GFA Companion

A dedicated RCS package designed to create Dialog boxes, Error boxes, Sliders and more to be merged into your GFA BASIC Interpreter programs.

for the Atari ST

MichTron

The GFA BASIC Companion

Dialog Box Source Code Generator
and
Language Tutorial
with
Source Code Libraries

User Guide

Written by John B. Holder,
Marathon Computer Press

Distributed By MICHTRON, Inc.
576 South Telegraph
☎: (313) 334-5700
BBS: (313) 332-5452

UPGRADES AND REVISIONS: If you return your information card, we will notify you if upgrades to ***THE GFA BASIC COMPANION*** become available. For minor upgrades and fixes, return the original disks to us with $5.00. For major revisions, the upgrade fee is typically 15-20% of the original suggested retail price.

FEEDBACK: Customer comments are VERY important to us. We think that the use, warranty and upgrade policies outlined above are among the fairest around. Please let us know how you feel about them.

Many of the program and documentation modifications we make result from customer suggestions. Please tell us how you feel about ***THE GFA BASIC COMPANION*** - your ideas could make the next version better for all of us.

The GFA BASIC Companion

User Guide

Published by MICHTRON, Inc.
576 South Telegraph
Pontiac, Michigan 48053
☎: (313)334-5700
BBS: (313)332-5452

Manual Written by John B. Holder
Book Design by Thomas L. Logan

ISBN 0-923213-78-3

Table of Contents

List of Diagrams . . . . *vi*
Author's Comments . . . . *ix*

Introduction 1

Getting Started 2
Configurations 3
About Using Desk Accessories 4
Making a Working Backup 5
Program File Shipping List 6

Part I: The Source Code 7

Technical aspects of the Generator 9
Running The Dialog Box Source Code Generator 14
The Desk Pull Down Menu 15
The Credits Box Option 16
The Error Box Option 23
The Function Key Box Option 25
The Help Text Box Option 28
The Custom Designed Option 32
Long Style Information Box Option 42
Radio Button Box Option 44
Text Only Box Option 51
Wide Style Information Box Option 55

Part 2: The Tutorial Driver 57

General Information 58
The "Desk" Section 60
The Tutor Option 61
The "Quit" Section 64
Printing the Tutorial Files 65
Additional Tutorial files and Support Material 66

Part 3: Modifying the Output Source Code 67

Developing The Generator 68
Moving Objects With The 4 Move Points 70
Changing Fill Patterns 73
A Word About Fonts 74
Product Development And Plans 75
Joining A National Billboard Service 76
Author's Closing Remarks 78

Index 79

List of Diagrams

Fig. 1: GFA Companion 2
Fig. 2: The Desk Menu 4
Fig. 3: Dialogs Menu 8
Fig. 4: Dialog Box Sample 10
Fig. 5: COMPAN.PRG 13
Fig. 6: Select Credit Box Option 16
Fig. 7: Press <RETURN> after each text line 17
Fig. 8: Selecting Font Style 18
Fig. 9: Type up to 12 lines of text 18
Fig. 10: Each Line can have a different Style 19
Fig. 11: Skip lines by clicking <RETURN> twice 20
Fig. 12: ...the 2nd time at the Font Style Box 20
Fig. 13: Type *Q or *q when finished 22
Fig. 14: Error Box 23
Fig. 15: Work Bench for Function Key Box 25
Fig. 16: Function Key Box Example 27
Fig. 17: Help Text Box example 28
Fig. 18: Up to 500 lines of Text 29
Fig. 19: Select Custom Designed 32
Fig. 20: Resolution Overlay Selection 34
Fig. 21: Custom Design Work Bench 34
Fig. 22: Drawing the box outline 35
Fig. 23: Drawing inside selectable squares 37
Fig. 24: A Three selection Custom Box 39
Fig. 25: Long Information Box Workbench 42
Fig. 26: You can ask for Help before starting 44
Fig. 27: ... And end up here 45
Fig. 28: For this Example choose 3 Buttons 46
Fig. 29: The Box Outline Screen 46
Fig. 30: Type in the first Button Box Name 47
Fig. 31: Press <RETURN> after each name 47
Fig. 32: Choose how the boxes are to be labeled 47
Fig. 33: Select Save 48
Fig. 34: Enter a Name for the new box 48

Fig. 35: You can have up to seven buttons in a box 50
Fig. 36: Selecting Text box size 51
Fig. 37: Choosing *Rag* or *Cent* 52
Fig. 38: Enter *Q or *q on a new line to exit 53
Fig. 39: Creating Wide Style Boxes 55
Fig. 40: The Tutorial Program 58
Fig. 41: Select *Read a Chapter* 61
Fig. 42: ...Then select a chapter 62
Fig. 43: A page from the Tutorial 63
Fig. 44: To Quit either program 64
Fig. 45: ...Click on Good Bye 64

AUTHOR'S COMMENTS

This program was designed to take the burden out of Dialog Box construction and usage for owners of GFA Basic. As it turns out, the program turned into a capable RCS that gives you many of the functions afforded by a classical RCS used by C programmers. When coupled with the source code library; this package will assist both the beginning GFA Basic programmer and the expert as well. Try to look upon the program as a springboard designed to catapult you headlong into the world of programming in GFA Basic. The concepts presented in the *program generated source code* and those that are contained in the source code libraries, are meant to show you some of the more subtle yet powerful capabilities of this wonderful language.

The tutorials bundled with their own custom presentation driver are targeted at the beginner to intermediate level GFA Basic programmer. Some Basic concepts are presented as well as some that will take you to a higher programming level. If you are new to GFA Basic, it will be to your advantage to utilize the Tutorials in conjunction with your Language Manual to shorten the learning curve associated with learning any new computer language. The manner of presentation is meant to be informative, yet it looks at programming in GFA Basic from a "middle of the road" perspective. Have fun with the language, and I truly hope that you have many entertaining hours with this product.

John B. Holder

Senior Software Engineer
Marathon Computer Press

Assistant Sysop on
GEnie's MICHTRON Roundtable

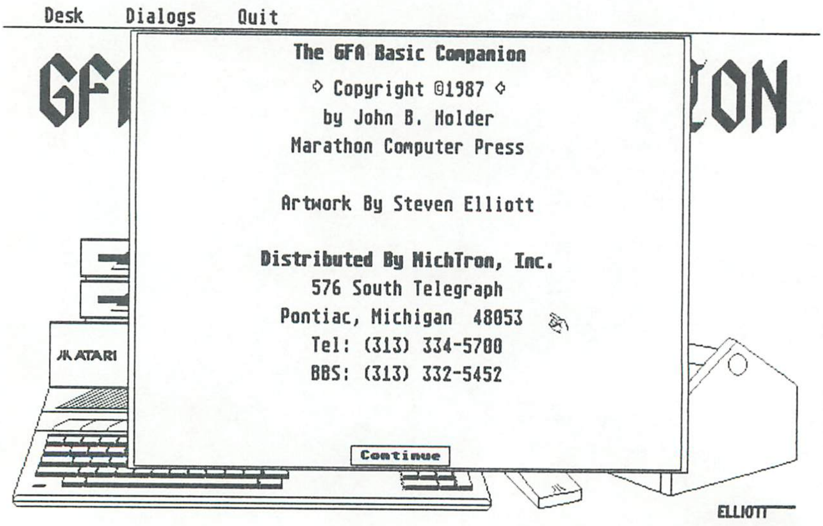
Introduction

Getting Started

This manual assumes that you know how to turn on your computer, format blank disks, copy files from the desktop, and delete files. If you are a first time user, or are unsure of how to accomplish any of the above tasks, please refer to the set of owner's manuals that were included with your computer.

In addition, it would be a good idea to browse through this reference prior to attempting to utilize the Dialog Box Source Code Generator. As with any program that creates and saves material to a floppy disk, you should always have a spare diskette handy in the event that you run out of room on a storage disk.

Fig. 1: GFA Companion



The GFA Basic Companion
◇ Copyright ©1987 ◇
by John B. Holder
Marathon Computer Press

Artwork By Steven Elliott

Distributed By MichTron, Inc.
576 South Telegraph
Pontiac, Michigan 48053
Tel: (313) 334-5700
BBS: (313) 332-5452

Configurations

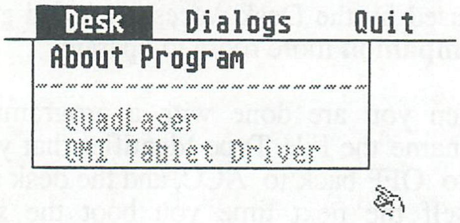
The Dialog Box Source Code generator is a LARGE program and uses a large part of the RAM (Random Access Memory) on your computer. If you are a 520 ST owner, you may want to turn off your desk accessories to give the program enough room to function properly. This may be accomplished by renaming your accessories from XXXX.ACC to XXXX.OFF and then rebooting your system. (The XXXX symbolizes your actual accessory file name. The only part you have to rename is the File Type Identifier ".ACC"). By doing this you will free the memory used by the Desk Accessories and give **The GFA Basic Companion** more room to operate.

When you are done with a programming session, simply rename the File Type Identifier that you previously changed to .OFF back to .ACC, and the desk accessory will install itself the next time you boot the system. This procedure may not be necessary with all 520 STs, but in the event that you have a lot of memory tied up with accessories, this will help. The lucky owners of 1040 STs will not normally have to worry about this unless they have unusually large RAM Disks installed.

About Using Desk Accessories

Neither COMPAN.PRG or TUTOR.PRG will allow you to use Desk Accessories from the main GEM pull down menu. This is because accessories are not needed or desired while either program is running. You will notice that all of the present (if any) desk accessories have been disabled. (They will appear in ghost font within the pull down menu bar.)

Fig. 2: The Desk Menu



Do not despair, if you have an extra formatted disk handy you should never need a desk accessory. This is planned for a number of reasons. The most prominent is to reduce the amount of code in the main program devoted to screen management thereby leaving more for the code generation schemes, and also to reduce the amount of GEM interaction with the program in an attempt to produce a faster running and less crash prone product. Once you have used the programs a few times, you'll see that Desk accessories are unnecessary.

Making a Working Backup

You are strongly urged to make a backup copy of the distribution diskette(s) that you have received in this package. The software is not copy protected so you may duplicate the diskettes in the normal way. If you have questions about how to make a disk backup, refer to your owner's manual that came with your computer.

A Note About Unprotected Software: You are being provided with unprotected distribution diskettes. That implies a lot of trust by MICHTRON, Inc. that you will honor our copyright. People have worked long and hard on this product and their livelihood is determined on how well this product sells. Please treat this software as you would any other copyrighted work, and remember that honoring software copyrights will ensure that developers will continue to provide you with software to get the most out of your computing investment. If you have any doubt about reproducing any portion of this package for other than personal use, please contact MICHTRON, Inc. for details. For more information on Unauthorized Software Duplication you may contact:

Software Publishers Association
Suite 1200 1111 19th Street, N.W.
Washington, D.C. 20036

Program File Shipping List

To check your diskette(s) for a listing of all files included with the release of the software, read or print out the README.1ST file contained in the main directory on your diskette(s). In addition to a shipping list, the README.1ST file will contain updates to this documentation and any other pertinent information concerning this program package. It would be a good idea to print out that file now if you haven't done so already.

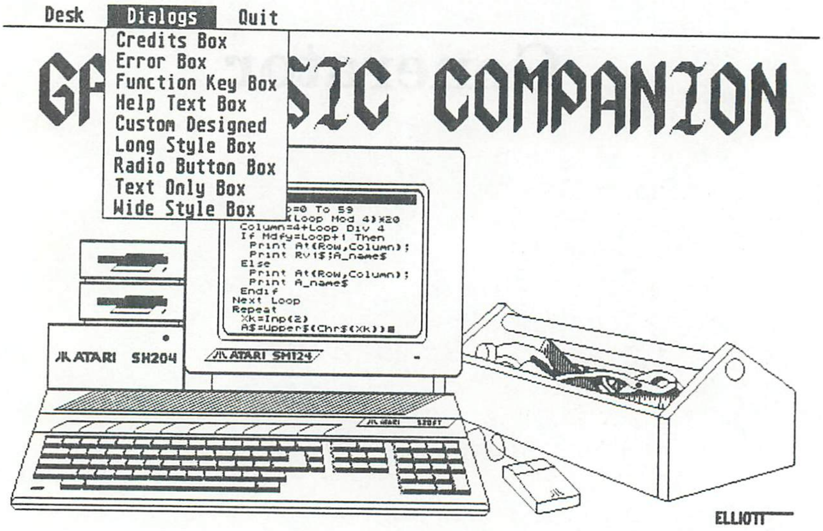
Part I

The Source Code Generator

Note:

This section of the owner's manual is dedicated to explaining the Dialog Box Source Code Generator and all of the options available from the main pull down menu. A section has been assigned to each option, and a special section on modifying the output source code is included at the end of Part 1.

Fig. 3: Dialogs Menu



```
          p=0 To 59
          (Loop Mod 4)*20
  Column=4+Loop Div 4
  If Mdfy=Loop+1 Then
   Print At(Row,Column);
   Print Rv1$;A_name$
  Else
   Print At(Row,Column);
   Print A_name$
  Endif
Next Loop
Repeat
  Xk=Inp(2)
  A$=Upper$(Chr$(Xk))
```

✢

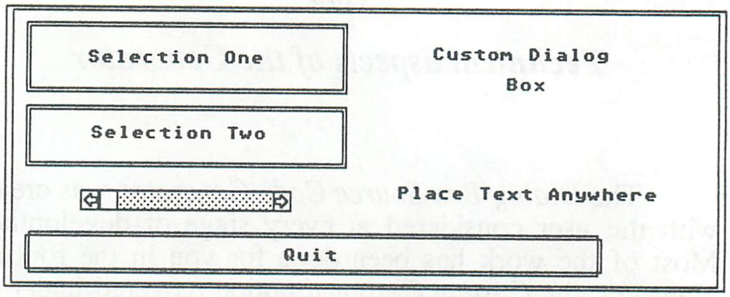
1.00

Technical aspects of the Generator

The *Dialog Box Source Code Generator* was created with the user considered at every stage of development. Most of the work has been done for you in the routines. However, the *Custom Designed* option (covered later) will give you total control over the creation of Dialog Boxes. By including options for quick Dialog Boxes and options for the "power user", this application should meet the needs of most GFA Basic Programmers. Many of the most common implementations of Dialog Boxes have been included in the options. Everything from an Error Box to Wide Style Information Boxes have been included.

