in its *content.*
7. The program code should be well written and commented for future revisions.

## Getting Started

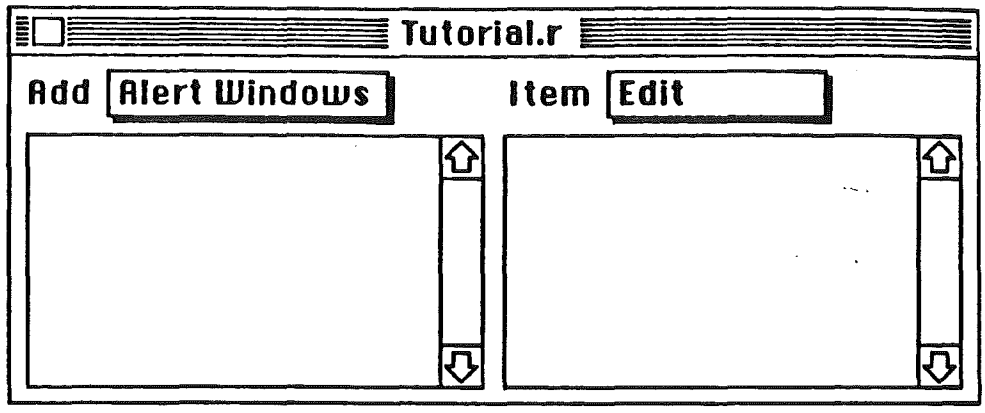
From the Finder, run `Genesys` (or the new `Genesys.2` from the last chapter). Afterthe Genesys credits window is removedyou'll be ready to create your program.

Pull down the **File** menu and select **New**. The resource chooser window will be displayed. For purposes of this tutorial, let's save the current file now and give it a name.

- *Note:* Although there is nothing in the file yet, you can still save the file as it contains information (the *resource file header*).

Pull down the **File** menu and select **Save As...**. A *Put File* dialog will appear allowing you to save the current file on any disk. Call the file `Tutorial.r`. After Genesys safely saves the file, the resource chooser window's title bar will reflect the new name. At any time during this tutorial, simply pull down the **File** menu and select **Save**, or press ⌘**S**, and any changes you've made will be saved.

- *Tip:* Files such as the one you'll be creating are best named with a "`.r`" at the end thus making it easier to recognize the file later as a resource file.

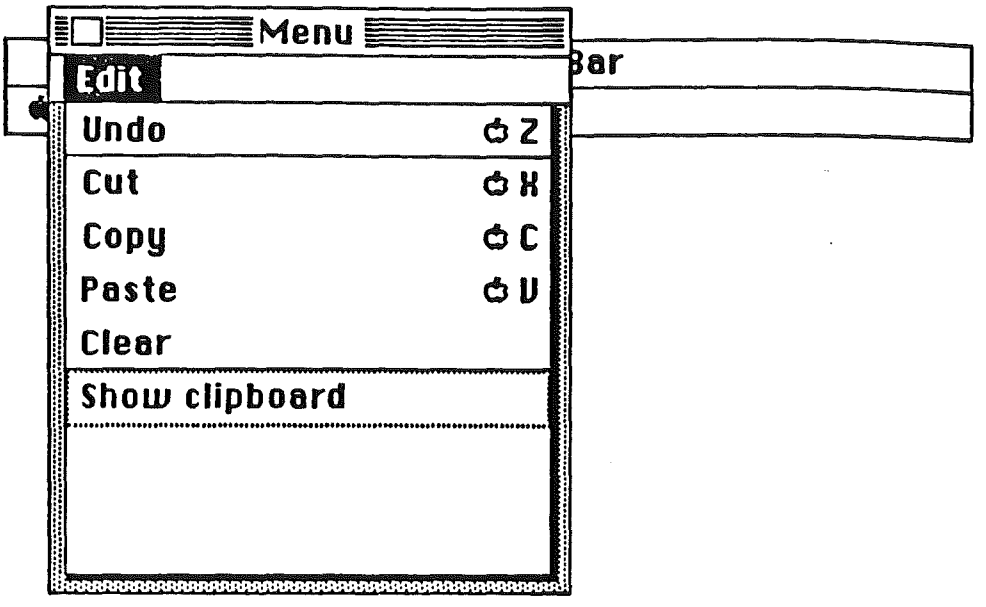



## Creating a Menu Bar

Let's create the program's menu bar. From the resource chooser window, click on the **Add** popup, and select **Menu Bar**. A Menu Bar is automatically created with all the standard items per Apple's *Human Interface Guidelines* and is ready for editing.

- *Tip:* Holding down the option key while adding a Menu Bar will create a empty Menu Bar. You can also make a copy of the Menu Editor (located in the `GEN.EDIT` folder), edit it with Genesys, and replace it. This allows changing the defaults the Menu Bar Editor uses. DO THIS ONLY ON A BACKUP COPY!

Select the Editor's  menu. Now select **Edit menu...** from the Genesys **Options** menu. If you followed the previous chapter, you should be able to change the **About...** menu item to **About Tutorial...** with a key equivalent of ⌘?. Use the **Edit item data...** menu item from the Genesys **Options** menu to add any special highlighting you may want to this menu.




Close the Menu Editor, click on the **Edit** menu, and select **Edit menu...** from the Genesys **Options** menu. Select **Add item** from the Genesys **Options** menu to add a menu item below the **Clear** menu item. Change this new menu item to **Show clipboard**, and add a key equivalent if you want it.

- *Warning:* If you use the same key equivalent in more than one menu, the system will only recognize and act on the first one it encounters.

Edit the **File** menu and add any menu items or special highlighting you want. Remember to consult Apple's *Human Interface Guidelines* when working with menus.

- *Tip:* You can change the order of menus or items by dragging them into position.

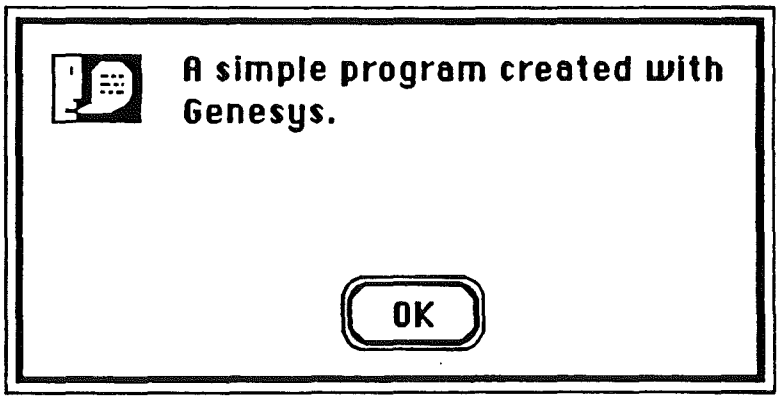
Close the Menu Editor. Select **Test menu bar** (⌘T) from the **Options** menu to "test drive" your menu bar. When you are done testing, click on the desktop or Genesys menu bar to return to editing. Make any further changes and close the Menu Bar Editor.

Press ⌘S or select **Save** from the **File** menu to save your work. If you'd like to return to this section later, select **Quit** (⌘Q).

## Creating an About Window

Most desktop programs have an "About" window showing the program title, author name and other information such as the amount of available memory left. The "About" window is displayed when the user selects **About...** from the  menu.

Our "About" window will be an Alert window containing some text, an icon, and a *default* **OK** button. To create this window, select **Add** from the chooser window's popup and select **Alert string**. A default Alert Window will appear on the desktop ready for editing.




- *Tip:* You can make a copy of the Alert Editor (located in the GEN.EDIT folder), edit it using Genesys, and replace the existing Editor. This allows you to change the defaults that the Alert Editor creates. MAKE SURE YOU DO THIS ON A BACKUP COPY!

The Alert Window Editor will allow you to edit the size, icon, text, and buttons for an Alert Window. Clicking on one of these items will select it for editing, as displayed by the red marching ants. Double-clicking or selecting **Edit...** from the **Options**

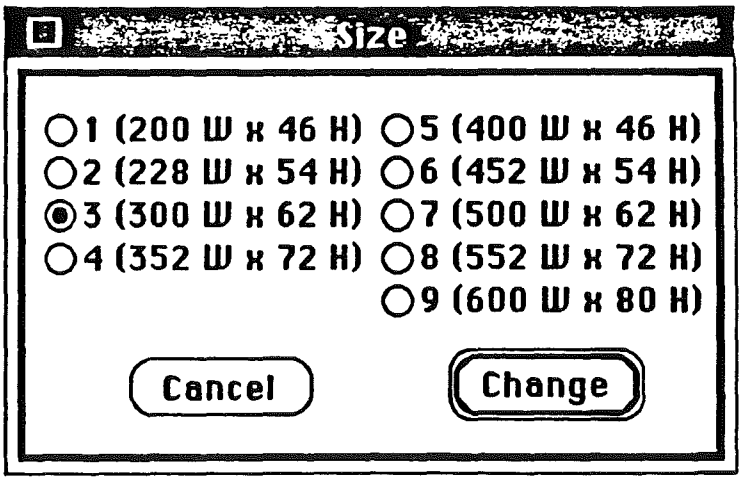
menu on a selected item allows you to edit that items attributes, such as the type of icon or button desired.

To create the Alert Window shown above, you need to remove the **No** and **Cancel** buttons, change the **Yes** button to read **OK**, change the size of the Alert Window, and change the message text. You may also decide to change the type of icon, or create your alert window with a different size or message.

- *Note:* Genesys will automatically use the first Alert Window it finds as the program's "About" window when generating program source code.

Select the interior of the Alert Window by clicking on it, or use the **Next item** (⌘>) or **Previous item** (⌘<) from the **Options** menu. The red marching ants will appear on the selected Alert Window. Now select **Edit size...** (⌘E) and the Alert Window Editor's Size window will be displayed.

- *Tip:* You can also double-click on the Alert Window's interior to edit the size.



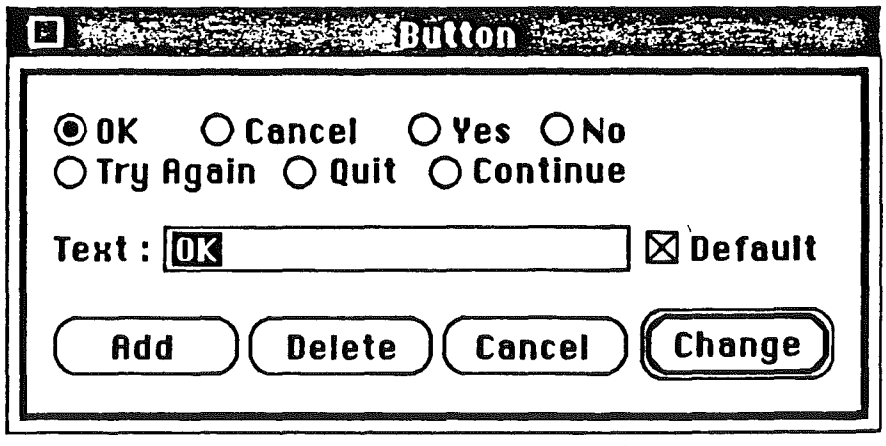

Select any of the nine standard Alert Window sizes, and the size of the underlying editing window will reflect the change. For the window you are creating, you will need size 3, which is approximately 300 pixels wide by 62 pixels high. When you are

finished sizing the Alert Window, click on the **Change** button (or press **return**) to accept the size change, or click on the **Cancel** button (or press **esc**) to cancel the changes made.

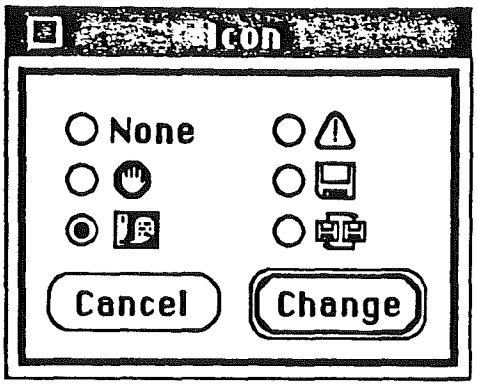
- *Tip:* You can use **Option** 1 through 9 keys to select sizes from the keyboard.

Select the message text using the mouse or keyboard. Type in the message text shown, or any text you desire. Full text editing features are available such as those you would expect to find in any word processor, as well as **Cut**, **Copy**, and **Paste** commands. When you are finished, click anywhere outside of the text editing area.

Editing the buttons is similar to editing any of the other items in the Alert Window. Double-click on the button to be edited or select it and use the **Options** menu. The Button editing window will be displayed, allowing you to select seven different default text strings, or supply your own. Selecting the **Default** check box allows you to specify that one of the buttons responds to the user pressing the **return** key, in which case, the button receives a double outline. Note that only ONE button may be the default button. You may add and delete buttons using the **Add** and **Delete** buttons, although you must have a minimum of one but not more than three buttons in an Alert Window. When you are satisfied with the changes you've made, select the **Change** button (or the **return** key) or discard the changes using the **Cancel** button (or **esc**). For your "About" window, delete the **Cancel** and **No** buttons, and change the **Yes** button to **OK**.



If you wish, you may also change the type of icon for your "About" window in the same fashion, by selecting or double-clicking the icon and specifying which icon you like, although you should use the default "Note" icon provided.



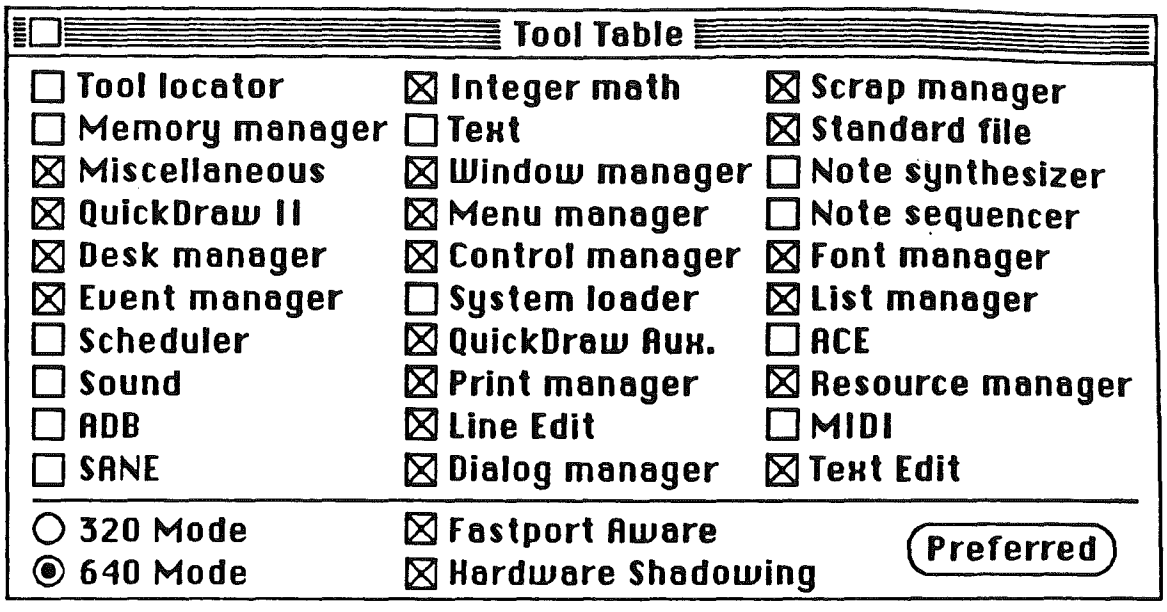

- *Note:* The Apple *Human Interface Guidelines* have much information about buttons, messages, and the type of icons to use in Alert Windows. Please refer to this important manual when designing elements of your user interface.

Select **Test alert...** (⌘T) from the **Options** menu to "test drive" your Alert Window. Clicking in any of the Alert's buttons returns you to the Editor. Make any further changes and close the Alert Window Editor.

Press ⌘S or select **Save** from the **File** menu to save your work. If you'd like to return to this section later, select **Quit** (⌘Q).

## Creating the Tool Table

Every desktop program must specify and load the necessary Apple IIGS *tools* it needs to operate properly. Prior to Genesys, this was a tedious process, as the specification and loading of tools requires knowledge and documentation of the tool numbers, versions, and the format required for each. Using the Genesys Tool Table Editor, this process takes seconds.




- *Note:* The Tool Table Editor does not make use of the **Options** menu.

To specify the Tool Table for your program, select **Add** from the Resource Chooser Window's popup and select **Tool table** from the menu list. A window will appear already set with the preferred tools needed by your program. Click on any additional tools you require, and whether your program will be operating in 320 or 640 mode. Close the window when you are finished.

- *Warning:* Certain tools require the presence of other tools to operate correctly. Since Genesys makes no presumptions about the tools you require or may load by other means, you should be thoroughly familar with the references on the Apple IIGS Tool Locator in Toolbox Volumes 2 and 3 before modifying a Tool Table. Genesys sets up a preferred Tool Table based on those needed by most programs and New Desk Accessories, as well as setting the Fastport aware and Hardware shadowing bits in the Tool Table.

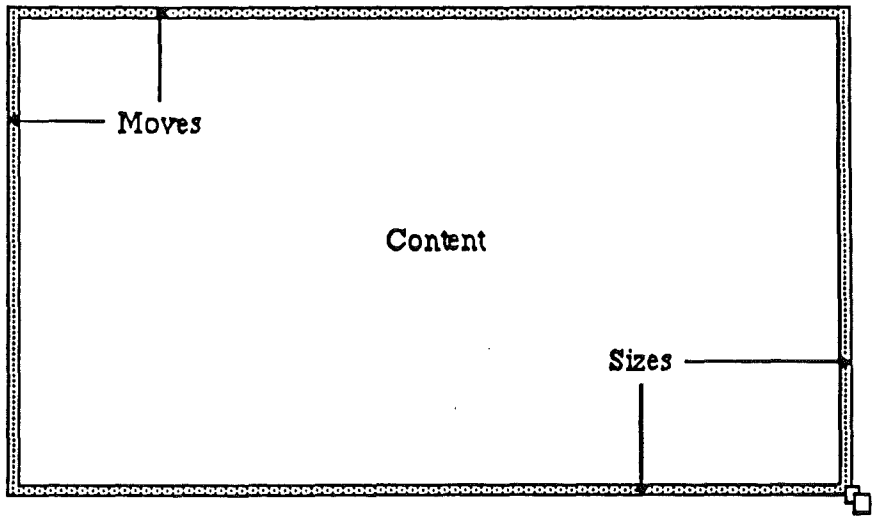
Press ⌘**S** or select **Save** from the **File** menu to save your work. If you'd like to

return to this section later, select **Quit** (⌘Q).

## Creating the Program Window

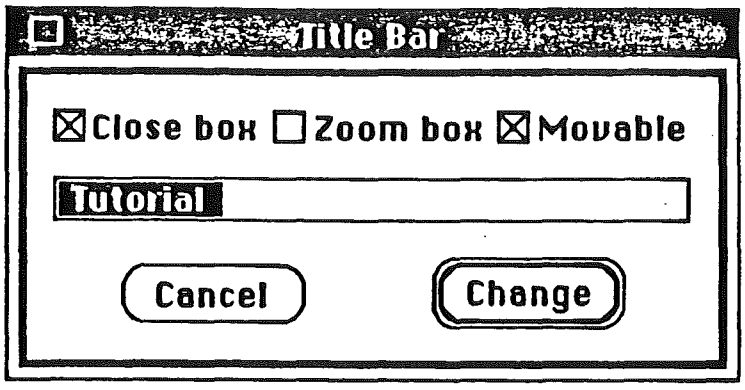
Our program window will be a *document* window that shows some of the more common *controls* used in a window. Select **Add** from the chooser window's popup and select **Window** from the menu list. A plain window will appear on the desktop.

The yellow border that surrounds the window allows you to move and size it. The right-bottom edges resize the window and the top-left edges move the window. The cursor changes over these areas to indicate these actions. Window positioning can also be accomplished by using the **Windows** menu from the Genesys menu bar allowing you to position the window on the desktop horizontally, vertically and both.



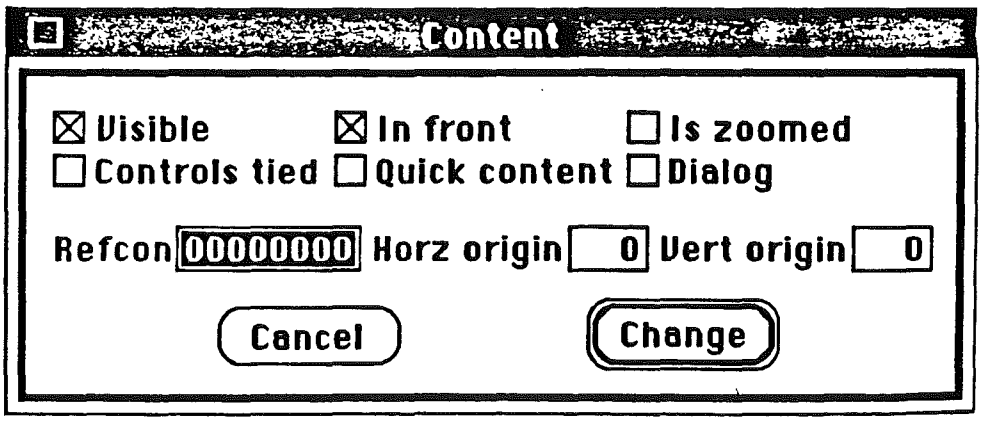

The Window Editor makes extensive use of the **Options** menu by allowing you to add elements such as a *title bar*, *close box*, *zoom box*, and *scroll bars* to the window being created or edited. These elements have additional parameters that can be edited by double-clicking or selecting **Edit...** from the **Options** menu.

Pull down the **Options** menu and select **Title bar**. A Title Bar will appear in the window. To edit this element, double-click on the title bar.




