FONTZ!™

NEOCEPT, Inc.

FONTZ!™

by *NEOCEPT, Inc.*

Table of Contents

Technical Support Policy*

NEOCEPT will provide unlimited free telephone technical support for all registered owners of FONTZ! In order to qualify for this, you must send in your warranty card, with all information included.

We regret that we are unable to provide an unlimited toll-free technical support telephone line, but it simply is not possible to do so and still maintain our policy of low priced software. However, if you do not wish to telephone, be assured that we will respond to any and all written requests for help from registered owners as quickly as possible.

In the interest of saving both ourselves and the user needless expense and trouble, we ask that you please restrict your questions to subjects which are either not specifically covered in this manual or which are not covered in sufficient detail. In the event that the answers to your questions are contained in the manual, you will be simply be directed to the proper section. So for your sake and ours, please read the manual thoroughly.

Also, please have your FONTZ! manual handy, along with your FONTZ! diskette, when you call. If possible, please have your computer booted up with FONTZ!, or be ready to run the program while talking to us.

NEOCEPT, Inc.
547 Constitution, Unit A
Camarillo, CA 93010

Technical Support Line: (805) 482-0313
Other information: (805) 482-4446
Monday-Thursday, 10:00am to 5:00pm PST.

* subject to change without notice.

Acknowledgements

I'd like to offer some very special thanks to some very special people. Without their help, this program might not have been possible.

First I'd like to thank Bernice Fulton, my mother. She originally loaned me the money I needed to purchase my Atari ST development system, way back in the olden days of Summer 1985, and has been very patient in waiting for me to pay her back. Thanks Mom!

I'd also like to thank Roger & Beverly Hillman, of Cal Com. I used to work at Cal Com's Southern California retail store, and they were often very helpful in many ways. Thanks Guys!

Jerome Broekhuijsen, Doug Keller, and Matthew Stern all deserve thanks for being kind enough to aid me in my research into the Macintosh font file format. Without their help, this very important feature of Fontz! might not have made it into the program.

I'd like to thank Harjit Singh for his general encouragement and friendship, and his occasional help. Thanks a bunch, Haji-Com!

I'd like to thank the officers and general membership of the Atari Computer Association of Orange County. This is an Atari users group I've been lucky enough to be associated with for about 4 years now. Many people in the club made helpful suggestions of some kind or another to me regarding this program.

Finally, of course, I'd like to thank my co-workers here at NEOCEPT, especially Shelby Moore III.

Thanks, Everyone!
Mike Fulton

Copyright Notice

NEOCEPT is taking a large risk in offering high-end products for the Atari ST at a fraction of the cost of similar products for other computers. We hope that by pricing these products to fit more people's budgets, we will sell more of them, allowing us to make up the difference with increased sales. This philosophy of bringing new technology to everyone by lowering the price and inproving the ease of use can only continue if we have the support of all of you Atari ST enthusiasts. To put it simply, if you illegally distribute or accept copies of this program, you are removing NEOCEPT's incentive to produce and upgrade the software products you need on the Atari ST. Remember, the Atari ST community is like a living organism, and for it to stay healthy and grow, we must all do our part. Please take note that we do not say this because we think that all Atari users are software pirates. We doubt very much if the percentage is much different than for any other computer, but since the Atari ST market is smaller, each individual pirate has a bigger impact. Help us help you -- be honest.

In the event you should sell your original copy of the FONTZ! program and documentation, you must either destroy all duplicate copies of the FONTZ! software and documentation, or give them to the new owner.

Shelby Moore III-President

Warning to Software Pirates

This program represents a lot of time and effort on the part of the programmer, and to sell or give away copies of this program is the same as stealing hard-earned money out of his pocket. If that fact, and the fact that it is illegal, doesn't deter you from distributing copies of FONTZ!, you may wish to know that the programmer is a large, strong man who will get quite angry and possibly very violent if he encounters someone with an illegal copy of his program. You might think, "He can't hit me or anything, that would be against the law!" My response to that line of thinking is that it's an convenient time to be worrying about the law. Think twice. Some bad things can happen when you break the law. Other things can get broken, too.

Introduction

Congratulations on your purchase of **FONTZ!**, a GEM Font Editor for the Atari ST computer. **FONTZ!** is designed to allow you to create and modify GEM text font files, as well as convert fonts into GEM format from other formats. These fonts can then be used with any Atari ST program which allows the use of standard GEM fonts.

As you read this manual, keep in mind that we have tried to keep things at a level where everyone will be able to keep up. However, if you are a beginner on the Atari ST or on computers in general, then you will probably run into things which you do not understand, or terms and expressions that you have not heard before. When this happens, you should please be patient. It is often very difficult to explain things in a way that puts each concept in order. Please just read on, and somewhere farther into the manual things should become more clear. Very often, it is not even necessary to completely understand something to be able to use that part of the program. Think of it as a math problem with several variables, until you get all the way to the end, you might not know what they all mean, but at the end, you will.

What is GEM?

Simply put, GEM is a program which is built-into your Atari ST computer. It is responsible for keeping track of windows, drop-down menus, dialog boxes, and graphics (including text) on the screen or other devices such as a printer. The ability of GEM that we are most interested in is that it can use a number of different sizes, styles, and typefaces of text characters. This text output ability provides a great deal of flexibility and power. The purpose of **FONTZ!** is to allow you to create and modify text characters to be used by GEM.

What is a Font?

The term *font*, and some other terms, are very often used incorrectly within the computer world when compared to what the same terms mean in the typographic world. So before we get too far along, let's define a few of the terms which relate to different styles and sizes of text, so that everyone will be sure that we are talking about the same thing.

The first term is *typeface*. This term is used to refer to any number of different-sized sets of letters which have a common design to all of the lines and curves that make up each character. A few examples of different typefaces are shown below.

This is Dutch This is Thames
This is Swiss This is Typewriter

The term *typestyle* is sometimes used in place of typeface, but this is not actually correct. The term typestyle actually refers to such special variations of a typeface, such as **boldface**, or *italics*. The different styles of text which GEM can print text with are shown below. (To keep from having to have a complete set of characters in each style, GEM creates these styles from the original style of the characters.) Typestyles are also sometimes refered to as *text effects*.

Plain Text Style **Boldface** *Italics/Skewed*
Underlined Outlined Light

A *font* is a complete set of characters of a given typeface in a single size and style. A *font family* is a set of different sized fonts of the same typeface and typestyle.

That should be enough to get you started. But if at any time in this manual we use a term which you do not understand, look in the index at the end of the manual to see if there are any other references to that term.

What Features Does FONTZ! Have?

One of the most important features of FONTZ! is the ability to load font files from the Apple Macintosh and Commodore Amiga computers and convert them into GEM format. Both computers, especially the Macintosh, have many different text fonts available, and now you can use those fonts on your Atari ST.

FONTZ! also allows Atari ST users to convert font files from several non-standard font file formats used on the ST. You can load and convert font files created for the original version of the DEGAS paint program, from Batteries Included. You can also load and convert font files which are in the format used by Hippopotamus Software for their HippoWord word processor. Finally, you can load font files which are used by the paint programs N-Vision, from Audio-Light Software, and Paintworks, from Activision, and convert them into standard GEM format. Of course, you can also load and edit fonts which are already in GEM format.

Other features of FONTZ! include the ability to scale fonts to different sizes and GEM devices. You have the ability to cut and paste entire characters, or even just parts of them! You can even merge all or part of a character into another character. FONTZ! allows you to control every single aspect of the font being edited, from its size to the special effects information used by GEM to create style variations like boldface and underlining. There are even several special drawing tools for drawing lines, circles, boxes, and more, to aid you in creating your characters. All this and more!

Make a Backup Disk!

FONTZ! is not copy-protected, for your convenience, and we suggest that the first thing you do is make a backup of your original FONTZ! disk. Refer to your Atari ST manual if you need instructions on how to do this. Once you are done, you should put your original FONTZ! disk away in a safe place and work only from your backup copy. Please do not abuse our policy of non-copy protected disks.

FONTZ! & Hard Disk Drives

If you own a hard disk drive, you will probably want to put FONTZ! onto it. You must copy the files **FONTZ.PRG**, **FONTZ.RSC**, and **SAMPLE.FNT** to the hard disk. Although FONTZ! will work from any disk drive and folder, I suggest putting the program in the same directory as your GEM font files. This will make things easier to work with.

The README.DOC File

Because some things can change between the time this manual was produced and the time when you actually purchase FONTZ!, there will be a file called **README.DOC** on your FONTZ! diskette which will contain any last minute information about the program. To read this information, simply load the file into your favorite word processor, such as **NEOCEPT's WordUp**, or print it to the desktop or your printer by double-clicking on the icon in the window, and then choosing "Show" or "Print" in the alert box that appears.

Loading FONTZ!

First start out by inserting your FONTZ! disk into one of your disk drives. If you have installed FONTZ! onto a hard disk drive, turn it on and let it come up to speed. After your disk drives are turned on, turn on your computer. If it was in drive A:, the FONTZ! disk will automatically open a window when the GEM desktop comes up. The FONTZ! program is named

FONTZ.PRG on your disk, so simply double-click on the icon with that name, and the program will load into memory and execute. If you are using a hard disk, then open a window for the drive and folder where you have put FONTZ!, and then double-click on the FONTZ! icon. (Of course, most of you have figured all of this out by yourself, and we didn't need to tell you any of it.)

FONTZ! is designed to be used only in medium or high resolution. If you try to load the program in low resolution, you will get a message telling you that it can't be done. Using high resolution on the monochrome monitor is the preferred method of using FONTZ!, but medium resolution also works quite well. If you have both monitors available, we suggest you use the program in medium resolution only when working on fonts designed for the medium resolution screen.

FONTZ! And GDOS

In case you don't know already, let me tell you that GDOS is the portion of GEM which is responsible for allocating areas of the computer's memory for font data and GEM device driver programs (which tell the computer how to access a device, such as a printer or the display screen) which will be loaded from disk, as well as making sure GEM graphics commands get to the correct device. This is all it does, and it is not actually responsible for using fonts once they are loaded. Currently, the version of GEM which is built into the Atari ST does not include a complete implementation of GDOS. It does not allow for loading either fonts or device drivers from disk, but simply handles screen graphics functions using only the ST's built-in fonts. In order to have the full capabilities of multiple fonts and devices through GEM, GDOS must be loaded from the AUTO folder of your boot disk at system start-up time.

For the most part, FONTZ! does not require that GDOS be loaded into your Atari ST. Most of the functions of FONTZ! will work in the same way regardless of if GDOS is present or not. However, there are a few functions in the program which do require that GDOS be loaded for them to work. And some other functions will not work at full power without GDOS loaded into the computer. These functions will have notes to this effect in their descriptions in this manual.

Getting Started

Once FONTZ! is loaded into memory, it will open its windows and put a title box on the screen. Simply click either of the mouse buttons anywhere on screen and this box will go away. Now you are ready to begin.

Your First Time With FONTZ!

The very first time you load FONTZ!, you may want to start out by exploring through the menus in order to gain a minimal level of familiarity with the program. Let's load a font into the program so that you have something to play with. Start out by moving the mouse up to the FILE menu. Click on the **Load GEM Font** option. A standard GEM file selector box will appear on the screen. Find the file named **SAMPLE.FNT** and select it by double-clicking the mouse on its name, or by typing in the name and hitting the **[Return]** key.

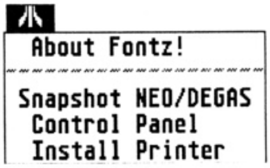
Once the sample font has been loaded, the program will display it, with the characters all in one long row, in the top window on your screen. Since the font is too large to fit in the window all at once, you may use the scroll bars and arrows of the window to look at different parts of the font.

Now you may want to simply take a tour of FONTZ! to see all of its features. Go through the menu choices and get yourself familiarized with all of the program's features. Have Fun!

FONTZ! Menus

Each of the many choices available in the FONTZ! menu bar are described in this section. Please note that some of the functions in the menus can also be performed by pressing a certain special key, for your convenience. In these cases, the key is named along with the menu entry. Also, some keyboard-only commands are described in the appropriate places.

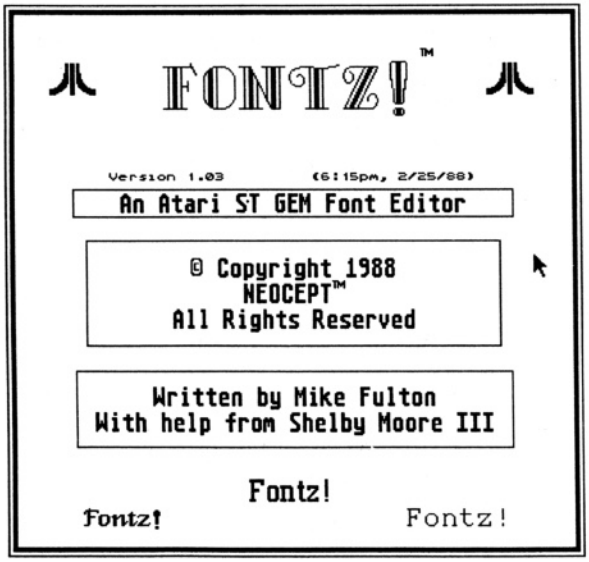
Desk Menu



This menu is at the top left corner of the screen. Any GEM desk accessories which are loaded into your system are accessible through this menu.

About FONTZ!

Choosing this menu choice will display the FONTZ! title box on the screen, as shown below. Simply click either mouse button and it will go away.



FONTZ!™

Version 1.03 (6:15pm, 2/25/88)

An Atari ST GEM Font Editor

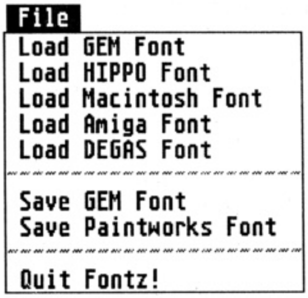
© Copyright 1988
NEOCEPT™
All Rights Reserved

Written by Mike Fulton
With help from Shelby Moore III

Fontz!
Fontz! Fontz!

File Menu

This menu contains all of the disk file operations which are available in FONTZ!



A Warning About Loading Font Files:

Do not attempt to load font files in other formats besides what the menu choice indicates, as this can cause the system to crash. Fontz! will always attempt to insure that the font is in the expected format before actually using it, however, it is possible for something to sneak through.

File Not Saved Warning

If you attempt to load a font file, either one already in GEM format or one in another file format, and your presently loaded file has not been saved since you last changed something, then an alert box will warn you that you haven't saved your font, as shown in figure #1. At this point, you can choose to load the new font anyway, or you can cancel so that you can save the font.

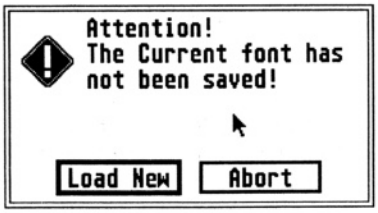

Attention!
The Current font has
not been saved!

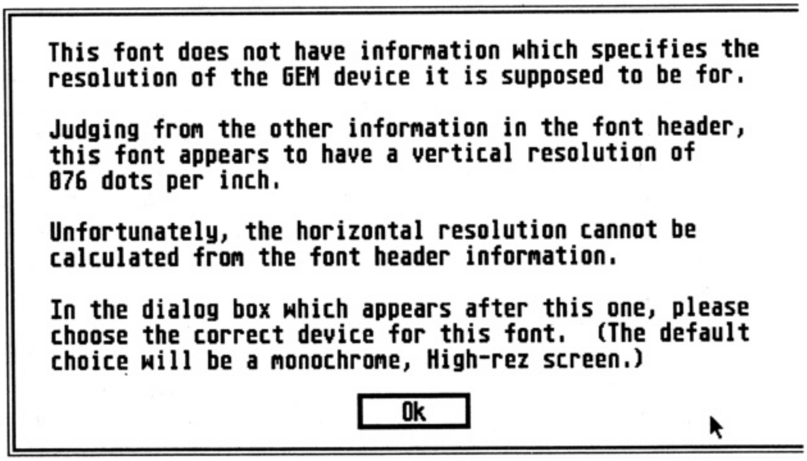
Figure #1, Font Not Saved Alert Box

Load GEM Font -- F9

This option lets you load either GEM fonts or fonts saved in the N-Vision/Paintworks format. Since these two formats are very similar to each other, the FONTZ! program is able to determine which of these formats the font is in and acts accordingly.