The benefits of the Generator are evident with the first use. Any applications generator reduces the amount of work that a programmer must do to bring an application to completion. By this we mean that instead of writing *Custom Dialog Box* routines for each program that you write, you can rely on **The GFA BASIC Companion** to successfully write those portions of your application for you. Anyone that has worked with GEM for very long will quickly learn to appreciate this luxury. And to top off the features of the Generator portion of this package, the output code is in 100% GFA Basic compatible .LST file format, ready to merge into your own programs. With this power at your fingertips, you can now create custom boxes to replace *Alert Boxes*, and easy to utilize *Radio Button Boxes* and the like. The possibilities are only limited by the capabilities of the GFA Basic Language and your imagination.

Fig. 4: Dialog Box Sample



It is hoped that this program package will change the way many programmers think about programming in GFA Basic on the ST. You will find that by using several of the resident commands in GFA Basic you can effectively create a versatile user interface that will mimic routines available from the GEM AES, but in a much easier to learn and change format. Making a change to a routine is as simple as loading a .BAS or .LST file into the GFA Basic editor and changing the portions desired in a real time mode and testing your changes immediately.

This is not possible with a Standard RCS like the one in the ATARI ST Developer's Kit, or others available on the open market. With these development tools you must enter the RCS editor, define the type of Dialog Box you want to design, assign objects to it, while all the time keeping track of the Tree Table, and then save the file. The classical RCS editor will save a .H (header file) that contains a series of equates relating to the objects and the index numbers, and a .DFN file (information to be used by the RCS at a later date if you desire to change or update the Dialog Box), and a compiled Resource file. If you desire to use the .RSC file, many specialized procedures must then be implemented into your program to load, manipulate, and utilize this .RSC file. What all of this means to the average GFA Basic programmer is a *WHOLE LOT OF WORK!*

With the options provided by the Generator, only the adventurous will probably ever desire to use a classical Resource Construction Set and suffer from the Migraine Headaches associated with using a .RSC file. Each generated routine that you create will contain comments concerning the usage and limitations of the box that you have created. Please refer to the .LST file that is created with each routine for full details prior to merging them into your own applications. Some of the routines are suitable for all resolutions available on the ST, while others are only suitable for Medium or High Resolutions. It will be up to you to decide the applicability of each routine as it pertains to your own application.

Also included, in the form of comments, are pointers on which variables may be updated to move the box around on the screen while retaining the ability to read & recover the user's interaction with the box. Typically these variables are:

Lx - The far left X coordinate of the Box
Rx - The far right X coordinate of the Box
Ty - The top limit Y coordinate of the Box
By - The bottom limit Y coordinate of the Box

If sliders are used:

Slx - The far left X coordinate of the Slider
Srx - The far right X coordinate of the Slider
Sty - The top limit Y coordinate of the Slider
Sby - The bottom limit Y coordinate of the Slider

This may not be true for each type of box generated by the program. Refer to the .LST file for full information in every case. If additional options are made available after the publication of this manual, they will be annotated in the

README.1ST file contained on the Master Distribution Diskette(s).

The routines for all but the *Custom Designed Box* option are pre-scripted within the Generator. They handle all of the Box drawing chores, text scrolling if available, and screen restoring functions. A good point to remember is that if you use any of the Dialog Box routines provided by **The GFA BASIC Companion**, you must allow 32000 (32K) bytes of storage for a *temporary use* video screen. This temporary screen is grabbed by the routines before the actual drawing routines take place, and then replaced before returning control of the screen back to the calling routine.

Following the screen replacement, the TEMPUSE$ variable is cleared automatically for you (provided you do not remove the line of code in the output file that accomplishes this function). If you are unsure if you will have enough memory for this function during the development stages of your application, just include the following line at the top of your program to ensure that the 32000 bytes of storage will be allocated for your program:

```
SGET TEMPUSE$
```

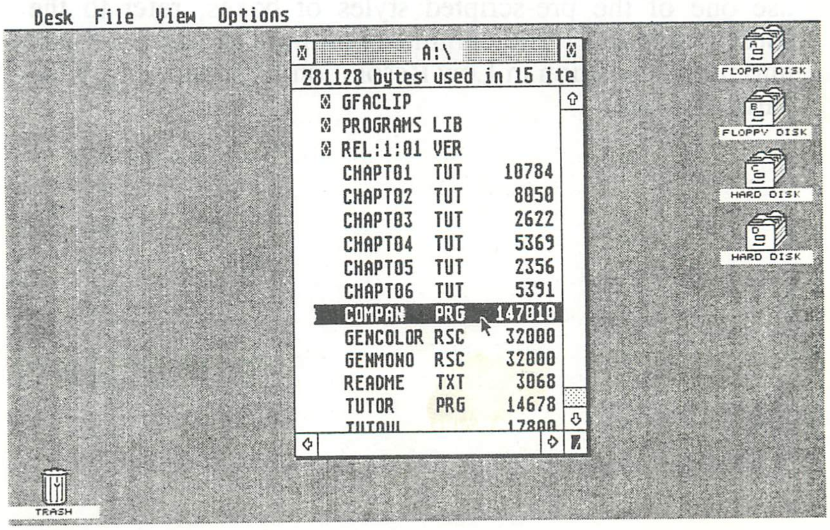
That is all there is to it. Now you will be assured of having the temporary screen buffer for all of the Dialog Box routines supplied by the Generator.

The *Custom Designed Dialog Box* development routine uses special graphics drawing routines to enable you to interactively create a dialog box on the screen for any resolution possible on the ST. In order to create a box for any resolution you will want to turn on the Resolution Overlay Lines before starting the routine. If you are familiar with the popular D.E.G.A.S. drawing program by Batteries Included, then you should feel right at home with this part of the program when drawing your boxes. A prime consideration when using this routine is that you have complete control over the placement of Boxes on the screen. It is a true WYSIWYG (What you see is what you get) type of editor. If you draw overlapping boxes, then the Generator will produce source code for that effect. A bit of experimentation will show you the results of this capability.

The Program *WILL NOT* alert you about overlapping boxes, shadow text out of the appropriate boxes, boxes that will interfere with menu bars, etc. It is up to YOU to decide just where you want your objects to appear on the screen. By giving you this option, you have the entire screen to work with. To create Dialog Boxes that will not interfere with a menu bar you will have to follow the directions contained in the part of this reference manual pertaining to *Custom Dialog Boxes*.

Well, since all of the preliminary pointers and warnings have been issued; Let's Get Started!

Fig.5: COMPAN.PRG



```
A:\
281128 bytes used in 15 ite
  GFACLIP
  PROGRAMS LIB
  REL:1:01 VER
  CHAPT01  TUT     10784
  CHAPT02  TUT      8050
  CHAPT03  TUT      2622
  CHAPT04  TUT      5369
  CHAPT05  TUT      2356
  CHAPT06  TUT      5391
  COMPAN   PRG    147010
  GENCOLOR RSC     32000
  GENMONO  RSC     32000
  README   TXT      3068
  TUTOR    PRG     14678
```

1.01
Running The Dialog Box Source Code Generator

To run the Generator double click on the Program Icon or Text Line for the program listed as COMPAN.PRG. Following this action, the program will be loaded into memory with all of the overlays that it requires and the opening screen will appear. You MUST be in either High or Medium Resolution for the program to run. If you attempt to run the program in Low Resolution, an Alert Box will appear and ask you to please place the machine into Medium Resolution before continuing.

Do not worry if you plan to design Dialog Boxes for Low Resolution, with Custom Designed Box routine's resolution overlays (discussed later) you will be able to effectively accomplish this task. Or if you are planning to use one of the pre-scripted styles of boxes, refer to the source code .LST file comments on how to move the box into proper position in Low Resolution (if supported by the particular box).

1.02

The Desk Pull Down Menu

By moving the mouse into the Desk area of the menu bar, the *About Program* option will appear along with a list of Desk Accessories that are currently loaded into your machine's memory. As mentioned earlier, all desk accessories have been disabled by the program for pertinent reasons.

By clicking on the About Program option in the menu, you will be presented with a Dialog Box (Created with **The GFA BASIC Companion** *Credits Box* Option) that will tell you the details of Copyright and Distribution for **The GFA BASIC Companion.**

You will notice that all desk accessories are represented in Ghost Text mode. This effectively disables the selection of all accessories during the execution of the Generator Program. They will be restored to full functionality with the return to the GEM desktop.

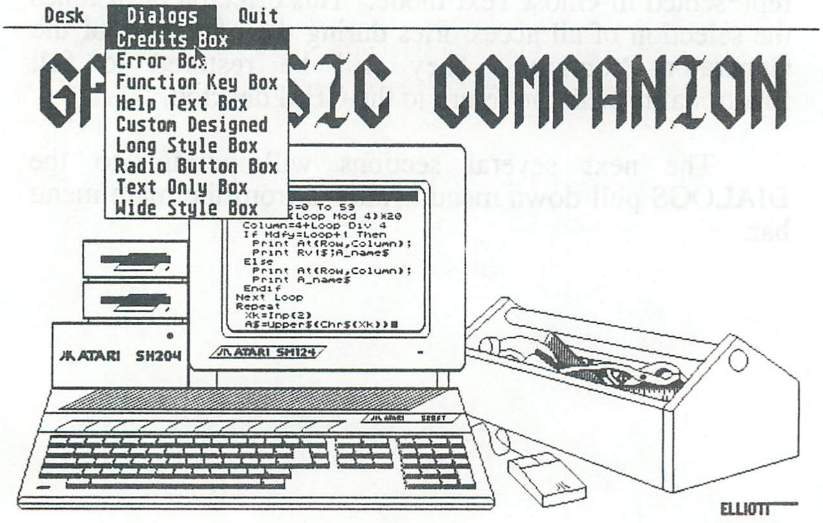
The next several sections will pertain to the DIALOGS pull down menu available from the main menu bar.

1.03

The Credits Box Option

If you click on this option from the pull down menu bar you are presented with an alert box that asks you if you really want to create a *Credits Box*. Once you click on yes, the box outline appears on the screen and prompts you to enter a title. Simply type in the title on the prompt line. As with all of the text entry prompt lines offered in the Generator, you may use the full keyboard combinations available (Control + key or Alternate + key to create the full range of screen printable characters).

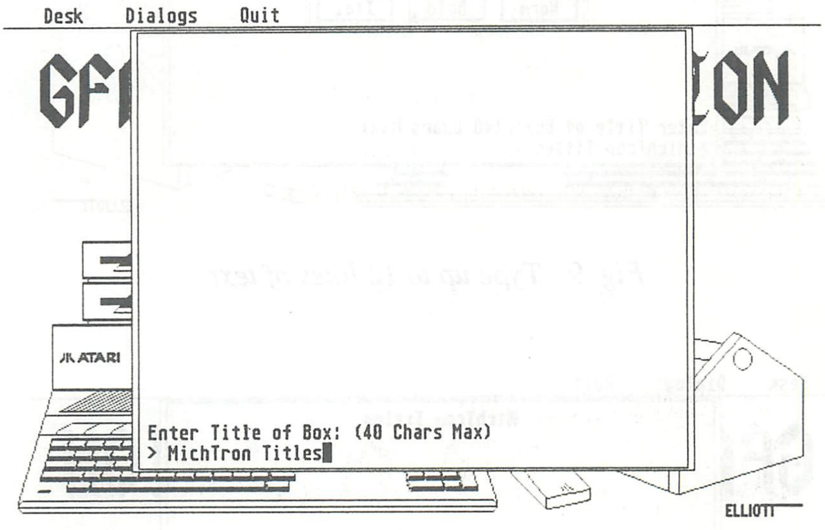
Fig. 6: Select Credit Box *Option*



```
    =0 To 59
    (Loop Mod 4)*20
  Column=4+Loop Div 4
  If Mdfy=Loop+1 Then
    Print At(Row,Column);
    Print Rv1$;A_name$
  Else
    Print At(Row,Column);
    Print A_name$
  Endif
Next Loop
Repeat
  Xk=Inp(2)
  A$=Upper$(Chr$(Xk))
```

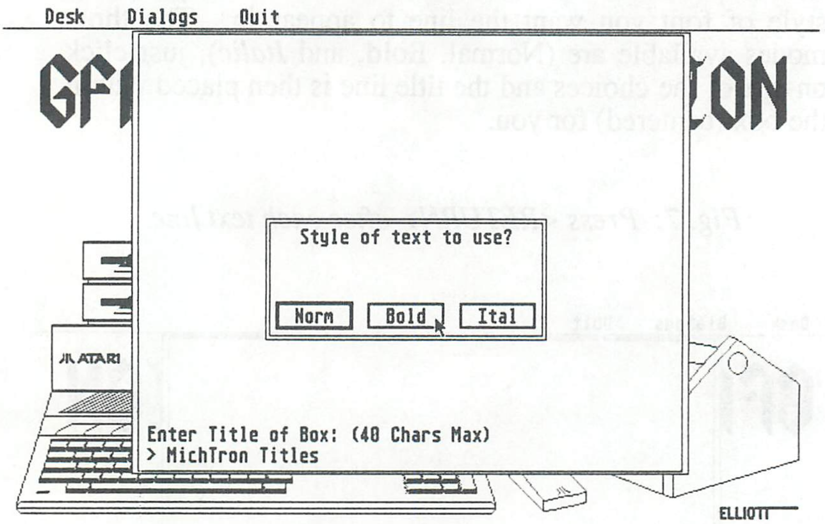
Following the text entry, press <RETURN> and you will be presented with another alert box asking you what style of font you want the line to appear in. The three modes available are (Normal, **Bold**, and *Italic*), just click on one of the choices and the title line is then placed within the box (centered) for you.

Fig. 7: Press <RETURN> after each text line



```
Enter Title of Box: (40 Chars Max)
> MichTron Titles
```

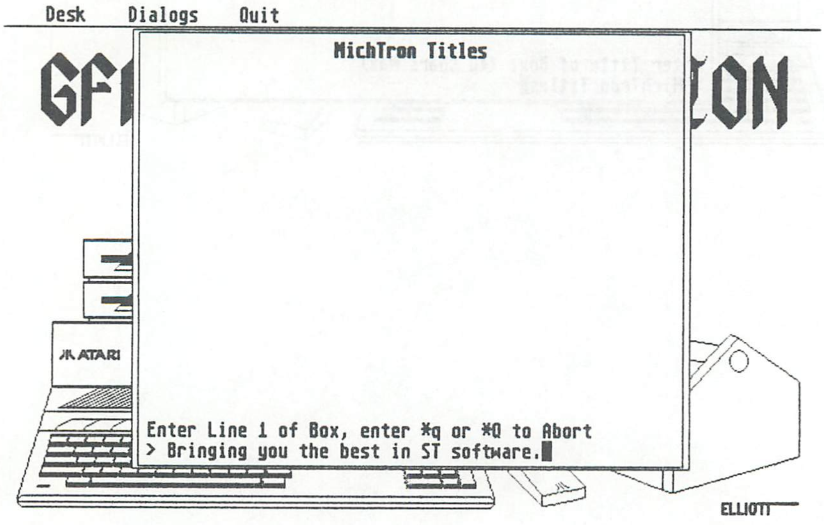
Fig. 8: Selecting Font Style



Style of text to use?