The Title Bar window allows you to specify the title for the window that appears in the title bar, whether the window has a close box or zoom box, and if the window can be moved on the desktop by the user or not. For our purposes, let's add a close box to the window, and rename the title of the window to `Tutorial`, using a space before and after the the title. Select **Change** to accept the changes just made.

Note that our window now contains TWO elements, the *Title Bar*, AND the *Content*. The content of a window is that area the program uses to display its information to the user. If you are following along, your content is that area below the title bar. If you were going to draw something to this window in your program, this is the area where drawing would occur, assuming you did things correctly.




Double-click in the content of the window to display the Content Editor. Under most circumstances, the defaults supplied will be appropriate with one exception; if you wish to create a dialog type window, you would click in the **Dialog** check box. A dialog window and standard window are functionally equivalent to each other, except a

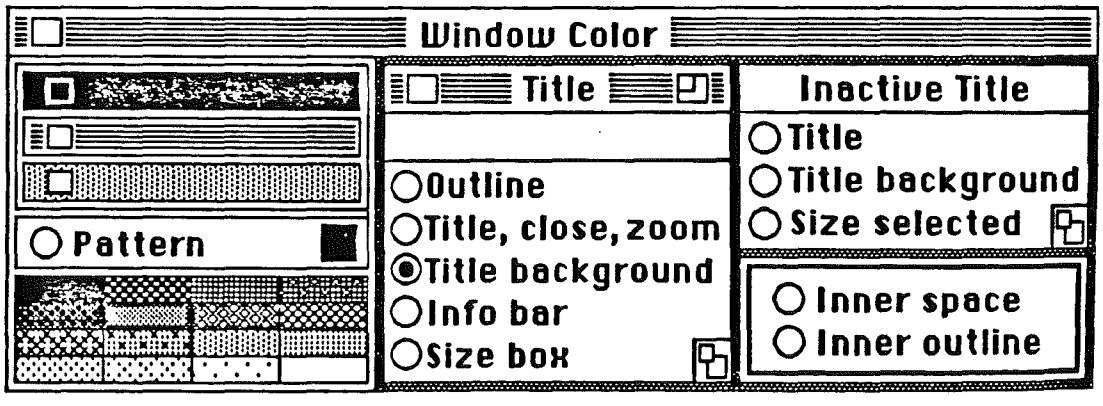
dialog window does not have anything but a content area. Select **Cancel** since no changes should be made to the content of this window.

- *Tip:* After selecting the window content, you can use the **Option** key and the arrow keys to fine positioning your windows.

### Window Colors

Your Apple IIGS is capable of displaying windows in many colors and the Window Color Editor allows you to change these colors. To edit the colors for your window, pull down the **Options** menu and select **Edit colors....**

The three small windows on the right against the desktop background display how your color selections will look. Each of the radio buttons in these windows apply the that window's state; active, inactive, and dialog. There is no need to set the colors for a window you are not using, such as setting the dialog window colors when you are using a standard document window.




The color palate allows you to select the color that will apply to a window element (the selected radio button). Select the **Title background** radio button and click on red in the color palate. Notice the title background in the active window becomes red. All other radio buttons for this Editor work in this fashion.

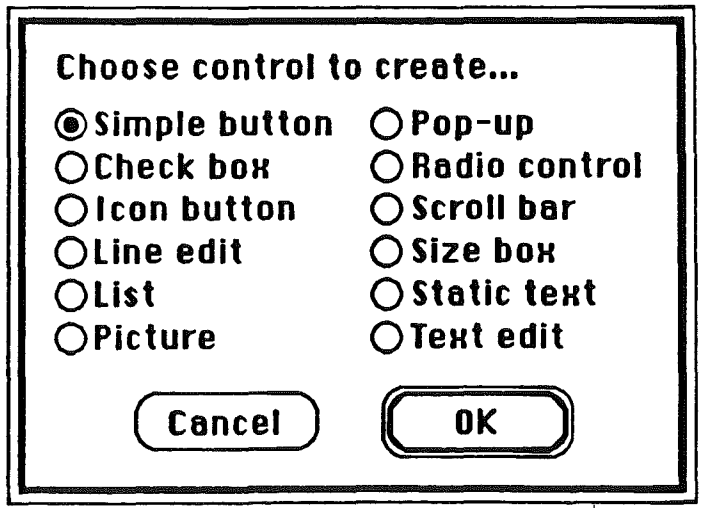
The **Pattern** radio button allows you to select the color that is used in conjunction with the title pattern selected. Select the **Pattern** radio button and click on white in the color palate. No change is made to the title bar this time as your window still has a filled title pattern instead of a checkered or lined pattern. Now click on the lined title bar above the Pattern button. Let's keep this color arrangement by closing the Window Color Editor using its close box and returning to the Window Editor.

- *Note:* PLEASE use discretion when creating or changing colors for your windows. Inappropriate window color combinations can ruin a good program!

---

### Controls

Pull down the **Options** menu and select **Add control...** (⌘A). A dialog window is displayed allowing you to select the type of control needed. Choose **Simple button** and click **OK** or press **return** to have the simple button added to the window content.



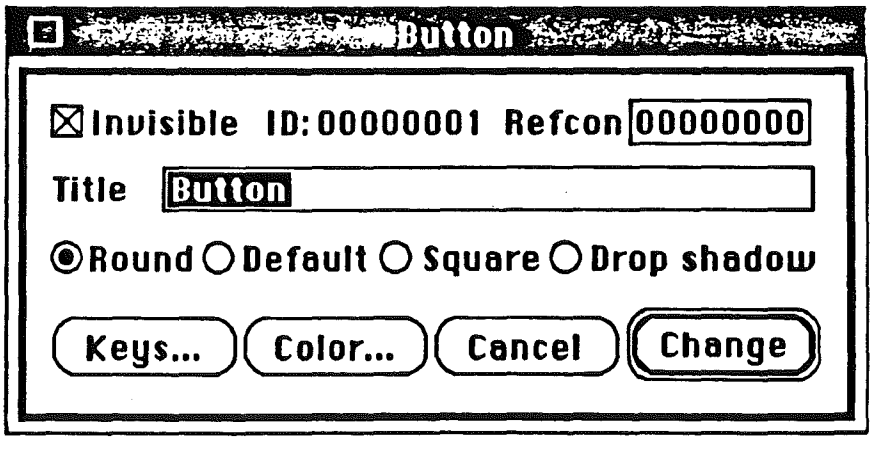

Moving and resizing controls works exactly like moving and resizing a window. Click on the control to select it. The red marching ants surrounding the control indicate it is selected. When the cursor is positioned near the bottom-right corner of the selected control, it changes to indicate that dragging will resize the control. When the cursor is

anywhere else in the control, the cursor changes to indicate that dragging will move the control. Simply clicking and dragging an unselected control will move it.

- *Tip:* After selecting a control, you can use the **Option** key and the arrow keys to fine positioning a control in your window.

Editing a control is like editing a window element. Double-clicking on a control allows you to to tailor that control's appearance. Double-click on the simple button you just added to your window. The Button editor is displayed allowing you to change its style, text, and other parameters. Different controls have different attributes for their appearance and the way they will be handled by a program, so be sure to explore each type of control's Editor.

- *Note:* Although you can specify a control as being invisible, Genesys will still display the control so you can continue to edit it.




Select Cancel in the Button Editor and continue using **Add control...** (⌘A) from the **Options** menu to add any additional controls to your window. Move, resize, and edit the controls as desired.

Select **Test window** (⌘T) from the **Options** menu to "test drive" your Window. Try clicking on the controls you've created to see how they operate. When

you are done testing, in the desktop, close box, or Genesys menu bar to return to the Window Editor. Make any further changes to your window and close the Window Editor.

- ***Tip:*** There may be instances where you won't have a close box defined for your window (such as a dialog window). Selecting **Close** (⌘W) from the **File** menu will close any open window.

Press ⌘S or select **Save** from the **File** menu to save your work. If you'd like to return to this section later, select **Quit** (⌘Q).

## Generating Source Code

Before continuing, let's get a basic understanding of what Genesys actually does when it generates source code. The heart of this process is the Genesys *Source Code Generation Engine* (SCG Engine). After specifying the type of source code to create, the SCG Engine looks for a matching *SCG Template* in the `GEN.LANG` folder, or prompts you for the Genesys *Templates Disk* containing the SCG Templates. After reading the SCG Template, Genesys generates source code *as specified in the SCG Template.*

What does this mean to you? It means you can edit the SCG Templates to generate source code in ANY format. You may prefer to have semi-colons delimiting your comments when generating assembly language, or `(* *)` comments in your Pascal code instead of `{ }`. You might want a particular header to be created in your files with copyright information, your name, and a section for program modifications. You could also define your own SCG Template for a language not provided with Genesys, or a new language under development.

- ***Warning:*** If you plan on modifying or creating Genesys SCG Templates, make sure you are thoroughly familar with the appendix on the *Source Code Generation Language.*

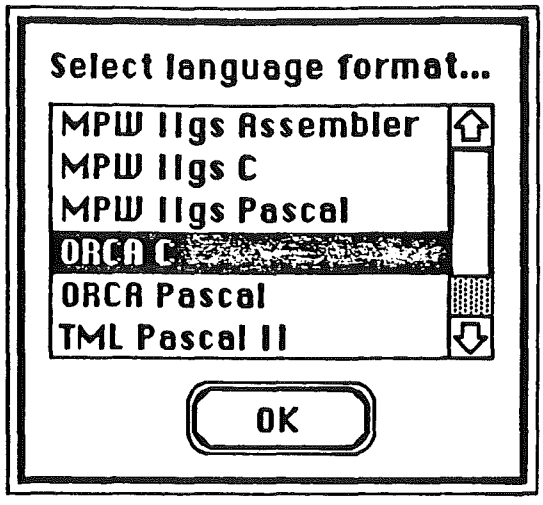
At this point a discussion of "names" is in order. For each item Genesys creates, a default name is provided such as **MENU_00000001**. Genesys allows you to rename these labels to something more appropriate such as **File Menu**. This feature allows you to name different items making it easier to remember what each item contains. This may seem trivial until you realize that Genesys USES THESE NAMES IN ANY SOURCE CODE GENERATED. Specifying the names for each of the items in your program allows Genesys to generate better source code for you.

- *Note:* The names are actually part of the file and are referred to as *Resnames*.

With that in mind, let's change some of the default names for the items you have created in the `Tutorial` program. Select **Alert String** in the Resource Chooser Window's types list and select **ALERT_00000001**, which is the Alert Window you previously created for an "About" window. Now select **Attributes** from the **Item** popup. The **Edit item attributes...** window will appear, allowing you to specify the resource ID, name, and resource attributes. Type in the new name **About Alert**. Click in the **OK** button and the new name appears in the items list for your Alert Window. Continue selecting types from the types list and editing the name for each of your items.

- *Warning:* Be very careful when changing the attributes of an item other than the name. Unless you know what you are doing, you can change the attributes of an item making it not function correctly in your program. Under most circumstances, you should only need to edit the item attributes name. Also note that you should NOT use duplicate names for different types of items, as your compiler may not be able to handle them in the source code that is generated. Genesys does not allow duplicate names for the same type.

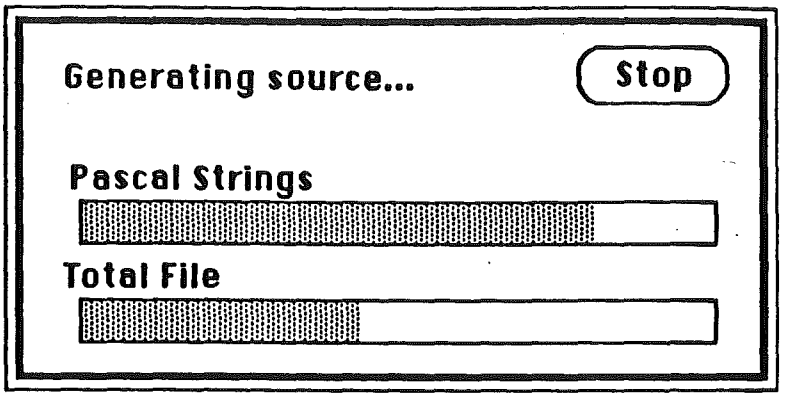
Choose **Generate Source...** from the **File** menu. Since this is the first time generating source, Genesys will ask you which language you would like generated. For our example, pick **ORCA C** by clicking on it and then the **OK** button (or pressing **return**).




- *Tip:* You can also select **Choose language...** from the **File** menu to choose or switch languages.

After choosing a language, a Put File dialog will appear. After selecting the disk and file name for the source to be generated (the filename is `TUTORIAL.CC`), choose the **Save** button. As explained earlier, the SCG Template for your chosen language will be read, and the SCG Engine will generate your program. The first dialog shows the template being processed, and the second dialog shows the source code being generated for each resource in the file.

If you wish to stop source generation, you can click on the **Stop** button, and Genesys will complete source generation for the current resource and abort the process.




After the source is generated, you can view it by selecting **View Source...** from the **File** menu and opening the source file. After you are done viewing the source code, select **Quit** from the **File** menu to quit Genesys.

- *Warning:* If you quit and Genesys asks if you would like to save the changes to your file, make sure you choose **Yes**, as the source code generated is based on the editing you've done.

## Compiling, Linking and Launching

- *Note:* Since we cannot possibly cover every compiler and linker, this section assumes you are using the ORCA environment and understand how to use it.

After returning to the finder or your program launcher, run the ORCA.SYS16 program and get to the disk/folder containing your generated source code. Compile and link the program. A typical compile and link statement would be:

```
COMPILE  TUTORIAL.CC KEEP=TUTORIAL
LINK     TUTORIAL    KEEP=TUTORIAL
FILETYPE TUTORIAL    S16
```

After successfully compiling and linking the source code, you need to attach the resource fork you've created to your program. One way to do this is to use the Apple utility DUPLICATE as follows:

```
DUPLICATE -R TUTORIAL.R TUTORIAL
FILETYPE TUTORIAL S16
```

The other method is to return to Genesys, open your resource file, and use **Save As...** under the **File** menu using the same name as the program (TUTORIAL). The Put File dialog will ask you if you want to replace the file, and you should choose **Yes** as Genesys will ONLY replace the resource part of your program, not the executable part.

Now launch your program. From the Finder, find the TUTORIAL program and double-click on it. From the ORCA Shell, simply type in:

```
TUTORIAL
```

Move your window around the screen, try the button, and make sure to select **About Tutorial...** to see your "About" window. You can also open any desk accessories and use **Close** from the File menu to close them. When you are done, select **Quit** from the **File** menu. Genesys automatically generated the code to handle your window, your **About Tutorial...** menu, and among other things, a quit routine allowing you to gracefully return to where you launched your program from!

## Making Program Modifications

Since your interface is already attached to your program, you can open your program with Genesys and make instant modifications to your windows, menus and other elements of the program. Changes to your program take an immediate effect.

If you decide to add other elements that aren't in your source code yet you can:

1.) re-generate the code,
2.) re-generate the code and paste the new code into the existing source code, or

3.) simply edit the existing source code and add the routines needed.

## Where to Go From Here

Although you have a general understanding of how one would edit and create a program with Genesys, there are many other features left to your own exploration. For example, although we manually created most of our program, you could have used **New application** under to **File** menu to create your standard Tool Table and program Menu Bar!

You should read the rest of this manual to become fully familar with all the features Genesys has to offer. The previous Chapters covered only a sampling of the Shell and Editor features. The Chapters that follow cover the Shell, Editors, and other facets of Genesys in complete detail. Included are appendices on writing your own Genesys Editors and Source Code Generation Templates.

Alot of time and effort was spent creating Genesys, and more time and effort will be spent supporting and adding new features. For those new to programming, we hope you will welcome Genesys as an invaluable tool to tailor your programs and cut down the learning curve to programming the Apple IIGS. For those already experienced, we hope Genesys will become an indispensable tool in your program development efforts.

# Chapter 5: The Genesys Shell

- *Note:* This chapter covers the Genesys Shell, and is intended as a reference. For detailed instructions on editing or creating programs with Genesys, please refer to the appropriate Chapters in this manual.

## Shell Overview

The Genesys program is actually a *Shell* for the Genesys Development System. It handles the desktop environment, menus, help system, source code generation engine, and access to desk accessories. The Shell provides the means to open and save a modified resource file. Features such as coordinate display, centering windows, and source code or text file viewing are also provided.

The Genesys Editors are loaded by the Shell from the `Gen.Edit` folder when Genesys is first launched. All interaction between the Shell and each Editor is seamless, and appear to the user as a single program. Source Code Generation (SCG) Templates are loaded from the `Gen.Lang` folder as needed. After specifying the type of language desired, the Shell loads the appropriate SCG Template and processes the resource file against the template.

This modular design allows Genesys to grow as new resource types are developed, as well as providing the advanced programmer with the ability to write his or her own Genesys Editor for custom uses. It also provides the Genesys user the means of creating or modifying source code generation to suit needs or personal tastes, as well as allowing new templates to be created as new languages are developed.

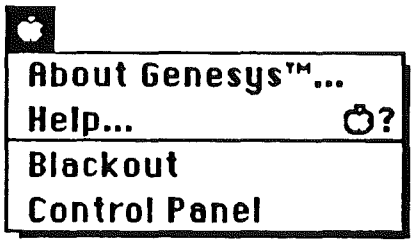
Genesys can be launched by any program launcher such as the Finder or APW Shell. After quitting Genesys, you are returned to the program that Genesys was launched from. Genesys makes sure the state of the system clipboard is left intact when it quits, so anything placed on the clipboard will be there when you return to Genesys, even after shutting off the power to your computer.

## Shell Menus

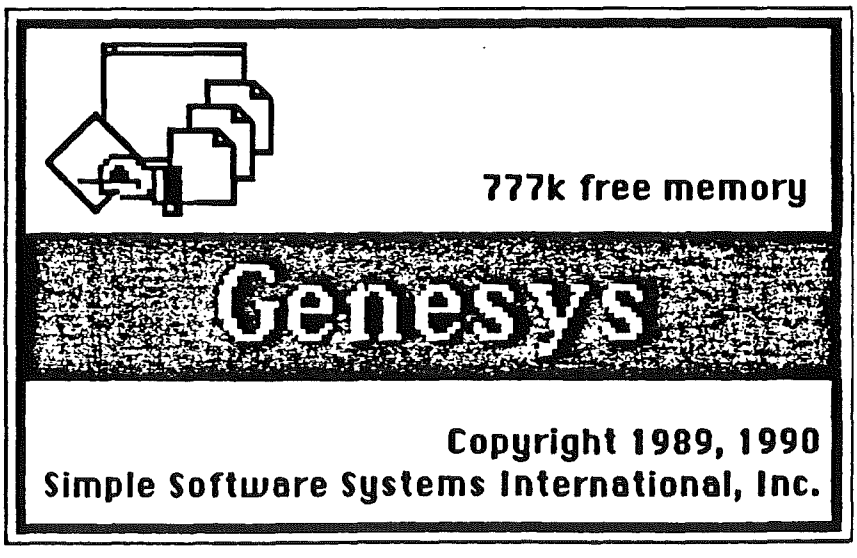
The Genesys menu bar contains the , **File**, **Edit**, **Layout**, **Windows** and **Options** menu. The **Options** menu is used strictly by each Genesys Editor, and changes depending on the Editor that is currently being used.

### Apple Menu

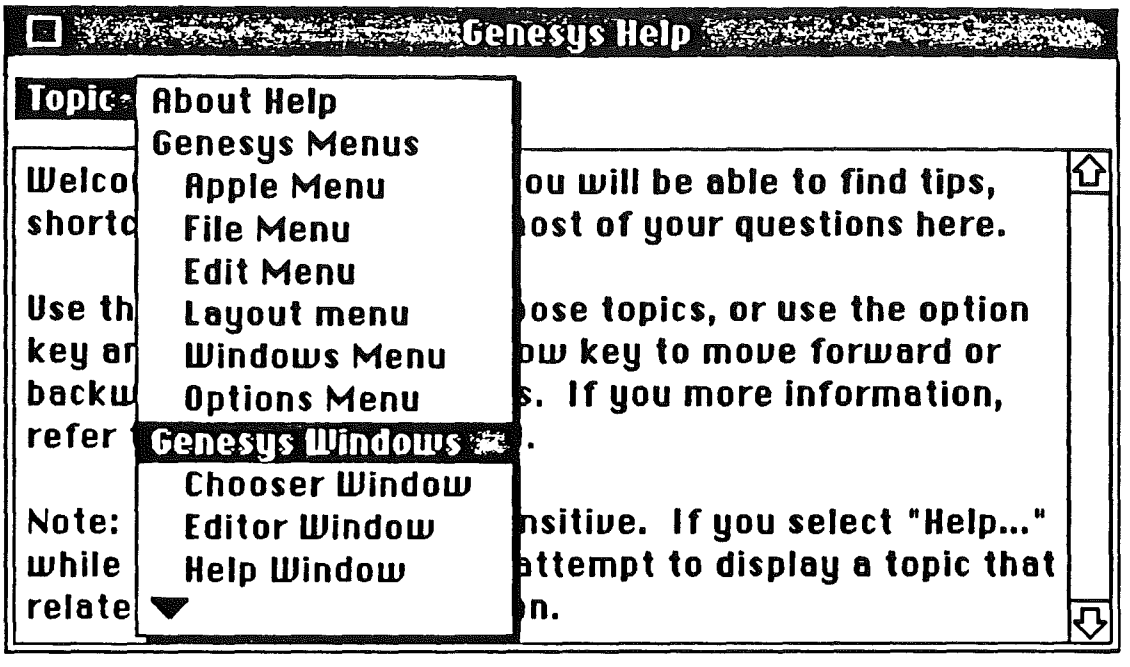
The  menu gives you access to program information and help. It also contains the *New Desk Accessories* that were loaded when the system was started.




**About Genesys™...** displays the Genesys "About" window with the version number of the shell or active editor, amount of free memory, and copyright information. This window is also displayed when Genesys is first run.