The user should keep in mind the fact that each GEM device, such as the monochrome and color display screens, or different types of printers, has its own resolution, and the font files for that device must match that resolution in order to produce correct results. GEM fonts created or edited and then saved with FONTZ! include information to indicate the resolution of the device the font is designed to be used with.

When you load a GEM font which does not have this information in it already, FONTZ! will inform you with a dialog box, as shown in figure #2.

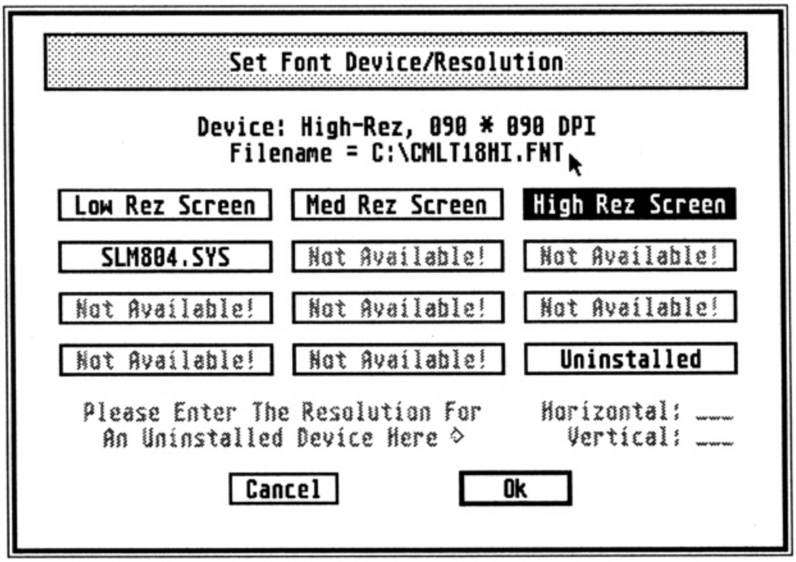

This font does not have information which specifies the resolution of the GEM device it is supposed to be for.

Judging from the other information in the font header, this font appears to have a vertical resolution of 076 dots per inch.

Unfortunately, the horizontal resolution cannot be calculated from the font header information.

In the dialog box which appears after this one, please choose the correct device for this font. (The default choice will be a monochrome, High-rez screen.)

Figure #2, Font Resolution Information Not Found Dialog Box



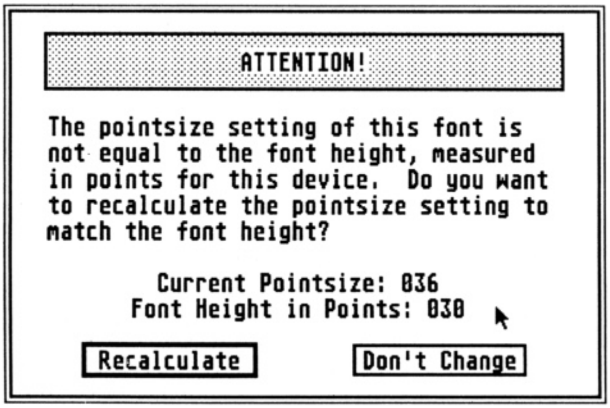
Device: High-Rez, 090 * 090 DPI
Filename = C:\CMLT18HI.FNT

Please Enter The Resolution For An Uninstalled Device Here ◇ Horizontal: ___ Vertical: ___

Figure #3, Set Font Device Dialog Box

After this, you will be asked in another dialog box to tell FONTZ! what the correct device is for this font, as shown in figure #3. The default device will be the monochrome screen. If this is not correct, and the correct device is shown as one of the choices, simply choose that device. For more information on this dialog box, see the **Set Font Device** menu choice description.

If the pointsize setting does not match the size of the font for the device you have chosen, then another dialog box will appear and show you the height of the font in points, and the current pointsize setting, and ask you if you want to reset the pointsize setting to match the font height, as measured in points. This is shown in figure #4.


ATTENTION!

The pointsize setting of this font is not equal to the font height, measured in points for this device. Do you want to recalculate the pointsize setting to match the font height?

Current Pointsize: 036
Font Height in Points: 030

Figure #4, Recalculate Pointsize Dialog Box

Load Hippo Font

This option lets you load font files which use the format used by the programs HippoWord and HippoPixel, from Hippopotamus Software, and automatically converts them into standard GEM format.

FONTZ! will always convert fonts from Hippo format into monochrome screen fonts. You can use the **Scale Font To A Different Device** function to create fonts for other devices once you have converted the font. See the section of this manual titled **Converting Hippo Fonts** for more information on this subject.

Load Macintosh Font

This allows you to load font files from the Apple Macintosh and automatically convert them into standard GEM format. However, you should know that there are a few conditions on your abilities to do this.

First of all, there are two kinds of font files commonly used on the Apple Macintosh, screen fonts and LaserWriter fonts. A

LaserWriter font file is basically a set of small programs written in the Postscript page description programing language, and is designed to be used with the Apple LaserWriter and other Postscript Laser Printers. At this time, FONTZ! cannot load this type of font.

Screen fonts can be loaded by FONTZ! with just one restriction: The font file must contain only one size of one typeface. Let me explain what I mean by that. Fonts on the Macintosh are kept in Macintosh Resource Files, which are very similar to the resource files used by GEM programs on the ST. However, these Macintosh Resource files can contain a few more different types of data than their GEM counterparts. Besides data such as icons, menus, or dialog boxes, these resource files are also used to contain Macintosh fonts. To be used with FONTZ!, Macintosh Resource files which contain more than one font or other types of information must first be separated into smaller files with only font information for one size and typeface each.

FONTZ! will always convert fonts from Macintosh format into monochrome screen fonts. You can use the **Scale Font To A Different Device** function to create fonts for other devices once you have converted the font. ***Note:*** See the section of this manual entitled **Converting Macintosh Fonts** to get more information about this subject.

Load Amiga Font

This option lets you load standard format Amiga font files and automatically convert them to GEM format.

FONTZ! will always convert fonts from Amiga format into monochrome screen fonts. You can use the **Scale Font To A Different Device** function to create fonts for other devices once you have converted the font. ***Note:*** See the section of this manual entitled **Converting Amiga Fonts** for more information on this subject.

Load DEGAS Font

This option allows you to load font files created with the font editor which was included with the original version of the DEGAS paint program, from Batteries Included, and automatically convert them into GEM format.

This option applies ONLY to fonts designed for use with the original version of DEGAS. Fonts which have already been converted with the font converter program included with the newer version, entitled DEGAS Elite, are in standard GEM format already, and should be loaded with the **Load GEM Font** menu option. As a general rule, older-style DEGAS fonts will have a file size of about 2050 bytes.

FONTZ! will always convert fonts from DEGAS format into monochrome screen fonts. You can use the **Scale Font To A Different Device** function to create fonts for other devices once you have converted the font. ***Note:*** See the section of this manual entitled **Converting DEGAS Fonts** for more information on this subject.

Save GEM Font -- F10

This option saves the font currently being edited as a standard format GEM font. In order to use this font, you must include the filename in the ASSIGN.SYS file and reboot your system with GDOS. (See the section of this manual titled **Using GEM Fonts.**)

Although GEM font files can legally use any valid TOS filename, Atari has come up with some general rules to follow in creating font filenames. The idea is that a standard way of naming font files will make them easier to deal with. The format for a font filename using the guidelines laid down by Atari is shown below.

AT*xxyyzz*.FNT

The 'AT' portion of the name indicates that the file is an Atari font. The '*xx*' portion of the name is an abreviation of the font's typeface name, such as the examples shown below. Also, sometimes both the 'AT' and 'xx' portions are used together to have an abreviation of the typeface name.

SS -- Swiss TR -- Dutch (Times Roman) TP -- Typewriter

The '*yy*' portion of the filename is used to indicate either the font's pointsize for proportional fonts, or the pitch (the number of characters which fit in one inch horizontally) for fonts which are monospaced, and must be an integer number ranging from 00 to 99.

The '*zz*' part of the filename is used to indicate which GEM device the font is supposed to be used with. Some current GEM devices have the following codes (Resolution in dots per inch horizontally and vertically shown in parenthesis):

"EP"	Epson FX/Compatible Printer	(120*144)
"SP"	Star NB/Compatible Printer	(180*180)
"NP"	NEC P6/P7/Compatible Printer	(360*360)
"LB"	Atari SMM804 Printer	(160*72)
"LS"	Atari SLM804 Laser Printer	(300*300)
"ME"	Atari GEM Metafile Device	(254*254)
" "	High or Low Resolution Screen	(90*90)
"LO"	Low Resolution Screen	(45*45)
"CG"	Medium Resolution Screen	(90*45)
"HI"	High Resolution Screen	(90*90)

Of course, these are just general guidelines, and you are not restricted to using them. If you want to use other rules for naming your fonts, go right ahead. However, if you use your own filename rules, you should keep a list with all the information about all of your fonts, and include this list if you distribute fonts that you name in your own way.

Save Paintworks Font

This option saves the font currently being edited in the format used by the Paintworks and N-Vision paint programs. You should note that not all fonts created or edited with FONTZ! can be used with these programs, however. Both programs allocate only a limited amount of memory to use to hold a font. Some fonts, such as those larger than 36 point, may require more memory than these programs make available. Unfortunately, there is no way to find out if a font is too large to be used with these programs except to try it.

Quit

This option allows you to leave the FONTZ! program, and return to the GEM Desktop. If you have not saved your work, then before actually leaving, you will be shown an alert box telling you so, as shown in figure #5. If you have not saved your work, you should cancel the QUIT and do so now, and then quit. If your work is saved, then the program will exit immediately.

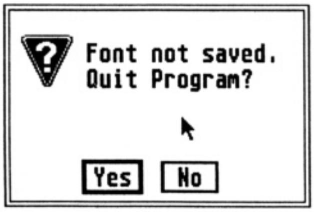
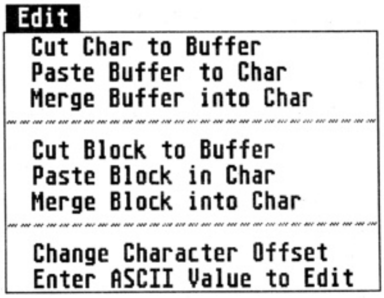

Font not saved.
Quit Program?

Figure #5, Font Not Saved Warning Alert Box

Edit Menu

This menu contains all the options in FONTZ! for cutting and pasting blocks or entire charcters, as well as options for changing the character being edited and the offset value of the currently edited character.



Cut Char to Buffer -- F1

This option copies the entire character currently being edited into a copy buffer. This can then be copied into another character, replacing it altogether. Or, it can be pasted into a character as a block. When you cut a character, a message will inform you that it has been saved in the cut buffer.

Paste Buffer to Char -- F2

This option copies the contents of the copy buffer into the character currently being edited. By using the cut option to cut a character to the copy buffer, and then this option to paste it to another character, you can copy characters to one another. The width of the character is always reset to the width of the copy buffer.

Merge Buffer Into Char -- F3

This option merges the contents of the copy buffer with the character currently being edited. The resulting character will have pixels set where either the original character or the copy buffer had pixels set. The width of the character is left unchanged if the character is wider than the contents of the copy buffer. If the copy buffer is wider than the character, the character is expanded to contain the entire copy buffer.

Cut Block to Buffer -- Shift F1

This allows you to copy a rectangular block of the character into the copy buffer. You move the mouse to the top left corner of the block, and then click the left mouse button. While holding the mouse button down, you move the mouse to the bottom right corner of the block, and release the button. While you move the mouse, the screen will show a rectangle outline to indicate where the block is located. If you decide to cancel, press the right mouse button and hold it down before you release the left mouse button.

Paste Block In Char -- Shift F2

This allows you to paste the contents of the copy buffer back into the character currently being edited. You move the mouse to the box of the character where you want the top left corner of the block in the copy buffer to go into the character, and click the left mouse button. Clicking the right mouse button cancels the operation.

If the the block contained in the copy buffer is wider than the area available between the box you have chosen and the right edge of the character, you will be asked if you want to cut off the block at the edge of the character, or if you want to make the character wide enough to accommodate the block. Simply choose whichever is appropriate for your needs.

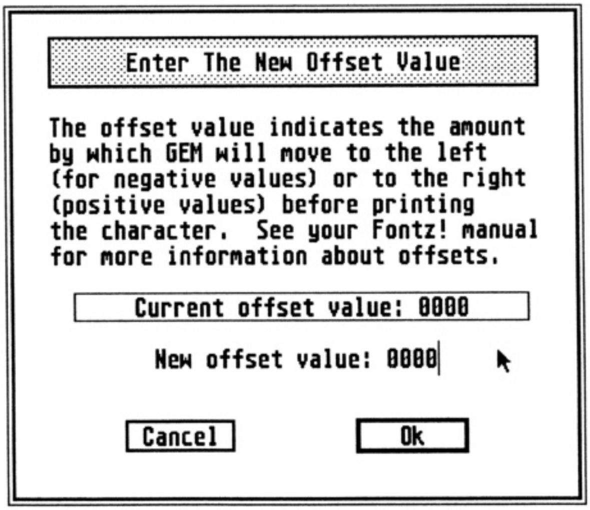
Merge Block Into Char -- Shift F3

This option works exactly the same as the **Paste Block Into Char** option, except that it merges the block in the copy buffer into the character, instead of replacing that portion of the character with the block. The resulting character will have pixels set where either the original character or the copy buffer had pixels set. Any pixels which were clear in both the original character and the copy buffer will be left clear.

If the the block contained in the copy buffer is wider than the area available between the box you have chosen and the right edge of the character, you will be asked if you want to cut off the block at the edge of the character, or if you want to make the character wide enough to accommodate the block. Simply choose whichever is appropriate for your requirements.

Change Character's Offset Value -- Alt-O

This brings up a dialog box, as shown in figure #6, which shows you the current offset value for the character currently being edited, and allows you to enter a new offset value. If the font being edited does not have a offset table enabled, this option is disabled. (The offset value indicates how far to the left or right to move before actually printing the character. See the ***GEM Font Definition*** section for more information about offsets.)


Enter The New Offset Value

The offset value indicates the amount by which GEM will move to the left (for negative values) or to the right (positive values) before printing the character. See your Fontz! manual for more information about offsets.

Current offset value: 0000

New offset value: 0000

Figure #6, Change Character's Kern Value Dialog Box

Enter ASCII Value to Edit -- F7

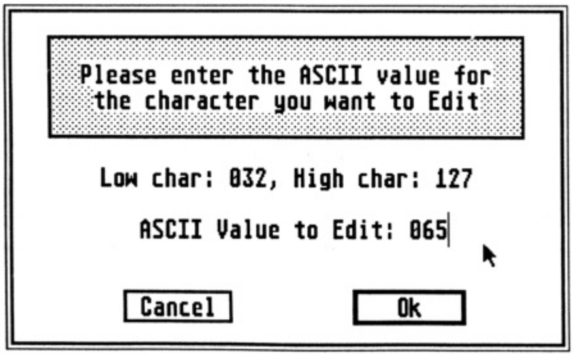
This option displays a dialog box, as shown in figure #7, which displays the lowest and highest characters in the font, and allows you to type in the ASCII value of the character that you want to edit. If you click on the **OK** button, the edit display will switch to the new character.

You can also choose characters with ASCII values ranging from 0 to 127 by simply pressing the corresponding key on the keyboard. Characters with ASCII values outside of this range must be selected with the menu choice or with the following method.

You also have the option of selecting a character to edit by double-clicking on it in the font display window. However, if a character has a width of zero, it can not be selected in this fashion unless it is the last character in the font or unless it is the first of several characters with zero width at the very end of the font.