Enter Title of Box: (40 Chars Max)
> MichTron Titles

Fig. 9: Type up to 12 lines of text

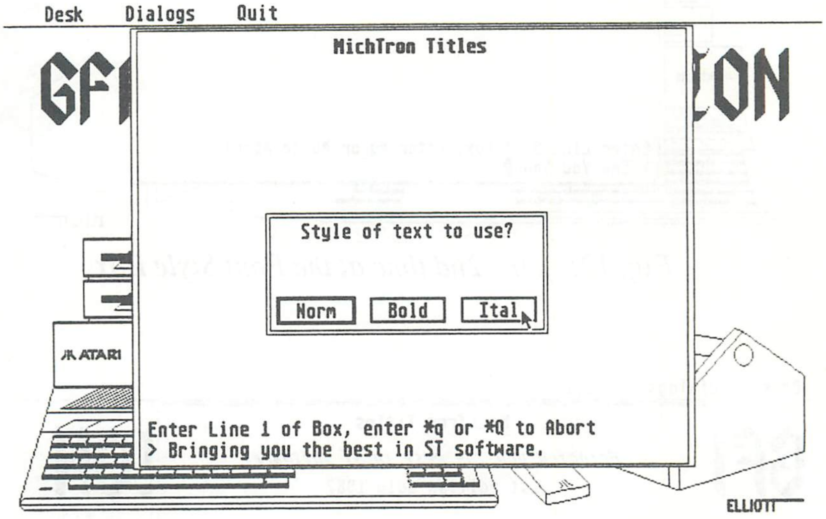


MichTron Titles

Enter Line 1 of Box, enter *q or *Q to Abort
> Bringing you the best in ST software.

Next, 12 additional lines are entered in much the same manner. If you do not desire to enter a line in the next position, just press the return key to insert a blank line and press <RETURN> again when the Font Style alert box appears.

Fig. 10: Each Line can have a different Style

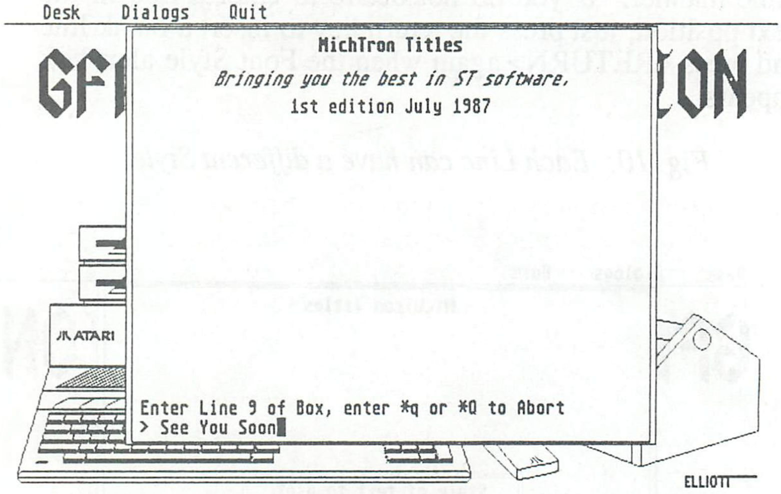


Style of text to use?

Norm Bold Ital

Enter Line 1 of Box, enter *q or *Q to Abort
> Bringing you the best in ST software.

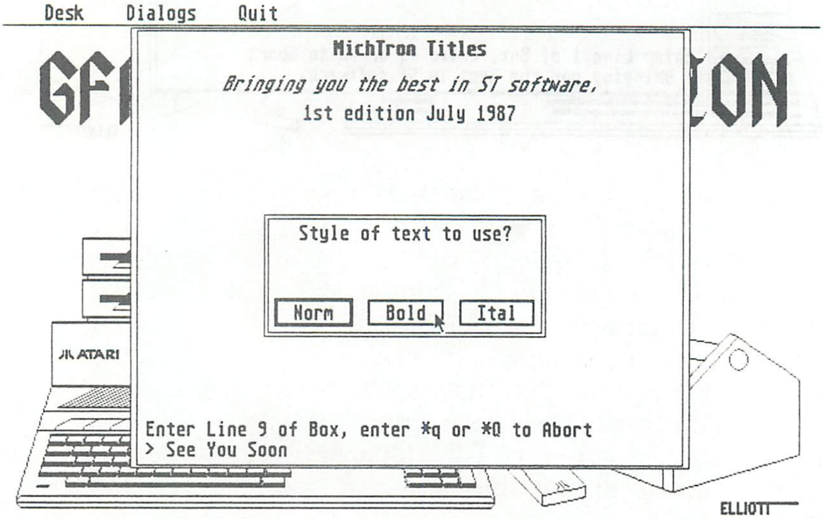
Fig. 11: Skip lines by clicking <RETURN> twice...



MichTron Titles
Bringing you the best in ST software,
1st edition July 1987

Enter Line 9 of Box, enter *q or *Q to Abort
> See You Soon

Fig. 12: ...the 2nd time at the Font Style Box



MichTron Titles
Bringing you the best in ST software,
1st edition July 1987

Style of text to use?

Enter Line 9 of Box, enter *q or *Q to Abort
> See You Soon

Following the Title and Text Line entry, you are given the chance to either save the box you have created, or to abort the function and return to the main menu. If Save is selected, the box you designed on the screen is saved in GFA Basic source code in the form of a .LST file, ready to be merged into your own program.

Modifications to the source code can be made easily, so if you make a minor mistake during the creation of the box there is no need to abort the entire box. Just fine tune the source code file if changes are desired. This also includes changing the font style. There is a special section devoted to tweening (modifying) source code at the end of this part of the manual. Since you are not given the chance to abort the *Credits Box* option once you have started entering text, you will have to at least press the return key a number of times before reaching the save or abort option. Since there are not a whole lot of lines available to enter (title + 12 text lines), this is not a great burden and it saves room for the rest of the program by not including the source code to abort the function from within the routine. Once you have created and saved your *Credits Box*, refer to the code listing to find out how to read the user input for continuing after the display. Either Mouse input, Keyboard input or a combination of both may be used.

To use the newly created routine, just use the merge option from the GFA Basic editor to load your .LST file. You will see the procedure named exactly as the file is itself. This enables you to generate specific file names that correlate to your own customized procedures. If you do not want to use the name supplied for the procedure(s) rename them with the editor. You must be sure to name each file and procedure distinctly. Two procedures with the same name are not allowed in the GFA Basic program format (or any other language that this author is aware of for that matter). To actually use or CALL your procedure you will have to use one of the following methods:

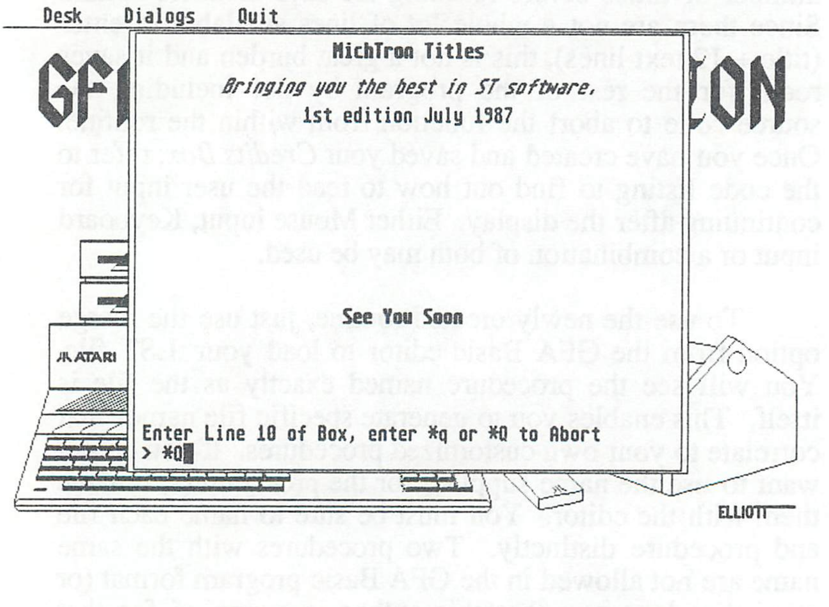
@MYROUTINE.LST

or

GOSUB MYROUTINE.LST

The MYROUTINE.LST would be whatever you have decided to call your *Credits Box* drawing procedure. That is all there is to designing a Credits Box. Pretty easy, Right? Well, even if it seems a bit complicated at first, you'll soon be designing attractive boxes with ease, and merging them into your own programs like a pro.

*Fig. 13: Type *Q or *q when finished*



MichTron Titles

Bringing you the best in ST software,

1st edition July 1987

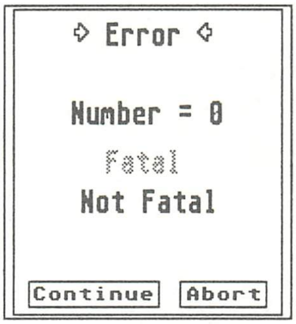
See You Soon

```
Enter Line 10 of Box, enter *q or *Q to Abort
> *Q
```

1.04
The Error Box Option

The *Error Box* option is a totally pre-scripted (already pre-determined) option that creates a Dialog Box that handles the display and interaction for error handling for you. It is infinitely more attractive than the Form Error Alert Box provided by the GEM AES.

Fig. 14: Error Box


◇ Error ◇

Number = 0

Fatal

Not Fatal

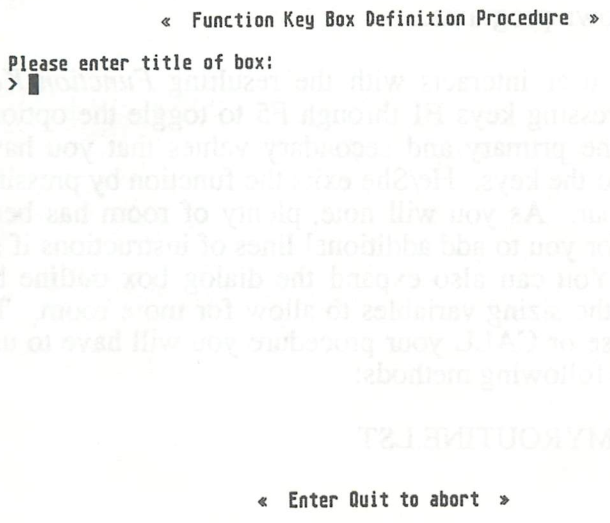
By choosing this option, an *Error Box* will be displayed for you on the screen and then you are prompted to either SAVE or ABORT the box. If Save is chosen, the source code listing to reproduce the box is saved in GFA Basic .LST file format ready for you to merge into your own program.

Refer to the source code listing to see how to interact with the box. There are examples of how to handle errors in the .LST libraries and the Tutorials. Beyond that, you are totally free to fine tune the *Error Box* to your own specifications. You may want to create several types of *Error Boxes* based on the generated error box source code, perhaps one for each major type of error. The next time a user suffers an error while using one of your programs, his dismay will be lightened by a really nice *Error Box*. Small compensation, Right?

1.05
The Function Key Box Option

The *Function Key Box* option provides an easy way of getting user input from the keyboard for up to 5 options. A popular implementation of this technique is to set or change program options from within a running application. The source code that is generated by this option will support only Medium or High Resolution Screens, however by the proper amount of tweening this style of box can be ported to Low Resolution screens (you have to do the work). Refer to the Source Code .LST libraries for examples of how this option might be used within a running application.

Fig. 15: Work Bench for Function Key Box



```
          « Function Key Box Definition Procedure »

Please enter title of box:
> █





                  « Enter Quit to abort »
```

Essentially, the generated procedure will read Function Keys F1 through F5 for you and toggle (flip) user options back and forth until the escape sequence is chosen by the user. So when the *Function Key Box* definition screen appears, you will have to define a Title for the Box, and Primary and Secondary values for the keys. If all five keys are not needed, you can press return for the values not wanted and delete the supporting source code from the generated .LST file. Using the same techniques contained in the *Function Key Box* procedure you can expand the covered procedures to the full 10 available function keys, or redefine the keys to any of those available on the keyboard.

Entering The Definitions:

When presented with the definition screen, you must simply enter the text for the title of the *Function Key Box*, and the primary and secondary definition for each of the five function keys offered. Following this series of text entry prompts, you are afforded the opportunity to either SAVE or ABORT the box as created. If SAVE is chosen, the box is saved in .LST file format, ready to be merged into your own programs.

The user interacts with the resulting *Function Key Box* by pressing keys F1 through F5 to toggle the options between the primary and secondary values that you have assigned to the keys. He/She exits the function by pressing the spacebar. As you will note, plenty of room has been reserved for you to add additional lines of instructions if so desired. You can also expand the dialog box outline by adjusting the sizing variables to allow for more room. To actually use or CALL your procedure you will have to use one of the following methods:

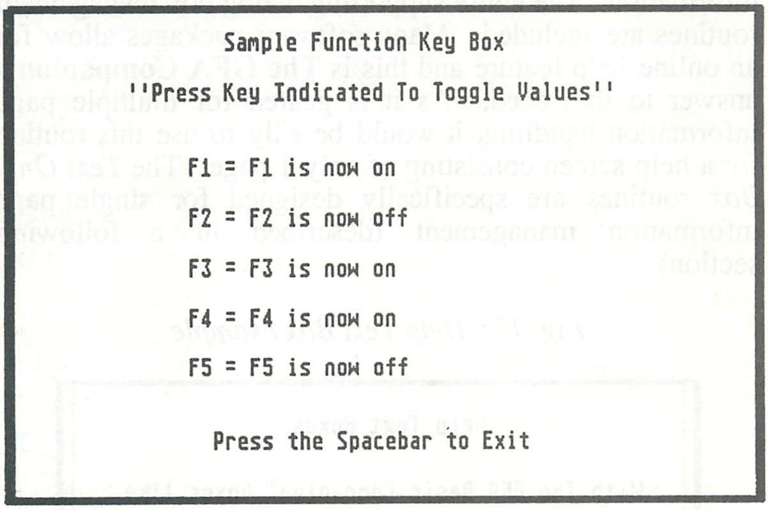
@MYROUTINE.LST

or

GOSUB MYROUTINE.LST

The MYROUTINE.LST would be whatever you have decided to call your *Function Box* drawing procedure. That's all there is to designing a *Function Box*. Not too bad, Huh?