**Help (⌘?)** gives you access to the Genesys context-sensitive help system. After selecting this menu item, a window will appear displaying a *Topic* pertinent to the action you are currently doing. For example, if you are working with the Menu Editor and select **Help** from the  menu, the Help Window will display information about the Menu Editor and how to use it.



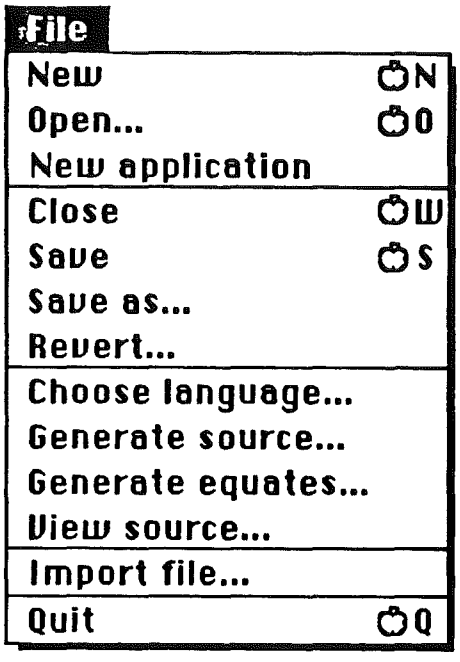

Selecting the **Topic** popup menu gives you access to all the topics in the Help System. You can also use the **option ->** and **option <-** keys to move forward and backward through topics.

**NDAs** (New Desk Accessories) follow **Help**. NDAs are loaded from the DESK.ACCS folder when you turn your system on. Example desk accessories include the **Control Panel** and SSSi's *Deskpak Desk Accessory* package.

- *Warning:* Many *shareware* and *public domain* desk accessories, *CDEVS*, and *Inits* work very well. However, should you experience strange or unexplained behavior while using Genesys (or any program), try removing or disabling these files and trying the program again.

### File Menu

The **File** menu provides items that deal with file related activities such as the opening and saving of your resource files.




**New** (⌘N) creates an empty resource file named `Untitled`, ready for the addition of interface elements using the **Add** popup menu from the Resource Chooser Window.

**Open...** (⌘O) allows you to open a file that was previously created with Genesys, or any file that contains a resource fork. Only files with resource forks can be opened.

- *Note:* As a precaution, Genesys creates a copy of the file you are editing in its `GEN.WORK` folder called `SCRATCHFILE`. Therefore, you don't actually modify a file until you choose **Save** or **Save As...**.

**New application** allows you to instantly create a program resource fork containing a standard menu bar and tool table.

- *Tip:* **New application** uses the file `DefaultFile` located in the `GEN.WORK` folder to create a default application. You can edit this file using Genesys to tailor the resources that **New application** creates for you. For example, you may want to add a standard "About" window, **Show clipboard** menu item to the **Edit** menu, and Clipboard window to this file that will be created automatically for you.

**Close (⌘W)** will close the frontmost window. If you close the Resource Chooser Window and the file has had changes made to it, you will be given an opportunity to save those changes.

- *Tip:* **Close** is especially handy when you have created windows with the Window Editor that don't have close boxes, such as a Dialog Window.

**Save (⌘S)** will save all changes made to a file. If the file is a new file, a Put File dialog will be displayed asking for a filename as if you had chosen **Save As...**.

- *Tip:* Use **Save** often to avoid unexpected loss of work.

**Save as...** will save all changes made to a file using the name specified in the Put File dialog. After saving the file, the Resource Chooser Window's title bar will reflect the new name.

- *Tip:* You can use **Save as...** to attach a resource fork to your program. Just specify the same filename as your program and answer **Yes** to replace the file. Genesys will only replace the resource fork, not the program fork. You can also use **Save As...** to make multiple copies of a file for backup purposes.

**Revert...** will restore the file being edited to the last saved version. This is a handy way to undo all the changes you've made to a file during an editing session.

**Choose language...** allows you to change the language that will be generated when selecting **Generate source...**.

**Generate Source...** will generate source code for the current file to a specified folder and filename. If a language hasn't been chosen, the **Choose language...** dialog will be displayed first.

**View source...** allows you to view ANY text or source file. This option is provided to allow viewing of the source code created by Genesys. Although the text can only be viewed, you may select and copy the text to the Clipboard if needed.

**Import file...** allows you to import any file that does not have resources into your existing file. You will be asked for the resource type, in hexadecimal, to import the file into. If you need to change the ID that Genesys creates, use the **Attributes** item from the Resource Chooser Window's pop-up.

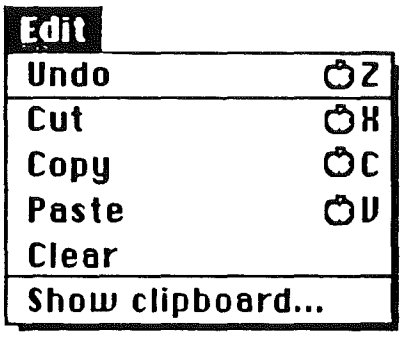
- *Warning:* Import file was designed to allow you to import code into code resources, text into text resources, etc. Genesys assumes you know what you are doing when you use this feature, and will allow you to import any file into any resource type.

**Quit (⌘Q)** terminates Genesys and returns to the launching program. If changes were made to the open file, you will be given an opportunity to save them.

- *Note:* Genesys always saves the Clipboard when it quits.

### Edit Menu

The **Edit** menu contains the functions necessary to interface with the system clipboard.




**Undo (⌘Z)** will undo the last operation performed, if possible.

**Cut (⌘X)** will remove the item selected and place it on the clipboard for subsequent pasting operations.

**Copy (⌘C)** will create a copy of the item selected and place it on the clipboard for subsequent pasting operations.

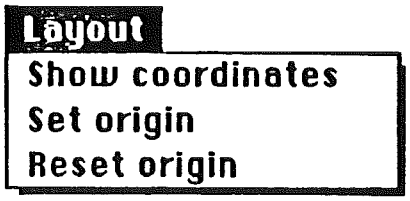
**Paste (⌘V)** will paste the clipboard's contents, if appropriate.

**Clear** will permanently remove the selected item.

**Show clipboard...** will display the system clipboard. Any changes made to the clipboard by cut or copy operations are immediately reflected in this window.

## Layout Menu

The **Layout** menu contains functions that will help you in laying out your program interface.




**Show coordinates** will display the horizontal (H) and vertical (V) coordinates of the cursor on the menu bar. The origin of the coordinates are based on the desktop's upper-left corner of the screen or an origin set using the **Set origin** menu item. After selecting **Show coordinates**, it will change to **Hide coordinates**, which will turn the coordinate display off.

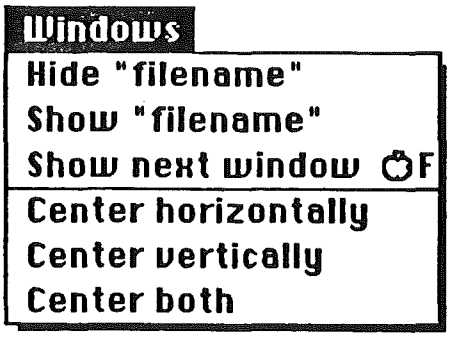
**Set origin** allows you to set the origin of the coordinate display. The cursor changes to a "target" cursor, allowing you to click on the new origin desired.

**Reset origin** resets the origin for the cursor display back to the upper-left corner of the desktop.

- *Tip:* Many Editors use the Genesys coordinate system to aid you in finely positioning objects. For example, when using the Window Editor and moving a control in a window, the Window Editor will show you the position of the control being moved relative to the window it is being moved in.

## Windows Menu

The **Windows** menu contains functions that affect how windows are displayed on the desktop.




**Hide** `"filename"` will hide the Resource Chooser Window (`"filename"` will be the same name as the one on the Resource Chooser Window). This menu item is useful when you need to work with a "clean" desktop.

**Show** `"filename"` will show the Resource Chooser Window (if it was hidden with the **Hide** `"filename"` menu item) and bring it to front.

**Center horizontally**, **Center vertically**, and **Center both** will center the frontmost window on the desktop accordingly.

- *Note:* Remember that positioning windows that you are creating or editing while using the Window Editor will affect the window's specified coordinate position.

---

### Options Menu

As specified at the beginning of this section, the **Options** menu is used strictly by each Genesys Editor, and changes depending on which Editor is currently being used. Note that an Editor is NOT required to have an **Options** menu.

Each Editor will construct its menu depending on its needs, although a guideline has been set for consistency. The general menu item structure is as follows:

**Edit** `xxx...` (⌘E) where `xxx` is the item to be edited. For example, using the

Alert Window Editor and selecting the alert's icon would change this menu item to **Edit icon...**. Most Editors have **Edit xxx...** menu item in the **Options** menu.

**Add xxx... (⌘A)** where **xxx** is the item that will be added. For example, if you were using the Menu Bar Editor, this menu item would read **Add menu...**. Most Editors will have an **Add xxx...** menu item in the **Options** menu.

The next group or groups of menu items are specific to the Editor. For example, the Window Editor has its groups of menu items to add title bars, close boxes and other elements pertaining to the window.

**Test xxx... (⌘T)** where **xxx** is the object that will be tested. Selecting this item would place the Editor in *test mode*, which would allow you to test the object as if it were in your program. For example, the Window Editor's menu item would read **Test window...**, and selecting the item would allow you to "test drive" the window and any elements of the window such as its controls. Clicking in the desktop or Genesys menu bar exits test mode and returns the Editor to *editing mode*.

**Next item (⌘>)** and **Previous item (⌘<)** allow you to select the next or previous object the Editor edits. For example, the Alert Window Editor would select the Alert's content, icon, message, and buttons respectively. Note that this menu item is functionally equivalent to selecting an object with the mouse.

An Editor is NOT required to have every menu item specified. Only those menu items pertinent to the Editor's function are used.

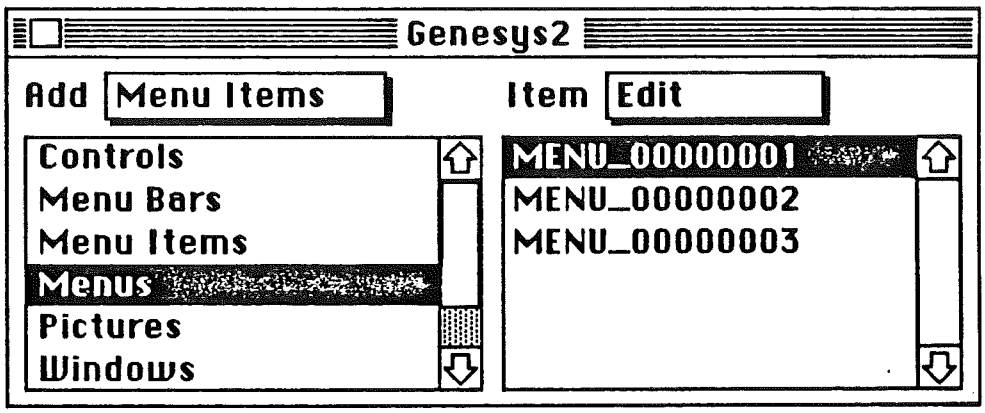
Refer to the next Chapter which covers each Editor in detail for their specific use of the **Options** menu.

## Shell Windows

Genesys uses several different types of windows that can be identified by their appearance. These windows and the functions they perform are described in the following sections.

### Resource Chooser Window

The Resource Chooser Window is displayed after selecting **New**, **Open...** or **New application** from the **File** menu. It is your key to creating, editing, and manipulating the resources in a file. The title bar contains a close box and the name of the file being edited or created, or UNTITLED if this is a new file. The content of this window contains two popup menus and two lists.




The left list shows the types of resources in the file (the types list), and the right list shows the items for a selected type (the items list). To display or edit menus, for example, one would select **Menus** in the types list which would display the defined menus in the items list, as shown above. Note that types and items are only displayed if they exist in the file being created or edited.

The **Add** popup is used to add types to the file being created or edited. This menu is constructed based on the Editors that are loaded when Genesys is launched. To add a new type, select the **Add** popup and the type of element you would like to add from the menu list. A default item is created and the Editor is automatically opened for you.

The **Item** popup selects an action for the currently selected resource in the items list. The action is invoked by selecting it from the **Item** popup, or by double-clicking on the item in the items list. There are currently five actions available:

**Attributes** displays a dialog allowing you to set the attributes or the name for the

selected resource. The Genesys Editors will create a resource with the proper attributes and a default name, but you may view or change these. Select the attributes and press **OK** to make the changes or **Cancel** to restore the original values and close the dialog.

- *Warning:* You should be thoroughly familar with the Resource Manager and the Memory Manager before changing the attributes (other than the name) for a resource.

**Data view** opens the HexASCII Editor allowing you to read the raw resource data in hexadecimal and ASCII text. Resource type, ID and size information is displayed for the resource also. See the Editors section for details on the HexASCII Editor.

**Delete** allows you to remove a resource from the file. A warning dialog is displayed that allows you to **Cancel** or **Delete** the action. Note that **Clear** from the **Edit** menu does the same thing.

- *Warning:* Many resources contain references to other resources. If you delete a resource that is required or referenced by another resource, unpredictable behavior may result. Be sure to have a backup copy of the file before deleting any resources from it.

**Edit** opens the appropriate Editor for the selected resource. Please refer to the following Chapter for specific details regarding Genesys Editors.

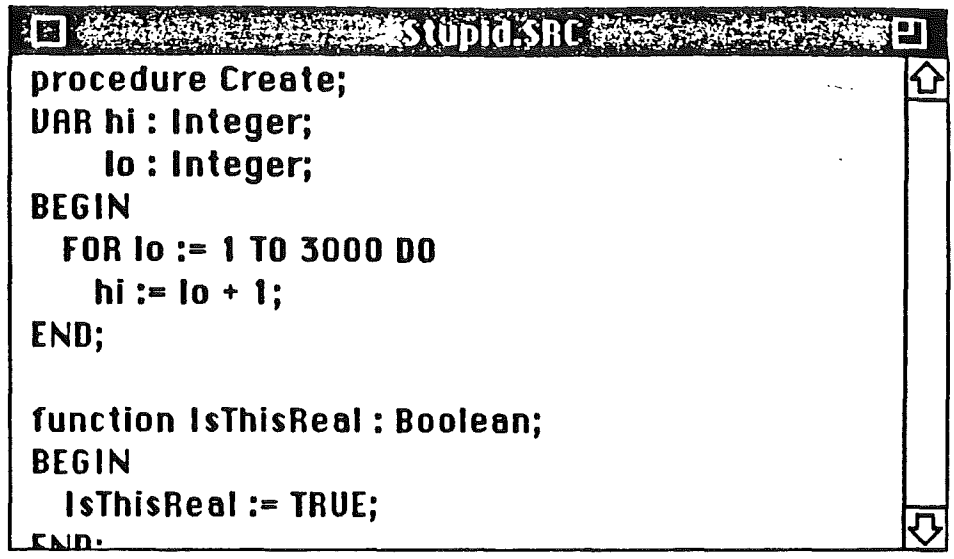
**Export** Allows you to export the currently selected resource to a file.

---

### View Source Window

The View Source Window is displayed when **View source...** is selected from

the **File** menu. After specifying the text or source file to view, a window is displayed with the text contained in the file.




The text in the window is not editable but you can still select text and copy it to the clipboard. Although this window is normally used to view the source generated by Genesys, you can use it to view any type of source or text file.

- *Note:* Some text files have their *high bits* set. The View source window will read these types of files too.

---

### Help Window

The Help Window displays and handles the Genesys Help System. It is used by selecting **Help** (⌘?) from the  menu. The Help System was covered earlier in this Chapter in the section covering the Genesys menus.

### Editor Windows

The most common Editor window has a standard black striped title bar. The is a standard editing window which is usually displayed by selecting **Edit** or **Add** from the Resource Chooser Window's popups.

An Editor with a red-striped title bar is identical to the normal editing window described above, but is displayed in response to a request by another editor. This type of Editor is referred to as a *Sub Editor*. In most cases, a Sub Editor works exactly like a normal Editor except you are restricted to the actions the Sub Editor performs. For example, the Window Editor will ask the Menu Editor to edit a popup menu. The Menu Editor will display its window as a Sub Editor (red-striped title bar). You are restricted to using the Menu Editor at this time. When you are through, close the Menu Editor to return to the Window Editor.

The last type of Editor window is referred to as a *Movable Modal Dialog*. This type of window is displayed by an Editor and operates much like a normal dialog window, except that a Movable Modal Dialog allows you to move the window on the desktop to see the window underneath that your editing actions would affect. A Moveable Modal Dialog has a solid red title bar, and restricts the actions that can occur to those pertaining to the dialog, much as a normal dialog window would.

For more information on the exact functions and window types an Editor uses, refer to the following Chapter on Genesys Editors.

# Chapter 6: The Genesys Editors

- *Note:* This chapter covers the Genesys Editors, and is intended solely as a reference. For detailed instructions on editing or creating programs with Genesys, please refer to the appropriate Chapter in this manual.

## Editor Overview

The Genesys Editors are separate files loaded by the Shell from the `Gen.Edit` folder when Genesys is first launched. All interaction between the Shell and each Editor is seamless, and appear to the user as a single program. Each Editor can represent one or multiple Editors, allowing Editors that handle common types and common code to be utilized. For example, the Menu Bar Editor is actually composed of three Editors: the Menu Bar, Menu, and Menu Item Editor.

When an Editor receives a request from the Shell to create or edit a resource, it displays a window containing a representation of the resource, and may or may not allow the user to change the resource. If the Shell cannot find an existing Editor to handle the editing request, it uses the *HexAscii* Editor to display the resource's data.

Since Genesys does not wish to restrict the user, most Editors allow any number of windows to be opened at one time. To make an open Editor active, simply click on its window to bring it to the front. The feature is handy when you would like to **Copy** and **Paste** parts of a resource from one type or file to another.

This modular design allows Genesys to grow as new resources are developed, as well as providing the advanced programmer with the ability to write their own Genesys Editor for custom uses.

- *Note:* If you wish to make your own Genesys Editor, please refer to the appendix *Writing Your Own Genesys Editors.*

## Common Editor Features

When a resource is created using the Resource Chooser Window's **Add** popup, the Editor creates default data for the resource, and displays a window containing a "live" representation of the resource. Many Editors have resource forks themselves that can be edited with Genesys to change the default data that Editor creates. Resource editing occurs using the Resource Chooser Window's **Item** popup and selecting **Edit**. The Editor validates the resource that is being created or edited to the best of its ability, and displays any errors that may or may not be correctable, depending on the reason for the error.

- *Warning:* System Software 5.0.3 corrects a bug that corrupts resources.

Genesys Editors use common techniques for editing. Windows that contain multiple lines of text work like any word processor, with full clipboard support. Objects that can be edited in the window are selected with the mouse or a keyboard equivalent. An object is shown selected by outlining it with a red selection marquee (*marching ants*). Double-clicking on a object usually produces additional editing options.

The Genesys **Options** menu is enabled with menu items particular to an Editor. If the Editor does not use the **Options** menu, it is dimmed. This menu is used strictly by each Genesys Editor, and changes depending on which Editor is currently active. A guideline for the **Options** menu is established as follows:

1.) **Edit xxx...** (⌘E) where **xxx** is the item to be edited.
2.) **Add xxx...** (⌘A) where **xxx** is the item that will be added.
3.) One or more menu items specific to the Editor.
4.) **Test xxx...** (⌘T) where **xxx** is the object that will be tested.
5.) **Next item** (⌘>) and **Previous item** (⌘<) select the next object.

Note that an Editor is NOT required to have every menu item specified. Only those menu items pertinent to the Editor's function are used.

## Remaining Sections

The remainder of this Chapter describes each Editor in detail. Each editor is presented in its individual section, allowing for easy section replacements as new features and Editors are added. Each Editor section is divided into three parts: **About the Editor** gives a general description of the Editor, **Using the Editor** describes specific features of the Editor, and **Tips** provides additional information that can help you become more effective while using the Editor.

- *Note:* Multiple Editors contained in a single file are treated together because they have similar features.

Each Editor section begins with the name of the Editor. Following this name is the type of resource that the Editor handles, the official Apple defined name (all names begin with a lowercase "r"), and its Apple defined hexadecimal definition. Apple has defined reserved resources from $8000-$FFFF, which are the types of resources Genesys Edits. All other types are defined by the user. An example of this format is:

**Control rControlTemplate $8004**

- ***Important Note:*** This following sections cover the Genesys Editors, and are intended solely as a reference. It is NOT the intention of this manual to duplicate the many volumes of information provided by Apple on what resources and interface elements you should use in your programs. If you are unclear on what resources and interface elements you need for your programs, refer to the appropriate *Apple IIGS Toolbox Reference* manuals and the *Apple Human Interface Guidelines*. A complete list of manuals and reference materials is listed at the beginning of this manual.

## Alert and Error Window Editors

### Contains:

| | | |
|---|---|---|
| **Alert Window** | **rAlertString** | **$8015** |
| **Error Window** | **rErrorString** | **$8020** |

### About the Alert and Error Window Editors

Alert Windows are used to provide a convenient and standard method to present the user with information. You can create Alert Windows to offer the user help, tips, or warnings, and to request permission from the user before performing a task.

Error Windows are used in the same fashion, but relate directly to notifying the user of GS/OS operating system errors. Although the `SYS.RESOURCES` file located in the `SYSTEM.SETUP` folder contains Error Windows for all GS/OS errors, you may want to add Error Windows that relate errors better tailored to your program. When the Window Manager tool call to display Error Windows is used, the program's resource fork is checked FIRST. In other words, adding Error Windows to your program overrides those in the `SYS.RESOURCES` file.