Edit Next Lowest Character -- Shift F9
Edit Next Hightest Character -- Shift F10

These keystroke commands save the changes to the current character, and then move up or down to the next character in the font to place it in the edit buffer.


Please enter the ASCII value for
the character you want to Edit

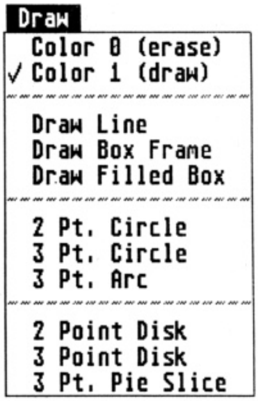
Low char: 032, High char: 127

ASCII Value to Edit: 065

Figure #7, Enter ASCII Value of Character To be edited.

DRAW Menu

This menu contains all of the drawing functions which are available in the FONTZ! program. The use of each function is described below.



Color 0 (erase)

This function sets the current drawing color used for all drawing operations to 0, or erase mode. A checkmark will appear in the menu next to this choice to indicate if it is active.

Color 1 (draw)

This function sets the current drawing color used for all drawing operations to 1 or draw mode. A checkmark will appear in the menu next to this choice to indicate if it is active.

Draw Line

Draws a straight lines between two points. After choosing this option, you move the mouse into the edit window. First you click with the left mouse button on one point and hold it down, and then a rubber line appears on screen until you move the mouse to the 2nd point and release the button. When you release the button, a line is drawn between the two points. To cancel, press the right mouse button before releasing the left button.

Draw Box Frame

This option draws an outline, or frame, of a box, instead of a filled one. Upon choosing this option, you will be shown a message telling you what to do, as shown in figure #8. You click the mouse on the top left corner of the box, and hold the left mouse button down until you have moved to the bottom right corner. When you release the mouse button, lines are drawn in the appropriate drawing color between all neighboring corners to make a box frame. To cancel, press the right mouse button before releasing the left button.

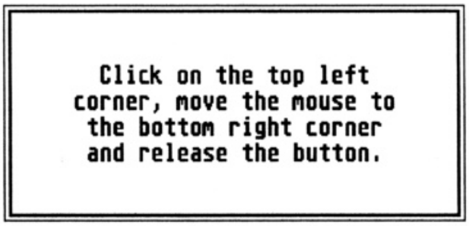

Click on the top left
corner, move the mouse to
the bottom right corner
and release the button.

Figure #8, Block Definition Message Box

Draw Filled Box

This option works like the Box Frame option, except that the box is filled in, instead of being an outline. The box will be cleared or filled, according to the current drawing color.

2 Point Circle

This option will allow the user to draw a circle within his character. You will be shown a message, as seen in figure #9, telling you what to do. Simply click on the centerpoint with the mouse and hold the left mouse button down. Now when you move the mouse a circle outline will be drawn to indicate where the circle will be drawn. The centerpoint of the circle may be anywhere within the character, and the circle itself may as big as can fit on screen. When you release the mouse button, the circle will be drawn in the current drawing color. Only those portions of the circle which would be inside the character will be put into the circle. To cancel, press the right mouse button before releasing the left button.

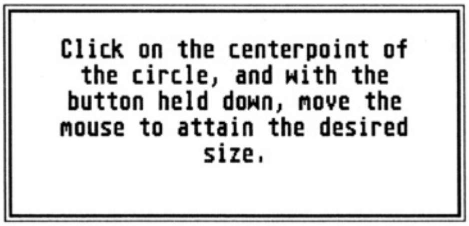

Click on the centerpoint of the circle, and with the button held down, move the mouse to attain the desired size.

Figure #9, 2 Point Circle Instructions Message

3 Point Circle

This option will draw a circle defined by 3 points along the circumference. A message box, as shown in figure #10, will instruct you to click on 3 points within the character which fall somewhere on the outside perimeter of the circle, in counter-clockwise order. A marker will be placed on screen to indicate each point as it is entered. After all 3 points have been entered, the program will compute the radius and centerpoint and draw the circle using the current drawing color. To cancel, press the right mouse button before entering the third point.

3 Point Arc

This draws an Arc between three points. You will be prompted to click on 3 points within the character which fall somewhere on the arc. A marker will be placed on screen to indicate each point as it is entered. After all 3 points have been entered, the program will compute the radius and centerpoint and draw the arc intersecting all three points, using the current drawing color. To cancel, press the right mouse button before entering the third point.

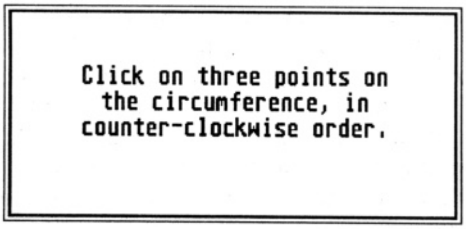

Click on three points on
the circumference, in
counter-clockwise order.

Figure #10, 3 Point Arc/Circle Instructions Message Box

2 Point Disk

This option is basically the same as the 2 Point Circle command, except that the interior of the circle is filled in, using the current drawing color, instead of being outlined only. To cancel, press the right mouse button before releasing the left button. This function can be slow, so please be patient. Hitting the right mouse button while drawing the disk will cancel, but the portion of the disk which has already been drawn will still remain.

3 Point Disk

This is basically the same as the 3 Point Circle command, except that the interior of the circle is filled in, using the current drawing color, instead of being outlined only. To cancel, press the right mouse button before entering the third point. This function can be slow, so please be patient. Hitting the right mouse button while drawing the disk will cancel, but the portion of the disk which has already been drawn will still remain.

3 Point Pieslice

This option is very similar to the 3 Point Arc option, except that the area between the arc and the centerpoint of the circle of which the arc is a part will be filled in, like a slice of pie. You will be prompted to click on 3 points within the character which fall somewhere on the arc. A marker will be placed on screen to indicate each point as it is entered. After all 3 points have been entered, the program will compute the radius and centerpoint and draw the arc intersecting all three points, using the current drawing color. To cancel, press the right mouse button before entering the third point. This function can be slow, so please be patient. Hitting the right mouse button while drawing the disk will cancel, but the portion of the disk which has already been drawn will still remain.

Fast Drawing Mode -- Shift F4

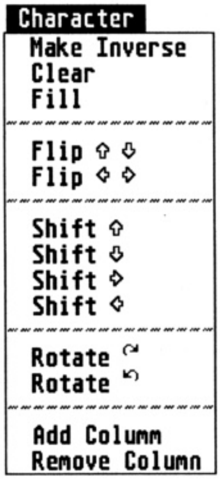
This is not a menu choice, but pressing this keystroke will turn off and on the instruction boxes described in this section for the various drawing functions. When you have gained some familiarity with how the drawing operations work, you can turn off the instruction boxes so that you do not have to wait for them to go away each time you choose a drawing function.

A Note Regarding Circle & Arc Draw Functions:

Because the screen locations representing the character in the editing grid must be scaled down to the appropriate locations within the character, and because the arc and circle drawing commands require several mathematical calculations to compute the centerpoint coordinates and radius values, there is sometimes a small amount of error in the points where the circles or arcs are drawn and where you wanted them to be drawn. This is unavoidable, and fortunately is usually only a pixel or two off at most. You may wish to experiment somewhat to become familiar with how the drawing routines work.

Character Menu

This menu contains a number of options for working on the individual character which is currently being edited. All of these operations, such as rotate, flip, shift, and so on, effect the entire character.



Make Inverse -- Alt-I

This option inverts all of the pixels in the the character into the opposite color. All of the pixels in the character which are set to 1 are set to 0, and all of the pixels which are set to 0 are set to 1.

Clear -- Clr/Home

This option sets all of the pixels in the character to 0, clearing the entire character.

Fill -- Shift Clr/Home

This option sets all of the pixels in the character to 1, filling in the entire character.

Flip Left-Right

This option flips all of the pixels in the character from the left side to the right side and vice versa, making the character into a mirror-image of the itself.

Flip Up-Down

This option works like the Flip Left-Right option, except in the vertical direction. All the pixels at the top are moved to the bottom, and vice versa, making the character into an upside down version of itself.

Shift Up -- Up Arrow

This option shifts all of the pixels in the character up by a specified number of rows. The very top row is moved to the bottom of the character, the next to the top row is moved to the top, and so on.. A dialog box will appear, as shown in figure #11, and allow you to enter the number of steps to shift the character. You can also change the direction of the shift by clicking on one of the arrows. Also, hitting the Up-Arrow key will shift the character by one step without calling up the dialog box.

Shift Down -- Down Arrow

This option works in a similar manner to the **Shift Up** function, except that it shifts in the down direction. Also, hitting the Down-Arrow key will shift the character by one step without calling up the dialog box.

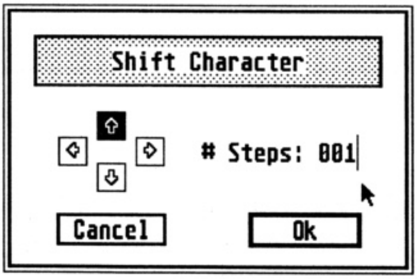


Figure #11, Shift Character Dialog Box

Shift Right -- Right Arrow

This option works in a similar manner to the **Shift Up** function, except that it shifts in the right direction. Also, hitting the Right-Arrow key will shift the character by one step without calling up the dialog box.

Shift Left -- Left Arrow

This option works in a similar manner to the **Shift Up** function, except that it shifts in the left direction. Also, hitting the Left-Arrow key will shift the character by one step without calling up the dialog box.

Rotate Clockwise -- Shift Right Arrow

This rotates the character in a clockwise direction, using the bottom left corner of the character as a base point. The top of the character is moved to the right side, the bottom to the left side. The left of the character is moved to the top, and the right of the character is moved to the bottom. If the character is not as wide as it is tall, then the top of the character is clipped off at the edge of the character when it is rotated.

Rotate Counter-Clockwise -- Shift Left Arrow

This rotates the character in a counter-clockwise direction, using the top right corner of the character as a base point. The top of the character is moved to the left side, the bottom to the right side. The left of the character is moved to the bottom, and the right of the character is moved to the top. If the character is not as wide as it is tall, then the bottom of the character is clipped off at the edge of the character when it is rotated.

Add Column -- Insert

This allows you to add a column to the character currently being edited. You will be shown an alert box, as shown in figure #12, and be asked if you want to add the column to the left or right side of the character, or if you want, you can cancel.

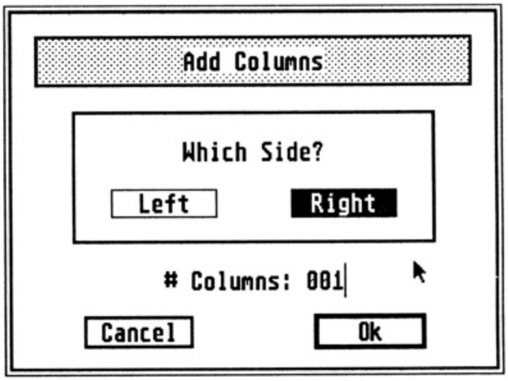
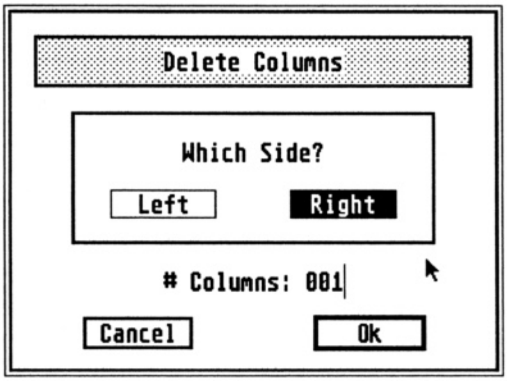


Figure #12, Add Columns Dialog Box

Remove Column -- Control-Delete

This allows you to delete a column from the character currently being edited. You will be shown an alert box, shown in figure #13, and be asked if you want to delete the column from the left or right side of the character, or if you want to cancel.

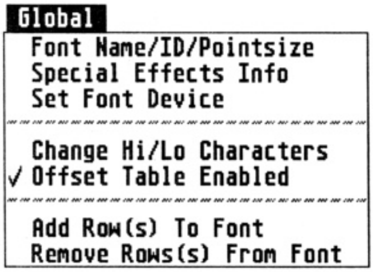


Columns: 001

Figure #13, Delete Columns Dialog Box

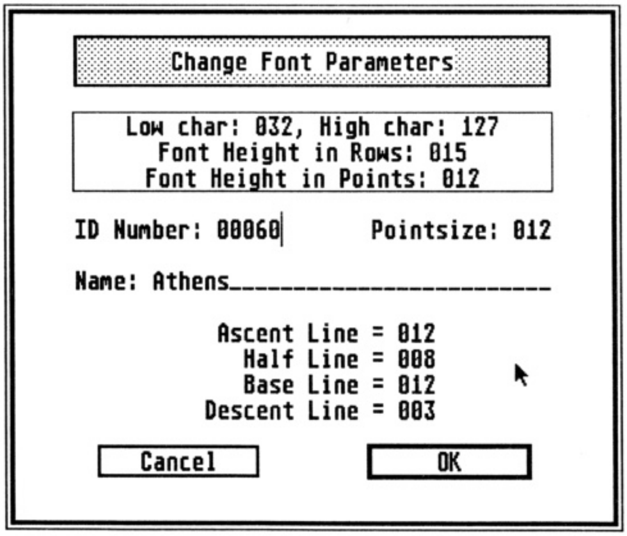
Global Menu

This menu contains options which affect the entire font which is being edited. You can change the special effects info, add or delete entire rows from the font, and more!



Font Name/ID/Pointsize -- F8

This brings up a dialog box, as shown in figure #14, which displays the settings for the font's name, ID number, and pointsize setting, as well as the font alignment line values. The range of characters is displayed for you, and the font height is shown in both rows and points.



Change Font Parameters

Low char: 032, High char: 127
Font Height in Rows: 015
Font Height in Points: 012

ID Number: 00060 Pointsize: 012

Name: Athens________________________

Ascent Line = 012
Half Line = 008
Base Line = 012
Descent Line = 003

Figure #14, Change Font ID#, Name, & Other Parameters Dialog Box

You can change the Font's ID number by typing in a different value. The font's typeface name is also shown, and can be changed. You can also change the pointsize setting to whatever you want. However, we very strongly recommend that you always have the pointsize setting match the font height, as measured in points. Otherwise, it can cause problems with some programs that expect the font height and pointsize settings to match.

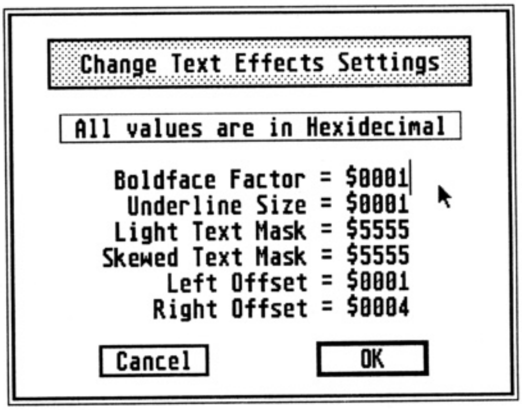
You can change the values for the font alignment lines simply by typing in new values. These are the ascent line, half line, baseline, and the descent line. (They are shown in the order they occur, as you go down from the top of the font. The baseline is measured from the very top of the font, and the other lines are offsets, in rows, from the baseline. See the section of the Fontz! manual entitled ***GEM Font Definition*** for more information on what these settings actually do.)

Special Effects Info -- Shift F8

This option allows you to set the values in the GEM font header of the font currently being edited which are used by GEM for special effects. A dialog box will appear, as shown in figure #15, and show you the current settings for the Boldface Factor, Underline Size, Skewed Text mask, and Light Text mask, as well as the right and left offset values (used for skewed text), and allow you to change them. Simply type in your desired new values. See the section titled ***GEM Font Definition*** for more information on what these settings actually do.