Fig. 16: Function Key Box Example

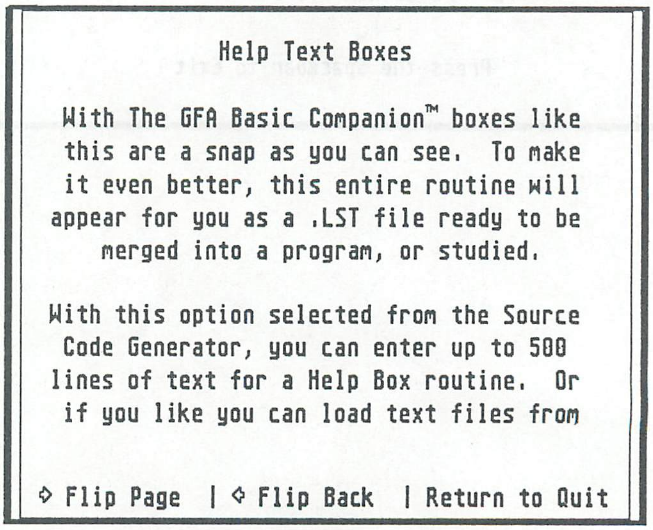


```
              Sample Function Key Box

       ''Press Key Indicated To Toggle Values''

               F1 = F1 is now on

               F2 = F2 is now off

               F3 = F3 is now on

               F4 = F4 is now on

               F5 = F5 is now off

             Press the Spacebar to Exit
```

1.06
The Help Text Box Option

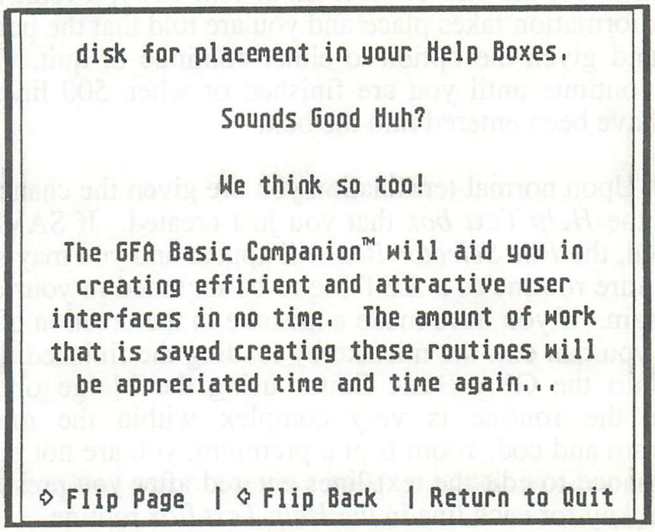
This specialized routine will create a complete online help routine for you to merge into your program. It is targeted at handling up to 42 (12 line) boxes of information. The entire supporting dialog box management routines are included. Many software packages allow for an online help feature and this is **The GFA Companion**'s answer to that need. As it is geared for multiple page information handling, it would be silly to use this routine for a help screen consisting of only 1 page. The *Text Only Box* routines are specifically designed for single page information management (described in a following section).

Fig. 17: Help Text Box example



```
                 Help Text Boxes

  With The GFA Basic Companion™ boxes like
  this are a snap as you can see.  To make
  it even better, this entire routine will
  appear for you as a .LST file ready to be
      merged into a program, or studied.

  With this option selected from the Source
  Code Generator, you can enter up to 500
  lines of text for a Help Box routine.  Or
  if you like you can load text files from

◇ Flip Page  | ◇ Flip Back  | Return to Quit
```

Fig. 18: Up to 500 lines of Text can be placed here.



```
disk for placement in your Help Boxes.

Sounds Good Huh?

We think so too!

The GFA Basic Companion™ will aid you in
creating efficient and attractive user
interfaces in no time. The amount of work
that is saved creating these routines will
be appreciated time and time again...

◇ Flip Page  | ◇ Flip Back  | Return to Quit
```

Upon clicking on this option from the pull down menu, you are given a choice to continue or abort as always. Next, an Alert Box will ask you if you would like some help. If you click on YES, a short description of the *Help Box* function is displayed within a dialog box for you. If NO is selected you enter the routine right away. The empty *Help Box* is drawn on the main screen for you just as it will appear when completed. Disregard the PRESS RETURN to QUIT portion of the screen as this is for the finished product and not for use during the creation of the routine.

A small area of the screen clears below the box and tells you to press any key to begin. Once you have done this, you may enter the word "quit" at any line prompt to exit the creation stage. This method of exiting is good only if the word "quit" is entered by itself on a prompt line, that way you can use the word quit in any sentence without exiting the function prematurely. As each line is entered by you it will be automatically centered and placed within the box. At the end of 12 lines of text entry, a redraw of the information takes place and you are told that the page is full and given the option to either continue or quit. This will continue until you are finished or when 500 lines of text have been entered into the box.

Upon normal termination you are given the chance to save the *Help Text box* that you just created. If SAVE is chosen, the *File Selector Box* will appear and you may save the entire routine as a .LST file to be included in your own program. If you have made a mistake in the creation of the box, you can edit the mistake by loading the finished .LST file into the GFA Basic Editor using the Merge option. Since the routine is very complex within the master program and code room is at a premium, you are not given the chance to edit the text lines entered after you press the return key for each line in the *Help Text Box* routine.

A bit of planning is in order so that you know where to break your pages of information and to ensure the formatting you desire. Once again, you can modify the source code within the editor following the creation of the routine. If you desire to create a very long *Help Text Box* routine and you do not want a line of code in your program for each line in your box, refer to the source code libraries to see how to load a text file from disk for use by the routine.

The user interacts with the *Help Box* by pressing either the right arrow key or the left arrow key. An information line appears at the bottom of the box telling them the keys that are supported within the routine. The left arrow key flips back to the previous page (if any), the right key advances to the next page (if any), and by pressing the return key the user leaves the Help feature. A bit of experimentation will demonstrate the functionality of this procedure to you very clearly. To actually use or CALL your procedure you will have to use one of the following methods:

```
        @MYROUTINE.LST
```

or

```
        GOSUB MYROUTINE.LST
```

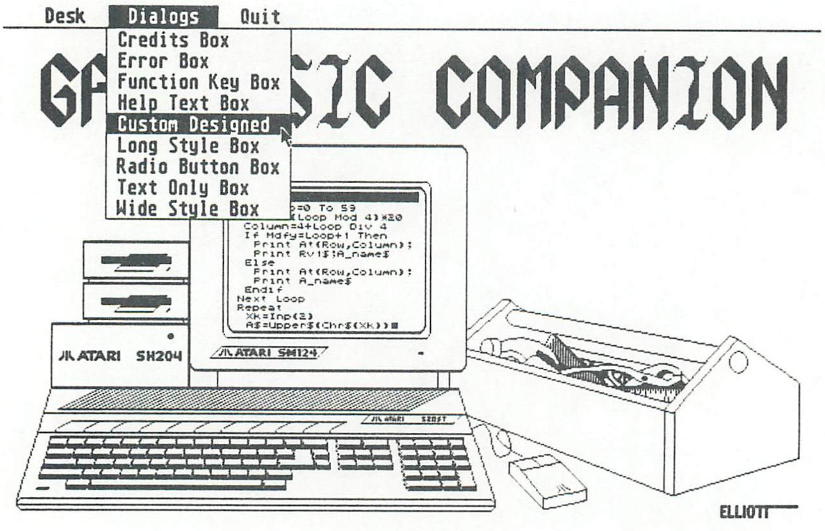
The MYROUTINE.LST would be whatever you have decided to call your *Help Box* Handling procedure. That's just about all there is to designing a *Help Box*. It's really not too complicated and surely adds a professional flair to any application.

1.07

The Custom Designed Option

This routine is the Flagship of **The GFA BASIC Companion's** Dialog Box Source Code Generator. With this option you are presented with a designer's screen that permits you to draw your own Dialog Box in a conventional manner, or you may choose to take a radically different approach to presenting a particular Dialog Box. If you are familiar with drawing boxes or adding text to a drawing on the screen with the popular D.E.G.A.S. program, then using this routine will be a snap for you.

Fig. 19: Select Custom Designed



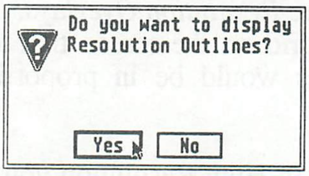
```
    =0 To 59
    (Loop Mod 4)*20
  Column=4+Loop Div 4
  If Mdfy=Loop+1 Then
    Print At(Row,Column);
    Print Rv1$;A_name$
  Else
    Print At(Row,Column);
    Print A_name$
  Endif
Next Loop
Repeat
  XK=Inp(2)
  A$=Upper$(Chr$(XK))
```

Following the selection of the *Custom Designed* option in the DIALOGS pull down menu, you are given a chance to continue or quit the function. If you choose to continue another alert box appears to ask you if you would like to display the *Resolution Overlays*. This option allows you to place a grid on the screen that tells you where the other resolutions would be in proportion to the current screen.

If you are in High Resolution you would see a series of lines appear on the screen. The dashed box in the upper left hand corner of your screen is where the Low Resolution Screen would fit, and above the solid line located at the middle of the screen is where the Medium Resolution Screen would fit. You are probably asking "Why do I want to see that?". Well, the answer is that, if you wish to design a box that works in both High and Low Resolution, you would have to make your box above the solid line located at the center of your screen. And if you wanted a Dialog Box that will fit in all Resolutions, then you must design the box within the dashed box in the upper left hand corner of the screen.

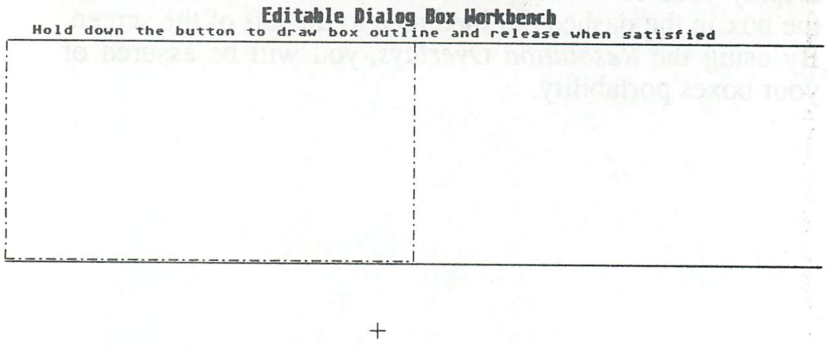
When in Medium Resolution, any box you design will work in High Resolution, but to be sure that you can display your box in all resolutions you will have to design the box in the dashed outlined area to the left of the screen. By using the *Resolution Overlays*, you will be assured of your boxes portability.

Fig. 20: Resolution Overlay Selection


Do you want to display
Resolution Outlines?

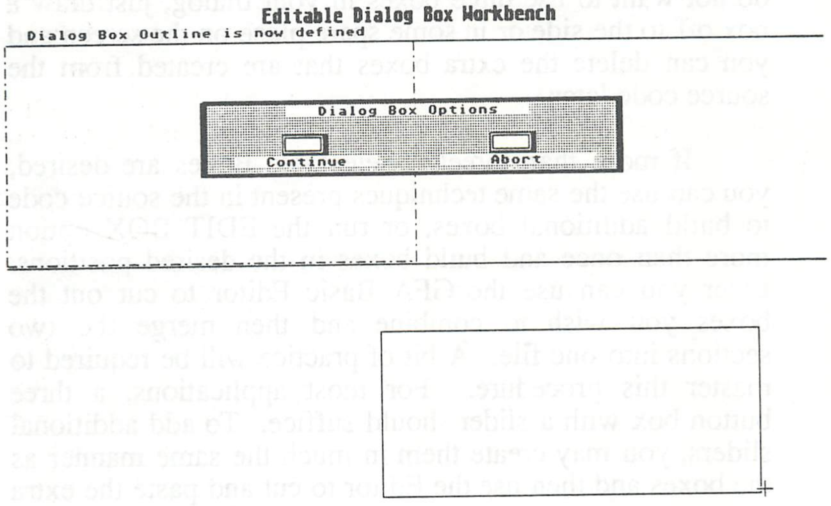
The next option available is the actual Work Screen itself, with a prompt for you to draw the outside outline of the box. To do this, locate the crosshair mouse cursor on the Work Screen where you would like the upper left hand corner of the box to start. When positioned, press down the left mouse button and hold. Next, move the mouse cursor to the location where you would like the bottom right hand corner to appear.

Fig. 21: Custom Design Work Bench


Editable Dialog Box Workbench
Hold down the button to draw box outline and release when satisfied

YOU MUST ALWAYS START AT THE TOP LEFT HAND CORNER OF WHERE THE BOX IS SUPPOSED TO BE, AND MOVE TO THE LOWER RIGHT HAND CORNER OF THE BOX BEFORE RELEASING THE MOUSE BUTTON.

Fig. 22: Drawing the box outline

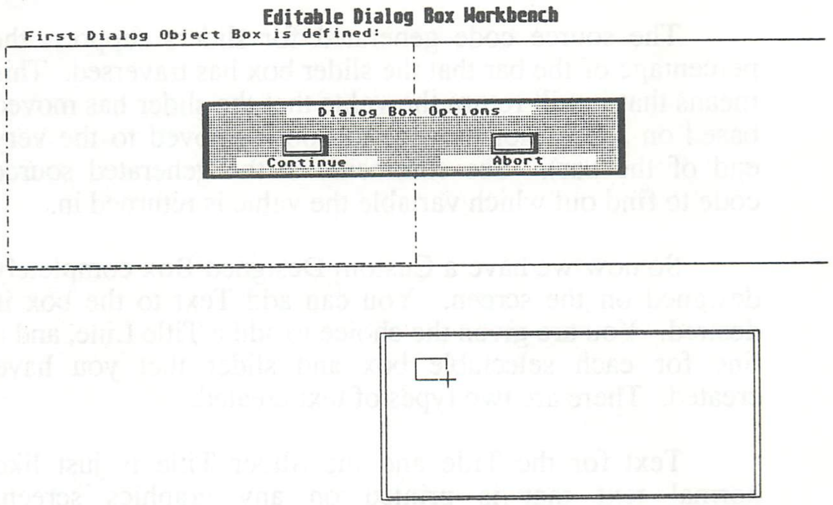

Editable Dialog Box Workbench
Dialog Box Outline is now defined
Dialog Box Options
Continue
Abort

By drawing in any other manner you will create a *MESS*. Once completed, you are presented with a special Alert Box asking you if you want to CONTINUE or ABORT. It is at this point you may choose to abort the current box if you messed up, and start over. This is the only place that you may restart the *Custom Designed Box* procedure. In every other situation, if you abort the current box you will be presented with the Main Menu Screen. If you are happy with the overall box outline, just select CONTINUE by clicking with the left mouse button.

For three similar boxes you will be presented with options in much the same manner as the above box. If you do not want to use three boxes in your dialog, just draw a box off to the side or in some spare place on the screen and you can delete the extra boxes that are created from the source code later.