Alert and Error Windows operate like modal dialog windows, are centered on the desktop, and consist of a specified size, optional icon, message, and one to three buttons that are set to the same size and spaced evenly in the window. The user must respond to the Alert or Error Window by clicking in a button before continuing. Since Alert and Error Windows are easy to use, they are highly suggested for use in your programs.

- *Note:* Alert Windows are NOT the same as Dialog Manager Alerts. Alert and Error Windows are a function of the Window Manager.

- *Note:* Since the Alert and Error Window Editors are functionally the same, all references to the Alert Window Editor apply to both Editors, unless noted.

The Alert Window Editor uses a movable window for editing allowing you to edit multiple Alert Windows at the same time. The Alert Window being created or edited is centered on a yellow background in the editing window. Items in the Alert Window are edited by double-clicking or clicking on the item and using the **Options** menu.

The Alert Window Editor uses the standard Genesys **Options** menu with the following notable exceptions:

The **Add icon... (⌥A)** menu item is only available when there is no icon present in the Alert Window.

**Choose font...** allows you to choose the font, style, point size and other attributes for the currently selected text. This menu item is only active while editing the message text.

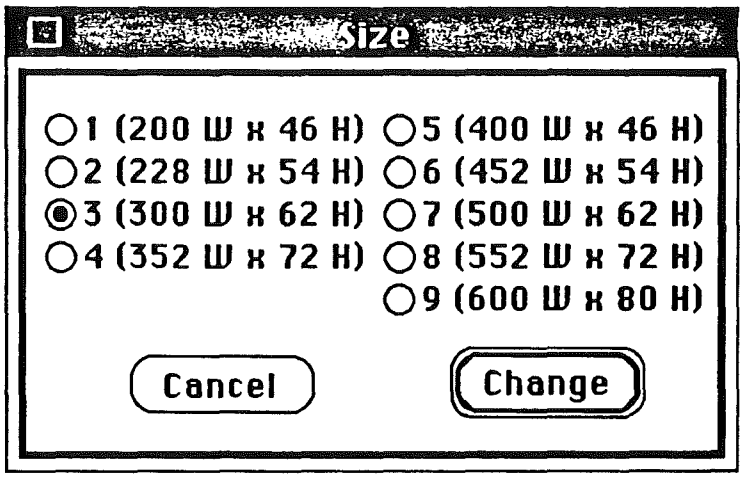
**Choose colors...** allows you to choose the foreground and background colors for the currently selected text. This menu item is only active while editing the message text.

**Left justify, Center justify, Right justify** and **Full justify** allow you to choose the justification for the text. Justification applies to all the message text. This menu item is only active while editing the message text.

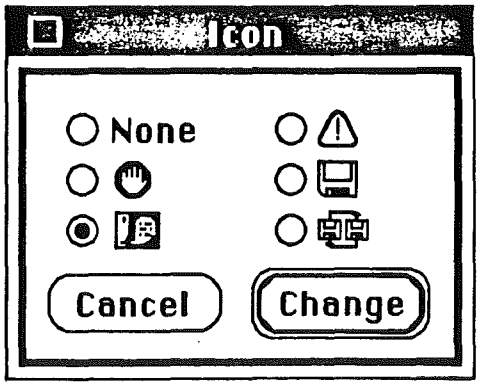
- *Warning:* The message text used in an Alert Window is of the LETextBox2 type. The System Software contains a bug so that LETextBox2 data will only work for small blocks of text with a reasonable number of styles.

Four types of objects within the Alert Window are editable: the **content**, **icon**, **message**, and **buttons**. Each of these objects are edited by clicking and selecting **Edit xxx...** from the **Options** menu or by double-clicking.

The Alert Window's **content** can be edited to specify one of nine sizes. Click on the size or use the **option 1** through **9** keys. The Alert Window is redrawn to show the new size. Select **Change** to keep the new size or **Cancel** to use the old size.

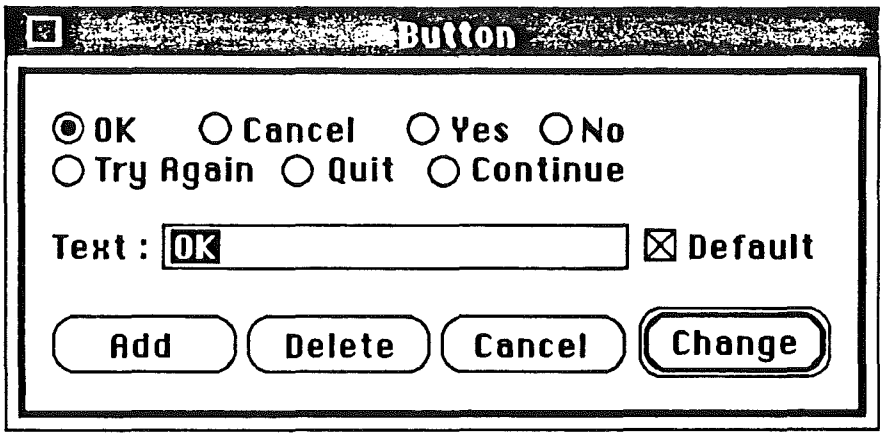



The **icon** can be edited to specify one of five types of icons, or none. The type of icon selected depends largely on the message to the user. For example, a message to the user to insert a disk in place of the one already there would require the use of the *disk swap* icon. After selecting an icon, the Alert Window is redrawn to reflect the new icon. Select **Change** to keep the new icon or **Cancel** to use the old icon.



The **message** text can be edited to specify your user message. The message is edited directly in the Alert Window to allow better representation of the resulting message. Full word processing type editing is supported including the **Cut**, **Copy**, and **Paste** menu items from the **Edit** menu, as well as fonts styling and justification from the **Options** menu. When editing is completed, click anywhere outside of the message area.

Each **button** can be edited to specify the default and text (use the predefined button titles or supply your own). The **Add** and **Delete** buttons are used to specify the number of buttons. **Change** keeps the changes and **Cancel** discards them.




- *Note:* When creating an Error Window, you will be asked to supply the ID that you are creating the Error Window for in hexadecimal. For existing error numbers and messages handled by the SYS.RESOURCES file, refer to the *Apple IIGS Toolbox Reference 3*, Chapter 52, Window Manager Update.

### Tips

• Using the largest Alert Window size (9), select the icon, message, and button(s) that best communicate your message. Then select the size that looks best.

• Dragging and dropping a button on another button will swap the button positions.

• Use the **esc** key or click in the yellow background to remove the marching ants.

• Many of the source code generation templates automatically use the FIRST Alert Window created in your program as the program's "About" window. If this is not suitable for your needs, change the source code generated or edit the Source Code Generation Templates.

• Create Error Windows that are more tailored to your program and user. For example, the SYS.RESOURCES file handles a disk full error by stating "The disk is full." It would be much better to create an Error Window with a disk icon stating "The disk is full. Please insert another disk in the drive and try again." with **Cancel** and **Try Again** buttons.

• DO NOT change the SYS.RESOURCES file. If you need tailored Error Windows, put them in your program.

## CDev Flags Editor

## Contains:

| | | |
|---|---|---|
| **CDev Flags** | **rCDevFlags** | **$8019** |

### About the CDev Flags Editor

The Apple IIGS Control Panel uses *CDevs* (Control Panel Devices). CDevs are "mini-programs" that are controled and accessed through the Control Panel desk accessory. Your system disk was shipped with CDevs for setting the Monitor, Printer, Modem and other parameters. These CDevs reside in the SYSTEM/CDEVS/ folder on your System Disk.

Each CDev contains at least 3 resources: The code that is called by the Control Panel to initialize and handle the CDev, the CDevs Icon, and the CDev's flags. The CDev Flags Editor allows you to create and edit these flags which control what type of events the Control Panel sends to the CDev, whether or not the CDevs icon is displayed at boot time, and text for the version, CDev and auhor names displayed when the user chooses **Help** in the Control Panel.

- *Warning:* Unless you are a programmer, you should not alter the CDev flags for a CDev. One exception is the flag that tells if the CDev will display the icon at boot time. If the item **Wants boot** is checked, the Icon is the resource MUST be EXACTLY 28 pixels wide. Refer to *Apple IIGS Filetype Note $C7 (199)* for complete information on CDevs and CDev flags.

## Control Editor

## Contains:

**Control** **rControlTemplate** **$8004**

### About the Control Editor

A new type of control was introduced with System software 5.0: the *extended control.* These new controls offer many new features, such as *keyboard equivalents* and special event handling. `TaskMaster` can now handle many of the control's events automatically. In fact, using extended controls and `TaskMaster`, it is no longer necessary to use the Dialog Manager's Dialog windows. These new extended controls offer many additional capabilities that are not available in the Dialog Manager, and they can be accessed from resources as well.

Controls are normally "tied" to a window, but they exist as stand-alone resources. Since it would be impossible to determine the type of window a stand-alone control belongs to, the Genesys Control Editor is not a graphical Editor. That function is left to the Window Editor which contains fully graphical representations of controls that can be moved and resized with respect to the window they belong to. Instead, the Control Editor displays the control's data in a moveable modal dialog. This format provides another method for editing controls: one based on actual values rather than pictorial representations.

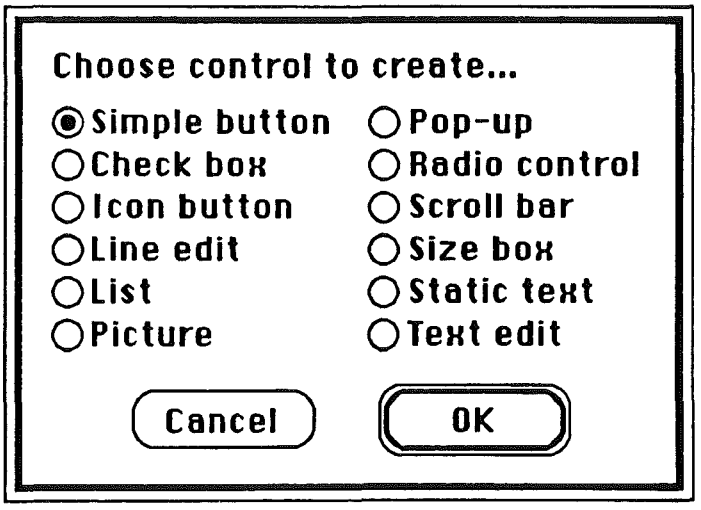
### Using the Control Editor

The Genesys Control Editor can edit and create all twelve types of extended controls: simple button, check box, icon button, line edit, list, picture, pop-up menu, radio button, scroll bar, size box, static text, and text edit. Although each control types editing dialog is tailored specifically for it, there are many common features between them.

- *Note:* In general, you can not tell what type of control a resource contains when you view it in the Resource Chooser Window's item list. Giving the resource a descriptive name like "Balloon Icon Button" can not only help you identify it easier, but also allows for richer and more descriptive source code generation.

### Creating a new Control

When you create a new control from the resource chooser window's **Add** popup, you will be presented with the **Choose control to create...** dialog shown below. Select the control to create and click **OK** to create and edit the new control. **Cancel** will close the dialog and return without creating any control.




- *Note:* You will not be presented with the **Choose control to create...** dialog when editing an existing control as the Control Editor determines what kind of control the resource contains and automatically opens the appropriate Editor.

### Control Editor Standard Commands and Features

All Control editing dialogs display the control's ID and *RefCon* field.

- *Warning:* The RefCon field is reserved for the programmer to use. The Control Editor will allow you to edit this field, but be very sure you know what you are doing if you change it.

**Cancel** Quit the control editing dialog without making the changes.

**Change** Quit the control editing dialog and keep the changes.

**Keys...** Opens a dialog that allows setting a keyboard equivalent for the control.

**Color...** Allows the colors for the control to be specified.

**Invisible** When checked, causes the control to be created as invisible.

**Title** A Line Edit field to change the control's title string. All of the button controls have title strings.

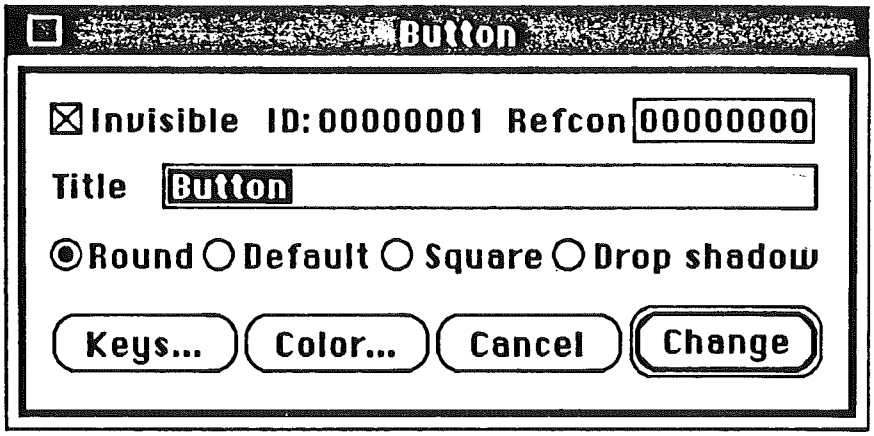
---

### Simple Buttons

Buttons are the most basic type of control. All Apple IIGS users should be familiar with how they look and operate. The button is normally given a descriptive title indicating its action, like **Cancel**. Clicking the button carries out its command. There are four styles of buttons, but they all work the same.

### Button Control Editor

In addition to standard features, the Simple Button Editor has four radio buttons that allow you to set the button style. The choices are: **Round**, **Default**, **Square** and **Drop shadow**. The **Default** setting creates a double outlined border. Default buttons are normally activated with the **return** key (you must set the **return** key yourself using the **Keys...** button).

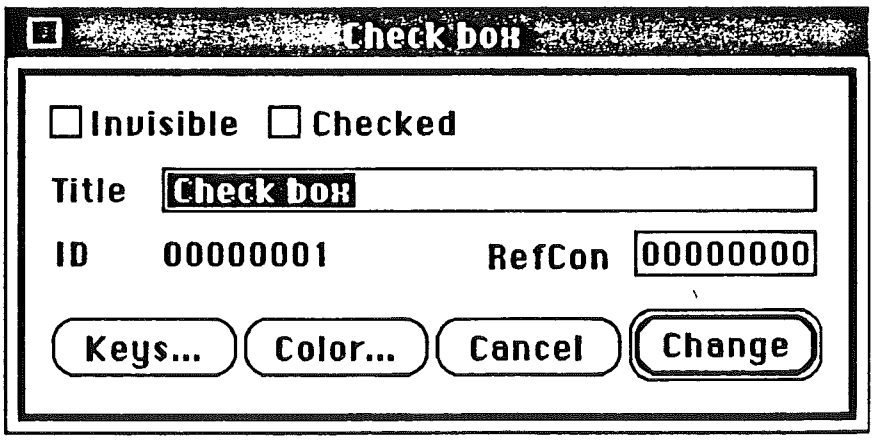



## Check Boxes

Check box controls are used to select or choose an option. They always have a title indicating what the option is. Clicking the control will toggle between the "checked" and "unchecked" settings. The active, or checked control contains an "X" in its check box. Check box controls can be created either checked or unchecked.

### Check Box Control Editor

Besides the standard Editor commands, the **Checked** check box allows you to select whether the control is created checked or unchecked.

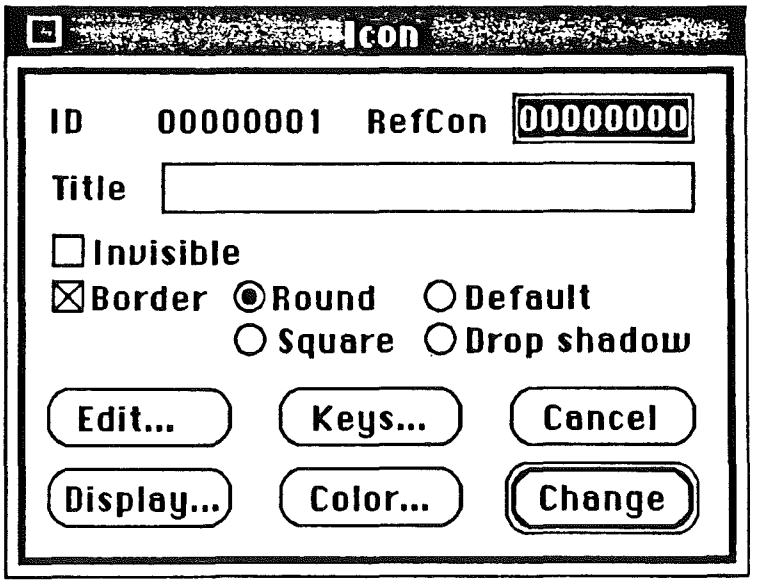


## Icon Buttons

Icon buttons work just like simple buttons, but display an icon in addition to an optional title. Icon buttons also have the same border options as simple buttons, or they can have no border.

## Icon Button Control Editor

There are several options in the Icon Button Editor. When the **Border** box is checked, the **Round**, **Square**, **Default** and **Drop shadow** buttons are activated. Choose the type of border you want your Icon to display by clicking one of the buttons. The **Edit...** button calls the Genesys Icon Editor to edit the control's icon. Refer to the Icon Editor section in this Chapter for complete details on using the Icon Editor. The **Display...** button is unique to the Icon Control Editor: it opens a dialog that lets you set the display mode for the icon.



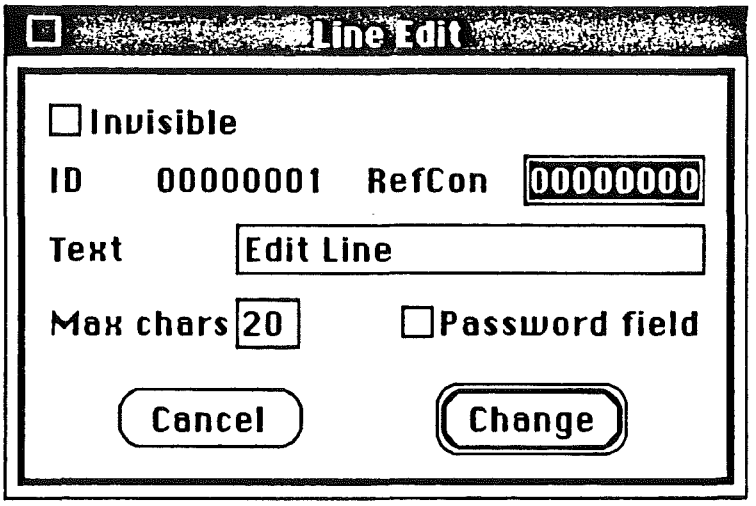

## Line Edits

Line edit controls provide an easy method of editing title or name strings. The control appears with a flashing caret if it is active (there can be only one active Line Edit at a time in a window). If a window contains more than one Line Edit control, the **tab**

key will select the next Line Edit and make it active. Line Edits usually display characters as they are typed at the keyboard, but a special "password" setting causes them to display an asterisk (*) instead. This feature is used for password entry protection if needed.

### Line Edit Control Editor

The Line Edit control Editor allows you to set the maximum number of characters the Line Edit handles and the default text, if any. Check the **Password** checkbox if you want the Line Edit to display asterisks (*) rather than the actual typed text.



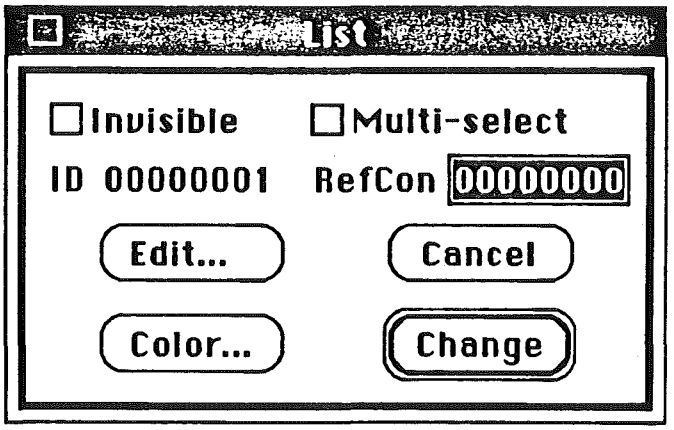

---

### Lists

List controls display items (*members*) in a scrollable list format. The control appears as a rectangle with a vertical scroll bar on its right side. A member is selected by clicking on it. Usually, only one member can be selected at a time, however, it is also possible for List controls to handle multiple member selections.

### List Control Editor

The **Multi-select** checkbox will allow multiple member selection. The **Edit...** button allows you to add and remove members to your list by calling the List Reference Editor. Refer to the section on the List Reference Editor for complete details.

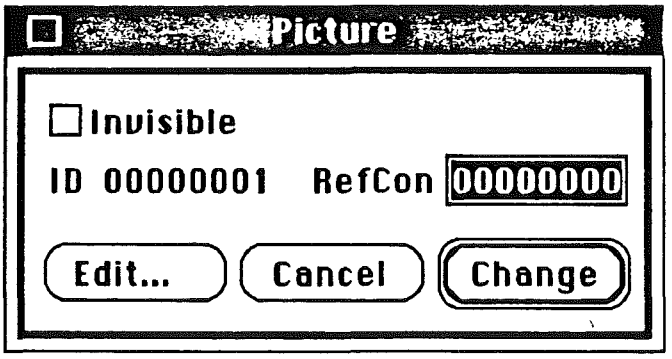



## Pictures

Use Picture controls to display pictures in your windows. Because they are extended controls, scrolling and redrawing are done automatically by `TaskMaster`. Picture controls do not return mouse clicks to a program and are used for display only.

## Picture Control Editor

The **Edit...** button will call the Picture Editor. Refer to the Picture Editor section for details on using the Picture Editor.