Set Font Device

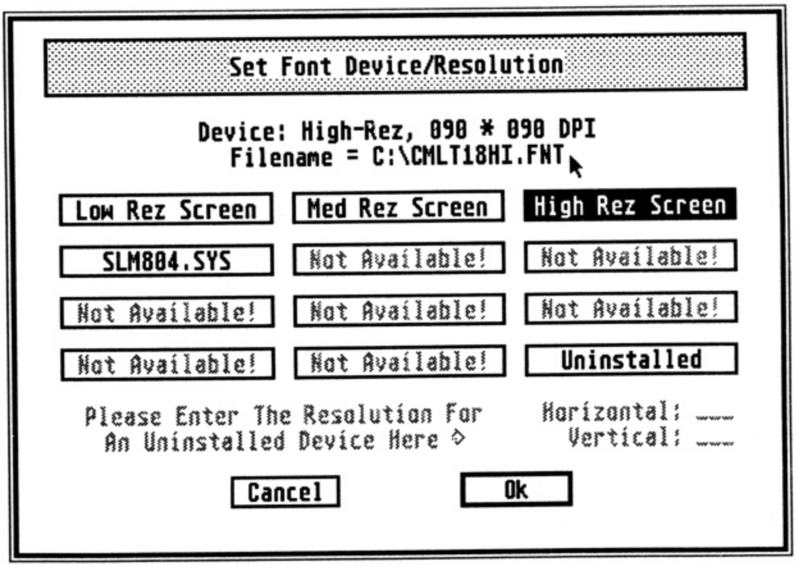
This option displays a dialog box, as shown in figure #16, and allows you to set the device for which the font being edited is intended. The name of each currently installed GEM device driver will be shown in its own separate button. The resolution of the currently selected device will be shown at the top of the box. You can also set the font to the resolution of a device that


Change Text Effects Settings

All values are in Hexidecimal

Boldface Factor = $0001
Underline Size = $0001
Light Text Mask = $5555
Skewed Text Mask = $5555
Left Offset = $0001
Right Offset = $0004

Figure #15, Text Effects Settings Dialog Box


Set Font Device/Resolution

Device: High-Rez, 090 * 090 DPI
Filename = C:\CMLT18HI.FNT

Low Rez Screen	Med Rez Screen	High Rez Screen
SLM804.SYS	Not Available!	Not Available!
Not Available!	Not Available!	Not Available!
Not Available!	Not Available!	Uninstalled

Please Enter The Resolution For
An Uninstalled Device Here ⇨

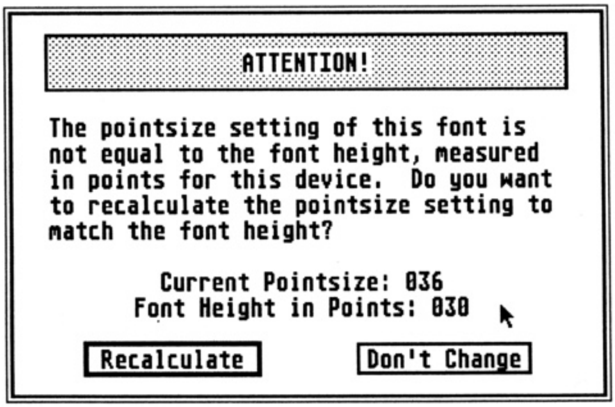
Horizontal: ___
Vertical: ___

Figure #16, Set Font Device Dialog Box

is not currently installed by clicking on the "Uninstalled Device" button and entering the resolution of that device, measured in the number of dots per inch. (Unless you click on the "Uninstalled Device" button, you are not allowed to enter the resolution.) When you are done, click on "OK" or "Cancel" to exit the dialog box.

If you choose a new device, FONTZ! calculates how many rows are required at the resolution of that device to make a font the same height as the font's current pointsize setting. If the font's height in rows does not match the result, meaning the font is too tall or too short for the pointsize setting, then another dialog box will appear, as shown in figure #17. This will tell you the calculated pointsize and the current pointsize, and ask you if you want to reset the pointsize setting to match the calculated pointsize. If you want, you can choose to leave the pointsize at the current setting. However, we do not recommend this unless you are certain that you know what you are doing.

If you change the ASSIGN.SYS file between the time you start your system and when you load FONTZ!, then the device entries in the **Set Font Device** dialog box may not be complete,


ATTENTION!

The pointsize setting of this font is not equal to the font height, measured in points for this device. Do you want to recalculate the pointsize setting to match the font height?

Current Pointsize: 036
Font Height in Points: 030

Figure #17, Match Pointsize Dialog Box

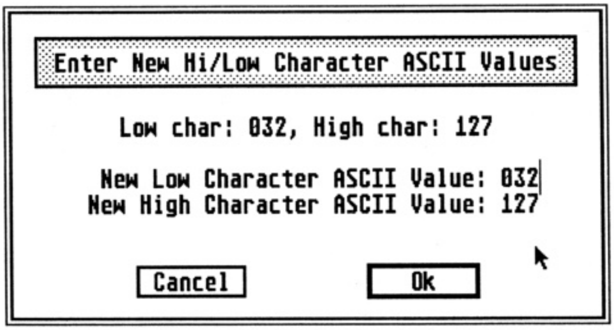
or completely accurate. Please always reset your system after changing the ASSIGN.SYS file. FONTZ! attempts to determine if the file has been changed, but this is not always possible.

Change Hi/Lo Characters In Font

This option displays a dialog box, as shown in figure #18, and allows you to change the lowest and highest characters in the font. If you change the high character to a higher value, or the low character to a lower value, then the font header is changed to indicate the new characters, but each of the new characters will have a width of zero pixels. Also, you cannot enter a low character value that is higher than the high character value, or vice versa. Doing so will cause an alert box to appear and warn you.

(The user should be aware that it will probably be necessary to use the **Enter ASCII Value To Edit** option to access some of these new characters, because of their zero pixel width.)

If you delete characters from the font, then the data is actually cleared and those characters are gone forever. It is therefore recommended that you delete characters only after saving a complete version of the font.



Enter New Hi/Low Character ASCII Values

Low char: 032, High char: 127

New Low Character ASCII Value: 032
New High Character ASCII Value: 127

Figure #18, Change High and LowCharacters in Font Dialog Box

Enable/Disable Offset Table

This option allows you to enable or disable the horizontal offset table for the font. A checkmark in the menu indicates when an offset table is enabled.

Most characters in most fonts will have offset values of zero. And in many fonts, all the characters have offset values of zero. In these cases, the offset table serves no purpose and just takes up memory and disk space, so this option allows you to enable or disable the font's offset table to avoid this. If no previous offset table exists and you enable one, a new table is generated with values of zero for each character. If the font you are editing had an offset table when you loaded it, but you later disabled it, then choosing this option again will restore it.

Add Row to Font -- Shift Insert

This allows you to add a row or number of rows to the font currently being edited. A dialog box will appear, as shown in figure #19, asking where to add the rows, and how many rows to add, from 0 to 999. Simply click on "Top" or "Bottom" and enter the number of rows. Now click on "OK" to add them, or on "Cancel". The user should note that adding or deleting rows from the font will affect the font height as measured in points. You may want to change the pointsize setting after adding or deleting rows from the font.

Delete Row from Font

This allows you to delete a row from the font currently being edited. A dialog box will appear, as shown in figure #20, asking where to delete the rows, and how many rows to delete, from 0 to 999. Simply click on "Top" or "Bottom" and enter the number of rows. Now click on "OK" to delete them, or on "Cancel". The user should note that adding or deleting rows from the font will affect the font height as measured in points. You may want to change the pointsize setting after adding or deleting rows from the font.

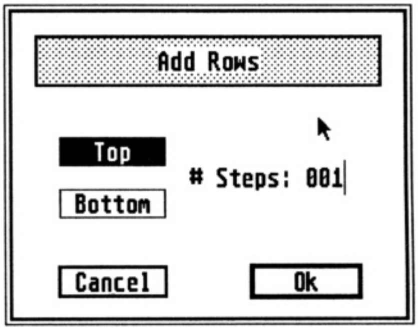


Figure #19, Add Rows To Font Dialog Box

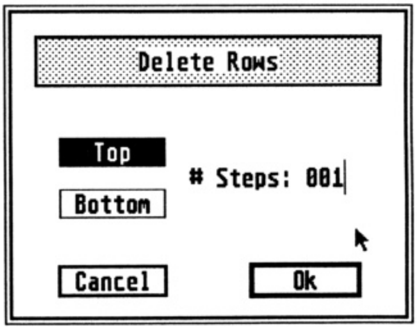
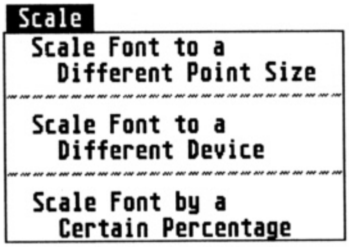


Figure #20, Delete Rows From Font Dialog Box

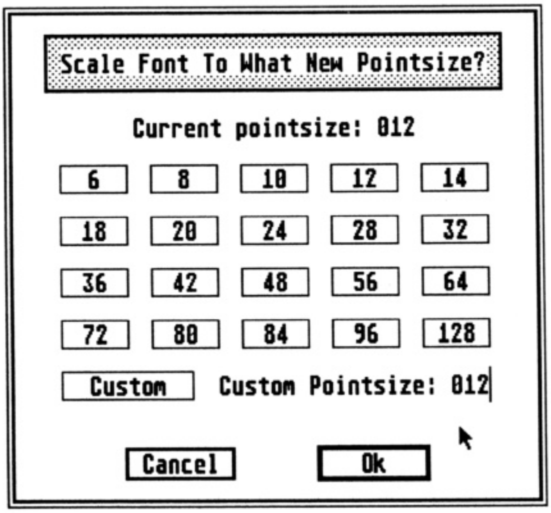
Scale Menu

This menu contains all of the special font scaling features of FONTZ! You can scale to different pointsizes, different device resolutions, or even by some arbitrary amount.


Scale
Scale Font to a Different Point Size
Scale Font to a Different Device
Scale Font by a Certain Percentage

Scale Font To A Different Pointsize -- F4

Using this function, you can scale the entire font to a different pointsize. A dialog box will appear, as shown in figure #21, and show you several choices of common sizes. You can either click on one of these sizes, or on Custom Pointsize. If you click on Custom Pointsize, then you can enter your custom pointsize and FONTZ! will scale to that size. Click on **OK** or **Cancel** to exit the dialog box.


Scale Font To What New Pointsize?

Current pointsize: 012

Custom Pointsize: 012

Figure #21, Scale Font To A Different Pointsize

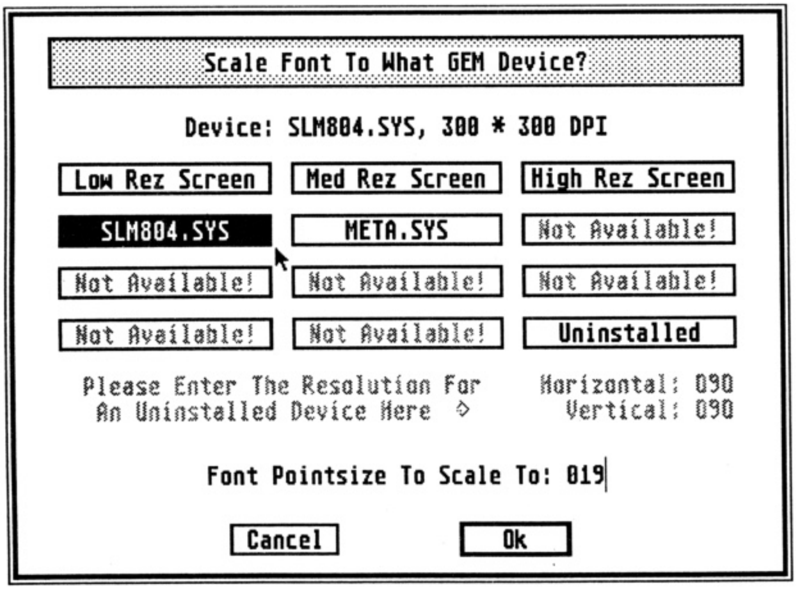
Once you have selected the new size, the font will be scaled in both the x-axis and y-axis by the ratio of the font height for the size you've selected over the current font height.

Scale Font To a Different Device -- F5

By using this function, you can scale the entire font to the resolution used by a specified device. This allows you to do things such as create a printer font from a screen font. (Or vice versa, for that matter.) A dialog box will appear, as shown in figure #22, with the current device highlighted and its resolution shown. Now you just choose the name of the device you want to scale to, and the resolution for that device will be displayed. If the device you want to scale to is not one of those listed, but you know the device resolution, then you can choose the **Uninstalled Device** button and the dialog box will change to allow you to enter the resolution for the device, for both the horizontal and vertical directions.

When you scale a font to a different device, you can also change the pointsize. This allows you to do things such as scaling a 24 point monochrome screen font to a 12 point printer font. Often, the amount of scaling can be reduced by using such methods, making the results more pleasing. If you want to change the pointsize as well as the device, simply type in the new pointsize at the bottom of the dialog box before you hit the **OK** button, otherwise simply leave it alone and the font will be scaled to the same pointsize for the new device.

When you hit the **OK** button, the font will be scaled to the correct size for the device and pointsize you have chosen. If you hit **Cancel** then the font will not be changed.



Scale Font To What GEM Device?

Device: SLM804.SYS, 300 * 300 DPI

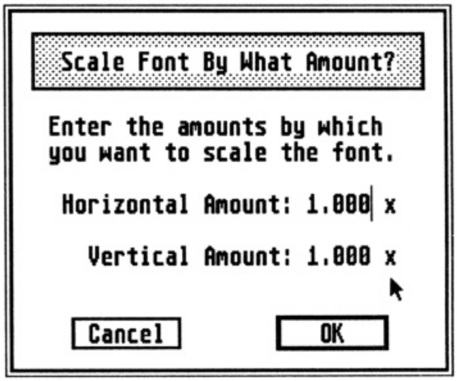
Please Enter The Resolution For An Uninstalled Device Here ◇ Horizontal: 090 Vertical: 090

Font Pointsize To Scale To: 019

Figure #22, Scale Font To A Different Device dialog box

Scale Font By a Certain Percentage -- F6

Using this function, you can scale the entire font by some arbitrary value in either or both directions. A dialog box will appear, as shown in figure #23, and allow you to enter in the amounts by which to scale the font. You can enter separate amounts for horizontal and vertical scaling. Vertical scaling will affect the font's pointsize setting, and it will be recalculated using the current device resolution and the new height of the font in rows. Click on **OK** to scale the font or on **Cancel** if you change your mind.


Scale Font By What Amount?

Enter the amounts by which you want to scale the font.

Horizontal Amount: 1.000 x

Vertical Amount: 1.000 x

Figure #23, Scale Font By A Certain Percentage Dialog Box

A Note Regarding The Scale Functions

First of all, you should realize that the scaling functions are a fast and easy way to create a starting point for a particular size of a font. Except in a few rare cases when scaling from one device and pointsize to a different device and pointsize, the resulting font after scaling will need to be touched up in order to be considered finished by most people's standards. However, you must consider that improving a font where the characters are already the correct size and general shape is much easier than creating an entire font from scratch.

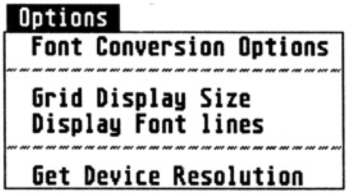
Also, it is an unfortunate reality that you cannot accurately scale fonts to exact pointsizes on lower-resolution devices. For example, the low-rez and medium-rez screen fonts are based on 45 dots per inch, vertically. If you wanted a 12 point font for either device, it would have to be 7.5 rows tall. (A point is 1/72" of an inch, a measurement used in typography & printing. And 12/72 points is equal to 7.5/45 rows.) Since you can't have half a row, you have to round off to the nearest integer number. But using 7 rows at 45 rows per inch is equal to only 11.2 points, or a pointsize of 11. And 8 rows at 45 rows per inch is 12.8

points tall, which rounds off to a pointsize of 13. See the problem? In cases like these, you'll have to accept that the smaller pointsizes may not be exactly the right height. This can occur with any device with a vertical resolution of less than 72 dots per inch, since a row will be larger than 1 point tall. In these caes, set the pointsize to what you need, even if the font height is slightly off.