If more than three(3) Selectable Boxes are desired, you can use the same techniques present in the source code to build additional boxes, or run the EDIT BOX option more than once and build boxes in the desired positions. Later you can use the GFA Basic Editor to cut out the boxes you wish to combine and then merge the two sections into one file. A bit of practice will be required to master this procedure. For most applications, a three button box with a slider should suffice. To add additional sliders, you may create them in much the same manner as the boxes and then use the Editor to cut and paste the extra code.

Fig. 23: Drawing inside selectable squares


Editable Dialog Box Workbench
First Dialog Object Box is defined:
Dialog Box Options
Continue
Abort

Adding a Slider:

Following the design of the overall box outline and selectable buttons, you are given the option to add a slider to your box. If selected, a small information box will appear in the top of the screen giving you a quick instructional run down on how it works. To begin, just press down the left mouse button and drag the slider to it's destination. When it is in the place you want, release the button to paste it down.

If you want to display a percentage information line above or below the slider later, leave an appropriate amount of space for it before you paste it down. With some practice, you'll have the slider routine down pat.

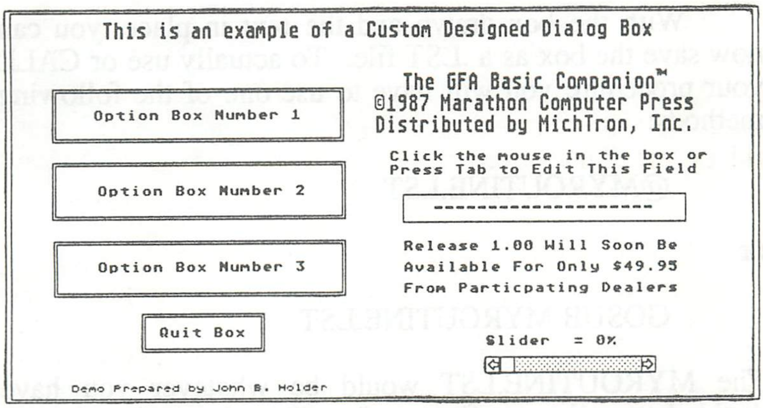
The source code generated for sliders supports the percentage of the bar that the slider box has traversed. This means that it will return the value that the slider has moved based on a 0% (not moved) to 100% (moved to the very end of the bar). See comments in the generated source code to find out which variable the value is returned in.

So now we have a Custom Designed Box completely designed on the screen. You can add Text to the box if desired. You are given the choice to add a Title Line, and a line for each selectable box and slider that you have created. There are two types of text created.

Text for the Title and the Slider Title is just like normal text that is printed on any graphics screen. However, the text for the three(3) selectable box buttons serves a dual purpose. The text placed within these boxes will invert itself when the option is chosen within a running application. What this means is that the box turns black and the text turns white so it can be seen when selected. If the option box is selected again, the box and text returns to it's normal state. See the examples in the source code libraries for a full understanding of the selectable box text.

You should keep this in mind when designing your box, and always place the text for the three selectable boxes INSIDE the selectable box that you are working on. You may however, decide to place it outside of the box. If the text is placed outside of a given box, you will be responsible for altering the generated source code to prevent it's inversion so it will be visible all of the time.

Fig. 24: A Three selection Custom Box with Slider and Text



This is an example of a Custom Designed Dialog Box

The GFA Basic Companion™
©1987 Marathon Computer Press
Distributed by MichTron, Inc.

Click the mouse in the box or
Press Tab to Edit This Field

Release 1.00 Will Soon Be
Available For Only $49.95
From Particpating Dealers

Demo Prepared by John B. Holder

To add text labels start typing on the keyboard. You will see the text appear at the mouse location. You are given complete freedom to use all of the keyboard combinations available. If you make a mistake, simply press the RIGHT mouse button to start over on the line you are currently working on. The Backspace key *DOES NOT* function here as it is a printable character, you must use the right mouse button to do a wipeout. Some interesting combinations are possible with this powerful text creation option.

When you are happy with the line you are currently working on, just move the mouse into the location that you want and press the LEFT mouse button to paste it down. There is *no* UNDO function so be sure it is in the right place before pasting it down. Otherwise you will have to start over from the beginning, or make minor alterations to the output source code following creation. These minor alterations are not that difficult to make, as you will see once you have used the option a few times. You will soon be placing the text like an expert.

With the box drawn and the text in place, you can now save the box as a .LST file. To actually use or CALL your procedure you will have to use one of the following methods:

```
@MYROUTINE.LST
```

or

```
GOSUB MYROUTINE.LST
```

The MYROUTINE.LST would be whatever you have decided to call your *Custom Box* procedure. Pretty easy after all isn't it?

Usage Variations:

This routine allows you to create a Dialog Box with complete freedom. If you so choose, you could even use the Generator to create 4 separate boxes and a slider located in various spots on the screen in an exploded sort of manner. Just remember that the first box that is drawn in this manner is not selectable since it was intended to be the master outline for the entire box. You could perhaps use it for a title box, it's your choice. See the source code libraries for an example of an exploded *Custom Designed Box*. Unlike the majority of routines created by the Source Code Generator, this option does not generate relocatable boxes for you.

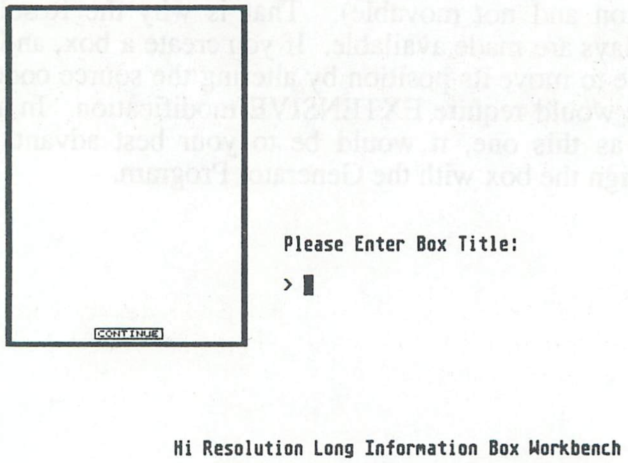
In contrast, the TWO BUTTON BOX option available in the *Radio Button Box* pull down menu will create a dialog box that can be moved anywhere on the screen by just changing the values in 4 positions (pointed out in the generated source code). The Custom Box option produces position dependent source code (set to a given position and not movable). That is why the Resolution Overlays are made available. If you create a box, and later decide to move its position by altering the source code; the move would require EXTENSIVE modification. In a case such as this one, it would be to your best advantage to redesign the box with the Generator Program.

1.08

Long Style Information Box Option

This handy little option will create a *Long* (vertical axis) *Information Box*. It is best suited for a quick information screen displaying a title and up to 10 lines of text with up to 18 characters in each. Great for a list of program commands.

Fig. 25: Long Information Box Workbench



```
Please Enter Box Title:

>
```

```
Hi Resolution Long Information Box Workbench
```

The option is selected from the Main Pull Down Menus under the DIALOGS sub heading. As with all of the other options available with **The GFA BASIC Companion**, you are given an option to CONTINUE or ABORT at the very start. If you select CONTINUE, you will have to finish the option prior to returning to the main menu. If you make a mistake and continue, just press the return key until the SAVE or ABORT Alert Box appears, you may then abort the function.

If you choose to continue, you are prompted to enter a title line for the box being created. Next, you must enter the text lines for the box. Up to 10 lines of text are allowed. With this option they are *not* centered automatically for you. It will give you a WYSIWYG display for your text lines, so plan ahead with this option. A small [CONTINUE] box is drawn for you at the bottom of the box and the supporting code is included in the source .LST file to read the user's response to continue. It is a pretty simple option, but wielded properly it can be a most useful type of Dialog Box. This box can also be positioned by moving or updating the four move points outlined in the .LST file. To actually use or CALL your procedure you will have to use one of the following methods:

```
@MYROUTINE.LST
```

or

```
GOSUB MYROUTINE.LST
```

The MYROUTINE.LST would be whatever you have decided to call your *Long Style Information Box* Handling procedure.

1.09
Radio Button Box Option

The *Radio Button Box* Option creates specialized Dialog Boxes for you to get from 2 to 6 user selected options by the use of a visually appealing interactive box. As this option creates all of the available boxes in essentially the same manner, only one description is offered in this section.

Fig. 26: You can ask for Help before starting...

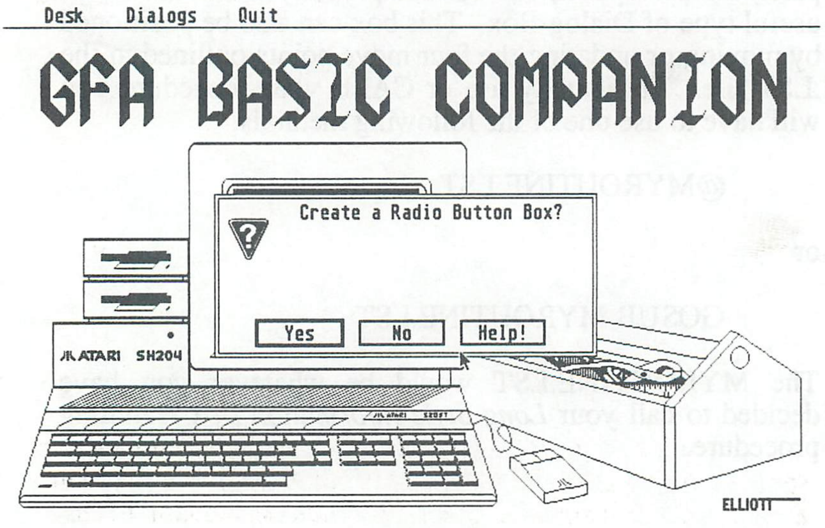
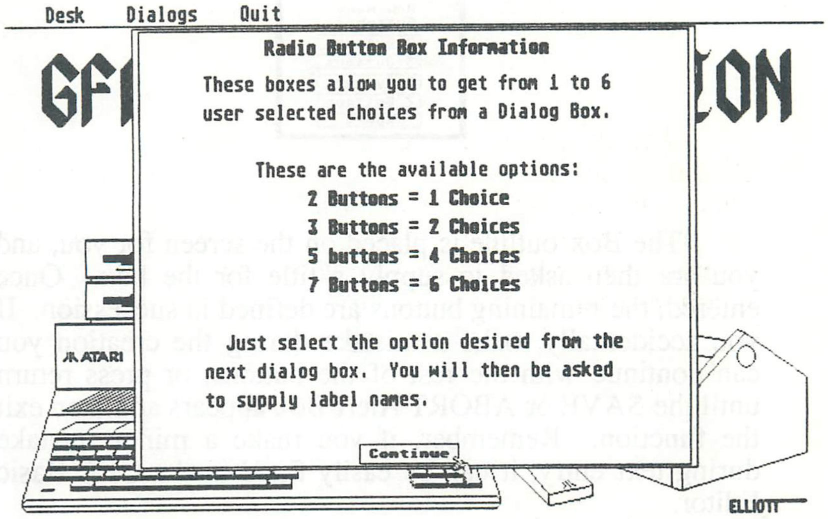


Fig. 27: ... And end up here



Radio Button Box Information

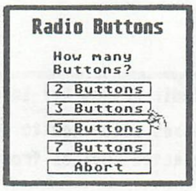
These boxes allow you to get from 1 to 6
user selected choices from a Dialog Box.

These are the available options:
2 Buttons = 1 Choice
3 Buttons = 2 Choices
5 buttons = 4 Choices
7 Buttons = 6 Choices

Just select the option desired from the
next dialog box. You will then be asked
to supply label names.

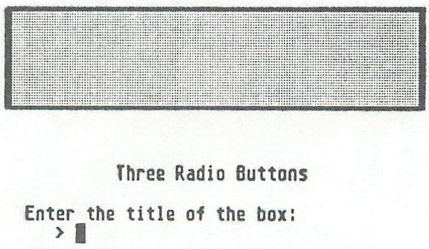
If you choose YES from the Alert Box, you will see a special Dialog Box that asks you if you would like to create a 2,3,5, or 7 Button Box. Or for a quick description of how many choices are available with the various boxes, consult the HELP option from the Alert Box prior to entering the option Work Screen.

Fig. 28: For this Example choose 3 Buttons



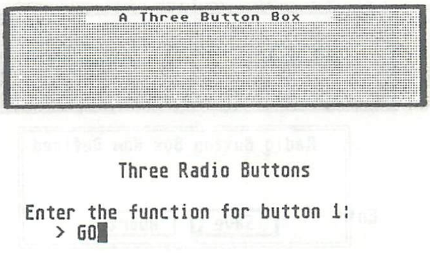
The Box outline is placed on the screen for you, and you are then asked to supply a title for the box. Once entered, the remaining buttons are defined in succession. If you accidentally make a mistake during the creation you can continue with the rest of the buttons, or press return until the SAVE or ABORT Alert Box appears and then exit the function. Remember, if you make a minor mistake during text entry, it can be easily fixed in the GFA Basic Editor.

Fig. 29: The Box Outline Screen



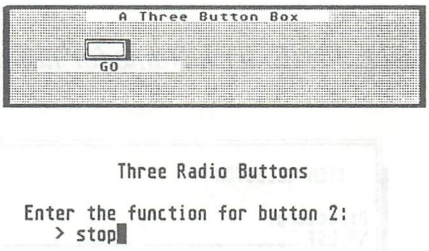
```
Three Radio Buttons

Enter the title of the box:
   > ▮
```

Fig. 30: Type in first Button Box Name



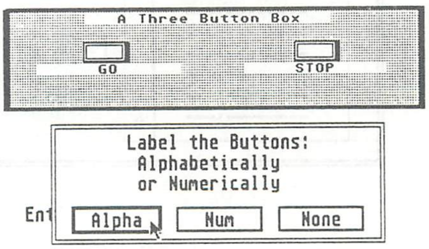
```
Three Radio Buttons

Enter the function for button 1:
    > GO
```

Fig. 31: Press <RETURN> after each name is entered



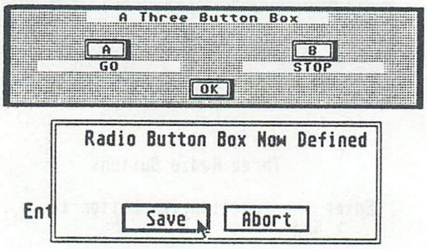
```
Three Radio Buttons

Enter the function for button 2:
    > stop
```

Fig. 32: Choose how the boxes are to be labeled



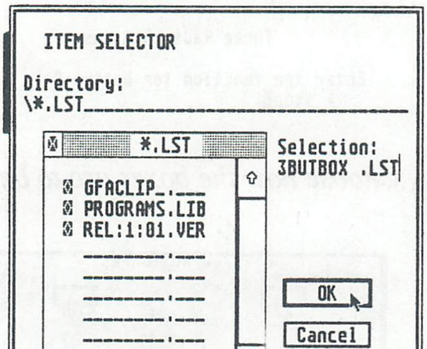
```
Label the Buttons:
Alphabetically
or Numerically
```

Fig. 33: Select Save



Radio Button Box Now Defined

Fig. 34: Enter a Name for new box



Directory:
*.LST

GFACLIP_.___
PROGRAMS.LIB
REL:1:01.VER

Selection:
3BUTBOX .LST

All text is automatically centered for you whether it is in the title block or beneath the buttons themselves. All supporting code to fill a button and invert the text label is included in the .LST file. See the source code listing for direction on how to read the variables that return the user's interactive choices. One variable is assigned for each button so it's really easy. Check the source code libraries for an example of how to use *Radio Button Boxes*.