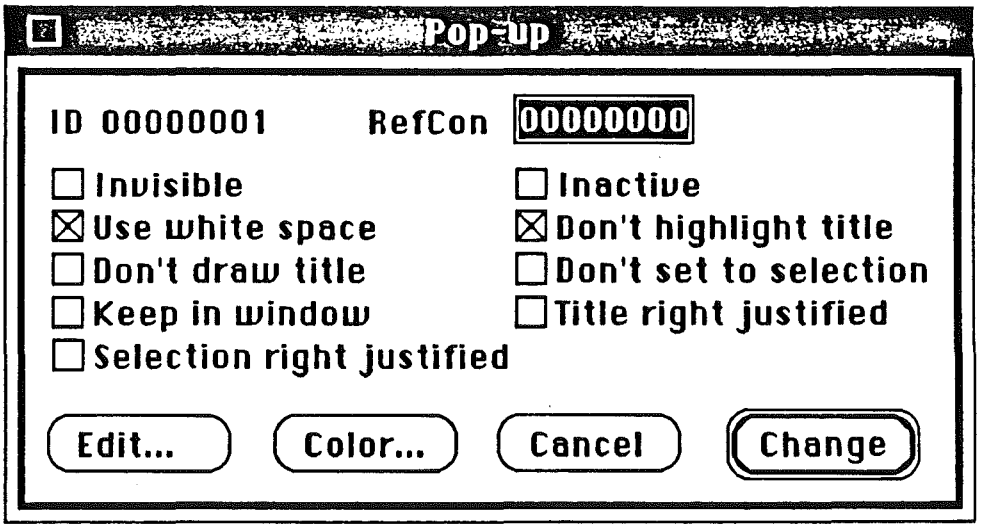


## Pop-ups

Pop-up controls are "menus in a button" and are an example of one of the powerful new extended control types. They offer the full capabilities of standard pull-down menus, but can be located anywhere within a window. Pop-ups are normally used to display a list of options or settings.

## Pop-up Control Editor

The Pop-up Control Editor allows you to control how pop-ups look and behave. Pop-ups can be created to extend beyond the top and bottom of their window or can be confined to the window. The **Edit...** button calls the Menu Editor, allowing you to create items for the Pop-up menu. Refer to the section on the Menu Editor for complete details.

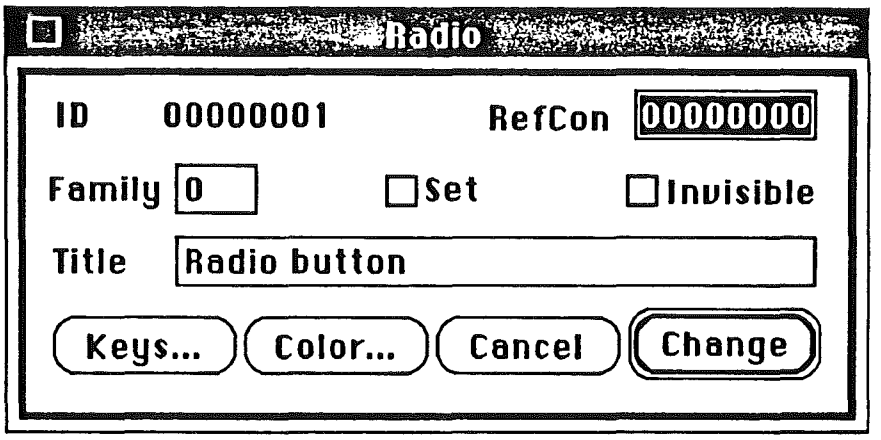



## Radio Buttons

Radio buttons are always arranged in sets of two or more and work just like the buttons on your car radio: only one button is "on" at a time. Radio buttons are useful for selecting one item from several different possibilities. Buttons that are related are called "families" and windows can contain more than one family of radio buttons.

### Radio Button Control Editor

The Radio Button Editor allows you to set or change the family number using the **Family** Line Edit. If you want the control to be created as the selected member of the family, check the **Set** checkbox. Only one member of a family can be set at a time.



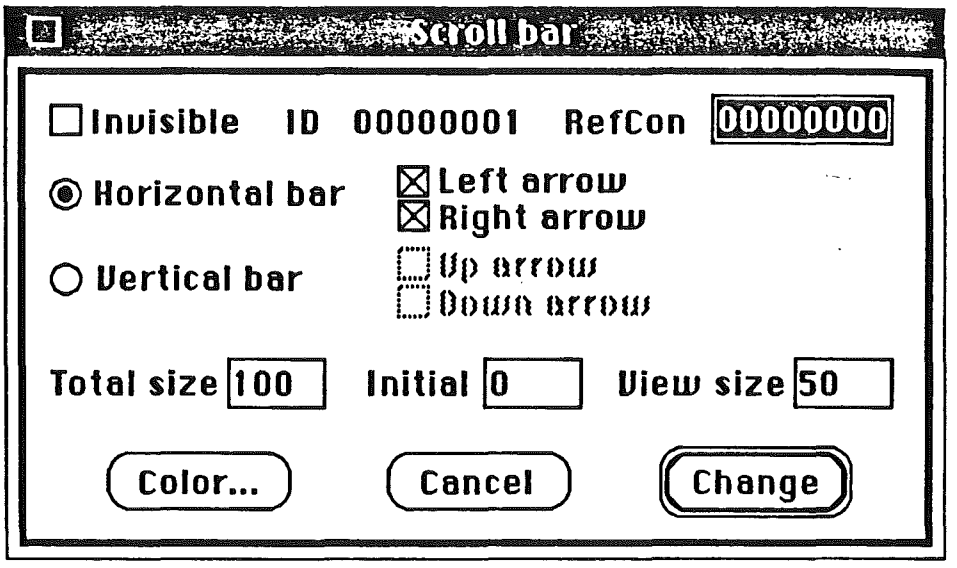

---

## Scroll Bars

Scroll bars are normally part of a window's frame but can also be created as stand-alone controls. The Window Manager automatically handles window frame scroll bars. Use the Scroll bar Editor to create stand-alone scroll bars.

### Scroll Bar Control Editor

Use the **Horizontal** or **Vertical** buttons to determine the position the Scroll bar will be created in. The **Left arrow** and **Right arrow** checkboxes control the drawing of arrows on the bar. The three line edits allow you to specify the **Total size** (in pixels) the scroll bar represents, the **Initial** location and the **View size**.



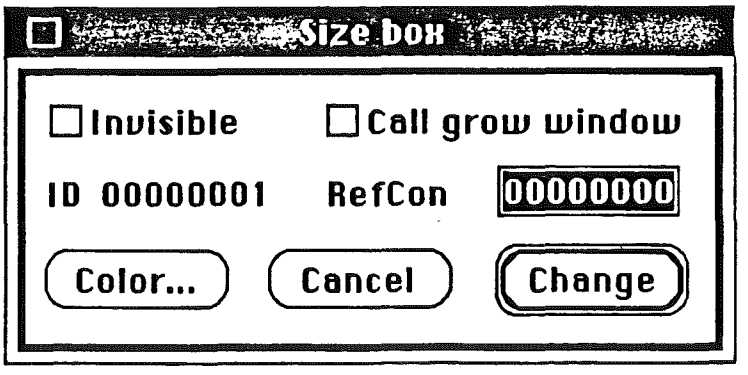

---

### Size Boxes

The Size box control is also usually associated with windows. Use the Window Editor to add a Size box control to the window. Size boxes can also be used within a window to control resizing an object, like a rectangle for instance.

### Size Box Control Editor

If checked, the **Call grow window** checkbox will allow `TaskMaster` to call `GrowWindow` when the control is repositioned.

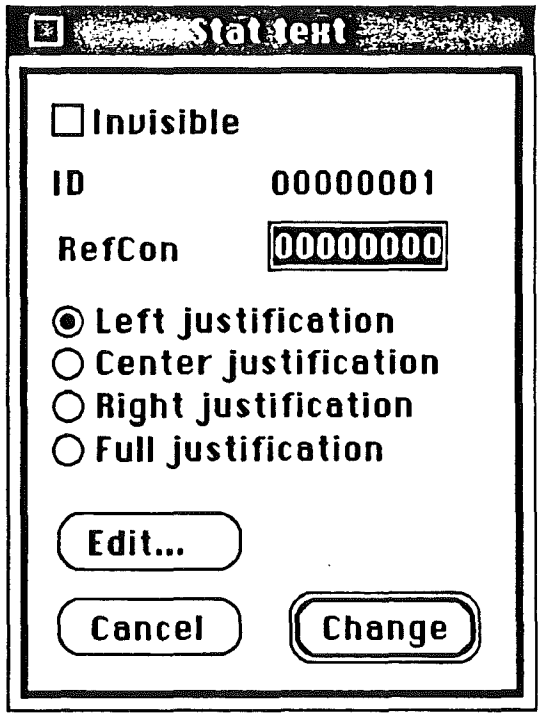


## Stat Text

Stat Text is an abbreviation for "static text": text that is displayed but can't be edited. These controls are useful for displaying messages and labels within a window.

### Stat Text Control Editor

Use the radio buttons to set how the text will be justified. The choices are **Left**, **Center**, **Right** and **Full**. Full justification is text that has straight left and right margins.

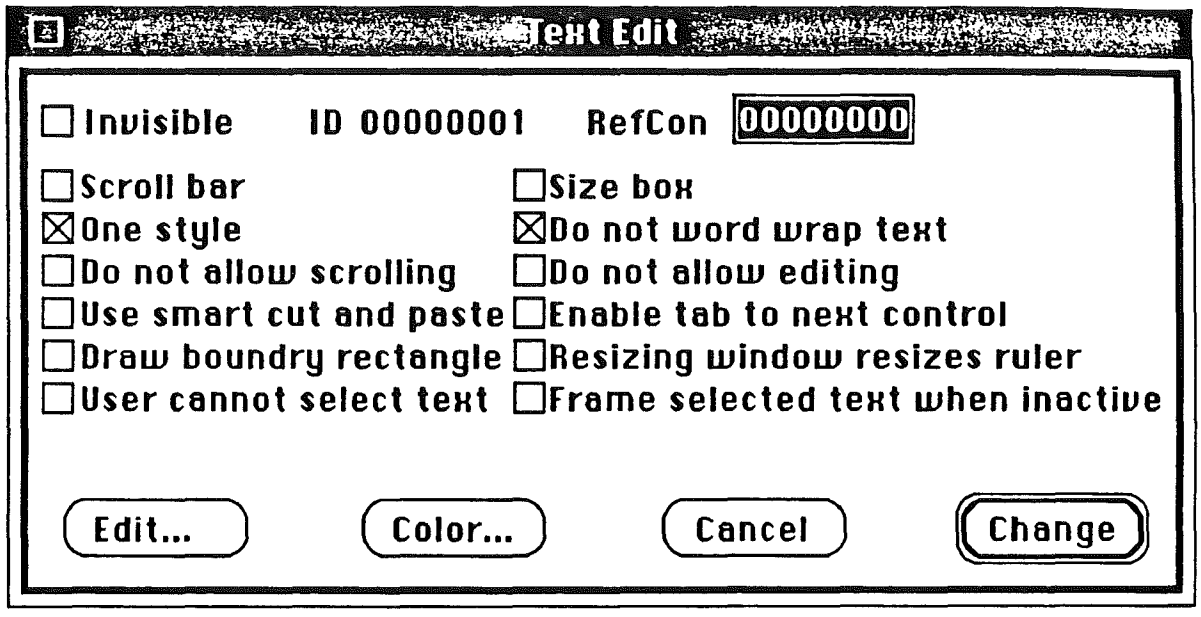



## Text Edit

Text Edit controls are a very complex new extended control type. They can best be described as an "instant word processor." Text Edits can occupy the entire window or only part of it. More than one Text Edit can be used in a single window, but only one Text Edit can be active at a time.

### Text Edit Control Editor

The checkboxes allow selection of the features Text Edit controls support. **Edit...** allows you to edit the text for the control by calling the String Editor. Refer to the section covering the String Editor for complete details.




---

#### Tips

• Although you can use the Control Editor to edit existing controls, it is always best to create controls that are tied to a window using the Window Editor. This "top-down" approach always yields the best results.

## HexASCII Viewer

## Contains:

### The default HexASCII Viewer

---

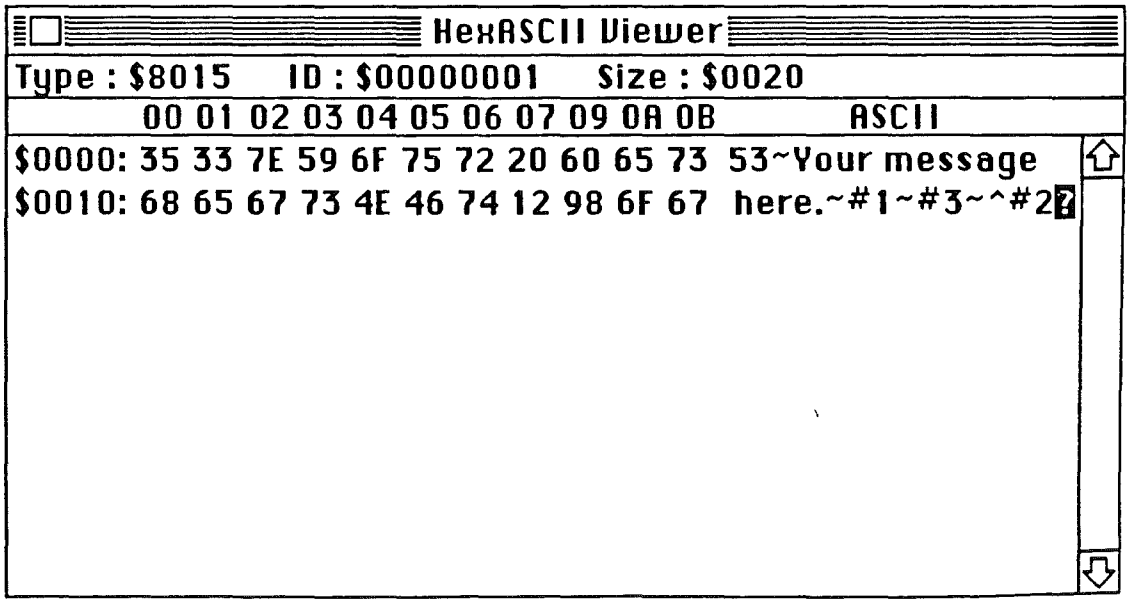
#### About the HexASCII Viewer

The HexASCII viewer displays a hexadecimal and ASCII representation of a resource, including the resource's type, ID, and size. This viewer is used to display resources whose format is unknown to Genesys.

---

#### Using the HexASCII Viewer

Although the HexASCII viewer is used automatically when no Editor exists, you may view ANY resource by selecting **Data View** from the Resource Chooser Window's **Item** popup.



## HexASCII Viewer

## Contains:

**The default HexASCII Viewer**

---

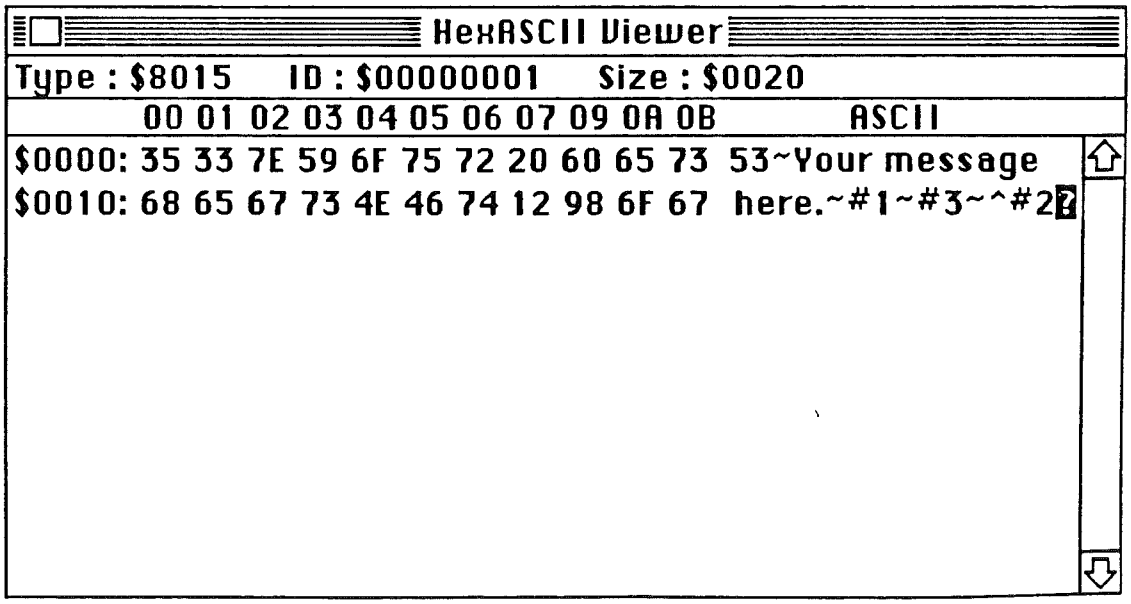
### About the HexASCII Viewer

The HexASCII viewer displays a hexadecimal and ASCII representation of a resource, including the resource's type, ID, and size. This viewer is used to display resources whose format is unknown to Genesys.

---

### Using the HexASCII Viewer

Although the HexASCII viewer is used automatically when no Editor exists, you may view ANY resource by selecting **Data View** from the Resource Chooser Window's **Item** popup.



## HexASCII Viewer

## Contains:

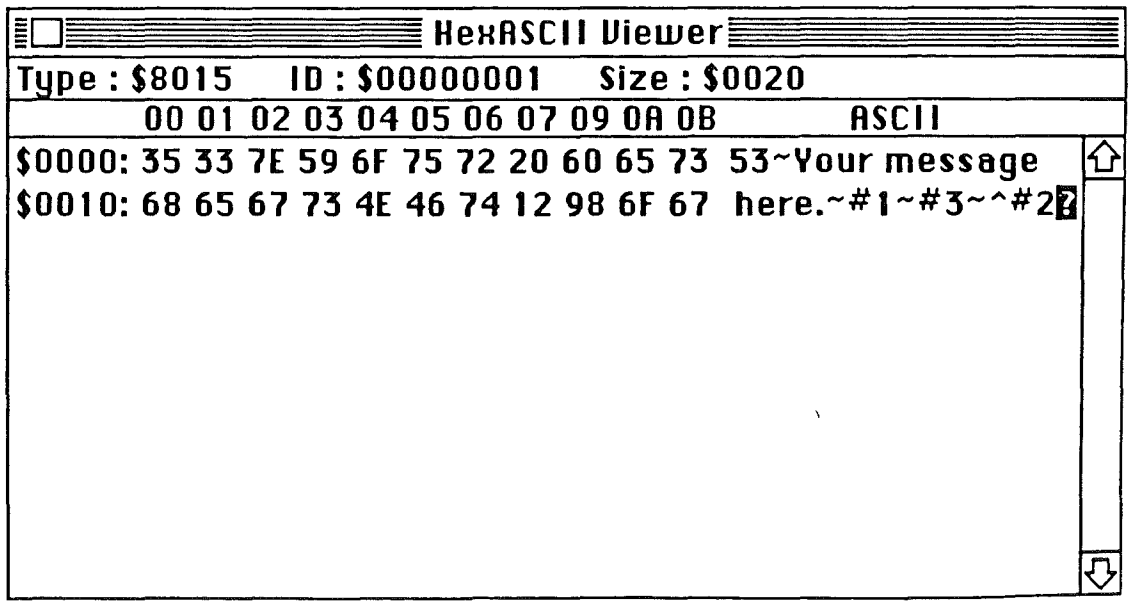
**The default HexASCII Viewer**

### About the HexASCII Viewer

The HexASCII viewer displays a hexadecimal and ASCII representation of a resource, including the resource's type, ID, and size. This viewer is used to display resources whose format is unknown to Genesys.

### Using the HexASCII Viewer

Although the HexASCII viewer is used automatically when no Editor exists, you may view ANY resource by selecting **Data View** from the Resource Chooser Window's **Item** popup.



## Icon and Cursor Editors

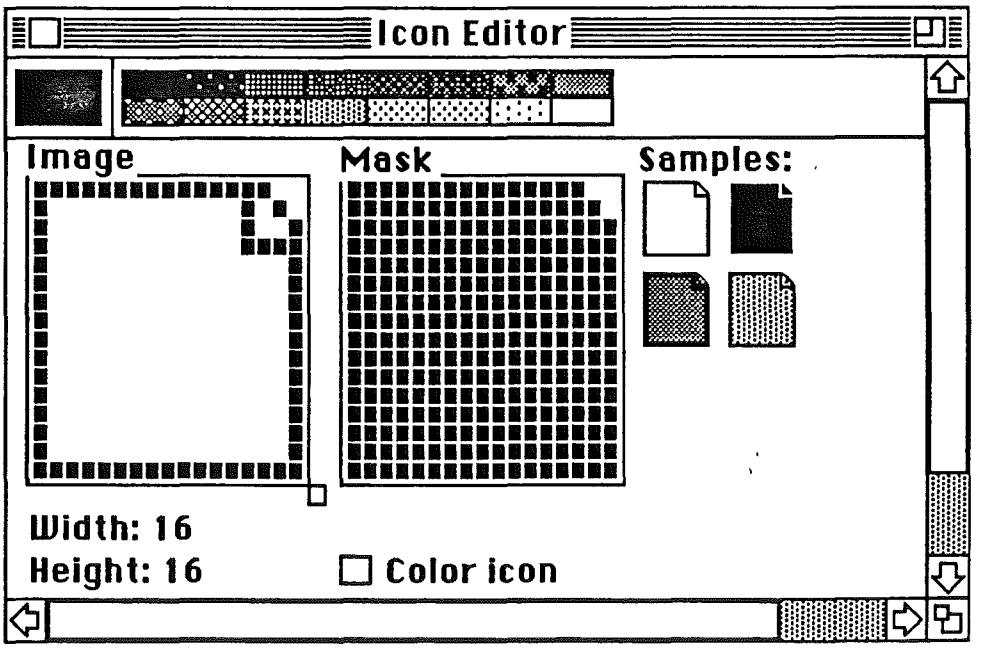
## Contains:

| | | |
|---|---|---|
| **Icon** | **rIcon** | **$8001** |
| **Cursor** | **rCursor** | **$8027** |

### About the Icon and Cursor Editors

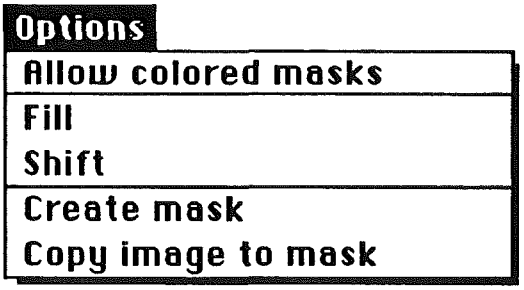
The Icon and Cursor Editors use a *fat bits* editing mode to edit icons and cursors. A full size sample of the icon or cursor is displayed to the right of the editing area. Both Icons and Cursors use an *Image* and a *Mask*. The image is the actual graphic representation of the Icon or Cursor that will be drawn, and the mask is applied to the image. Refer to the *Apple IIGS Toolbox Reference 2* for icons and cursor details.