One thing you should always do after scaling a font is check the special effects information. The best settings for these values are not always directly proportional to the font size, especially the Boldface factor, so scaling these values will not always provide the best results. Check them by hand using the **Special Effects Info** function and change them if you think it is necessary.

It is best to avoid scaling a font down if it is possible. This is because information is lost when you are scaling down and the results can be unpredictable. If you are using fonts which have thick strokes to the characters, or if you only scale down by a small amount, then scaling to a smaller size can be successful, but otherwise, it is best to avoid scaling down to smaller sizes. When you are enlarging a font, the worst side effect is that the result may be blocky and require touching up. For this reason, we suggest that you start out with creating smaller sizes and work your way up, rather than doing things the other way around.

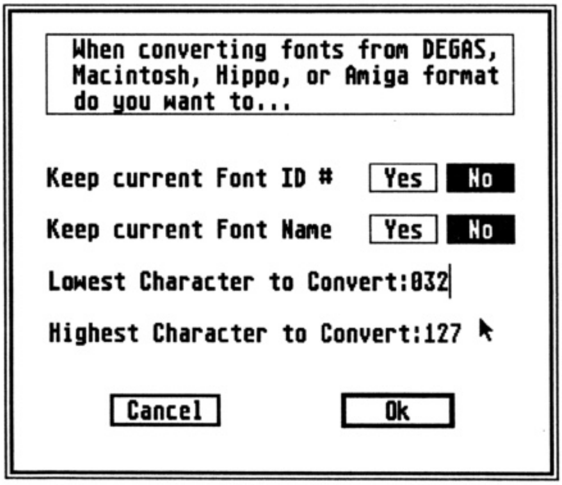
Options Menu



This menu contains choices which allow you to change several settings about how the program works.

Font Conversion Options

This option brings up a dialog box, as shown in figure #24, which lets you set several options for converting fonts from non-GEM formats. You can set the lowest and highest characters to be converted by simply typing in their ASCII values. The default setting is to convert characters with ASCII values from 32 to 127. You can also choose if the font name and ID number should be reset or not whenever a font is converted. Setting this option to keep the font name and ID number of the previous font is useful when you are converting several font sizes of the same typeface into GEM format. It saves you from having to re-enter the font's name and ID number for each size.


When converting fonts from DEGAS, Macintosh, Hippo, or Amiga format do you want to...

Keep current Font ID # Yes [✓ No]

Keep current Font Name Yes [✓ No]

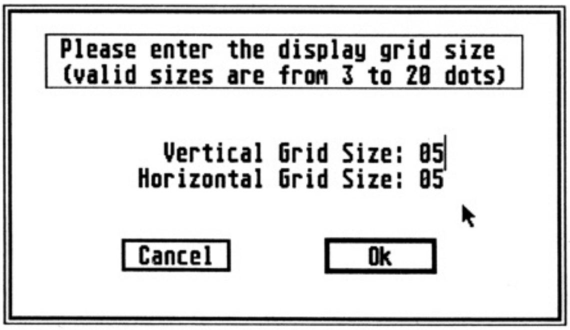
Lowest Character to Convert:032

Highest Character to Convert:127

Figure #24, Font Conversion Options dialog box

Grid Display Size -- Alt-G

This option displays a dialog box, as shown in figure #25, which allows you to set the size of the boxes used in the edit grid. You can set the boxes to be any size from 3 to 20 pixels wide and tall. Since the width and height of the boxes can each be set separately, you can adjust the size of the editing grid to fit your own tastes.


Please enter the display grid size
(valid sizes are from 3 to 20 dots)

Vertical Grid Size: 05
Horizontal Grid Size: 05

Figure #25, Edit grid size adjustment dialog box

Display Font Lines -- Alt-L

This option toggles the display of the ascent line, halfline, baseline, and descent line in the font display window. An example of the display window, with and without these lines displayed, is shown in figure #26. (These terms are all defined in the section titled **GEM Font Definition.**) An example of the Edit window with these lines shown is shown figure #27.

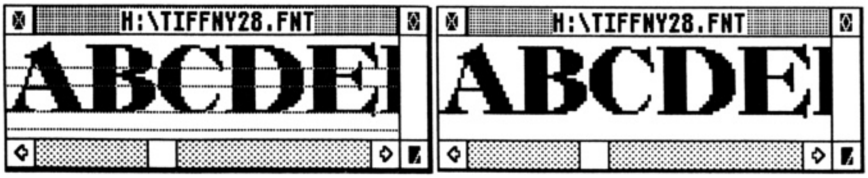


Figure #26, Display window with and without font lines.

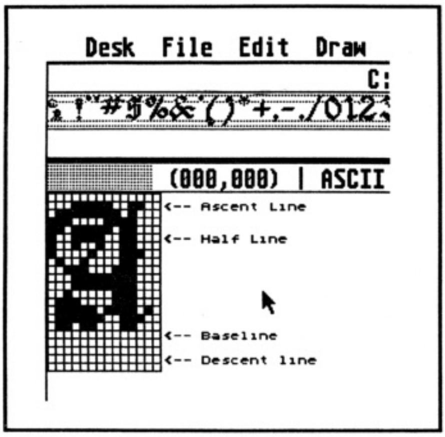
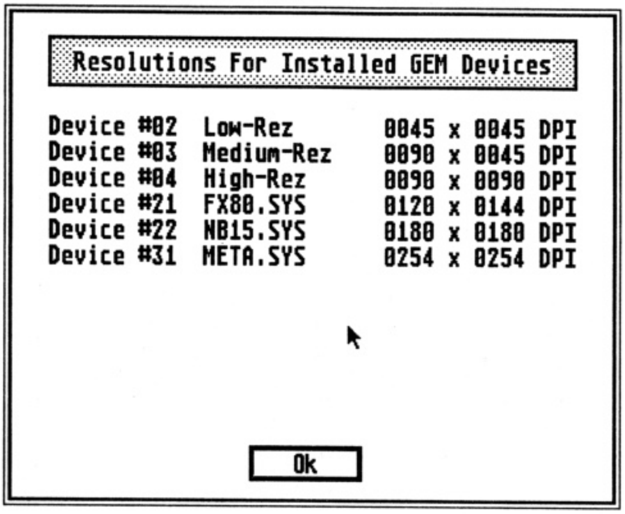


Figure #27, Font Lines in Edit Window

Get Device Resolution

This option allows you to determine the resolution, in dots per inch, of all of the currently installed GEM devices, including the display screen, printers, or plotters. For this option to work with anything but the screen devices, GDOS must be loaded into your system, and the device you wish to test must be named in your ASSIGN.SYS file. Also, the device driver file must be in the correct drive and directory. A dialog box will be displayed, as shown in figure #28, and show you the resolution of each device, expressed as a number of dots per inch, both horizontally and vertically. When you are done looking, click on the OK button to exit the box.

If you change the GEM device driver entries in the ASSIGN.SYS file between the time you boot your system and loading FONTZ!, you may get an error message, as shown in figure #29, when you load the FONTZ! program. If you do, some of the currently installed GEM devices may not appear in the dialog box shown in figure #8, or in the dialog boxes for the **Set Font Device** and **Scale Font To A Different** Resolution

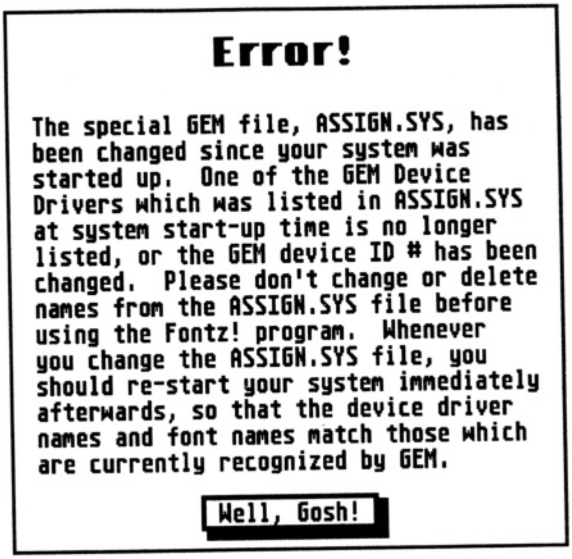


```
Resolutions For Installed GEM Devices

Device #02  Low-Rez        0045 x 0045 DPI
Device #03  Medium-Rez     0090 x 0045 DPI
Device #04  High-Rez       0090 x 0090 DPI
Device #21  FX80.SYS       0120 x 0144 DPI
Device #22  NB15.SYS       0180 x 0180 DPI
Device #31  META.SYS       0254 x 0254 DPI
```

Figure #28, Display GEM Devcies Resolution Dialog Box

functions. As a general rule, any time you alter the ASSIGN.SYS file, you should reset your computer system so that the new configuration in the file becomes recognized by the GEM VDI system. Otherwise, programs which try to use the information in the ASSIGN.SYS file will not have the current information. See the section of this manual titled **The ASSIGN.SYS File** for more information.



Error!

The special GEM file, ASSIGN.SYS, has been changed since your system was started up. One of the GEM Device Drivers which was listed in ASSIGN.SYS at system start-up time is no longer listed, or the GEM device ID # has been changed. Please don't change or delete names from the ASSIGN.SYS file before using the Fontz! program. Whenever you change the ASSIGN.SYS file, you should re-start your system immediately afterwards, so that the device driver names and font names match those which are currently recognized by GEM.

Figure #29, ASSIGN.SYS File Changed Dialog Box

FONTZ! Keyboard Commands

Pressing a key on the main part of the keyboard will save the currently edited character, and then move that character into the edit buffer. Combinations with the [Control] key and another key will move that control character into the edit buffer, after saving the currently edited character. Shown below is a summary of all of the available keyboard commands in FONTZ! Aside from these functions, all other keystrokes should either move a new character into the edit buffer, or produce no effect.

F1 -- Copy edit buffer to copy buffer
F2 -- Replace edit buffer with copy buffer
F3 -- Merge copy buffer into edit buffer
F4 -- Scale font to new pointsize
F5 -- Scale font to new device and/or pointsize
F6 -- Scale font by percentage
F7 -- Enter ASCII value of character to edit
F8 -- Edit Font's Typeface name/Pointsize/ID number
F9 -- Load GEM font
F10 -- Save GEM font
Shift F1 -- Copy block of edit buffer to copy buffer
Shift F2 -- Replace portion of edit buffer with copy buffer
Shift F3 -- Merge copy buffer into portion of edit buffer
Shift F4 -- Toggle fast mode for drawing/cut & paste functions
Shift F8 -- Edit Special Effects parameters
Shift F9 -- Save edit buffer, move to next lower character
Shift F10 -- Save edit buffer, move to next higher character
Enter -- Save edit buffer to character, continue editing
Undo -- Undo changes to edit buffer since character saved
Insert -- Insert column(s) into edit buffer character
Shift Insert -- Insert rows into font
Clr/Home -- Clears all pixels in edit buffer
Shift Clr/Home -- Sets all pixels in edit buffer
Up Arrow -- Shift all pixels in edit buffer up 1 row
Down Arrow -- Shift all pixels in edit buffer down 1 row
Right Arrow -- Shift all pixels in edit buffer right 1 column

Left Arrow -- Shift all pixels in edit buffer left 1 column
Shift Right Arrow -- Rotate character 90 deg. counter-clockwise
Shift Left Arrow -- Rotate character 90 deg. clockwise
Control Delete -- Delete columns from edit buffer
Alternate-G -- Change Edit Grid size
Alternate-I -- Inverse edit buffer
Alternate-L -- Show/Hide font alignment lines
Alternate-O -- Edit character offset value

Many of these keystrokes are the same as some of the GEM drop-down menu choices. In these cases, the drop-down menu will show you what the cooresponding keystroke for that command is. But in order to shorten things, symbols are used to indicate the Control, Shift, and Alternate keys. These symbols are explained below.

^ = Control Key
▣ = Alternate Key
Shft = Shift Key

Creating a New Font

Creating a new font can mean one of several things. It might mean creating a new size of an existing typeface. It might mean creating a printer font from a screen font, or vice versa. Or, it might mean creating an entirely new typeface and size. If you are interested in creating a new size of an existing typeface, then skip down to the section titled **Creating A New Size**. If you are interested in creating a font for a different device, such as making a printer font from a screen font, then skip down to the section titled **Creating Fonts For Different Devices.** Still with us? Good! If you want to create an entirely new typeface and size, read on.

When FONTZ! is first loaded, you will need to load a font file before you can do anything. Even if you want to create an entirely new font, instead of working on an old one, you must load a font to start from. There are several reasons for this, but it all comes down to the fact that it's much easier to start from an existing font than generate all the information for a new font from scratch, for both the program and the user.

The SAMPLE.FNT file on your FONTZ! disk is a very basic, simple font, designed to be used only as a starting point when creating new fonts. By scaling it to different pointsizes and/or devices, you will be able to make it into a good starting point for any size font you want to create from scratch.

Another way of creating a new font is to convert a font from another file format into GEM format. Obviously, the closer you start to what you want your finished result to be, the less work you'll have to do. See the sections of this manual on converting fonts for more information.

Creating A New PointSize

If what you want to do is create a new size of font from an existing font, then you can do it in one of two different ways. First, if you know what size you want to create, you can use the **Scale Font To A Different Pointsize** command to scale the font to the new size. See the section on the **Scale** menu choices for more information.

If you don't know what pointsize you want to end up with, you can use the **Scale Font By A Certain Percentage** command. This will let you scale the font by a value that you enter into the dialog box. You can also change the relative width by entering a different value for the horizontal scaling, in case you want to make the characters wider overall or narrower overall. See the section on the **Scale** menu for more information.

If you used **Scale By A Certain Percentage** with different values for the horizontal and vertical directions, then you have, in effect, created a new typeface. For example, you might scale a 18 point "Swiss" font by 1.0x vertically and 1.5x horizontally, producing what you might call "Swiss Extended" or "Swiss Wide" instead of "Swiss". In fact, if you do something like this, you should definitely give the result its own unique font name and font ID number.

Creating Fonts For Different Devices

Each font is designed to be used with a device that has a certain resolution. This because if printers, for example, used the same fonts as the screen, it would not take full advantage of the printer's higher resolution. (You may wish to go take a look at the sections titled ***Using Fonts*** and ***The ASSIGN.SYS File*** before continuing with this section.) FONTZ! will allow you to take a font intended for one device, and scale it to the appropriate size for a different device.

Let's say that you have a 24 point Dutch screen font, and you need to create a 24 Point Dutch printer font that matches it. You can do this by choosing the **Scale Font To A Different Device** choice from the **Scale** menu. This will allow you to choose a different device from among those which are currently installed, or you can enter the resolution of a device which is not currently installed. If you want, you can also change the pointsize for the font to be created. When you are done making your choices, FONTZ! will create a new font which will be the correct size for the device and pointsize you have selected.

Notes On Scaling In General

Once the font has been scaled, it will probably need to be fixed up a bit. This is because scaling a font only makes sure that everything is the right size; it will get the correct general shapes of the characters, but it cannot do the fine detail work to make everything as pretty as it could be. You will have to edit the character information to make things look better. The degree to which this will be necessary depends on the taste of the person doing it, and by how much the font was scaled.

In order to reduce the amount of retouching that a font will require after scaling, it is recommended that you always try to scale to a larger size if possible. Scaling to a smaller size is possible, but it causes a loss of information which can be difficult to fix. If you must scale downward, try to do so by as little as possible, and please remember that parts of characters that are only a dot or two in width or height may be lost in scaling.

After you have scaled a font to a different device and/or pointsize, it is recommended that you do not change the width of a character unless you are certain you know what you are doing. This is because the idea is for the different sized fonts of the same typeface to be proportionally correct to one another. If you change the width of characters, then things might not be correct from one size or device to another any longer.