Technical Tips on Dealing With Radio Button Boxes:

Radio Button boxes are presented in a smaller size in High Resolution, and a larger size in Low Resolution so you may wish to do some scaling. See the section in this manual that covers object scaling for further details. The key to success is to expand each counterpart equally.

If you want to lengthen a box vertically you must add equal amounts to the "Ty" and the "By" variables. See the source code library for an example.

If you choose you may change the text style presented in the box by experimenting with the DEFTEXT command wherever it appears in the source code listing. Fill patterns may be changed as well. Just experiment with the DEFFILL command that appears in the listing at the start of the .LST file. If the DEFFILL command is altered in the body of the procedure, the inversion of the key tops may not appear normal. To actually use or CALL your procedure you will have to use one of the following methods:

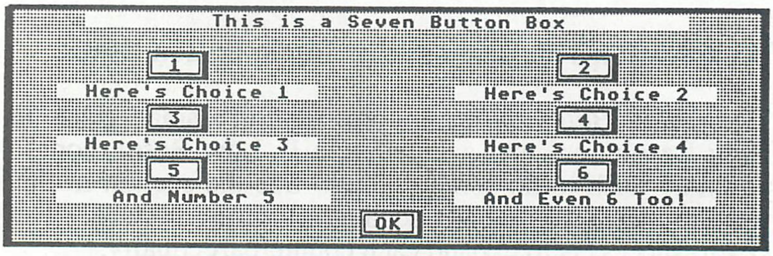
```
@MYROUTINE.LST
```

or

```
GOSUB MYROUTINE.LST
```

The MYROUTINE.LST would be whatever you have decided to call your *Radio Button Box* Handling procedure. Experimentation is the name of the game here. Have fun playing with *Radio Button Boxes*!

Fig. 35: You can have up to seven buttons in a box.



```
              This is a Seven Button Box

        1                                    2
   Here's Choice 1                      Here's Choice 2
        3                                    4
   Here's Choice 3                      Here's Choice 4
        5                                    6
   And Number 5                         And Even 6 Too!
                         OK
```

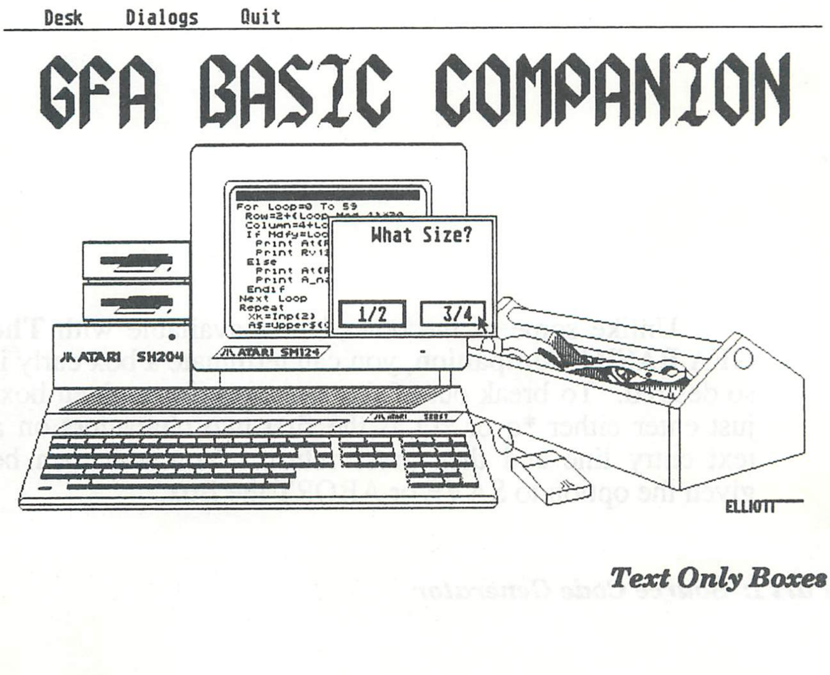
✢

1.10
Text Only Box Option

This option will create a Dialog Box on the screen designed to display text in a Centered or Ragged (Left Justified) manner. This is most useful when you need to display a fairly long description of a function that your program will accomplish, but you really do not need a *Help Text Box*, and the *Wide Style* or *Long Style Boxes* are not sufficient.

Enter this option by Selecting the *Text Only Box* option from the Dialogs pull down menu. Following your choice to continue, a small Alert Box appears and asks you which size is appropriate for your need. The two sizes are 1/2 and 3/4.

Fig. 36: Selecting the size of the Text box



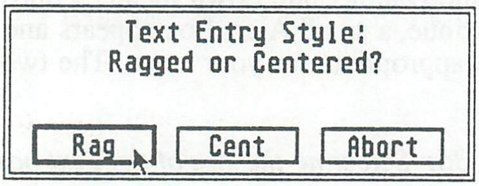
```
For Loop=0 To 59
  Row=2+(Loop
  Column=4+Lo
  If Mdfy=Loo
    Print At(R
    Print Rv1
  Else
    Print At(R
    Print A_n
  Endif
Next Loop
Repeat
  Xk=Inp(2)
  A$=Upper$((
```

The specifications for the two types is as follows:

1/2 Size Box = 16 Lines by 40 characters across
3/4 Size Box = 18 Lines by 61 characters across

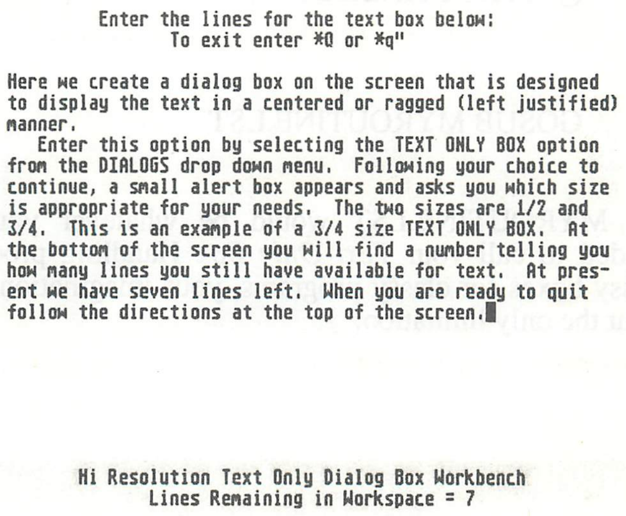
If RAG is selected you are on your own when it comes to formatting your text lines. In other words you have a WYSIWYG type display in the box. Just the opposite, if CENT is chosen for text placement, each line will be automatically centered for you on output.

Fig. 37: Choosing Left Justified (Rag) or Centered (Cent)


Text Entry Style:
Ragged or Centered?

Unlike some of the other boxes available with **The GFA BASIC Companion,** you can terminate a box early if so desired. To break out of the text entry for a given box, just enter either *q or *Q as the first two characters on a text entry line and then press return. You will then be given the option to SAVE or ABORT the Box.

*Fig. 38: Enter *Q or *q on a new line to exit*



```
               Enter the lines for the text box below:
                      To exit enter *Q or *q"

Here we create a dialog box on the screen that is designed
to display the text in a centered or ragged (left justified)
manner.
   Enter this option by selecting the TEXT ONLY BOX option
from the DIALOGS drop down menu.  Following your choice to
continue, a small alert box appears and asks you which size
is appropriate for your needs.  The two sizes are 1/2 and
3/4.  This is an example of a 3/4 size TEXT ONLY BOX.  At
the bottom of the screen you will find a number telling you
how many lines you still have available for text.  At pres-
ent we have seven lines left. When you are ready to quit
follow the directions at the top of the screen.█




             Hi Resolution Text Only Dialog Box Workbench
                   Lines Remaining in Workspace = 7
```

With the output source code .LST file you will find that the box management consists solely of temporary screen management by the manipulation of the TEMPUSE$ variable that stores the original screen for you, and a suggestion of how to read the user's input to continue with the program. The suggestions are to use the MOUSEK or A=Inp(2) examples located at the end of the procedure. It's your call. A good idea is to use the last line in the box to tell the user what to do to exit. For example:

Press Any Key to Continue...

A nice modification to this type of box is to use a fill pattern different from the Deffill 0,2,8 that draws the Dialog Box for you at the start of the procedure. If you are using a color monitor, try using green or red to fill the box. You'll need to use an XOR mode for the text too. See a demo in the source code library. To actually use or CALL your procedure you will have to use one of the following methods:

```
@MYROUTINE.LST
```

or

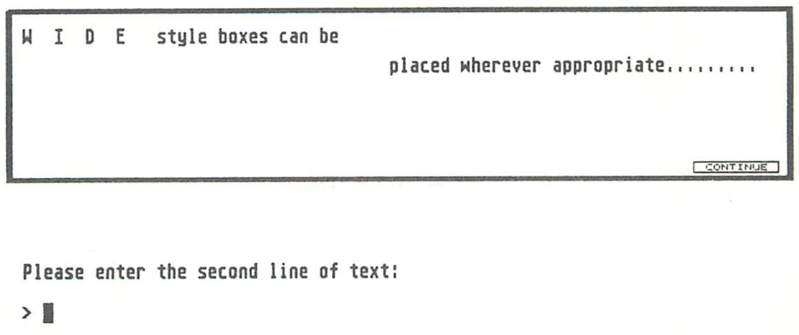
```
GOSUB MYROUTINE.LST
```

The MYROUTINE.LST would be whatever you have decided to call your *Text Only Box* Handling procedure. Classy boxes for classy programs, your imagination is just about the only limitation.

1.11
Wide Style Information Box Option

This Option creates a *Wide* (horizontal) *box* to display information within a running application. It is an indispensable tool when you do not want to obscure the entire screen with an information box, or if you only have a couple of long lines in your description. By adjusting the move points in the source code listing, you can effectively place the box anywhere on the screen that it properly fits. You may decide to use it at the bottom of the screen or the top. A small [CONTINUE] Box is drawn for you at the bottom right of the Box. This may be left for effect or removed from the source code .LST file. As with all of the *Text Information Boxes*, refer to the .LST file to see the recommended methods of reading user input to continue with the program after the box has been displayed and read.

Fig. 39: Creating Wide Style Boxes



```
W I D E  style boxes can be
                                   placed wherever appropriate.........

Please enter the second line of text:
>
```

Box Specifications: You are permitted to enter a title for the box, and up to 4 text lines consisting of a total of 70 characters per line. It is a good idea to include on the last line how the user is supposed to exit the function, such as press a key or mouse button. With this style of box you have a true WYSIWYG display. No centering is done for you. Plan ahead and format your lines appropriately. To actually use or CALL your procedure you will have to use one of the following methods:

@MYROUTINE.LST

or

GOSUB MYROUTINE.LST

The MYROUTINE.LST would be whatever you have decided to call your *Wide Style Box* Handling procedure.

Part 2

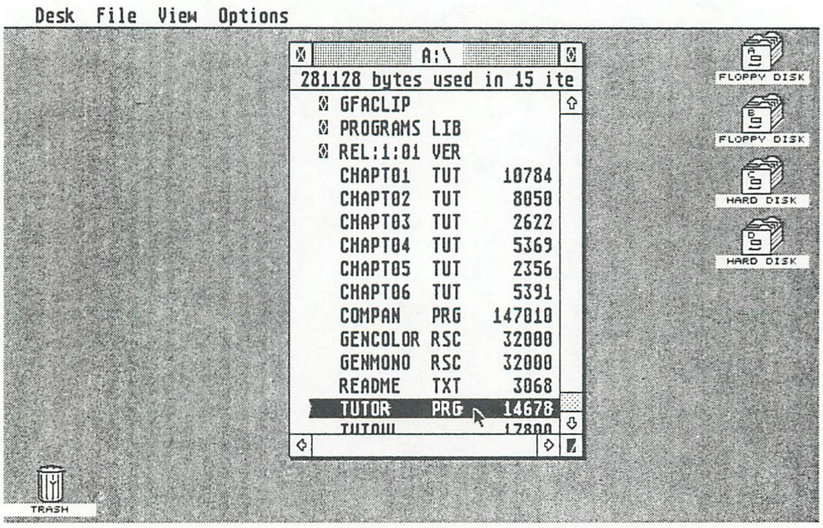
The Tutorial Driver

2.00

General Information

The second feature of **The GFA BASIC Companion** covered in this reference manual is the program that presents the Tutorials for you on your monitor. Since the program is very easy to use, not too much space will be spent in the manual to describe its usage. Just a quick break in here, and then it's off to the GEM desktop to experiment for yourself.

Fig. 40: The Tutorial Program



```
A:\
281128 bytes used in 15 ite
  GFACLIP
  PROGRAMS LIB
  REL:1:01 VER
  CHAPT01  TUT    10784
  CHAPT02  TUT     8050
  CHAPT03  TUT     2622
  CHAPT04  TUT     5369
  CHAPT05  TUT     2356
  CHAPT06  TUT     5391
  COMPAN   PRG   147010
  GENCOLOR RSC    32000
  GENMONO  RSC    32000
  README   TXT     3068
  TUTOR    PRG    14678
  TUTOUL          17800
```

Starting the Driver:

To run the Tutorials, just double click with the mouse on the Icon or Text Line for TUTOR.PRG. You will be presented with the same screen as the Source Code Generator so the setup will be familiar to you. That's about as far as the likeness goes though. You'll notice right away that the Menu Bar has changed. The second Field is no longer DIALOGS, but is now THE TUTOR.

2.01
The "Desk" Section of The Pull Down Menu Bar

The Desk Section will appear much in the same way as in the Source Code Generator. Once again you will note that all Desk Accessories have been disabled. There is really no valid reason to have access to any resident utilities or applications since you just came to look and read.

By clicking on the *About Program* option you will be presented with the credits for the Tutorial Driver Program.