### Using the Icon Editor



The Icon Editor works like most graphic programs for the Apple IIGS. The arrow cursor changes to a pencil when it is moved over the **Image** or **Mask** area. Pressing the mouse button draws pixels. The drawing color is selected from the *palette* at the top of the window. The current color is displayed at the left.

The Icon Editor uses an **Options** menu dissimilar from most other Editors as the Icon itself is an object and does not consist of editable objects.




**Allow colored masks** allows a colored mask rather than black and white.

**Fill** fills the icon with the current color. The cursor will change to a paint bucket when it is positioned over the **Image**. Click the mouse to fill.

**Shift** allows you to reposition the icon image. The cursor will change to a four-headed move cursor when it is positioned over the icon editing area. Click and hold the mouse button down while dragging the icon to its new position. The mask will also be repositioned accordingly.

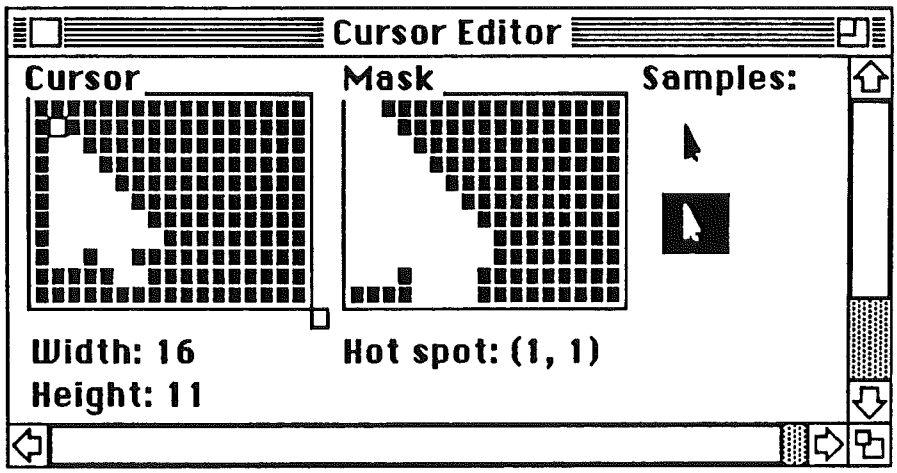
**Create mask** constructs a mask from the icon image area.

**Copy Image to mask** copies all non-white pixels from the **Image** area to the **Mask** area.

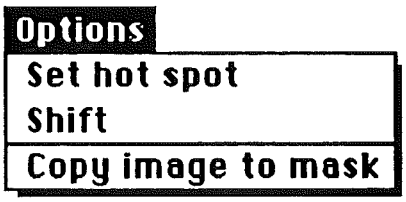
**Fill Mask** fills the entire mask rectangle with black or the selected color if the **Allow colored mask** option is checked.

### Using the Cursor Editor

The Cursor Editor is very similar to the Icon Editor. Only features specific to the Cursor Editor are covered. For complete editing details, refer to the Icon Editor covered previously.




The Cursor Editor also uses an **Options** menu dissimilar from most other Editors as the Cursor itself is an object and does not consist of editable objects. Only menu items specific to the Cursor Editor are covered. For details on other menu items, refer to the Icon Editor menu items covered previously.




Every Cursor must have an "active" area called a hot spot. For example, the tip of the pencil cursor is its hot spot. The Cursor Editor displays the hot spot with a red square which can be reset using the **Set hot spot** menu item. The cursor will change to a cross when it is positioned over the **Image** area. Click the mouse where you'd like the new hot spot to be.

### Tips

• The drawing color can be changed by holding down the ⌘ key and clicking on an existing pixel in the **Image** area.

• You can fill the Image by holding down the **option** key while over the **Image** area and clicking to fill.

• You can shift the Image by holding down the **Shift** key while over the **Image** area and dragging the image to its new location.

• You can use an Icon Editor such as `IconEd` or `DiceEd` to **Copy** finder icons to the clipboard and **Paste** those icons within Genesys into your program's resource fork using the Genesys Icon Editor.

## Menu Bar, Menu, and Menu Item Editors

## Contains:

| | | |
|---|---|---|
| **Menu Bar** | **rMenuBar** | **$8008** |
| **Menu** | **rMenu** | **$8009** |
| **Menu Item** | **rMenuItem** | **$800A** |

### About the Menu Editors

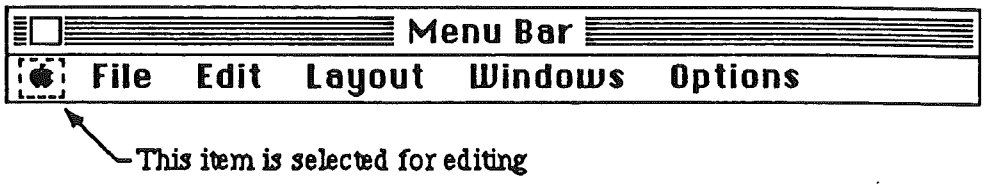
Every desktop program should contain a Menu Bar. A Menu Bar contains Menus which, when clicked on, pull the Menu down to display Menu Items. The Menu Editor provides full capabilities to edit or create Menu Bars, Menus and Menu Items, and is also used by the Window and Control Editors to edit popup controls.

If you are creating menus for a new application, you should start with the Menu Bar Editor and create your program's menus and menu items from there, although any of the Editors are suitable for customizing a program's menus. This "top down" approach always yields the best results.

### Using the Menu Bar Editor

The Menu Bar Editor looks like a real menu bar, only its menu bar is contained in a window. When you **Add** a new menu bar to the file, the Editor normally generates the new menu bar with a full complement of standard menus and menu items, as defined by the *Apple Human Interface Guidelines*.

Menus normally have two spaces before and after their titles to separate them on the menu bar, and Genesys creates its menus this way. These spaces are recommended, but you are free to change them if you'd like. The only exception is the  menu, which is created by using the "@" symbol without leading or trailing spaces. The Menu Manager toolset will draw the rainbow colored  icon instead of the "@" character.




Some menus may appear disabled (dimmed) to denote that its menu items are not available. The Menu Editor allows you to create and edit these disabled menus.

The Menu Bar Editor uses the standard Genesys **Options** menu already described at the beginning of this Chapter, although minor clarification is required.

**Edit menu...** becomes active when a menu is selected. This command opens the Menu Editor for the selected menu, which allows you to edit the menu and menu items. See the next section for information on using the Menu Editor.

**Add menu** adds a new menu to the menu bar. Normally, the menu will be added to the end of the menu bar. If a menu is selected however, the new menu will be inserted before it.

Selecting **Clear** from the **Edit** menu or using the **delete** key will remove the selected menu from the menu bar.

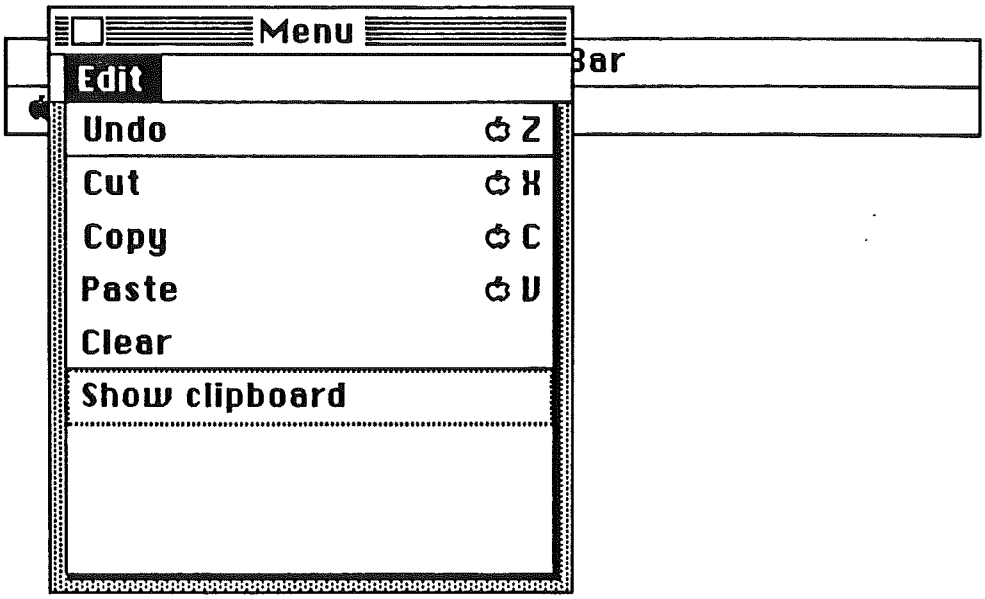
---

### Using the Menu Editor

The Menu Editor has many features in common with the Menu Bar Editor which won't be covered again.

The Menu Editor supports scrolling exactly like the Menu Manager in system 5.0. You can drag menu items and scroll simultaneously to position an item at the top or bottom of a long item list.

Double-clicking a menu item creates three side-by-side line edits for you to specify the item's mark character, text and keyboard equivalent (see the Menu Item Editor). Functionally, no difference exists editing an item with the Menu or Menu Item Editor.




The Menu Bar Editor uses the standard Genesys **Options** menu already described at the beginning of this Chapter, although minor clarification is required.

The first menu item changes to reflect the current selection. If the selection is the menu title, **Edit menu data...** is displayed, if it is a menu item, **Edit item data...** is displayed. In either case, a moveable modal dialog is used. See the sections: *Menu Data Dialog* and *Item Data Dialog* below for descriptions.

**Add item** will add a new item to the menu. Normally, the new item will be added to the bottom of the menu unless a menu item is selected, in which case the new menu item will be inserted before it.

Selecting **Clear** from the Edit menu or using the **delete** key will remove the selected menu item from the menu.

---

### Using the Menu Item Editor

Menu items are made up of three parts: the mark character, the menu item text, and the keyboard command equivalent. Double-clicking an item opens three line edits that allow you to create or change these parts.

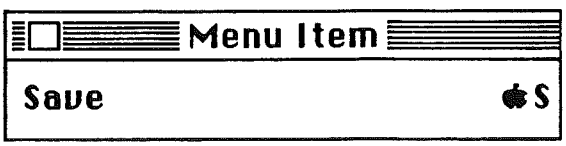
mark character: The mark character is optional and, if present, is displayed to the left of the item's text. Four special characters are available from the keyboard:

| | |
|---|---|
| **control Q** | ⌂ |
| **control R** | ✓ |
| **control S** | ◆ |
| **control T** | ⌘ |

item text: The item text should describe what the menu item does. Menu items that open a dialog normally have a trailing ellipsis like: **Open...**. A menu item may appear dimmed to inform the user that it is not available at that time. Items can also appear in several styles as described later. There are two kinds of item separators. One type is created with the **Divider** checkbox in the **Item Data** dialog. This type of divider does not take-up extra space in your menus. The other type, a separator, can be created by entering a hyphen (-) as the item's text. Make sure that you disable this type of separator.

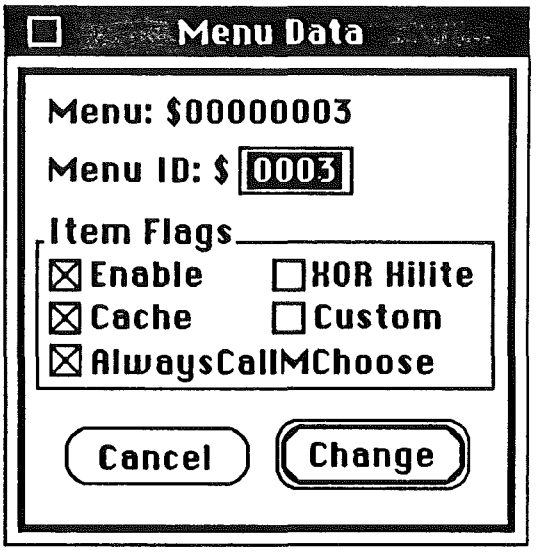
keyboard equivalent: The keyboard equivalent appears next to the ⌂ symbol and is optional. Key equivalents invoke the menu item using the keyboard. The keyboard equivalent line edit will allow you to insert two characters. The first character, normally uppercase, is the one that will be displayed on the menu. The second character is usually the lowercase equivalent. If you only want one key to activate the item's command, input it twice.

The Menu Item Editor uses one menu item for the **Options** menu. Selecting **Edit item data...** opens a moveable modal dialog. The *Item Data Dialog* section that follows describes this dialog.



### The Menu Data Dialog

The Menu Data dialog has five check boxes to control how menus are created. **Enable** will create the menu enabled. A disabled menu appears in dim text. **Cache** utilizes the Menu Manager's cacheing scheme for very quick menu drawing which should normally be set. The other three checkboxes are special features added for experienced programmers to create "tear away" and custom menus.

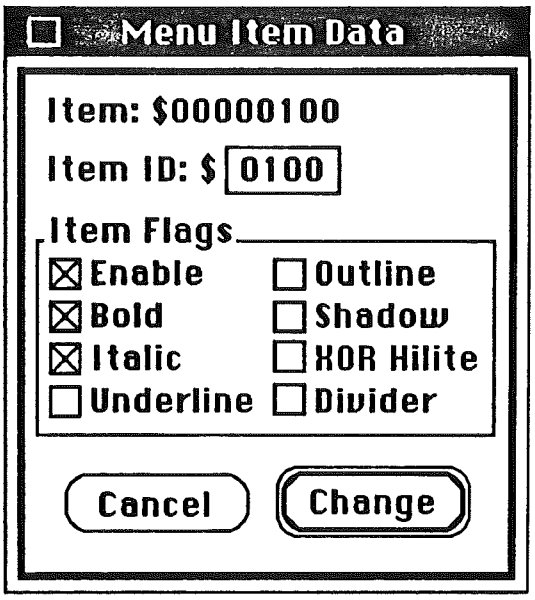



The menu ID field is usually based on the menu's resource ID, but not always. You may change the ID if needed. After you have made your changes, select **Cancel** to restore the original values or **Change** to keep the changes.

### The Item Data Dialog

The Item Data dialog is similar to the Menu Data dialog, but provides control of how menu items are displayed. The eight checkboxes select the styles and attributes an item may have.

- *Note:* The System Software won't allow underlining of the system font.




As you select the checkboxes, the Editor redraws the item to reflect the changes. After you have made your changes, select **Cancel** to restore the original values or **Change** to keep the changes.

---

### Tips

• Hold down the **option** key when selecting **Menu Bar** from the Resource Chooser Window to create an empty menu bar.

• The default menu bar, menus and items are contained in the Menu Editor's resource fork. You can customize these items to suit your needs.

• Menus and Menu Items can be repositioned by dragging them with the mouse.

• A menu or item can be deselected with the **esc** key or clicking outside the selection.

• Use the **Test menu...** item to test your menu bar.

• For best results, remember to build your program's menu bars starting with the Menu

Bar Editor and working down from there.

## Picture Editor

## Contains:

| | | |
|---|---|---|
| **Picture** | **rPicture** | **$8002** |

### About the Picture Editor

Pictures, as opposed to bitmaps, are a collection of drawing commands that tell *QuickDraw II* how to draw a picture. Pictures can be used by themselves or with Picture Controls, as covered in the section on the Control Editor.

The Genesys Picture Editor, although simple, relys heavily on the Clipboard. Since different people use different methods to create pictures, the Picture Editor assumes you have access to either a Scrapbook desk accessory such as SSSi's Super Scrapbook and/or a Paint Program.

### Using the Picture Editor

The Picture Editor will allow you to create or edit any picture resource. To change the picture the resource contains, simply use your favorite paint program or scrapbook desk accessory to place the desired picture on the Clipboard, and use the **Edit** menu to **Paste** the picture into the Picture Editor.

To edit an existing picture, simply use the **Edit** menu again to **Copy** the existing picture to the clipboard, and **Paste** the picture into your paint program or Scrapbook desk accessory.

### Tips

• You can use Genesys to edit the default picture this Editor creates.

## String Editor

### Contains:

| | | |
|---|---|---|
| **C String** | **rCString** | **$801D** |
| **C1 Input String** | **rC1InputString** | **$8005** |
| **C1 Output String** | **rC1OutputString** | **$8023** |
| **Pascal String** | **rPString** | **$8006** |
| **Text** | **rText** | **$8016** |
| **Text Block** | **rTextBlock** | **$8011** |
| **TextBox2 String** | **rTextBox2** | **$800B** |
| **W String** | **rWString** | **$8022** |

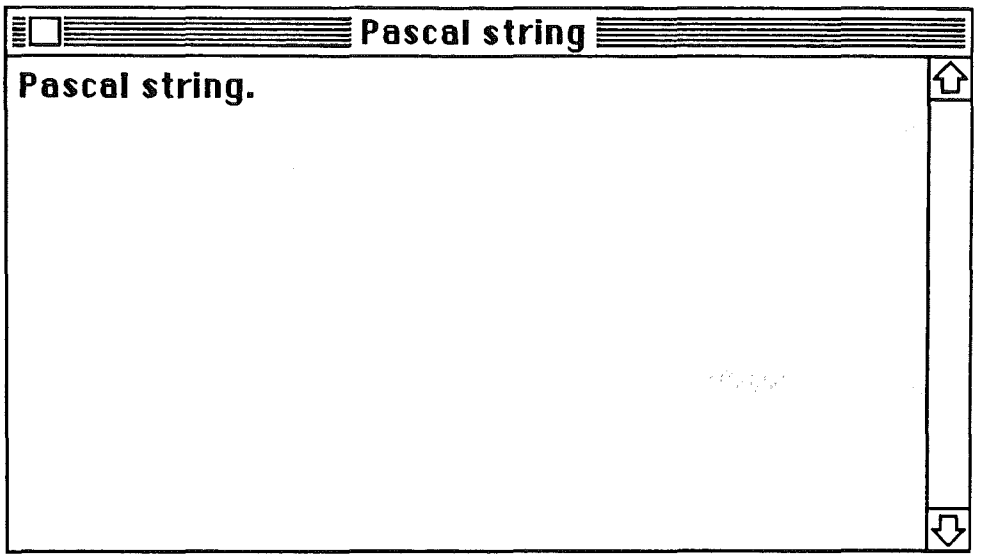
### About the String Editors

The String Editor allows creating and editing a variety of string types as follows:

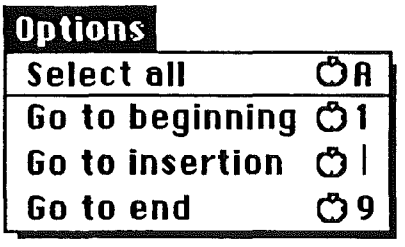
| | |
|---|---|
| **C string** | 1 to 65,535 characters including a null ($0) terminator. |
| **C1 Input string** | GS/OS Class 1 input string with leading word length and 1 to 65,535 characters. |
| **C1 Output string** | GS/OS Class 1 output string with word buffer size, word length, and 1 to 65,535 characters. |
| **Pascal string** | Byte length followed by 1 to 255 characters. |
| **Text** | Unformatted block of text of up to 65,535 characters. |
| **Text Block** | Unformatted block of text of up to 65,535 characters. |
| **LETextBox2 string** | String of 1 to 32,767 characters for LETextBox2 tool call. |
| **W string** | Word String with word length and 1 to 65,535 characters. |

## Using the String Editors

The String Editors all look and work alike. Each String Editor fully supports the **Cut**, **Copy** and **Paste** menu items from the **Edit** menu.




The String Editor uses an **Options** menu dissimilar from most other Editors as the String itself is an object and does not consist of editable objects.



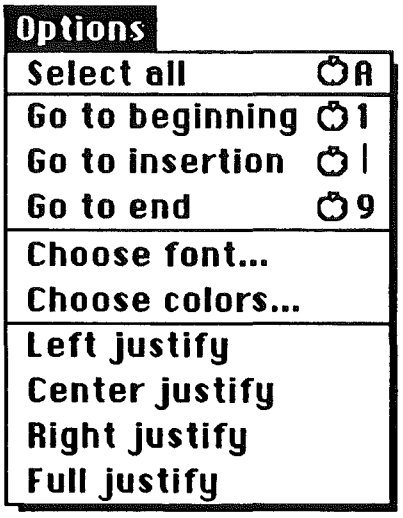

**Select all** will select all text in the editing window.

**Go to beginning** will scroll the window to the beginning of the text and place the insertion cursor before the first character.

**Go to insertion** will scroll the window to display the area where the insertion cursor currently resides.

**Go to end** will scroll the window to the end of the text and place the insertion cursor after the last character.

Additional items are available for the **LETextBox2 String** Editor, which also supports fonts, styles, colors and justification.




**Choose font...** allows you to choose the font, style, point size and other attributes for the currently selected text.

**Choose colors...** allows you to choose the foreground and background colors for the currently selected text.

**Left justify, Center justify, Right justify** and **Full justify** allow you to choose the justification for the text. Justification applies to all text in the window.