When you scale a font to a different pointsize and/or device, it is often worth your time to investigate all of the different completed fonts of that same typeface (fonts of different devices as well as different pointsizes) before deciding which font to scale. For example, if you want to create a 12 point Typewriter printer font to match your 12 point Typewriter screen font, you may discover that scaling your 18 point Typewriter screen font to a 12 point Typewriter printer font provides better results than scaling the 12 point Typewriter screen font. Or if you want to create a 24 point Swiss screen font, you may discover that scaling your 18 point Swiss printer font provides better results than scaling the 12 point Swiss screen font. Experiment with scaling all of the available fonts to the desired size and device before you decide on which one to keep. This way, you can reduce the amount of retouching to the absolute minimum required to make the font look correct.

Creating And Using A Logo

A logo is actually nothing more than a special character which has been changed to have a special design or picture instead of a letter. By including a logo as part of a font, you can use it by simply typing in that character where you wanted the logo to appear. (Because a logo is a character within a GEM font, it is subject to the same rules as GEM fonts. See the section titled **Using Fonts** for more information about this.)

First of all, you should choose a character to contain the logo. You can choose a character from an existing font and include the logo as part of the overall font. Or you could give the font containing your logo its own unique name and id number, and delete the regular characters from it so that only the logo remains. (This has the advantage of not taking over a character in your regular font. You can also have more than one logo in a font. Since a logo is just a character, you could have as many different ones in a font as you have characters available.)

If you decide to include a logo character in a font which also has regular characters in it, choose a character that is not commonly used in most text, avoiding numbers, letters, and punctuation marks. You might try brace or bracket characters. The character should also be able to be typed in using regular keystrokes available from any program capable of using the font. Also remember that the logo will be restricted to the overall height used by the rest of the font, unless you add rows to the entire font.

If you put your logo or logos into a font by themselves, then it is suggested that you start with the letter "A" and go up from there, since it will be easily remembered and accessed. You should also delete the other characters from it to avoid wasting memory and disk space. Finally, give the font its own font id number and font name, such as "Logos".

To create your logo, simply choose the character which will contain it and get it into your editing window. (If you choose a character with zero width, you should add some columns to it right off the bat, or you won't be able to edit anything!) Once you have the character in your edit window, all you have to do is edit the dots in it to create your logo design. All of the drawing and editing tools of FONTZ! can be used for this purpose. You can even cut blocks or characters to be copied into your logo. When you are done, make sure that your design is saved into the font by choosing another character to edit. Or you can simply save your font from the FILE menu.

Don't forget that printer fonts must also contain versions of the logo if the logo is to be used on a printer. You can do this using the following method. Create your logo for the screen font. Save this out to disk if you haven't already. Now scale the font to the printer device. If your font contains only logos and you haven't previously created and saved a printer font, then you can save it now as a printer font (or after touching up the details) and you are done. If your font contains regular characters also, and you do not have a printer font already, save now and you are done. If your font contains regular characters and you already have a printer font for it, then save the font using a temporary name like TMP.FNT. Go to your logo character, and cut it into the copy buffer using the F1 key. Now load your printer font and go to the same character and paste in your logo using the F2 key. Save the printer font. If you have more than one logo, then reload the TMP.FNT file, and repeat for each character. Have fun!

Converting Macintosh Fonts

Although the FONTZ! program converts Macintosh font files into GEM font format and sets all of the GEM font parameters automatically, there are certain parameters which may need to be adjusted by the user in order to be provide the best results. This is because some of the parameters in the GEM font definition do not have exact counterparts in Macintosh fonts, and vice versa.

First of all, Macintosh font files do not have Top Line and Bottom Line definitions. Instead, the Ascent Line is the topmost line of the font data, and the Bottom Line is the bottommost line. When FONTZ! converts a Macintosh font into GEM format, the Top Line and Ascent Line values are set to the same number, as are the Bottom Line and Descent Line values.

With GEM fonts, the blank spaces between the Top Line and the Ascent Line and between the Descent Line and the Bottom Line are used to determine the spacing between different lines of text. But since the Macintosh Font does not have Top Line and Bottom Line parameters, there is another value used to specify the space between lines. But in many Macintosh fonts, this value, called the Leading value, is set to zero and the blank space used to separate lines of text is actually built into the font information. In this case, the Ascent Line and Decent Line values may not actually show up on the correct rows of the font, and you should change them as required. Otherwise, you may need to add some empty rows to the top and/or bottom of the font in order to keep lines of text printed in that font from sitting right on top of each other.

Also, the Macintosh font definition does not use the Half Line value that GEM fonts do. FONTZ! attempts to generate a value for the Half Line setting, but it is merely an approximate value, and may need to be changed by the user to be correct. Remember that the Half Line should fall across the top of most lowercase letters.

One very important thing to remember is that the Macintosh fonts do not have a pointsize parameter. This is generated by the conversion routine from the height of the font. But it will not match the pointsize used on the Mac. Keep this in mind.

Macintosh fonts do not have special effects information such as underline or bold face values. During the conversion, these values are generated for the GEM font using constants or according to the size of the font being converted, but they may need changing. Use the Special Effects Info option to change these values.

Finally, Macintosh fonts are designed for a screen resolution of 72 dots per inch (dpi) horizontally and 72 dpi vertically. But GEM fonts for the High resolution Atari ST screen are based on 90 dpi horizontally and 90 dpi vertically. For the most part, you do not need to worry about this small difference, because FONTZ! automatically sets the pointsize to be correct for the Atari ST screen. Since the dots are square on both the Mac and the ST, the only difference will be the pointsize setting.

Some of you may be asking the question, "Where do I get Macintosh fonts?" The answer is that there is a large number of places. First of all, if you have a modem, you should be able to find a fair selection of fonts on Macintosh-based bulletin board systems. Also, commercial telecommunications systems such as Genie or CompuServe that have Macintosh sections often have literally hundreds of public domain or shareware Macintosh fonts. Another idea is to find a Macintosh User's Group in your area that has a public domain software library.

There are a few more things to keep in mind when you download font files for the Macintosh. Macintosh fonts are stored in Macintosh Resource Files. These are similar to the *.RSC files used by the Atari ST, except they can contain more different types of information, including fonts and desk accessory programs. A program for the Macintosh called **Font/DA Mover** can be used to separate Macintosh fonts and desk accessories from a resource file into separate files. This is important, because FONTZ! expects a Macintosh font file to be a Macintosh resource file with just one font of one size and style in a file. Anything else may not work or all but the first font may be ignored. Finally, there are some special file compression programs which are commonly used on the Macintosh, such as **StuffIt** or **PackIt III** (which are somewhat similar to **ARC** on the ST), and if you download font files which have been compressed using one of these programs, FONTZ! will not be able to recognize it as a Macintosh font file.

Another route to obtaining Macintosh fonts is the Magic Sac cartridge, from Data Pacific. This is a Macintosh emulation cartridge for the Atari ST. Combined with Data Pacific's Translator One disk drive adapter, you can run Macintosh software and read and write Macintosh disks on your Atari ST. You could obtain the Mac fonts using the Magic Sac, and then after making sure there was only one font size and style per file, you could use a file conversion utility to transfer the files to ST disk format.

The method of transfering font files which used in creating and testing FONTZ! was to have two computers connected together via modem or null-modem cable. One would be an ST using an ST terminal program supporting Xmodem protocol, such as Flash or Interlink. The other would be a Macintosh or ST running Magic Sac using a terminal program for the Macintosh with MacBinary Xmodem protocol (often simply called MacBinary, this is a special version of Xmodem for the Macintosh which makes sure that Macintosh-created files are recognized correctly.) At this point, you simply download the file(s) from the Mac computer to the ST computer using the MacBinary Xmodem protocol. If you only have access to one computer, you might upload the file using Magic Sac to another computer running a BBS, switch to ST mode, and then download it back again, as long as the font files are uploaded from the Macintosh using MacBinary Xmodem. Note that the MOVER program included with the Magic Sac for transfering files does not use MacBinary format, and so will not work correctly in moving font files to be used with FONTZ!. Another program for use with the Magic Sac called Transverter is supposed to support MacBinary file transfers, but we have not tested it at this time. Future version of FONTZ! may recognize either file format. Please see your README.DOC file on the FONTZ! disk for more information about this.

Please respect the fact that there are several companies which market fonts for the Macintosh. These fonts are not public domain, and even if you legally obtain them, you cannot legally distribute them without permission from the company.

Converting Amiga Fonts

Although the FONTZ! program converts Amiga font files into GEM font format and sets all of the GEM font parameters automatically, there are certain parameters which may need to be adjusted by the user in order to be completely accurate. This is because some of the parameters in the GEM font definition do not have exact counterparts in Amiga fonts, and vice versa.

First of all, Amiga font files do not have Top Line and Bottom Line or Ascent Line and Descent Line definitions. The Baseline parameter is given, however, so the Top Line and Bottom Line values for the GEM font are determined using this value and the total height of the font. The Ascent Line is set to the same value as the Top Line, and the Descent Line is set to one less than the Bottom Line. Also, the Amiga font definition does not use the Half Line value that GEM fonts do. FONTZ! attempts to generate a value for the Half Line setting, but it is merely an approximate value, and may need to be changed by the user to be correct.

Amiga fonts do not have special effects information such as a light text mask value or skewed text mask value. During the conversion, these values are generated for the GEM font using constants or according to the size of the font being converted, but they may need changing. Use the Special Effects Info option to change these values.

Finally, one very important thing to remember is that the Amiga fonts do not have a pointsize parameter. This is automatically generated by the FONTZ! conversion routine from the height of the font to be correct for the Atari ST monochrome screen device. Usually the filename of the Amiga font file will contain the pointsize, so keep in mind that the pointsize of the converted font will most likely not match the pointsize used on the Amiga, even though the actual size of the characters is not changed.

Where do you get Amiga fonts? As with Macintosh fonts, the answer is that there is a large number of places. First of all, if you have a modem, you should be able to find a fair selection of fonts on Amiga-based bulletin board systems. Also, commercial telecommunications systems such as Genie or CompuServe that have Amiga sections often have literally hundreds of public domain or shareware Amiga fonts. Another idea is to find a Amiga User's Group in your area that has a public domain software library.

Unlike Mac fonts, Amiga fonts come in files of one size and style each, like the Atari ST. Also unlike Mac fonts, Amiga font files which have been compressed to save disk space and file transfer time often use a format which is compatible with the ST. A public domain program called ARC is the most popular such file compression utility on both the Amiga and Atari ST. If you download files which have been ARC-ed on the Amiga, you can use ARC on the ST to de-ARC them. These files usually end with ".ARC"

As with the Macintosh, there are several companies which market fonts for the Amiga. Please respect the fact that these fonts are not public domain, and even if you legally obtain them, you cannot distribute them.

Converting DEGAS Fonts

Although the FONTZ! program converts DEGAS font files into GEM font format and sets all of the GEM font parameters automatically, there are certain parameters which may need to be adjusted by the user in order to be completely accurate. This is because DEGAS fonts are very simple, and do not have the many parameters of the GEM font definition. These parameters have to be created on the fly by the conversion routine, and you may wish to change them.

First of all, DEGAS Fonts are always 16 lines high, so FONTZ! uses constant values to set the GEM font parameters. The Baseline and Top Line settings are always set to the 13th line from the top. The Ascent Line value is set to 12. The Half Line value is set to 7. The Descent Line value is set to 2. The pointsize setting is always set to 13. The special effects information is also set, but you may want to change it.

Where can you get DEGAS fonts? If you owned the original DEGAS program, you probably already own several. Typesetter ST and Typesetter Elite, from Xlent Software, also used DEGAS-style fonts, and came with several styles. Besides this, many Atari ST BBS systems have DEGAS style fonts in their download sections, and most Atari ST user's groups will have public domain software libraries with at least some DEGAS format fonts.

Converting Hippo Fonts

Although the FONTZ! program converts Hippo font files into GEM font format and sets all of the GEM font parameters automatically, there are certain parameters which may need to be adjusted by the user in order to be provide the best results. This is because Hippo fonts do not have many of the parameters of the GEM font definition, and these parameters have to be created on the fly by the conversion routine, and you may wish to change them.

First of all, Hippo font files do not have Top Line and Bottom Line or Ascent Line and Descent Line definitions. The Baseline parameter is given, however, so the Top Line and Bottom Line values for the GEM font are determined using this value and the total height of the font. The Ascent Line is set to the same value as the Top Line, and the Descent Line is set to one less than the Bottom Line.

Also, the Hippo font definition does not use the Half Line value that GEM fonts do. FONTZ! does generate a value for the Half Line setting, but it is merely an approximate value, and may need to be changed by the user to be correct.

Hippo fonts do not have special effects information such as a light text mask value or skewed text mask value. During the conversion, these values are generated for the GEM font using constants or according to the size of the font being converted, but they may need changing. Use the **Special Effects Info** option to change these values.

Finally, one very important thing to remember is that the Hippo fonts do not have a pointsize parameter. The Hippo font assumes the height of the font in rows is the same as the pointsize, but since this is not true for GEM fonts, FONTZ! calculates the correct pointsize setting for the Atari ST monochrome screen.

Where can you obtain Hippo format font files? FONTZ! has this conversion option mostly for the benefit of users of the HippoWord word processor. If you use this program now, or in the past, then you should have a number of fonts in this format, and now you can convert them to GEM format for use with programs such as NEOCEPT's **WordUp** word processor. Unfortunately for Hippopotumus Software, this program and font format never caught on with the larger portion of the Atari

ST world, so it may be difficult to locate fonts in this format on bulletin board systems or commercial telecommunications systems, or in user's group's public domain software libraries.

Using Fonts

How do you go about using fonts created and edited with **FONTZ!** with your programs? There are three main requirements. Number one, you must boot your system with **GDOS.PRG** in the **AUTO** folder of your startup disk, with the **ASSIGN.SYS** file in the main directory of the same disk. The number two requirement is that the font filename be listed in the **ASSIGN.SYS** file along with the appropriate device. Lastly, the program must be capable of loading and using standard GEM fonts. Many different types of programs load and use GEM fonts, but the most common applications are word processors, desktop publishing, and drawing packages of various types. If a program does not use GEM fonts, then you will probably not be able to use fonts created or converted with FONTZ! with that program. (Programs can either use GEM to perform output, or access the output device directly. Unless a program uses GEM to perform its output, it will probably not use GEM fonts. And unless the publisher rewrites the program to do so, it is extremely unlikely that there is any way to make it use GEM fonts.)

1) Some of the fonts I have installed are not available from my program.

This can be caused by one of four things. You may not have enough memory left over after the program is loaded for GEM to be able to load all of the fonts named in the **ASSIGN.SYS** file. When this happens, GEM will load as much as it can. Some devices, such as the Atari SLM804 Laser Printer, require a large amount of memory to work (over 1 megabyte for the SLM804!), and this will reduce the number of fonts that you can load. There isn't much you can do about this except to remove any unneccessary desk accessories which might be present and restart your system. Some of the fonts you have listed in the **ASSIGN.SYS** file may not be required for this program, so you could change your **ASSIGN.SYS** file so that GEM will not load them, leaving more memory for the fonts you really do need. Or, you could add more memory to your computer system.

Another possbility is that the fonts may have been loaded into memory without any problems, but your program might only recognize a certain number of different typefaces (fonts with the same ID number). For example, if a program has only 5 menu spaces for typeface names, then any typefaces past #5 will not be available while in that program. You should contact the publisher of that program regarding the problem to see if an update is available.

Three, while the font's filename might be listed in the **ASSIGN.SYS** file now, was it there when the computer was turned on or last reset? The **ASSIGN.SYS** file is only read once, when the computer is turned on or reset. If you make changes to it, they do not take affect until you restart your system. Also, make certain that all of the font filenames in your **ASSIGN.SYS** file are spelled correctly and match the files you actually have available.