✢

2.02

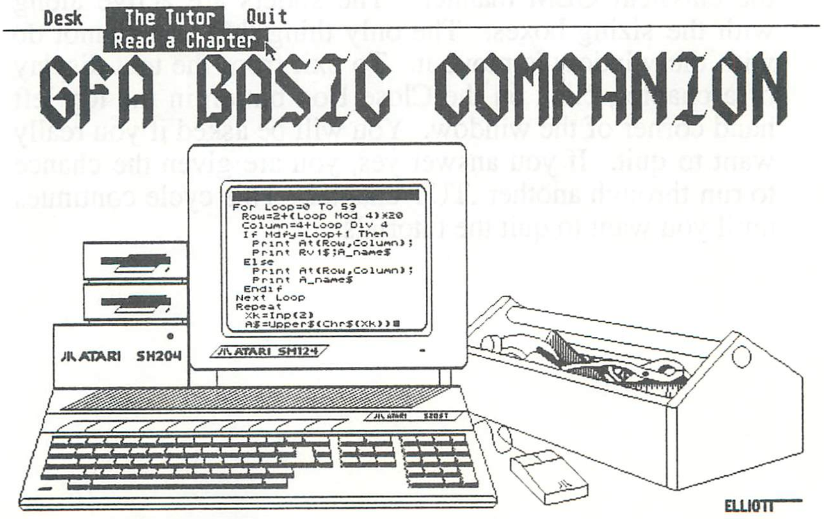
The Tutor Option

Under THE TUTOR, you will find a single option that clearly states its purpose which is to read a chapter. Click on it to load the overlay.

Reading a Chapter:

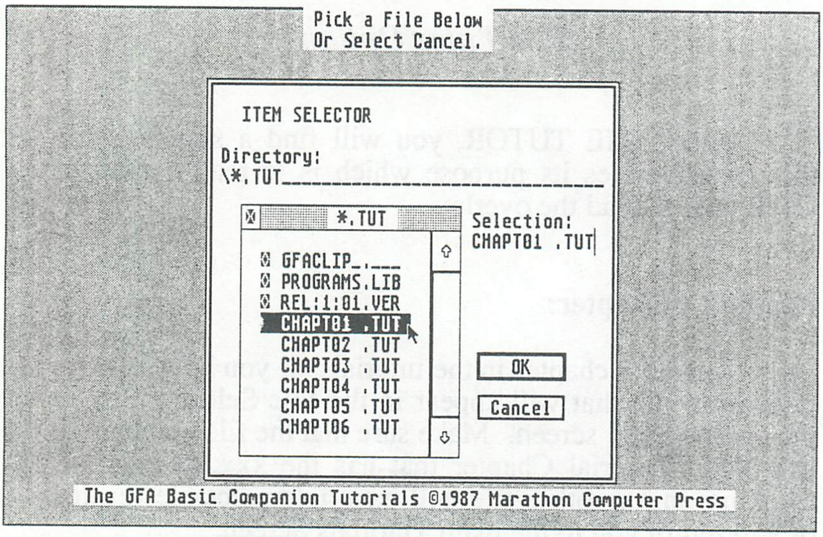
To read a chapter in the tutorials all you have to do is pick out a file that will appear in the File Selector Box in the center of the screen. Make sure that the file you choose is a valid Tutorial Chapter that has the xxx.TUT suffix. Clicking on any other file will have unpredictable results, or will return you to the main Tutorials Screen.

Fig. 41: Select Read a Chapter *from The Tutor menu...*



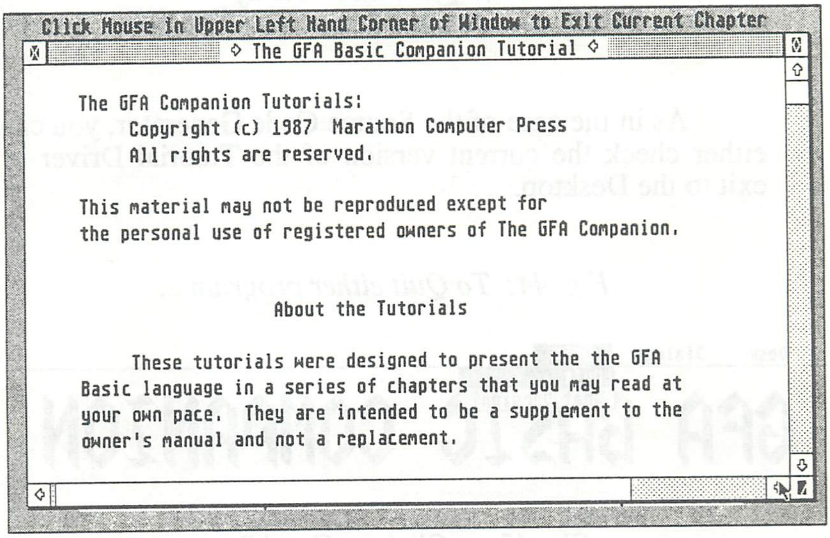
```
For Loop=0 To 59
  Row=2+(Loop Mod 4)*20
  Column=4+Loop Div 4
  If Mdfy=Loop+1 Then
    Print At(Row,Column);
    Print Rv1$;A_name$
  Else
    Print At(Row,Column);
    Print A_name$
  Endif
 Next Loop
 Repeat
  Xk=Inp(2)
  A$=Upper$(Chr$(Xk))
```

Fig. 42: ...Then select a chapter


```
Pick a File Below
Or Select Cancel.

ITEM SELECTOR
Directory:
\*.TUT

*.TUT                Selection:
                     CHAPT01 .TUT
GFACLIP_.___
PROGRAMS.LIB
REL:1:01.VER
CHAPT01 .TUT
CHAPT02 .TUT
CHAPT03 .TUT
CHAPT04 .TUT
CHAPT05 .TUT
CHAPT06 .TUT
                     OK
                     Cancel

The GFA Basic Companion Tutorials ©1987 Marathon Computer Press
```

Following the selection of a chapter, a GEM window will appear on your screen and text will appear in the window. You are free to scroll through the file at will in the classical GEM manner. The sliders are active along with the sizing boxes. The only thing that you cannot do with the window is move it. To exit from the text display of a chapter, click on the Close Box button in the top left hand corner of the window. You will be asked if you really want to quit. If you answer yes, you are given the chance to run through another .TUT chapter. This cycle continues until you want to quit the tutorials.

Fig. 43: A page from the Tutorial


Click Mouse in Upper Left Hand Corner of Window to Exit Current Chapter

◇ The GFA Basic Companion Tutorial ◇

```
The GFA Companion Tutorials:
    Copyright (c) 1987  Marathon Computer Press
    All rights are reserved.

This material may not be reproduced except for
the personal use of registered owners of The GFA Companion.


                    About the Tutorials

    These tutorials were designed to present the the GFA
Basic language in a series of chapters that you may read at
your own pace.  They are intended to be a supplement to the
owner's manual and not a replacement.
```

2.03

The Quit Section of The Pull Down Menu Bar Section

As in the case of the Source Code Generator, you can either check the current version of the Tutorial Driver or exit to the Desktop.

Fig. 44: To Quit either program ...

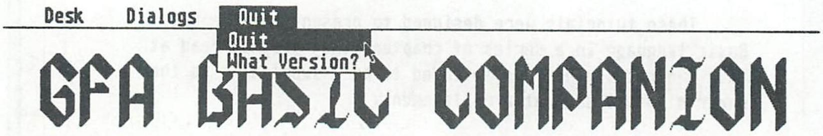
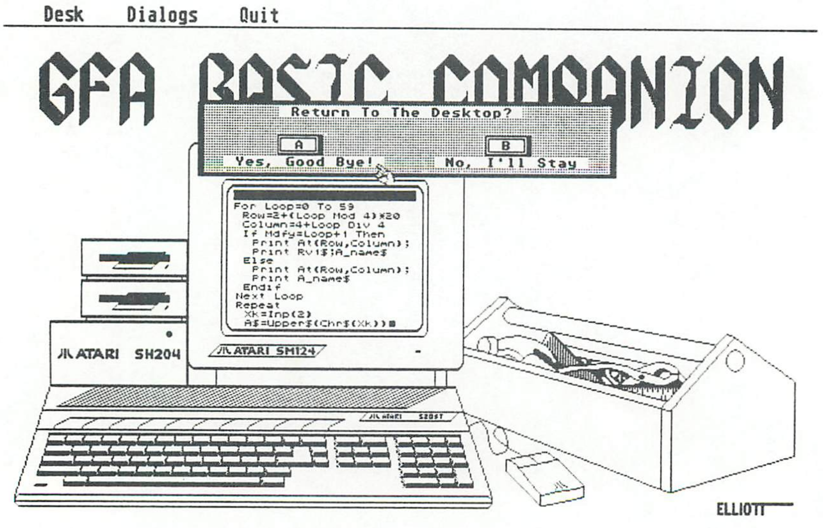


Fig. 45: ...Click on Good Bye ...



```
For Loop=0 To 59
  Row=2+(Loop Mod 4)*20
  Column=4+Loop Div 4
  If Mdfy=Loop+1 Then
    Print At(Row,Column);
    Print Rv1$;A_name$
  Else
    Print At(Row,Column);
    Print A_name$
  Endif
Next Loop
Repeat
  Xk=Inp(2)
  A$=Upper$(Chr$(Xk))
```

2.04

Printing the Tutorial Files

Since the Tutorial files are stored in ASCII rather then binary format, you may, if you wish, print out hard copies of the files. Please remember it is for your use only. The Tutorials are not Public Domain, they are Copyrighted material and it is illegal to give them away, or otherwise distribute them. This is meant to be a convenience to you, please do not abuse it.

Any Word Processor that reads ASCII files can load and print the xxx.TUT files for you. First Word and WordWriter ST can both do this. Refer to your Word Processor owner's manual for details. If you do not have a word processor, you can also list the files by double clicking on a .TUT file from the Desktop and selecting the Print option in the Alert Box.

2.05

Additional Tutorial files and Support Material

Additional Tutorials may be included in a separate directory by the name of GFATIPS. Files that are located in that folder are Help Files and Source Code that have appeared on GEnie (General Electric Network for Information Exchange) and were written by the author of this software package. These files are not intended to work with the Tutorial Driver. Just print them out if desired in the normal manner. They have been included on the release diskette(s) as an added bonus just in case you do not have access to GEnie, or a modem.

Part 3

Modifying the Output Source Code

3.00

Considerations Taken When Developing The Generator

Before we get into the thick of things, it might be a good idea to fill you in on some of the insights that were the driving factors when the Source Code Generator was developed.

First and foremost, the newly generated code should be able to support either a color or monochrome system. Next, it should do so with a minimum of difference in the source code output. With this in mind, the problem was tackled.

Based on those guidelines, some more questions arose and were answered with as few trade-offs as possible. Some were:

Q. *What font supports all three resolutions?*

A. The 8x8 font looks normal in all resolutions, so that is the universal font produced by the Generator. Some combinations do not appear readable in all resolutions, but this one is an exception.

Q. *Should every box support all resolutions?*

A. It's pretty hard to support everything in three resolutions, so the boxes should function at least in high and medium resolution. Some Dialogs created with a classical RCS will appear strange in low resolution too, so it's not just a problem specific to some of the boxes created by **The GFA BASIC Companion**. But since you are provided with source code for the object, you can tween it to your own specifications.

Q. *Won't there be a placement problem with the objects on different resolution screens?*

A. Yes. That is why the 4 point movement concept arose. With the boxes that can be used in all resolutions, you may move the box around by updating only 4 points (Lx,Rx,Ty,By variables that are imbedded in the final output source code).

Q. *With the wide range of differences in the three available resolutions, won't there be differences in the size of the objects? What if I want larger or smaller boxes than the Generator produces?*

A. Yes, there are differences in the size of the objects created by the Generator. Since the name of the game is compatibility with either a color or monochrome system, the middle resolution is the target field for source code generation. No modification at all is necessary with medium resolution objects. Some of the objects are not compatible with low resolution or must be modified, and all of the objects will function in high resolution but you may desire to modify them to reduce the stretch effect created by using a small font and the reduced height of the objects. Examples of these types of modifications are located in the source code libraries.

3.01
Moving Objects With The 4 Move Points

Let us look at an example of object movement. We'll say that we have just created a TWO BUTTON *Radio Button Box*, and we want to use it in both Medium and Low Resolution. A simple modification is necessary. First, locate the section of the source code near the top of the listing that appears like this:

```
Lx=154
Rx=480
Ty=75
By=130
```

To have the box function in both resolutions properly add the following lines of code just beneath the above listing:

```
If Rez=0 then
    Lx=14
    Rx=315
' What we did here was first move the left margin
over
' to an acceptable position for low resolution, and
next
' we will squeeze in the sides of the box by a few
' pixels so it will fit on the low resolution screen
' with (320 x 200) pixels available.
Endif
```

To handle the pixel squeeze operation, move down through the listing a few lines until you locate the outside dimensions of the TWO BUTTON Box. It will look something like this:

```
Sget Tempuse$
Deffill 1,2,2
Deftext 1,0,0,6
Defline 1,3
Pbox Lx,Ty,Rx,By
Box Lx,Ty,Rx,By
```

For the resolution squeeze add the following code in the appropriate places:

```
' *** picking up from the example above ***

Defline 1,3
' Add the next 4 lines to the existing code.
' If Rez=0 then
     Pbox Lx+10,Ty,Rx-10,By
     Box Lx+10,Ty,Rx-10,By
Else
'
' The next two lines were already there, so leave
them. Pbox Lx,Ty,Rx,By
Box Lx,Ty,Rx,By
'
' Add this next line.
Endif
```

When you get used to the movement of objects, it'll be second nature to move them around with ease from within your own programs.

With that behind us, let's take a look at an easier situation: a minor adjustment to a *Wide Style Information Box* so it will be located at the bottom of a High Resolution Screen instead of at the top, as is standard with the source code output.

Locate this following snip of code in the .LST output file for your newly created *Wide Style Information Box*:

```
Lx=14
Rx=624
Ty=15
By=85
```

Add the following code just beneath the above lines:

```
If Rez=2 then
Ty=Ty+300
By=By+300
Endif
```

This results in the box appearing at the bottom of the screen if the procedure is called in High Resolution and will continue to appear at the top of the screen in Medium Resolution. Neat huh?

Not in an attempt to mislead you, the desired modifications can become quite extensive. For example, if you chose to modify a SEVEN BUTTON *Radio Button Box* so that it is twice as tall in High Resolution and uses the normal font for High Resolution (DEFTEXT 1,0,0,13), then the required changes will prove to be fairly extensive. Since modifications of that type are beyond the scope of this manual, a lengthy example is best left to the source code libraries.

It is suggested that you first become acquainted with the use of Dialog Boxes, and then learn how to make minor changes to the source code before jumping headlong into major resolution modifications to a .LST file.

3.02

Changing Fill Patterns

Changing the fill patterns in the Generator produced Dialog Boxes is really very simple. Anywhere you see a DEFFILL in the .LST file, you can change it at will with any of the fill patterns listed in the GFA Basic Manual or those you define yourself.