- *Warning:* The System Software contains a bug. LETextBox2 data will only work for small blocks of text with a reasonable number of styles.

### Tips

• You can use the String Editor to tailor the strings for menus and controls.

• You can edit the String Editor in the `GEN.EDIT` folder to change the default strings each editor creates.

• You can use **View Source...**, **Copy**, and **Paste** to load in and place the text from any file into your program's resource fork.

• PLEASE use discretion when changing colors and fonts using the LETextBox2 editor. Inappropriate color and font combinations can ruin a good program!

## Tool Table Editor

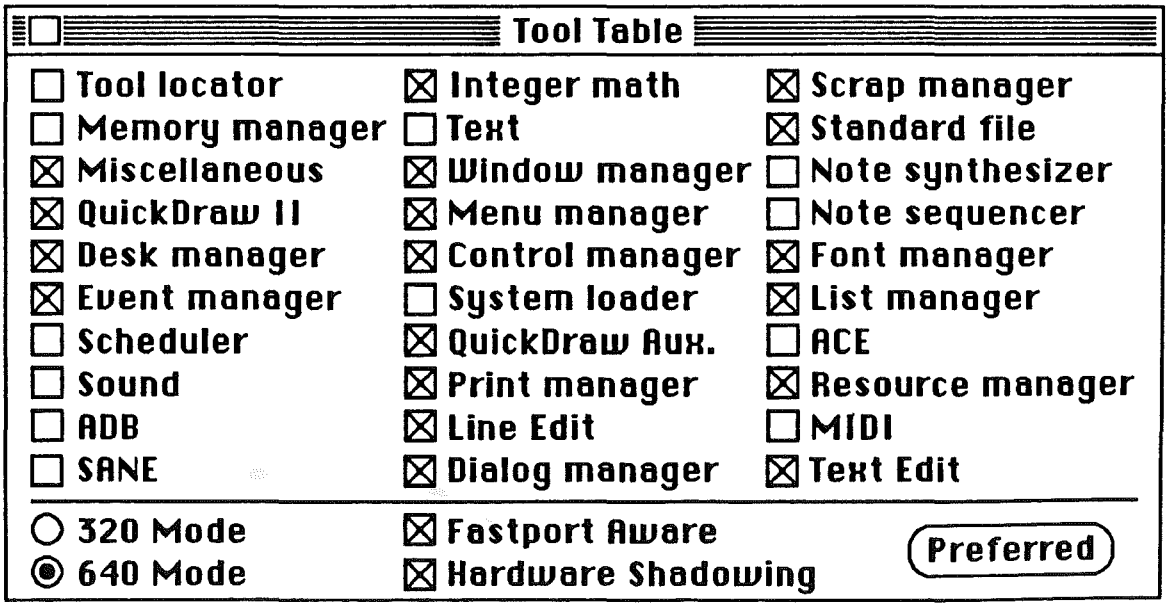
## Contains:

**Tool Table** **rToolStartup** **$8013**

### About the Tool Table Editor

The Tool Table Editor creates the Tool Startup resource required by the `StartUpTools` call. In addition to setting the *toolsets* that will be created at startup time, you may select the graphics mode (320 or 640) and *fastport* and *hardware shadowing* modes that your program will execute in.

`StartUpTools` is a deceptively simple but powerful tool call. Toolsets must be started in a very particular order. If they are not, strange and unexpected behavior will result. `StartUpTools` insures that the tools passed to it in the Tool Startup record are started in the correct order.



### Using the Tool Table Editor

Make your selections by clicking in the check boxes of the tools your program requires. The **Preferred** button will select the preferred set of tools for you. The default settings for **Fastport aware** and **Hardware shadowing** are checked.

- *Warning:* Many of the source code generation templates expect a tool table to be defined. If they don't find one, they may generate a tool table for you that might not contain everything you need, or may return an error when the StartUpTools call is made. Always define a tool table for each and every program you create.

### Tips

• Make sure to refer to the *Apple IIGS Toolbox Reference Volumes 2* and *3* when selecting or not selecting tools in the preferred tool setup that Genesys uses. Many tools have dependencies between each other that you'll need to be aware of.

## Two Rects Editor

## Contains:

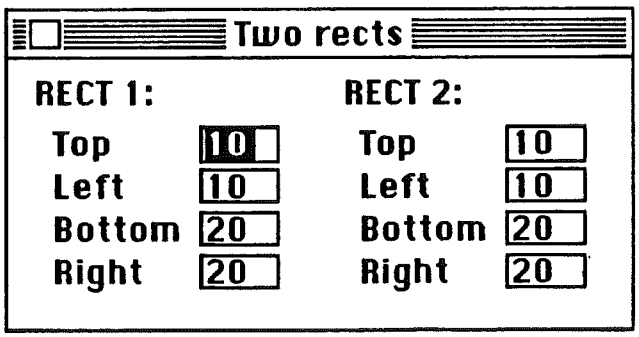
| | | |
|---|---|---|
| **TwoRects** | **rTwoRects** | **$801A** |

### About the Two Rects Editor

The Two Rects editor is used to create and store two rectangles. Desk accessories such as the Control Panel use the Two Rects resource to keep the window coordinates that it uses for 320 and 640 mode.

### Using the Two Rects Editor

The Two Rects Editor displays both rectangles contained in the resource. Type in the values for the rectangle in the line edits. No validation is performed on the rectangles. If the Top coordinate is greater than the Bottom coordinate Editor assumes you know what you are doing. If you type in nonsense characters, such as `PQ@#`, the editor will set these values to `0`.



## Tips

• Use the Two Rects editor to keep your 640/320 mode window coordinates.

• You can edit the Two Rects Editor's resource fork to modify the rectangle values used when creating a new Two Rects.

## Window Editor

## Contains:

| | | |
|---|---|---|
| **Window** | **rWindParam1** | **$800E** |

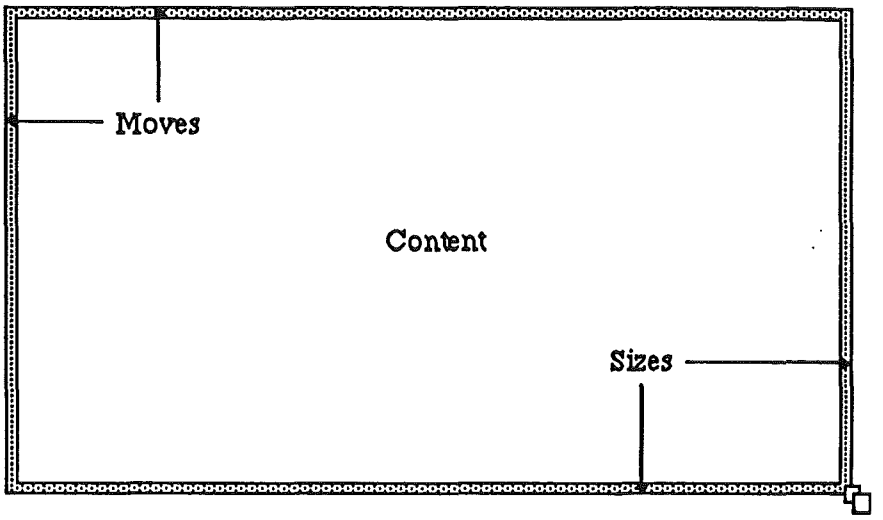
### About the Window Editor

The Window Editor allow the creation of colorful windows and dialogs containing all twelve of the new System 5.0 extended controls. Windows play a major part in the operation of a program as most information and user manipulation is done with windows.

### Using the Window Editor

Although the Window Editor uses the Control and Window Color Editors for the creation and editing of controls and window colors, only those features specific to the Window Editor are covered here. Refer to the sections covering the Control and Window Color Editors for complete details on using these Editors.

The only visual difference between an actual window created or edited using the Window Editor and an actual window in a program is the yellow border that surrounds the window. This border is used by the Window Editor to allow window moving and resizing when no title bar or size box is part of the window. The right-bottom edges of the yellow border resize the window and the top-left edges of the yellow border move the window. The cursor changes over these areas to indicate these actions. Window positioning can also be accomplished by using the **Windows** menu from the Genesys menu bar allowing you to position the window on the desktop horizontally, vertically and both. All other features of a window created by the Window Editor look and work the same as real program windows: you can close it by its close box, drag it by its title bar, zoom it by its zoom box and resize it by its size box.




The Window Editor makes extensive use of the **Options** menu by allowing you to add elements such as a title bar, close box, zoom box, and scroll bars to the window being created or edited. These elements have additional parameters that can be edited by double-clicking.

**Edit Colors...** is always active allowing you to edit the window colors at any time by calling the Window Color Editor. All changes made to the Window Colors are reflected in the Window Editor after closing the Window Color Editor.

The **Add control...** menu item is also always active, displaying the **Choose control to create...** dialog. Refer to the Control Editor for complete details. Removing a control is accomplished by choosing **Clear** from the **Edit** menu. For a complete description on control manipulation, see the section later on Editing Controls.

**Zoom Window** allows you to zoom the window between the size of the zoom rect and its previous size. This menu item is functionally equivalent to using a zoom box on a window's title, but is provided for windows without a zoom box.

**Set zoom rect** sets the window's maximum zoom size. Setting the window's desired position and size on the desktop and selecting this menu item will record this position as the position and size to use when zooming the window with a zoom box.

**Clear zoom rect** becomes active if a zoom rect for the window has been set to

anything other than the default zoom rect, which is the size of the desktop. Most programs use the size of the desktop as the zoom rect for a window.

**Title bar** adds or removes the title bar.

**Close box** adds or removes the close box and is only active if the window contains a title bar.

**Zoom box** adds or removes the zoom box and is only active if the window contains a title bar.

**Info bar** adds or removes the info bar.

**Right scroll bar** adds or removes the vertical scroll bar.

**Bottom scroll bar** adds or removes the horizontal scroll bar.

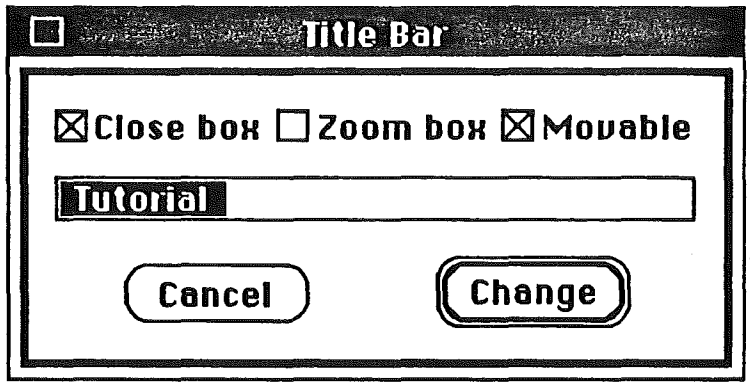
**Size box** adds or removes the Size box. This item is only inactive when at least one scroll bar is present. Removing the size box removes the scroll bars.

- *Note:* If you use right and bottom scroll bars, the size box is added automatically. The Window Manager requires a size box if both scroll bars are present, and does not allow only a size box without at least one scroll bar present.

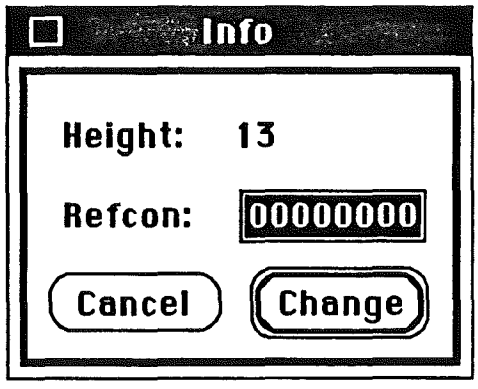
### Editing Window Elements

The window's content, title bar, info bar, scroll bars and size box can be selected by clicking. Marching ants highlight the selection. Double-clicking, pressing **return** or selecting **Edit...** from the **Options** menu displays a moveable modal dialog with additional editing features.

The **Title Bar** window allows you to specify the title, close box, zoom box, and if the window can be moved on the desktop. Make sure to add a space before and after the title if using a lined or dithered title bar.

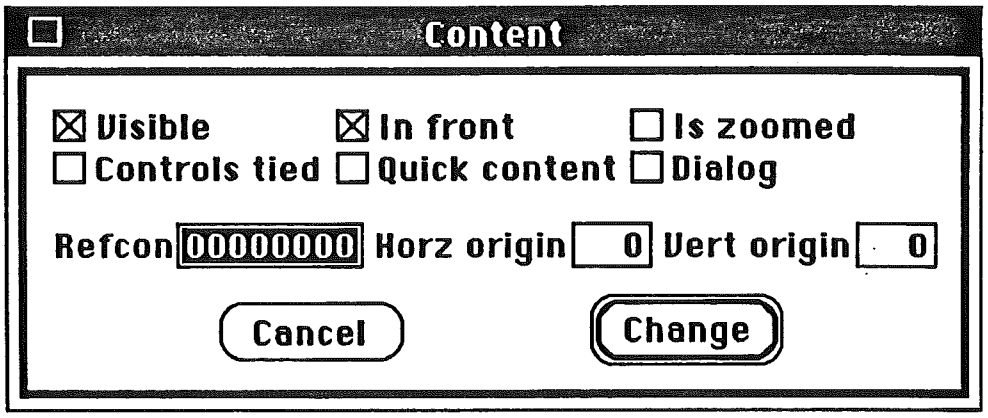



The **Info** window allows you to specify the RefCon for the info bar, and shows the height of the current info bar. To change the info bar's size, move the cursor over the info bar in the window until the cursor changes to a vertical move cursor. Click on the info bar and move it vertically until the info bar is set at the height desired.

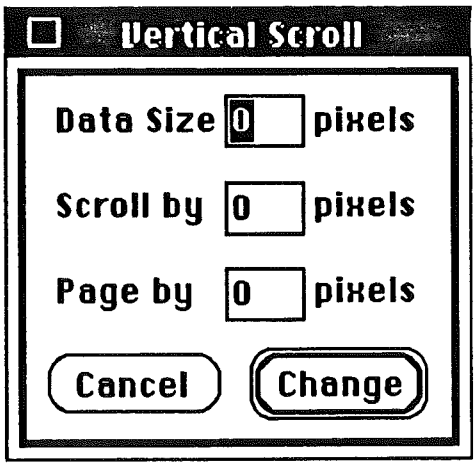



The **Content** window allows you to specify various attributes for the content of your windows. Under most circumstances, the defaults supplied will be appropriate with one exception; if you wish to create a dialog type window, you would click in the **Dialog** check box.

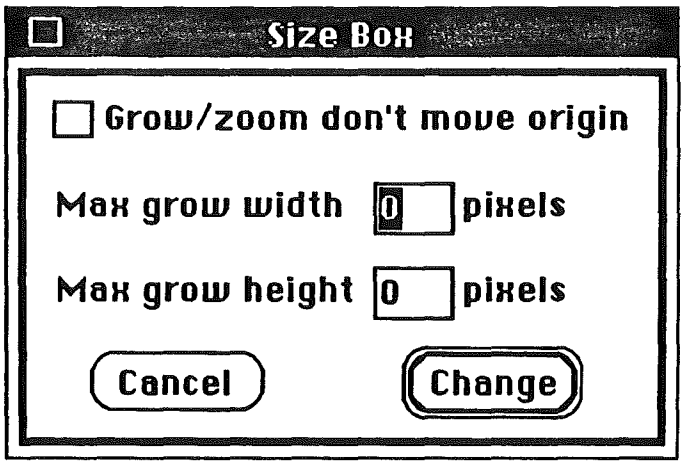
A dialog window and standard window are functionally equivalent to each other, except a dialog window does not have anything but a content area. If the window is specified as NOT being in front or visible, the Window Editor will show the yellow border area in a pattern rather than yellow.




The **Horizontal** and **Vertical** windows allow you to specify the *data size* that the scroll bars will represent, and how much the scroll bars will *scroll* and *page* by. Scrolling is accomplished by clicking in the arrows on the scroll bar. Paging is accomplished by clicking in the checkered area.




The **Size Box** window allows you to specify the maximum grow height and grow width that the window will grow when the user resizes the window with the size box, as well as specifying whether or not the grow and zoom boxes move the windows origin or not.




### Editing Controls in a Window

Clicking on a single control selects that control for editing shown by the red marching ants around the control. Clicking and dragging a control allows you to move the control, and leaves the control selected for further editing. When a control is selected the cursor will change to a size cursor over the bottom-right corner of the control allowing the control to be resized and a move cursor over the top-left corner of the control allowing the control to be moved. Double-clicking allows you to further define a control using the Control Editor. Refer to the Control Editor for complete details on editing controls.

---

### Tips

• After selecting the window content or control you can use the **Option** key and the arrow keys to fine positioning your windows and controls.

• Turning the coordinates on from the Genesys menu bar will show you the actual coordinates when sizing and moving a window, control or the info bar. The window's coordinates are based on the desktop's origin (upper-left corner) and a control is based on the window's origin (upper-left corner of the window's content). When moving the info bar, the vertical coordinate shows the height of the info bar.

## Window Color Editor

## Contains:

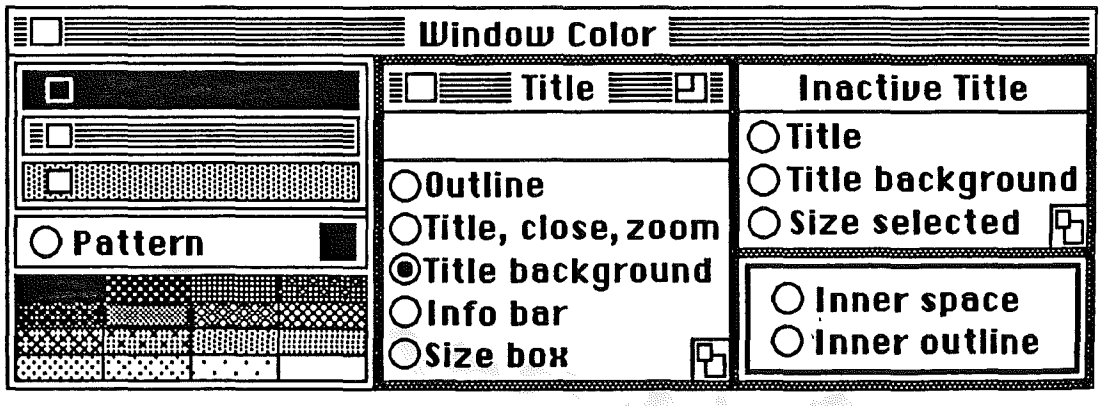
**Window Color rWindColor $8010**

### About the Window Color Editor

Your Apple IIGS is capable of displaying windows in many colors and the Window Color Editor allows you to create or edit these colors. Window colors are actually kept in a resource referred to as a *window color table.*

### Using the Window Color Editor

The three small windows on the right against the desktop background display how your color selections will look. Each of the radio buttons in these windows apply the that window's state; active, inactive, and dialog. There is no need to set the colors for a window you are not using, such as setting the dialog window colors when you are using a standard document window.




The color palate allows you to select the color that will apply to a window element (the selected radio button). For example, selecting the **Title background** radio

button and clicking on red in the color palate will change the title background in the active window to red. All other radio buttons for this Editor work in this fashion.

The **Pattern** radio button allows you to select the color that is used in conjunction with the title pattern selected. For example, selecting the **Pattern** radio button and clicking on white in the color palate followed by a click on the lined title bar above the **Pattern** button would result in a white lined title bar.

- ***Warning:*** PLEASE use discretion when creating or changing colors for your windows. Inappropriate window color combinations can ruin a good program!

---

### Tips

• Although you can use the Window Color Editor directly to create or edit window color tables, it is best to use this Editor from the Window Editor. See the Window Editor section for complete details.

# Appendix A: Source Code Generation Language

The Genesys Source Code Generation (SCG) Language provides the means for Genesys to create source code in virtually any language from the resources in any file. The information on this language is furnished here to allow you to customize existing, or write new language template files for generating source code from within Genesys.

## SCG Template Files and Language Types

An SCGL template file consists of an Apple source file (type $B0) with the auxtype field set to match that of the target language. Several of the languages available for the IIGS do not use the Apple source filetypes, nor do they have assigned language numbers set in the auxtype field. For these languages, SSSi has assigned a psuedo-language number so the Source filetype can be used. Currently the language descriptions are hard-coded within the Genesys shell and are:

| Language | Auxtype | | Output file |
|---|---|---|---|
| Resource Info Report | $FE | (254) | TXT |
| APW/ORCA ASM65816 | $03 | (3) | SRC ASM65816 |
| APW C | $0A | (10) | SRC APWC |
| APW REZ | $15 | (21) | SRC REZ |
| Lisa 816 Assembler | $83 | (131) | TXT |
| Merlin16 | $FF | (255) | TXT (MSB set) |
| Micol Advanced BASIC | $90 | (144) | TXT |
| Micol MACRO Assembler | $91 | (145) | TXT |
| MPW IIgs Assembler | $80 | (128) | TXT |
| MPW IIgs C | $81 | (129) | TXT |
| MPW IIgs Pascal | $82 | (130) | TXT |
| ORCA C | $08 | (8) | SRC CC |
| ORCA Pascal | $05 | (5) | SRC PASCAL |
| TML Pascal II | $9E | (158) | SRC TMLPASCAL |

Languages with auxtypes greater than $7F (127) will force the code generator to produce 7 bit ASCII output to a TXT file (type $04). Merlin16 is currently a special case extension: setting the auxtype value to $FF (255) will still produce a text file, however, the output will be set to 8 bit ASCII (MSB SET).

- *Note:* The **CTRL** command (described later) allows the state of the character's MSB to be forced set or clear. The default setting for Merlin16 files may go away in the future, so please set the MSB with a **CTRL** in all Merlin templates.