Four, all of your font files and device driver programs must all be together in the same folder on the disk, and this disk and folder must be named in the ASSIGN.SYS file. If they are not, GEM will not find them. See the section entitled ***The ASSIGN.SYS File*** for more informtion about this.

2) The name of one (or more) of my fonts is listed twice in my program.

GEM keeps track of different fonts by two ways: font id number and pointsize. Normally, all of the different sizes of the same typeface will have the same font id number. Or at least they are supposed to. But if for some reason they do not, then you can get the same font listed twice in your menu. Check to see what sizes are available for each of the two entries for that font, and then check those sizes with FONTZ! to make sure that all of the sizes have same font id number and font name. (For consistancy, you should make sure the font names match exactly, in regards to uppercase & lowercase letters.) Change the font id number of any sizes that are incorrect. You may want to keep a list of the sizes you have for each font, with the font name and font ID number. A template for keeping such a list can be found at the end of this manual. Feel free to make photocopies of this page of the manual in order to keep track of your fonts. If you keep track of what fonts you have from the very beginning, you'll save yourself a great deal of confusion later.

Another possibility is that you have a font which does not have the correct name. Try looking at both of the fonts to see if they are the same typeface, or different ones. If they are different sizes of the same typeface, then the problem is the font id number, but if they are different typefaces, then the problem is that one (or more) of the fonts has the wrong typeface name. You will have to use FONTZ! to look at the names and font id numbers and change them if necessary. Only fonts with the same id number should have the same typeface name.

3) I have all my fonts installed, but some programs don't use them at all, or don't use some of them.

First of all, if your program does not seem to use your installed fonts at all, then chances are that this program does not use standard GEM fonts. Some programs which use multiple fonts use their own format for fonts instead of the GEM format. In this case, you should refer to the manual for that program to determine the options available to you. Also see the answer to question #1 for more information.

The ASSIGN.SYS File

The ASSIGN.SYS file is a special file used by GEM to keep track of the names of the available GEM devices, and the fonts which go with them. This section has some technical information about GEM and the ASSIGN.SYS file for those who are interested.

What's a GEM device, you ask? A GEM device can be several things, such as a printer, a plotter, or the display screen. The GEM system that is in the Atari ST has built-in programs for running the standard display screens. However, when it comes to other devices, such as printers, there could be many different kinds, such as the Atari SLM804 Laser Printer, Epson LQ-1000, or Atari SMM804 Dot Matrix Printer. Each of these printers has its own protocol for accomplishing its functions. And since GEM must be able to run a variety of such devices, it's impractical to have built-in programs for each one. It would simply take up too much memory.

To get around this, GEM is capable of loading programs which tell it how to run these various sorts of output devices. Such programs are called device drivers. When GEM needs to use a device, it loads the device driver program from disk into memory. This way, the program does not take up memory until it is required.

GEM graphics output is always printed with the graphics mode of that device, how else? And GEM graphics text is also done the same way. Think of each character of text as being a little picture that gets put into the page individually. And since different devices can have very different resolutions, each one must have its own set of font files for proper output of text, with the characters especially designed for the resolution of that devicie. Otherwise, the character data would have to be scaled to be the correct size or the characters would be printed in the wrong size, compared to everything else on the page. Either method results in a loss of quality.

Obviously, GEM must have a way of keeping track of what device driver programs are available, as well as knowing which font files go with which device. This is where the ASSIGN.SYS file comes in. It is a simple text file in a fairly simple format which lists all of this information for GEM. GEM reads this file when your system is started at powerup or reset time, and stores the information for later use. The format of this file is the subject of the rest of this section.

ASSIGN.SYS File Format

The first thing you do in the ASSIGN.SYS file is specify the disk drive and directory where the device driver programs and font files will be found. This is done with a line at the very beginning of the file like this:

```
    Path = C:\GDOS.SYS\
```

This directory path can be a maximum of 64 characters long, and must be the very first non-comment line within the ASSIGN.SYS file. (Comments begin with ";" and you can start a comment anywhere in the line, including at the very beginning. Everything on a line following a comment marker is ignored by GEM when it reads the file.)

The next thing you have to do is to identify your GEM devices. Each particular type of device has a certain range of numbers which it uses for identification, allowing more than one device of the same type to be available at once by using different ID numbers within the proper range. The various devices and their ranges are shown below.

```
    01-10 -- Screen Device
    11-20 -- Pen Plotters
    21-30 -- Printers, including Laser
    31-40 -- Metafile
    41-50 -- Polaroid Palette Camera
    51-60 -- Graphics Tablets
```

You identify a device in the ASSIGN.SYS file by starting a new line with the ID number of the device, followed by the complete filename of the device driver program. You can also use one of two optional modifiers after the device ID. These will affect how GEM will obtain that device driver program. Valid modifiers are:

r - Load Driver at GDOS Initialization and keep in memory.
p - Device driver is located in Atari ST ROM chips

After a device driver name, you list all of the font files which will be associated with that device id number, each on a separate line. All of the non-comment lines following will be interpreted as font filenames, until you reach another device driver id number and filename or the end of the ASSIGN.SYS file.

An example of a ASSIGN.SYS file is shown below. Notice that all of the device id numbers are in numerical order. Also notice that the screen device id numbers have a "p" added to them, indicating that these device driver programs are built into the computer, and do not need to be loaded from disk. The example has three different fonts installed, all from the same typeface (Swiss, in this case), for the default screen, high resolution screen, and Epson FX-80 printer. Note that since the 02 & 03 SCREEN.SYS devices have no fonts listed after them, no loaded fonts will be available to these devices.

```
Path = C:\GDOS.SYS\
;---------
01p SCREEN.SYS      ; Default Screen fonts
ATSS12HI.FNT
ATSS18HI.FNT
ATSS24HI.FNT
;---------
02p SCREEN.SYS      ; Low Rez Screen
03p SCREEN.SYS      ; Mes Rez Screen
04p SCREEN.SYS      ; Hi Rez Screen
ATSS12HI.FNT
ATSS18HI.FNT
ATSS24HI.FNT
;---------
21 FX80.SYS         ; Epson FX-80 & compatibles
ATSS12EP.FNT
ATSS18EP.FNT
ATSS24EP.FNT
;---------
31 META.SYS
;
```

GEM Font Definition

GEM fonts have certain attributes which you should be aware of when you are creating or editing them. This section will give you a basic working knowledge of these attributes.

Font Alignment Lines

First of all, there are several kinds of special lines which help to define a font, and which are used to help align individual characters and lines of text with one another. These lines are, strictly speaking, imaginary ones. But even though you can't see them, they affect the whole font and individual characters nonetheless. GEM can use any of these lines in aligning text output, so it is important that the values for them must be set correctly. It is important that you read the definitions for all of these values if you intend to change any of them.

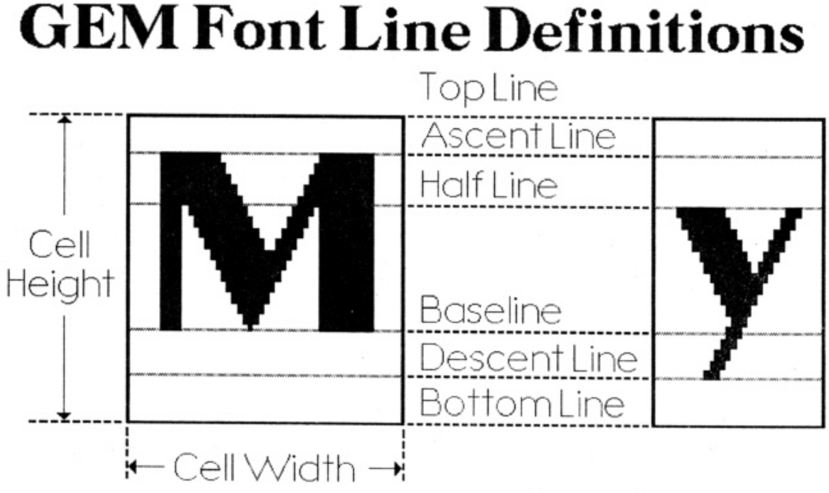


Figure #30, GEM Font Line Definitions example

The lines you need to be familiar with are the Ascent line, the Descent line, the Half line, the Baseline, and also the Top Line and Bottom Line. See figure #28 for an example of these lines in a typical font.

TOP LINE: The Top Line is the very top row of the font data. In the GEM font header information, the setting for the Top Line is equal to the distance, in pixels, from the top row of the font to the baseline row of the font, including the baseline row. (See BASELINE, below.) The actual height of the font, in pixels, can be determined by adding together the Top Line and Bottom Line values.

BOTTOM LINE: The Bottom Line is the very bottom row of the font data. In the GEM font definition, the setting for the Bottom Line is equal to the distance, in pixels, from the top row of the font to the baseline row of the font, not including the baseline row. (See BASELINE, below.)

BASELINE: The Baseline is an imaginary line which runs against the bottom, or base, of all or most of the characters in a font, except those characters which have descenders. Figure #28 shows where the baseline is in relation to the other lines and the main part of the character.

With a GEM font, the Baseline and Top Line parameters both specify how far, in pixels, the Baseline row is from the very top row of the font. In the GEM font file header definition, the Baseline setting is not given by itself, but is determined from the Top Line setting, which gives the distance in pixels from the top of the font to the Baseline. So, the Baseline and Top Line definition values are the same, because they both specify the distance between one another.

In FONTZ!, the Top Line/Baseline parameter may be changed in the **Change Font ID#/Name/Pointsize** dialog box, where it is shown as "Baseline."

ASCENT LINE: The Ascent line is an imaginary line which runs against the highest point of all or most of the capital letters in a font, along with some lower case letters such as "bld".

In a GEM font definition, the Ascent Line parameter specifies how far, in pixels, the Ascent line is from the baseline, including the baseline row. This value must be less than or equal to the Top Line/Baseline setting.

Please note that because the Ascent Line value is the distance from the baseline, reducing or increasing the value for the Baseline parameter will affect where the Ascent Line falls within the font. If you want the Ascent Line to remain where it is when moving the Baseline, you must also increase or decrease it by the same amount as the Baseline.

DESCENT LINE: The decent line is an imaginary line which runs against the lowest point of all the letters in a font which have descenders, such as the lower case letters "ypgq".

In the GEM font definition, the Descent Line parameter specifies how far, in pixels, the Descent Line is from the Baseline, not including the Baseline row itself. This value must be less than or equal to the Bottom Line setting.

Please note that because the Descent Line value is the distance from the baseline, reducing or increasing the value for the Baseline parameter will affect where the Descent Line falls within the font. If you want the Descent Line to remain where it is when moving the Baseline, you must increase it when decreasing the Baseline, by the same amount, or decrease it by the same amount as the Baseline's increase.

HALF LINE: The Half Line is an imaginary line which runs against the highest point of most of the lower case letters in a font, such as "geqrop", and so on.

In the GEM font definition, the Half Line parameter indicates how far, in pixels, the Half Line row is from the Baseline row of the font, including the baseline. This value should be less or equal to the Ascent Line.

Please note that because the Half Line value is the distance from the Baseline, reducing or increasing the value for the Baseline parameter will affect where the Half Line falls within the font. If you want the Half Line to remain where it is when moving the Baseline, you must increase or decrease it by the same amount as the Baseline.

Special Note: The Ascent, Descent, and Half lines are counted as offsets from the baseline because this is the GEM standard for this. It would be easier for some people if these lines were defined as being a certain position down from the top of the font. However, for those people who were more knowledgable, this would be a problem. Please remember to change these other values if you change the baseline parameter.

Character Cell

Besides the font alignment line definitions, there are other parameters which affect how GEM uses a font. The character cell is basically a box which is wide enough and tall enough contain any letter within the font. See figure #30 for an example of these parameters. These values are automatically maintained as required by FONTZ!, and the user does not need to worry about setting them.

CELL WIDTH: This value defines the width of a box which can contain any single character in the font, from the widest one to the thinest one, including any positive horizontal offset values (see the **GEM Font Header** section for more information about horizontal offsets).

CELL HEIGHT: This value defines how tall the character cell is. The parameter is equal to the height of the font.

CHARACTER WIDTH: This value defines the width, in pixels, of the portion of a character cell which is actually used by a particular character.

CHARACTER HEIGHT: This value defines the height, in pixels, of the portion of a character cell which is actually used by a particular character.

GEM Font Header

For any programmers who may be interested, this section will explain the differences between the original GEM font specification from Digital Research, and the extended GEM font specification created by NEOCEPT. If you aren't a programmer, this section won't make any sense at all, but the information is not really important for non-programmers anyway. This extended definition was created to address some of the shortcomings in the original definition which became apparent during the creation of FONTZ!. First, let's show a C language definition of the modified font header structure defintion.

```
/*
 *  FONTDEFS.H -- Font structure definitions
 *  Written By Mike Fulton
*/

#define FONT_MAGIC (0xFEED)

typedef struct gemfont
{
int  face_id;          /* id # for font */
int  pointsize;        /* Font's official pointsize setting */
char font_name[32];    /* Name for font */
int  low_char;         /* Lowest char within font */
int  hi_char;          /* Highest char within font */
int  top_line;         /* Distance from top of font to baseline */
int  ascent;           /* Distance from baseline to ascent line */
int  half_line;        /* Distance from baseline to half line */
int  descent;          /* Distance from baseline to descent line */
int  bottom;           /* Distance from font bottom to baseline */
int  widest;           /* Width in pixels, of widest character */
int  cell_width;       /* Width in pixels, of box to hold all chars */
int  left_off;         /* Left offset for skewed text */
int  right_off;        /* Right offset value, used for skewed text */
int  bold_mask;        /* Shift mask for bold text effect */
int  underline;        /* Height of underline, in pixels */
int  light_mask;       /* Mask for light text */
int  skew_mask;        /* Shift mask for skewed text effect */
int  flags;            /* Font format flag word */
long h_offset;         /* Offset to kern table, if present */
long char_offset;      /* Offset to X-position/width table */
long char_addr;        /* Offset within file to start of raster data */
int  form_width;       /* Width, in bytes, of one row of font */
int  form_height;      /* Height, in pixels, of font (starts at 1) */
long next_font;        /* Pointer to next font.  (0 in disk file) */
int  magic;            /* Magic number should = $FEED */
int  pwidth;           /* If MAGIC # valid, Pixel width in Microns. */
int  pheight;          /* If MAGIC # valid, Pixel height in Microns. */
} GEMFONT;
```

Now let's look at all of the different parameters in the font header, and define what each one is supposed to mean.

int face_id;
This parameter is used to contain the font ID number that GEM uses to make sure that different pointsizes of the same typeface are recognized as all one typeface. Each different size of the same typeface must have the same *face_id* setting.

int pointsize;
This contains the pointsize setting of the font. This is the height of the font, measured in points, as it would appear on the device the font is intended for. A point is 1/72 of an inch.

char font_name[32];
This is a 32-byte string containing the Typeface name of the font. All 32 bytes must be included in the header, but the name does not have to use all 32 bytes as long as it is null-terminated (ends with a byte set to zero).

int low_char;
This is the ASCII value of the lowest character within the font. It must be lower than or equal to the *hi_char* setting.

int hi_char;
This is the ASCII value of the highest character within the font. It must be higher than or equal to the *low_char* setting.

int top_line;
This is the distance, measured in rows, from the topmost row of the font to the baseline of the font.

int ascent;
This is the distance, measured in rows, from the ascent line row of the font to the baseline of the font.

int half_line;

This is the distance, measured in rows, from the half line row of the font to the baseline of the font.

int descent;

This is the distance, measured in rows, from the descent line row of the font to the baseline of the font.

int bottom;

This is the distance, measured in rows, from the bottom row of the font to the baseline of the font.

int widest;

This parameter contains the width of the widest character to be found within the font.

int cell_width;

This parameter contains the width of a box which could contain any single character within the font. (The actual width of the widest character, plus any positive horizontal offset value for that character.)

int left_off;

This parameter is used in creating the skewed text style. Basically, it tells how many pixels are added to the left of the character when skewed text is used.

int right_off;

This parameter is used in creating the skewed text style. It tells how many pixels are added to the right of the character when the skewed text style is used.

int bold_mask;

This is the number of times to move to the right and reprint the character when the bold (sometimes called *thickened*) text style is used. The character is printed once at the specified position, and if the bold style is active, it moves one pixel to the right of the previous position and prints the character again. It repeats this as many times as this value tells it to. Note that this can make a character into an unreadable blob when the bold style is used if this value is incorrectly set to too large an amount.

int underline;

This value contains the height of an underline to be drawn under the characters, at the descent line, when the underline style is active. For example, if the underline value is 5, then a line 5 pixels tall will be drawn at the descent line underneath the character.

int light_mask;

This value contains a mask which will be used when printing light style text. A logical AND operation is performed using the character and the mask value, and the result is what gets printed on screen. The mask is logically shifted to the left for each successive row of the font, so that it does not remain constant from one row to the next.

int skew_mask;

This value contains a mask that is used, along with the left offset and right offset values, to determine how and where to shift a character when the skewed text style is used.

int flags;

This value is a bit-mapped flag for several parameters about the font.

```
bit value description
 0    1  Set if default system font
 1    2  Set if horizontal offset table used
 2    4  Set if font raster in Motorola format
         clear if font raster in Intel format.
 3    8  Set if font is mono-spaced.
```

int h_offset;

In a disk file, this is the offset from the start of the file to the horizontal offset table, if one is present. If no horizontal offset table exists, then this value should be ignored. Once loaded in to memory, this value is changed to a pointer to the memory address where the horizontal offset table was loaded.