Some special considerations:

It is HIGHLY recommended that you only change the fill pattern for the body of a box and not the DEFFILL 0,2,8 and DEFFILL 1,2,8 that occurs within most of the boxes to fill and clear the Radio Buttons on Selectable Object boxes. By sticking to this, you will mimic GEM dialogs completely. But if you really want to be different, go ahead and modify the color of the fill but *not* the style. For example DEFFILL 0,2,8 for the clearing routine and DEFFILL 2,2,8 for the solid fill. It's up to you. Just experiment a bit and do not get frustrated with results.

Also, with some styles of boxes (*Custom Designed* to name one) some of the text lines interact with user input. The percentage line on a slider for example. It must be in Graphmode 1 (replace) for it to function properly. This can mess up your solid fill pattern. Once again, try out all changes *Before* including them in your own programs.

3.03

A Word About Fonts

You have the freedom to change the fonts used in all of the Dialog Boxes, however, the 8x8 font is really the only one that looks pretty much normal in all three resolutions. Additionally, with the change in font style, you may have to do modifications to the source code with varying difficulty. As always, experiment with your ideas Before you include procedures with Font changes incorporated. It is best when you do your changes to use a conditional If Rez=(Whatever 0-2), (The changes), and Endif. That way you will not affect the portability of the code produced by **The GFA Companion's** *Source Code Generator*.

3.04

Product Development And Plans

Depending on the interest and popularity of this product in the GFA Basic community, some future developments for the Generator could be the incorporation of Custom Icons and the inclusion of Conditional Resolution adjusted source code output. If you enjoy this product and know of someone that also owns GFA Basic, why not mention it to them. Or better yet, show them the results of your work. Just be sure to abide by the Licensing Agreement with this package, and above all, Please Honor Our Copyright.

Software Piracy hurts everyone in the end. If you are reading these words from a photocopy, then you are most likely in receipt of a pirated copy and you should visit your nearest dealer or write to MICHTRON for a legal copy. Along with the satisfaction that you are not breaking the law, you will receive customer support and reasonable upgrades for this product from MICHTRON if you purchase a copy and send in the Registration Card.

3.05

Joining A National Billboard Service

If you own a modem you might want to check into joining GEnie. GEnie is the General Electric Network for Information Exchange. It is a major online billboard service that allows you to communicate with other programmers with common interests, and you can download Public Domain software from the RoundTable Libraries.

There is an ATARI ST RoundTable with thousands of ST programs available, and best of all a MICHTRON Product Support that houses extensive libraries (Lots of GFA Basic Programs), Conference rooms (For Group Discussions), and a Product support Bulletin Board for you to air your questions to the *MichTron* staff concerning any of the Products that we publish.

For more information on GEnie, and a free trial run of the service follow these instructions:

1. Set your modem for Half Duplex (local echo) at either 300 or 1200 Baud.
2. Dial 1-800-638-8369. (Free call) When you connect, type the letters HHH and press return.
3. A prompt that looks like this will appear - U# When you see that prompt, type in XJM11957,GENIE and press return.

The system will fill you in on the details from that point on. If you are interested in joining up, just break out your MasterCard, Visa, AMEX, or Checking Account information and join immediately ONLINE! If you choose to join, you will be joining thousands of avid computer enthusiasts all over the country, and the world. An added feature of GEnie is that it sports some of the lowest connect charges going! Over less than half of Compuserve during non-prime time. It really is a wise investment for getting the most for your computing dollar. To get to the MICHTRON Product Support RoundTable after you have received full access to GEnie, just type in *MICHTRON* at the top page following your log in. Then settle back and enjoy the programs & support that are available to you at the press of a key. Have fun!

3.06
Author's Closing Remarks

I truly hope that you enjoy this program package as much as I enjoyed putting it together for you. I also hope that you will enjoy a lot of saved hours of work, and a more professional user interface for your programs as a result of this product.

Furthermore, this was not intended to serve solely as an RCS for GFA Basic, but also to change the way you think about GFA Basic in general. I hope that by your careful examination of the code produced by the Source Code Generator, and the code contained in the libraries that you will begin to realize the overwhelming possibilities of the GFA Basic Language. In my opinion it is truly one of the most remarkable Interpreter and Compiler combinations to ever be produced for a home computer.

Index

INDEX

*q 52
.LST file 21, 26, 49, 72
.TUT file 65
4 Move Points 70
4 point movement concept 69
520 ST owners 3
8x8 font 68, 74
About Program 60
Adding a Slider 37
ATARI ST RoundTable 76
Backup copy 5
Changing Fill Patterns 73
COMPAN.PRG. 4, 14
Credits Box Option
    12 lines 19
    drawing procedure 22
    insert a blank line 19
    Saving 21
    Selecting Font Style 18
    Title and Text Line entry 21
Custom Designed Option 12
    add a slider 37
    add Text 38
    add text labels Option 39
    continue or abort 36
    designer's screen 32
    interactively created 12
    minor alterations 40
    more than three(3) Selectable Boxes 36
    no UNDO function 39
    Resolution Overlays 12, 33
    special Alert Box 36
    text inversion 38
    three similar boxes 36
    Title Line 38
    Usage Variations 41
    wipeout 39
    Work Screen 34
DEFFILL 49, 53, 73
DEFTEXT 49, 72
Desk accessories 3, 15
Desk Pull Down Menu 15
    About Program 15
    Accessories 15
Desk Section 60
Development tools 10
Dialog Box 9
Dialogs pull down menu
    Credits Box 16
    Custom Designed 32
    Error Box Option 23
    Function Key Box option 25
    Help Text Box Option 28
    Long Style Information Box 42
    Radio Button Box 44
    Text Only Box 51
    Wide Style Information Box 55
Edit box option 36
Error Box option 23
    handle errors 23
    how to interact 23
    Save or Abort 23
Exit to the Desktop 64
Exploded Custom Designed Box 41
File Type Identifier 3
Font style 17
Form Error Alert Box 23
Function Key Box Option
    5 options 25
    definition screen 26
    drawing procedure 27

Function Key Box Option (Cont.)
  Primary and Secondary values 26
  save or abort 26
  set or change program options 25
  Title 26
Function Keys 26
GEM desktop 15
General Electric Network for Information Exchange 76
GEnie 76
GFA Basic editor 21
Ghost Text 15
GOSUB MYROUTINE.LST 22, 27, 31, 40, 43, 49, 54, 56
Help Text Box Option
  500 lines of Text 29
  breaking pages 30
  dialog box management 28
  exiting 29
  Handling procedure 31
  multiple page 28
  online help 28
  supported keys 30
  user interaction 30
Long Style Information Box 42
  10 lines of text 42
  18 characters in each 42
  Handling procedure 43
  quick information screen 42
  Save or Abort Alert Box 43
  title line 43
  vertical axis 42
Main menu bar 15
Merge option 21
Modifications 21
MOUSEK 53
Moving Objects 70
MYROUTINE.LST 22, 26, 31, 40, 43, 49, 54, 56
Object scaling 49
Overlapping boxes 13
Percentage information line 38
Placement problem 69
Position dependent source code 41
Pre-scripted 12
Product Development 75
Quit Section 64
Radio Button Box 44
  2 to 6 user selected options 44
  Handling procedure 50
  HELP option 45
  interactive 44
  invert the text label 49
  object scaling 49
  Save or Abort Alert Box 46
  supporting code 49
  Technical Tips 49
  text centered 49
  text style 49
Rag (Left Justified) 52
Random Access Memory 3
RCS 10
README.1ST file 12
Resolution Overlays 12, 33, 41
Resource Construction Set 11
RoundTable 76
Running application 25
Screen replacement 12
SGET TEMPUSE$ 12
Shadow text 13

Slider 36, 38
Source Code
  modifying 21
Source Code .LST libraries 25
Source Code generator 3, 8, 9
  accessories 15
  benefits 9
    100% GFA Basic
      compatible 9
    amount of work avoided 9
    easier to learn 10
    unlimited capabilities 9
    works with GEM 9
  developed 68
  extensive modifications 72
  font supports 68
  modification 70
Source code libraries 41, 49, 69,
    72
Standard RCS 10
Stretch effect 69
Temporary use video screen 12
  32000 (32K) bytes 12
  calling routine. 12
  TEMPUSE$ 12
TEMPUSE$ 12, 53
Text Only Box 51
  *Q 52
  1/2 Size Box 52
  3/4 Size Box 52
  Centered 51
  display text 51
  Ragged (Left Justified) 51
  Save or Abort 52
  terminate a box 52
  two sizes 51
THE TUTOR 59
The Tutorial Program 58
TUTOR.PRG 4, 59
Tutorial Driver 60
  .TUT chapter 62
  Additional Tutorials 66
  Close Box button 62
  Desk Accessories 60
  main Tutorials Screen 61
  Printing 65
  read a chapter 61
  sizing boxes 62
  sliders 62
  text display of a chapter 62
  The Tutor 61
  The Tutor menu 61
Tweening 21, 25
Two Button Box 41
Wide Style Information Box 55
  Box Specifications 56
  display information 55
  Handling procedure 56
  obscure the entire screen 55
Work Bench 25
WYSIWYG display 52, 56

GFA-VECTOR

by GFA Systemtechnik

The incredible power of the *GFA BASIC* interpreter is increased dramatically with the introduction of the three dimensional graphics program, *GFA-VECTOR*.

Now you can create and manipulate astounding images and optical effects, and place them into your own *GFA-BASIC* programs. And since *GFA-VECTOR* creates pictures written entirely in machine language, you are able to rapidly update the screen, thus allowing the creation of real time animations!

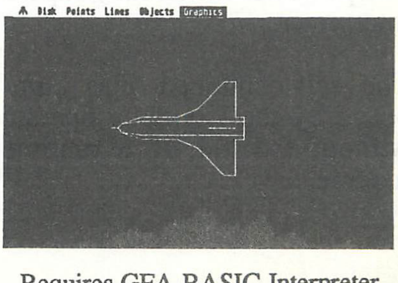
GFA-VECTOR lets you create objects two ways. First, points and lines can de defined by entering their coordinates with the keyboard. Or the objects can be created interactively on the screen with the aid of the 3D-graphics editor. With this potent editor an object can be viewed from the top, front, or side, and modified effortlessly with the cursor.

GFA-VECTOR provides a concise tutorial and a number of interesting examples that make the program easy to learn and simple to use, that even an amateur artist can use it.

Using dual screen pages, *GFA-VECTOR* is able to render a virtually flicker-free image. Objects can also be overlayed. using the OR or XOR mode. Graphics can be further enlarged, reduced, or animated from within a *GFA BASIC* program.

Objects can be revolved along any of the three axes in one degree increments through any order of rotation. These objects can be scaled in 511 increments from 1/64th to 8X normal size. Each object file can have up to 1024 defining points and 1024 defining lines. Up to 32 objects can be stored in a single file.

It's time to put action and adventure into those dull computer sessions, and *GFA-VECTOR* will make your BASIC programs live and breath in ways you never dreamed possible.



Requires GFA-BASIC Interpreter

Available for the Atari ST

GFA BASIC Compiler

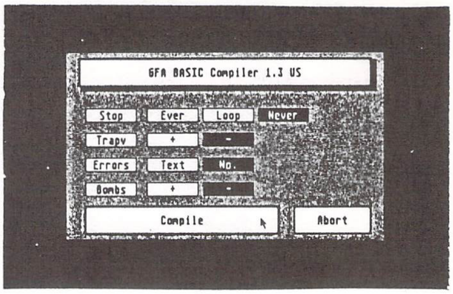
by GFA Systemtechnik

MichTron has now moved into the fast lane of the language race with an exciting new product from GFA. Presenting the GFA Compiler, a fast, 2-pass compiler which can convert your finished GFA BASIC programs into compact, faster running machine language files.

GFA BASIC has been drawing rave reviews from around the world, becoming the standard for basic programmers everywhere. Now with the new GFA Compiler you have one of the most powerful and easy to use languages ever developed.

GFA Compiler is totally compatible with the GFA BASIC Interpreter. The Compiler utilizes some of the most advanced programming techniques available to compile any GFA BASIC program in seconds, enhancing the speed and power of the already superb interpreter.

The self-contained files composed by the Compiler operate without the aid of a runtime module.



GFA Compiler utilizes all GEM features including mouse controls, windows and drop-down menus. In no time you will be converting your BASIC programs into faster running, compiled programs.

Another benefit that adds to the value of this package is that no royalties need to be paid when selling a program compiled with the GFA BASIC Compiler.

Requires GFA BASIC Interpreter

Available for the Atari ST

$79.95

GFA Companion

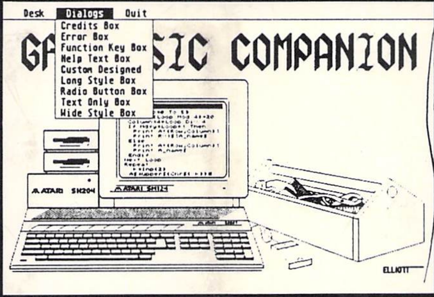
by Marathon Computer Pres

GFA BASIC owners now have a new, incredibly useful tool available that will cut the time and annoyance often (if not always) associated with programming in GEM with BASIC. Already the possessors of the best BASIC available for any computer, GFA users can now build Radio Button Boxes, Dialog Boxes, Help Text Boxes, Sliders, Error Boxes, and more with The GFA BASIC Companion.

The GFA BASIC Companion Dialog Box Source Code Generator produces quick Dialog Boxes easily so even the newest programmers can give their programs an elegant, professional look, but the Custom Design Option gives the total control and creativity necessary to satisfy the hungriest of "power users".

You'll find that by using several of the GFA BASIC resident commands you can create a versatile user interface that will mimic routines available in the GEM AES, but in an adaptive format that is easier to learn and change.

These objects, are stored in GFA BASIC's ASCII.LST file format so they can be studied, modified, or merged into a GFA BASIC program. Altering one of these routines is simply a matter of loading the .BAS or .LST file into the GFA BASIC Editor and changing the desired portions in a real time mode that allows immediate testing of your changes. This isn't possible in a standard RCS, such as the one included in the ATARI ST developers kit. Think of the time saved, and the headaches that will be cured by such an interface.



Other attractions of The GFA BASIC Companion include an extensive online Tutorial that may be viewed from a window, or printed out for further study.

The GFA BASIC Companion is an exciting addition to your library of GFA BASIC products.

Req

MichTron
576 S. Telegraph, Pontiac, MI 48053
Orders and Information (313) 334-5700

GFA Companion
Atari ST Dialog Box Construction Program
by John B. Holder
MichTron
Copyright 1987
1422

GFA Companion
Tutorial Disk
MichTron
Copyright 1987

GFA Companion
Library Disk
by John B. Holder
MichTron
Copyright 1987
1422