The templates for any one language can reside in one or more files. The file's type and language values are used to determine which template files will be accessed during generation. There must be at least one template for each supported language.

## Registers

The SCG Language makes heavy use of storage registers. These registers are based on the storage structures of programmable calculators. There are 256 registers in the Genesys code generator. Each is 4 bytes wide and can contain any signed or unsigned long value or pointer. They are addressed in the SCGL by register names R0 through R255. For example, to load register 26 with the decimal value 1024, the template would contain the following command line:

```
LOAD R26 1024
```

Registers `R0`, `R10` through `R19`, and `R200` through `R255` are reserved by the SCGL. You must NEVER change the contents of these! Registers `R1` through `R9` are used for the sequential storage of output arguments used by the **WRITE** command. See **WRITE** and the ANSI C formatted output section for a complete description of the output registers. All other registers (`R20..R199`) are available for the template program to use in any way it chooses.

**Reserved register usage:**

R0 After any **WRITE** operation, this register will contain an integer count of the number of arguments processed.

| | |
|---|---|
| R10 | Pointer to the start of the current template. |
| R11 | Current resource memory address. |
| R12 | Current resource size in bytes. |
| R13 | Current resource type. |
| R14 | Current resource ID. |
| R15 | Running count of the resources of the current type. |
| R16 | Count of characters returned by a **READ** text operation. |
| R17 | Data flag (usage varies by command). |
| R18 | Total count of resources of the current type. |
| R19 | Current resource attributes. |

## Command Structure

SCGL command syntax is limited to the following forms:

| | |
|---|---|
| Form 1 | COMMAND |
| Form 2 | COMMAND register |
| Form 3 | COMMAND argument |
| Form 4 | COMMAND register MODIFIER |
| Form 5 | COMMAND register MODIFIER argument |
| Form 6 | COMMAND 7 bit_ascii_text |
| Form 7 | COMMAND register 7 bit_ascii_text |

In all cases, the line must begin with a command. The command may have leading space or tab characters, however, blank lines are not allowed. Lines are terminated by a single carriage return ($0D).

Commands and modifiers may be upper, lower or mixed case.

Register references consist of the letter 'R' or 'r' followed by a decimal integer number between 0 and 255 inclusive.

Arguments may consist of a register reference, or an unsigned long decimal or hexadecimal number. Hexadecimal numbers may start with the character '$' or in C convention, the sequence '0X' or '0x'. Binary format is not permitted. When a register reference is given, the value contained in the register is actually used.

Form 6 and Form 7 structures are special. Refer to the **WRITE** and **PTEXT** commands respectively for more information.

## Template Structure

A Genesys SCGL template must contain at least one **TYPE/DONE** command pair. The template must also contain a **TITLE/DONE** pair to provide a header and/or terminator to the output source file. See the description for **TITLE** and the *Example SCGL Template* section later in this appendix.

The **TYPE** command is the starting point for the template program and the **DONE** command is the ending point. An example template that outputs the string "Copyright 1989, 1990 by SSSi, Inc." for each ICON resource ($8001) in a work file would look like this:

```
TYPE $8001
WRITE
Copyright 1989, 1990 by SSSi, Inc.\n
DONE $8001
```

This example is larger than the minimum template (**WRITE** and text are optional).

- *Warning:* There should always be a null template defined for a language. The null template is used to handle undefined or unimplemented types. The null template can consist of merely a **TYPE** 0 / **DONE** 0 command pair, a full hex dump template like the one found in the ORCA/M SCG template file, or anything in between. If the **TYPE** is $8014 you MUST use a **DONE** 0, not **DONE** $8014.

Template files may not contain blank lines. If you want to add blank space or comments to your SCGL Template file, see the command description for #.

## The Commands

The commands and modifiers presented below are listed alphabetically by primary function. Occasionally, a command may have a list of modifiers. Unless otherwise noted, one modifier must be selected - no items are optional. In the case where the syntax is not the same for all modifiers, the command will appear more than once. After the modifiers required by a command have been satisfied, comments may be placed before the <CR> delimiter. However, you may not place comments on the argument line for **WRITE** or **PTEXT** commands as the SCGL will print the comment as if it were the desired output!

*# your comment goes here*

The # command must be followed by a space character in the body of a SCG Template.

| | |
|---|---|
| **BIT** register AND argument <cr> | register = register AND argument |
| NOR | register = register NOR argument |
| OR | register = register OR argument |
| SL | shift register left by argument bits |
| SR | shift register right by argument bits |
| XOR | register = register XOR argument |

**BIT** allows bitwise modification of the register by the value of the argument. The following truth tables apply:

| | AND--- | | NOT--- | | OR---- | | XOR--- | |
|---|---|---|---|---|---|---|---|---|
| register | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |

**CTRL** register argument — Special action based on argument

**CTRL** provides a means to set conditions within the SCGL and code generator. Typically the argument action occurs on or uses the register, however that is not always the case.

| argument | action |
|---|---|
| 1 | Set output ASCII msb to 1. Register unaffected. |
| 2 | Reset output ASCII msb to 0. |
| 3 | Equivalent to **DONE** 0 if register contains zero. |
| 4 | Equivalent to **DONE** 0 if register is non-zero. |
| 5 | Converts a byte value in the register into a pointer to the equivalent character in an ASCII string. If the character is non-printable, the conversion is not performed and R17 is set to 1. |
| 6 | Converts a byte value in the register into a pointer to the equivalent character in an ASCII string. If the character is non-printable, the string returned contains the non-printable character (default [.]). |
| 7 | Specifies the quote exception character (default ["]). |
| 8 | Specifies the non-printable character (default [.]). |

**DEC** register argument — register = register - argument

**DIV** register argument — register = register / argument

| | | |
|---|---|---|
| **IF** register | EQ argument | Do if register is equal to argument |
| | GE | greater than or equal to |
| | GT | greater than |
| | LE | less than or equal to |
| | LT | less than |

| | | |
|---|---|---|
| | NE | not equal to |
| **AND** | " " | |
| **OR** | " " | |
| **ELSE** | | Do if last **IF** comparison was false |
| **END** | | End of **IF** and **ELSE** |

The **IF** family provides a rich set of comparison options. An **IF** test can be extended by immediately following it with the optional **AND** or **OR** extensions. For example, The following syntax would be constructed to execute code based on `R40` containing `1` and `R43` and `R44` each containing `2` or `3`:

```
IF    R40 EQ 1
AND   R43 EQ 2
OR    R43 EQ 3
AND   R44 EQ 2
OR    R44 EQ 3
commands located here would be executed
END
```

If the above conditions were met, all commands following the conditional up to an **ELSE** or **END** would be executed. Otherwise, all lines to the **ELSE** or **END** would be ignored. This works on the rule of "skip if false". An **ELSE** merely inverts the conditional result and then applies the same rule.

| | |
|---|---|
| **INC** register argument | register = register + argument |
| **INDEX** register argument | register = register[argument] |
| **LOAD** register argument | register = argument |
| **LOOP** register | register = current template address |
| **CONT** register | current template address = register |
| **MOD** register argument | register = register % argument |
| **MUL** register argument | register = register * argument |

**NAME** register argument — register = ->item name string

If the argument is zero, the name of the current item being processed is returned. If the argument is a resource type value, the resulting name of that type/ID pair is returned. If the argument is a resource type value, the **NAME** statement must be preceded by a **READ** Rxx LONG.

Syntax: **NAME** Rxx 0

Syntax: **READ** Rxx LONG
**NAME** Rxx $xxxx

**PTEXT** register<cr> — register = -> (Pascal) text_string
*#your text here*

**PTEXT** allows strings to be preloaded into registers for more sophisticated insertion into output data. The text can not contain any formatting characters and must start with a delimiter character (# recommended).

| | | |
|---|---|---|
| **READ** register | BYTE | register = byte from resource item |
| | WORD | register = word from resource item |
| | LONG | register = long from resource item |
| | CSTR | register = -> text_string |
| | GSOS | register = -> text_string |
| | PSTR | register = -> text_string |
| | NSTR argument | register = -> text_string |

**READ** is used to return data from a resource. The BYTE, WORD and LONG modifiers return a 1, 2 or 4 byte value in the target register and move the resource pointer immediately after the data read.

The four string functions all work in a similar manner, returning a pointer to a C style text string. CSTR processes C strings having a NULL byte terminator. PSTR processes Pascal Strings using the value of the first byte to determine string length. GSOS handles Class 1 GS/OS string structures with a leading length word. Last, NSTR accepts a length argument to process a specified number of characters from the resource. It is possible for a string to contain a non-printable or control character. In

the case where such a character is encountered, the string is terminated after the last regular character and R16 is set to the count of characters read stops
**READ** is the only command that affects the pointer to the present position in the current resource. **READ** R1 BYTE increments the pointer by one byte, **READ** R1 WORD increments the pointer by two bytes, etc. When you read a string, the pointer is incremented by the length of the string plus any length or null bytes.

**TYPE** argument argument = resource type or NULL

**TYPE** marks the beginning of a template definition for source code output. **TYPE** must always have a matching **DONE** statement to terminate the definition. The argument is either the resource type the template handles (like $8001 for rIcon resources) or NULL ($0000) for a template that handles any undefined resource types. Every SCGL file must have a **TYPE 0/DONE 0** pair that handles undefined resource types. See *Example SCGL Template* below for more information.

**TITLE**

**TITLE** is used to place a title, or header, and/or ending sections in a source code or equate output file. **TITLE** has no arguments but makes use of R17 for additional information. R17 values and their meanings are:

| | |
|---|---|
| 1 | beginning of source code segment |
| 2 | ending segment |
| 3 | beginning of equates segment |
| 4 | ending of equates segment |

The **TITLE** command must have a matching **DONE 0**. The *Example SCGL Template* section below provides an example of **TITLE/DONE**.

**WRITE** <cr>
*your output goes here* sprintf format text

**WRITE** is used to write output to the source file. The output text can contain up to 255 characters. Formatting characters are limited to the following:

| | |
|---|---|
| \r | carriage return |

| | |
|---|---|
| \n | new line (converted to carriage return) |
| \t | tab |
| \xNN | insert hex value NN |

Output conversion specifiers:

| | |
|---|---|
| % | All output conversion specifiers must start with a % character. |
| 0 | If a value is shorter than the minimum field width, it is padded with zeros. |
| .<integer> | A period followed by an unsigned integer defines the minimum field width. Characters are right justified and padded with spaces. |
| l | All numeric output must have a lower case "l". |
| d | Sets output to decimal format. |
| X or x | Sets output to hexadecimal format. Upper or lower case "X" sets upper or lower case for the characters A through F. |
| s | Outputs a string stored in a register. |
| p | Outputs a **PTEXT** string. |

Consecutive conversion specifiers in a single **WRITE** statement output the values stored in registers 1 - 9 in ascending order.

Conversion specifier examples:

| | |
|---|---|
| %0.4lX | Output a hex value with a min. field width of four places, padded with zeros if necessary. The hex characters will be upper case. |
| %.2ld | Output a decimal value with a min. field width of two places, padded with spaces if necessary. |

## Example Template

```
#=================================================================
#  A SCGL template for generating generic assembly language source
#  code output.  It provides examples on using the TITLE, TYPE 0
#  and TYPE $8014 commands.
#  Copyright 1989, 1990
#  Simple Software Systems International, Inc.
#  All Rights Reserved.
#=================================================================
TITLE
IF R17 EQ 1             put in header for source generation
      WRITE
RGLOBAL DATA\n
      WRITE
*****************************************************************\n
      WRITE
* Genesys created Assembly Language data structures\n
      WRITE
* Copyright 1898, 1990 Simple Software Systems International, Inc.\n
      WRITE
*****************************************************************\n
ELSE
IF R17 EQ 2             put in end
      WRITE
\n    END;  RGLOBAL data\n
ELSE
IF R17 EQ 3             put in header for equate generation
      WRITE
*****************************************************************\n
      WRITE
* Genesys created Assembly Language equates\n
      WRITE
* Copyright 1989, 1990 Simple Software Systems International, Inc.\n
      WRITE
*****************************************************************\n
END
```

```
DONE 0
#
# Handler for all otherwise unsupported resource types
#
TYPE 0
IF R15 EQ 1              put in a header if this is the first one
      LOAD R1 R13        get resource type
      WRITE
\n*\n* Resource type $%0.4lX\n*\n
END
NAME R1 0                set the label ptr
IF R12 EQ 0              if the resource is NULL
      WRITE
\n* Item %s is an empty resource\n
      DONE 0             done with this one
END
WRITE
%s ANOP\n
LOAD R20 R12             get size of resource for a counter
#
LOOP R30                 handle each line of output
LOAD R21 16              with 16 bytes per line
WRITE
      DC    H'
#
LOOP R31                 handle the line of output
READ R1 BYTE             get a byte from the resource
DEC R20 1                dec chars/counter
IF R20 EQ 0
      WRITE
%0.2lX'\n\n
      DONE 0
END
DEC R21 1                dec line counter
IF R21 GT 0              same line
      WRITE
%0.2lX,
```

```
      CONT R31          do next byte in this line
#
ELSE
      WRITE
%0.2lX'\n
      CONT R30          go start another line
#
END                     done
DONE 0
#
# Handler called by Generate equates... dispatch
#
TYPE  $8014
IF R15 EQ 1             put in header if this is the first one
      WRITE
\n
END
NAME R1 0               get resource label
LOAD R2 R14             get resource ID
WRITE
%s GEQU $%lX\n
DONE 0
#
# End of SCGL Example
#
```

# Appendix B: Writing Your Own Genesys Editors

## Special Note

The format for writing your own Genesys Editor has not been released yet. Genesys is still in a state of change, and any Editor writing that you would do currently would most likely not work with a later version of Genesys.

As soon as the format for Genesys Editors has been solidified, you will be notified and sent this Appendix in its entirety.

# Appendix C: Updates and Support

## Release Notes

Certain features are omitted or added to any software product at the last minute due to time constraints, system software bugs, and other factors beyond the programmer's control. To make sure you are aware of these situations, the Help System in Genesys contains a section titled **Release Notes**. You can access the Help System by selecting **Help** from the  menu in the Genesys menu bar.

If you are already using Genesys and have upgraded to a new version, the release notes also contain notes on the major features that have been added to Genesys since the last released version.

## Reporting Problems

If you have a question or problem to report, you may contact us using any of the following methods:

| | |
|---|---|
| Phone: | **(404) 928-4388** |
| Mail: | **SSSi, Inc.**<br>4612 North Landing Drive NE<br>Marietta, Georgia 30066 U.S.A. |
| GEnie: | Product support board **SSSi.** |
| America Online: | Keyword **SSSi**, or address electronic mail to **SSSi.** |
| MCI Mail: | Address electronic mail to **SIMPLE.** |
| AppleLink: | Address electronic mail to **D6862.** |

We realize that Genesys, resources, and programming in general can be a lonely process if you have no one to turn to, and we are glad to help you with your efforts in any way we can. Make sure you carefully prepare your questions and problems beforehand. It would help us greatly if you can tell us how to recreate any problems you are having, although we realize this is not always possible.

## Registration

The front of this manual contains a registration sheet that needs to be completely filled out and sent in to SSSi. We realize that sending in registrations for software can be thoroughly boring, but it is VERY IMPORTANT that you do so. We cannot offer you product support if we do not have your registration, and more importantly, we will not be able to notify you of additions, updates, or new products.

## Acknowledgements



### Credits


Genesys was designed and developed by Michael Hackney, Kevin Seidule, and Marc Wolfgram.

We would like to thank all of the beta testers who helped test, test, and re-test this product. Your encouragement, ideas, support and suggestions are invaluable.

Special thanks to Paul Doty, Allan Reichert, Paul Elseth, and all the folks at Apple.

And *thank you* for purchasing Genesys.

## The Fine Print

# Glossary of Terms

**:** Refers the the multi-colored apple menu where new desk accessories are available.

**:** Refers to the open-apple key on the keyboard.

**320 Mode:** An Apple IIGS screen size of 320 pixels wide by 200 pixels tall.

**640 Mode:** An Apple IIGS screen size of 640 pixels wide by 200 pixels tall.

**Apple IIGS:** An incredibly advanced and fantastic computer of the Apple II family. Kind of like a Macintosh, only better.

**APDA:** The Apple Programmers and Developers Association. They offer Apple and third party documentation, development systems and utilities. There is an annual membership fee, but the services they provide are invaluable. You can contact them at (800) 282-2732.

**ASCII:** An acronym for American Standard Code for Information Interchange. It is the standard for representing alphanumeric characters on computers.

**attributes:** Determines how memory blocks are allocated and maintained.

**BASIC:** Acronym for Beginners All-purpose Symbolic Intruction Code.

**bit image:** A collection of bits in memory that have a rectilinear representation, such as an icon.

**boot:** Start up. When you turn on your computer you boot.

**boundry rectangle:** The rectangle that defines an item's boundry, such as a control rectangle.

**bug:** An error in a program that causes it not to work as intended. Programmers hate bugs.

**C:** High-level programming language.

**classic desk accessory (CDA):** Desk accessories that work in the "text" environment and are available to both Desktop and ProDOS 8 applications available by pressing **-control-esc**.

**Clipboard:** The holding area for material the user last cut or copied which can later be pasted.

**close box:** Icon in a window's title bar used to close a window.

**compiler:** Used to convert a language written in a high-level language to machine instructions.

**computer:** A device used to run Genesys.

**configuration:** The total components that make up a computer system.

**content:** The area below a window's title bar where a program displays information to the user.

**context sensitive:** A system that takes appropriate action based on what you are doing. Used here it refers to the Genesys Help System which is context sensitive.

**control:** A object in a window for controlling an action such as a button.

**Control Panel:** A desk accessory used to change system parameters such as the system time.

**control panel device (CDEV):** Located in CDEVS folder and accessed from the Control Panel.

**coordinates:** X and Y locations on the screen.

**copy:** To duplicate something by selecting it, such as copying something to the clipboard.

**copy protect:** A fragile method used by paranoid publishers to make a disk uncopyable.

**crash:** An unexpected cease of operation. One of many methods used by a program to make a publisher and/or programmer paranoid.

**cursor:** A graphical pointer, such as the arrow cursor.

**cut:** To remove a selected object and place it on the clipboard.

**data fork:** One of two parts of a file that contain the program. This data is not available for the user to modify. Before System 5.0, all Apple files were simply a data fork.

**debug:** To remove bugs from a program. See bug.

**default:** A preset value or response.

**default button:** Button that responds by pressing the **return** key. Usually double outlined.

**dialog window:** A modal window used to interact with the user.

**double-click:** Act of quickly pressing and releasing the mouse twice on an object.

**drag:** To click and hold the mouse on an object and move the mouse, which should move the object.

**easter egg:** A surprise, usually an icon, picture, or really neat feature hidden in a program. Genesys has none of these. Ok it does, but we can't tell you what it is.

**equates:** An equate is a label that references an object that is normally referenced by an identification number. For instance, if an icon has an ID of 1, you can form an equate like:

```
in Assembly:  MyRedIcon   GEQU        1
in C:        #define     MyRedIcon    1L;
in Pascal:    CONST       MyRedIcon = 1;
```

**fastport:** An improvement to QuickDraw introduced with System Software 5.0.

**fat bits:** A type of graphics editing that displays magnified, or fat, pixels.

**fork:** One of two parts of a GS/OS file. Prior to System 5.0 files had only one fork, the data fork. Under GS/OS a file can also have a resource fork to store program data.

**hexadecimal:** The base 16 numbering system quite popular with programmers.

**INIT file:** A GS/OS file that is located in the `System.Setup` folder of the startup disk.

**keyboard equivalent:** A keyboard command that activates a control.

**keyboard translation:** Information that the operating system uses to determine how to display or handle characters typed at the keyboard. This information is stored as a table and can be selected with the Control Panel NDA.

**marching ants:** Red dashed lines that surround an item when it's selected. These small dashes constantly "march" around the item, giving the illusion of ants.

**modal:** A type of dialog window that requires a response from the user before proceeding.

**new desk accessory (NDA):** A "mini-application" available from the  menu. An example is the Control Panel supplied with System 5.0. This desk accessory allows you to change system settings like the screen border color and mouse click speed.

**pixel:** A "picture element": the small rectangular or circular colored dots that make up a graphic.
**pointer:** A reference, or address, to a memory block.
**programmer:** A geeky looking animal that cannot survive without pizza and caffine.
**public domain:** A program that has been donated to the public by its author. You do not pay for the right to use this type of program. However, public domain programs can be copyrighted and cannot be sold or used for commercial applications. Check with the author of the program for details.
**resource:** Data stored in the resource fork of a file. Many types of resources are defined by Apple to store commonly used information like menus, windows and controls.
**resname:** A type of resource that allows other resources to be named.
**resource chooser window:** A Genesys window that displays a list of resource types and IDs.
**resource fork:** One of two parts of a file that contains resources. This data is stored by resource type and resource ID and can be located by the Resource Manager Toolset.
**resource ID:** Unique identification numbers given to resources within the same type.
**resource type:** Information stored in a file's resource fork is "contained" in groups called types.
**SCG Template:** A Genesys Source Code Generation Template.
**shadowing:** An improvement to QuickDraw // introduced with System software 5.0.
**shareware:** An inexpensive program that you can try out and share before purchasing it. Many shareware programs can be found on online services. *Please support the shareware authors!*
**shell:** An application that oversees other components of a programming environment. A familiar example is the Apple Programmer's Workshop (APW) command shell or the Genesys Shell.
**support:** The service received when one fills out and returns the registration card (hint).
**system startup disk:** The floppy disk or hard disk that contains the GS/OS operating system and important files, like the Tools, Fonts and NDAs. You use a startup disk to start your computer.
**thermometer bars:** Horizontal bars that an application displays during a time consuming process.
**toolset:** The built-in software routines or tools that programmers can use to create powerful and easy to use programs.
**WYSIWYG:** An acronym for "What You See Is What You Get".