The horizontal offset table contains an entry for each character in the font which specifies how many pixel spaces (positive or negative values are allowed) to add to the character position before actually printing the character. The *flags* parameter must have the proper bit set for this table to be active.

long char_offset;
In a disk file, this is the offset from the start of the file to the start of the character offset table, which contains the horizontal position of each character within the font raster data. For example, if the character "a" starts at 653 pixels from the left edge of the font raster, then the character offset table will contain 653 for that character. Each entry in the table is an integer value (2 bytes). The table starts at the lowest character in the font, and contains one more entry than the number of characters in the font. You can determine the position of an entry in the table by subtracting the *low_char* value from the ASCII value of the character you are interested in. You can find the width of any character by subtracting the value for that character from the value of the character that follows it. (Which is why the table has one extra entry at the end.)

Once loaded into memory by GEM, this value is changed to a pointer to where GEM loaded this table.

long char_addr;
In a disk file, this is the offset from the start of the file to the start of the actual character information for the font. The font data is stored as a ST machine-specific single-plane raster form and must be an even number of bytes long. See a GEM VDI programming reference for a description of this format. Once loaded into memory by GEM, this value is changed to a pointer to where GEM loaded the character data.

int form_width;

This parameter specifies the width, in bytes, of the character information raster form.

int form_height;

This parameter specifies the height, in rows, of the character information raster form. The total size, in bytes, of the character information form can be determined by multiplying this value by the *form_width* value.

long next_font;

In disk file, this value is not important and is usually set to zero. Once loaded into memory, this value contains a pointer to the next font header currently loaded into the computer's memory.

Note: The following parameters are the ones added by NEOCEPT to create the extended GEM font format.

int magic;

This is a magic number parameter which is used to determine if the extended font header is valid or not. If the extended header is to be considered valid, this value must be set to $FEED.

int pwidth;

If the *magic* parameter is valid, then this contains the width, measured in microns (1/1000 of a millimeter), of a pixel on the device for which the font is intended. For some resolutions, this value may be rounded off slightly.

int pheight;

If the *magic* parameter is valid, then this contains the height, measured in microns (1/1000 of a millimeter), of a pixel on the device for which the font is intended. For some resolutions, this value may be rounded off slightly.

Now let's take a look at the original GEM font header specification from Digital Research so that we can see what the differences are.

```
/* FONTDEFS.H -- Font structure definitions */

typedef struct gemfont
{
int  face_id;          /* id # for font */
int  pointsize;        /* Font's official pointsize setting */
char font_name[32];    /* Name for font */
int  low_char;         /* Lowest char within font */
int  hi_char;          /* Highest char within font */
int  top_line;         /* Distance from top of font to baseline */
int  ascent;           /* Distance from baseline to ascent line */
int  half_line;        /* Distance from baseline to half line */
int  descent;          /* Distance from baseline to descent line */
int  bottom;           /* Distance from font bottom to baseline */
int  widest;           /* Width in pixels, of widest character */
int  cell_width;       /* Width in pixels, of box to hold all chars */
int  left_off;         /* Left offset for skewed text */
int  right_off;        /* Right offset value, used for skewed text */
int  bold_mask;        /* Shift mask for bold text effect */
int  underline;        /* Height of underline, in pixels */
int  light_mask;       /* Mask for light text */
int  skew_mask;        /* Shift mask for skewed text effect */
int  flags;            /* Font format flag word */
long h_offset;         /* Offset to kern table, if present */
long char_offset;      /* Offset to X-position/width table */
long char_addr;        /* Offset within file to start of raster data */
int  form_width;       /* Width, in bytes, of one row of font */
int  form_height;      /* Height, in pixels, of font (starts at 1) */
long next_font;        /* Pointer to next font.  (0 in disk file) */
} GEMFONT;
```

As you can see, there is actually very little difference between the two specifications. The entire content of the original specification is left unchanged in the new version, and is simply added to at the end to make up the new extended font header definition.

The additions to the font defintion were introduced to overcome one significant shortcoming in the original defintion, namely that there was no way to determine what device resolution a font was supposed to be used with. The only way that fonts were designated as being for different devices was by using different filenames which specified the device. Unfortunately, this method is prone to tampering from ignorant or malicious users and still does not specify the actual resolution of the device.

There are three new parameters in the font definition. First of all, there is a magic number which serves to determine if the font has the extended header or not. Since the portion of the font following the header is usually the character x-position/width table, and we don't want to accidentally get values from here as our two other parameters, there needs to be a way to determine if the new parameters are valid. Therefore, we use a "magic" number of $FEED to indicate that the font has an extended format header. If the magic number of the font is set to $FEED, then the following two parameters indicate the width and height, respectively, of a pixel on the device for which the font is intended.

By including information for the pixel size in the font, it makes it far easier to keep track of a number of things. While some of these might be of use mainly to a font editor program such as FONTZ!, they also have some uses by more general applications programs. First of all, it makes it possible to detect large discrepancies between the actual size of a font and its specified pointsize, an important thing for an editor. It also allows an application to make sure that the correct fonts are installed. It would also allow a program designed to create ASSIGN.SYS files to determine which fonts would work correctly with a particular device. I have seen some strange printouts from people who had set up their ASSIGN.SYS file so their printer was using the same fonts as their screen display.

Having this information in the font header facilitates scaling and transfering fonts from one pointsize and device to other pointsizes and devices. For example, the standard 24-pin printer resolution is 180x180 dots per inch, while the screen resolution is 90x90dpi. Obviously, this means fonts for a 24-pin printer need to be twice as wide and twice as tall as screen fonts to be correct for that device. When you have a number of devices to keep track of, it gets very difficult to keep track of the fonts for each. Let's say you wanted to create a 24 point screen font by scaling a 12 point printer font. Having the device resolution for the font allows you to determine that no scaling is even necessary, because a 12-point font at 180x180dpi is the same size as a 24-point font at 90x90dpi. Knowing the device resolution allows you to scale between different devices and pointsizes simultaneously, which means less manipulation of the font raster is required. Of course, the information regarding the device & resolution could be entered by the user. But it would be necessary for it to be entered each time a font is loaded by a program which used this information, and why should the user have to put up with keeping track of this when it could easily be included in the font in the first place?

Of course, because this specification is a bit different from the original, there may be concerns about compatibility. However, we have discovered that as long as the offsets to the character offset table, horizontal offset table, and raster data are set correctly to account for the different header size, GEM does not seem to care about the additional parameters. We have tested fonts for both the screen and various printers (Epson FX, Citizen 224, NEC P7, & Atari SLM804) with the extended format with WordUp, DEGAS Elite, Easy-Draw, Paint-Pro, and Microsoft Write, as well as with test programs which we have written ourselves. In all cases, we have not discovered a single point of incompatibility with our new extended font header format. (We do, however, regcognize the possibility of a problem if a program was attempting to load fonts itself using constants for the positions of the various parts of a font instead of using the offset values from the header. But at this time, the only program with this possibility that we know of is the Paintworks/N-Vision program, and it wants only GEM fonts with the header in Motorola format. So, when the FONTZ! font editor saves a font for use with Paintworks, it reverses the header into Motorola format, strips the horizontal offset table and extended portion of the header, saves the font this way, and then restores this information afterwards.)

Mike Fulton, Author of FONTZ!

Why FONTZ!??? A Note From The Author

During the creation of this program, some people have asked me why I was creating a GEM font editor. Well, if you are interested in the answer, set your time machine back to January 1984. I was just starting to gain a decent level of programming skill with my 8-bit Atari 400 computer when Apple had the famous "Super Bowl" commercial for their new Macintosh computer. It was very intriguing, so shortly thereafter I went down to my local Apple dealer to have a look. If you didn't happen to see it, you missed a beauty. It depicted the Macintosh as being the reason the year 1984 wasn't going to be like the famous novel *1984*. As far as I know, the 1984 Super Bowl telecast was the only time it was ever shown.

One reason I was fascinated by the Macintosh was the way it let you use many different sizes and styles of text. I've been a photographer longer than a computer programmer, and I'm often involved in putting together multi-projector slide presentations. To create title slides, I had been using sheets of rub-on letters and individually rubbing each letter of each title onto a sheet of paper. While this could produce excellent results, it was very time consuming and difficult. The possibilities that the Macintosh provided were very interesting.

There was just one problem. The Macintosh, although being hailed by Apple as "the computer for the rest of us," was priced out of the range of many of us, including myself. There was a special student discount at several colleges across the country, but my school was not included and I was out of luck.

In July 1984, Jack Tramiel purchased Atari from Warner Communications. It wasn't long before rumors began to surface that Atari would announce a computer that would compete with the Macintosh as far as speed and power were concerned, but be priced far below the Mac's price level. The Atari ST turned out to be that computer, and its price was at a level I could handle.

At the time of its release, there were promises that GEM Paint, GEM Draw, and GEM Write would soon be out for the Atari ST. These programs would supposedly do with GEM on the ST as much as or more than the programs I'd envied on the Macintosh. However, time dragged on, and to this day, these programs have not been released in the U.S. for the Atari ST.

Eventually, Easy-Draw, an object-oriented drawing package from Migraph, was released. It was the first program commercially available for the ST to use the standard GEM font format. For me, Easy-Draw had one very significant shortcoming: only one typeface was included with the program, although it supported up to five different typefaces in several sizes. Still, the promise of more typefaces to come gave me something to look forward to. A few other ST programs had been allowing multiple fonts, such as DEGAS and TYPESETTER ST, but these were using fonts converted from the Atari 8-bit computers, and they were incompatible with the GEM standard and other programs. (And actually, while they allowed the use of different fonts, you could still only have one loaded at a time.)

Another program which could use standard GEM fonts was PaintPro, from Abacus Software. It came out in August 1986, but did not include any fonts with the package. Nothing else appeared until an improved version of the DEGAS paint program was released in the fall of 1986, titled DEGAS Elite.

DEGAS Elite used standard GEM fonts, but to allow you to use the older DEGAS fonts with the new program, a special conversion program was included. It let you load in an older-style DEGAS font and save it out in standard GEM format. You even had a choice of creating fonts with various combinations of double and half width and height. However, the larger sizes were simply expanded versions of the smaller ones. And while the original DEGAS program included an editor for its fonts, DEGAS Elite did not have such an editor for the new fonts after they had been converted to GEM format.

Another problem was that even if you liked the fonts converted to GEM format, the program created screen fonts only. If you wanted to use them with a program to create printouts, such as Easy-Draw, they would not print correctly, because of the resolution differences between the screen and the printer.

At this time, I'd had my Atari ST for about a year and a half, and the font situation was nowhere as good as it should be. Although GEM on the ST has virtually the same multiple font capabilities as the Macintosh, there was only a few programs and fonts to take advantage of them. The biggest problem seemed to be that software companies whose programs used fonts wanted to do things their own way, and be incompatible with everything else, rather than use the compatible, and usually more capable, GEM font format. Maybe this was because of the small number of fonts available in GEM format. Maybe they simply couldn't figure out the right way to do things, because although the documentation has always been there, it isn't always that easy to understand. I don't know why they didn't use GEM fonts right off, but they didn't. (At the least, I think they should have made fonts which followed the GEM standard, even if they used them their own way, instead of using GEM. That way the fonts at least would be compatible with future programs.)

I do know that a lot of people seemed to blame the GDOS portion of GEM for their own problems and misunderstandings. I know I blamed a number of things on GDOS myself for awhile, when a program I was working on didn't work right. But I later found out that there were other things I was doing wrong, and GDOS had nothing to do with my trouble. Certainly GDOS did have some initial problems, but not nearly to the extent suggested by the amount of complaints. The mere fact that there were already a few programs available which used GEM fonts should have suggested to people that their problems, or at least some of them, were their own fault, and not that of GDOS. This goes for those people who complained about there not being a "finished version" of GDOS as well.

At any rate, I got tired of waiting for someone else to supply fonts and font editors, and so on. "The best way to get something done is to do it yourself." Right? Whatever. Anyway, I started looking into figuring out the GEM font format. This was an aspect of programming the ST that I hadn't yet looked into. As I said before, the documentation wasn't always easy to understand, but I eventually gained a fair level of understanding about how things were supposed to work.

About this time I began to think about converting Macintosh fonts over to GEM format. After all, if there were all of these Macintosh fonts available, it would be foolish to simply ignore them. With this in mind, I bought the best book I could find about programming the Macintosh and started learning as much as I could about its font format. Eventually, I had the method of converting Macintosh fonts to GEM format figured out.

Somewhere down the line, Hippopotamus Software had released a word processor called HippoWord which used multiple fonts, using their own method of doing things. To take advantage of the fonts in this format, I started work on a program to convert Hippo fonts to GEM format.

While I was figuring out how to convert Hippo fonts, I got the idea that I may want to create a single program to handle all of these conversions, instead of several smaller ones. That would make it easier and faster to do a number of conversions at the same time, and make the program more efficient. So, I decided to integrate all of my small programs into a somewhat larger one. I took the main portion of one of my previous programming efforts to use as a shell, and plugged my font conversion routines into it. I called this early program FONTZ!

After I got this program together and working, I started thinking of making my own font editor program. The only problem was that I didn't really know the best way to approach this type of program. For one thing, this whole deal was starting to get far beyond the level of a simple program intended for personal use, which was how I had originally envisioned the conversion programs.

I started out by making a list of features that I'd want to see in a character editor. I sat down with some paper and just started writing things down. I asked friends for their opinions. Eventually, I had a general idea of what types of features should be in the program.

Throughout the creation of FONTZ! there have been some features which were thought up and implemented into the program in just a few minute's time. Other features which originally took many days or even weeks to completely figure out and implement have been removed in favor of something else. All in the interest of improving the program overall, of course. So many changes have been made in this program that I long since lost track of them all. Since it is my practice to keep a current listing printout of a program that I am working on, this program has kept my printer very busy. I've used up several boxes of printer paper and quite a few printer ribbons as well.

I hope you find this program useful. It's certainly been an interesting experience creating it. I suppose the next few months after FONTZ! is released will indicate what kind of impact it will have. At the time I write this, several long-awaited products which use GEM fonts are expected to be out soon, including NEOCEPT's own WordUp word processor. Hopefully, FONTZ! will allow users to get the most out of these programs. At the very least, it can allow practically every Macintosh font and Amiga font to be used on the ST. Since this amounts to hundreds of different sizes, styles, and typefaces, it should go a long way in eliminating the over-extended absence of a large number of fonts on the ST.

Thanks,
Mike Fulton, Author of FONTZ!

GEM Fonts

Font Filename	Font Name	Font ID Number	Font Pointsize

Index