

FontLab 2.5

for Windows

*The professional typeface
creator, editor and designer for
the Windows environment*

Foreword

Thank you for purchasing FontLab 2.5 for Windows (FLW). We have tried to create the ultimate solution for the development and modification of fonts. We hope you will be satisfied with our product and find it to be powerful, versatile, and easy-to-use.

In version 2.5, we tried to meet the additional demands of our customers and look forward to being of service to you. In this version, we support work with TrueType fonts. With this new feature, you can now create high-quality fonts in a format which is supported by Windows 3.1, and have the largest possible market. In FLW 2.5, you can import and export TrueType fonts and manage their headers as well.

FontLab is mainly oriented towards creating Adobe Type 1 fonts. With such an orientation, FLW lets you manipulate all PostScript structures and create fonts with the highest possible quality. If you are an experienced developer of PostScript fonts (or any vector images described in the PostScript language), you will be comfortable with the many special features of FLW that provide full control over all parts of a Type 1 font. If you are only beginning your work with fonts and just want to make some modifications of existing typefaces, you will be able to achieve your goal after a few FLW sessions.

Another use of FLW is to create logos and special symbols. FLW provides excellent tools for drawing black-and-white pictures with high quality and high speed, and it is more flexible in working with vector graphic objects than any special drawing program on today's market. FLW lets you create single images and save them as .EPS files (in the Encapsulated PostScript format), or form libraries of special symbols as Type 1 fonts. In version 2.5, we include several special features, for example the **Palette** window, to simplify the creation of logos and special symbols. Now you can choose from 20 predefined graphical billets, and create your own library of graphical elements.

Getting Started

You can start working with FLW fairly quickly. We recommend that you complete these steps:

- ❶ Fill in the registration card and mail it to us. By becoming a registered user, you get privileges such as discounts on upgrades to future versions of FontLab and information about related products.
- ❷ If you have not already done so, install Microsoft Windows 3.1 or higher. Learn the basics of Windows before starting to work with FLW. You will find many specific Windows terms in the FLW manual, so please read your "Microsoft Windows User's Guide" attentively to feel comfortable reading this manual.
- ❸ We strongly recommend purchasing and installing Adobe Type Manager 2.0 or later. This program allows you to work with PostScript fonts in Windows, so after installing ATM you will be able to use the fonts created by FLW in the Windows environment. You can create TrueType fonts without ATM, but if you plan to create PostScript Type 1 fonts, it is necessary.
- ❹ Install FLW following the instructions contained in the next section.
- ❺ Study this manual and experiment with FLW constantly. This book will guide you from performing simple operations to manipulating complicated PostScript structures, but without practice you may soon forget the techniques. Moreover, we wrote this book with the assumption that you have a computer nearby and are able to perform described actions right after reading about them.

We tried our best to make FLW's interface the most intuitive and easy-to-use. And, being a Windows application, FLW incorporates the features common to all programs written for Windows. Therefore, we hope that it will not be difficult for you to begin working with the system.

You will be able to create simple fonts after several FLW sessions. But, if you want to create PostScript fonts and are not already familiar with the Adobe Type 1 font format, you'll have to spend some time studying various PostScript structures to develop quality fonts. FLW provides all the tools needed to control every part of a Type 1 font, but you must understand the PostScript ideology before trying to use these sophisticated features. For a serious font developer, we strongly recommend the manuals published by Adobe (see the "Bibliography" section) which cover all aspects of the Type 1 format in full detail.

If you are going to create TrueType fonts, it is not necessary to study the internal structure of PostScript fonts (though it would be very useful for a better understanding of digital fonts). Support for TrueType fonts is included in Windows 3.1 and doesn't require any additional programs or skills. However, FLW is better suited to work with PostScript fonts and, especially for those, we can provide maximum possible quality.

Installing FLW

System Requirements

To work with FLW successfully, you need:

- ① An i286 or better computer with at least 2 Mb of RAM (it is possible to work with FLW on a 286-based CPU with 2 Mb of RAM, but you will not be able to manipulate some complicated fonts). An i386 or higher processor based computer is recommended. We strongly recommend using a computer equipped with a mathematical co-processor (remember that i486 DX or DX/2 processors already have a co-processor, so you don't need to install one).
- ① Microsoft Windows 3.1.
- ① At least a color VGA-resolution monitor (higher resolutions are preferable) supported by Windows 3.1. Color capability is especially important since the only visual difference between some FLW objects is their color.
- ① Any pointing device supported by Windows.
- ① A printer or plotter supported by Windows is highly recommended. (You will find it useful in several ways: FLW lets you print samples of characters, strings, and code tables; since there are no PostScript monitors for PC, the only way to check the results of your work is by examining printed samples produced by FLW).

We also recommend purchasing and installing Adobe Type Manager 2.0 or later since it lets you work with PostScript fonts in Windows (output them to any device supported by Windows, including your screen and printer). FLW also supports work with TrueType fonts, including import and export of them. Nevertheless, it is first and foremost a superb tool for creating and managing PostScript fonts.

Installing FLW

Microsoft Windows 3.1 must be installed and working before starting the installation of FLW.

The installation procedure is quite simple and takes about 2 minutes. Run it directly from the Windows Program Manager by selecting **Run...** from the **File** menu and entering the following in the **Command Line** string of the dialog box:


A:SETUP then click on **OK**

If your FLW disk is not in the **A** drive, use the appropriate drive letter in the above command.

After executing the installation program, follow the instructions that appear on your screen. You'll have to specify the FLW directory (or just click on **OK** to accept the default name). If it doesn't exist, the program will ask you to confirm its creation. After this, enter the serial number on your FLW disk #2 in the dialog box which appears. The setup program copies files to your hard disk. In the process of installation, the program will ask you some questions. Carefully read all instructions and do exactly what you are asked. Once the installation is complete, an FLW program group will be created containing the FLW icon.

Starting FLW

To start FLW:

- ➊ Run Windows by entering **WIN** at the DOS prompt.
- ➋ Double-click on the FLW icon .
- ➌ From the **File** menu of **FontLab**, select **New** if you want to start working on a new font, choose **Open...** to edit an existing file, or **Import** to work with a file saved in the Type 1 or TrueType format.

First Tricks

Before you learn all the features of FLW and study how to use their full power, we have to show you how easy it is to work with FLW and get professional results. To do this, we will discuss several examples of FLW uses in this chapter.

Stylizing

Besides having powerful editing features and the ability to control all parts of fonts, FLW gives you a wide choice of tools to modify existing typefaces. Using these tools, you can create new unique fonts. All these features are available directly from FontLab and later we will discuss how to access them, but now let's look at Stylizers, the simplest way to make new fonts. Stylizers are precompiled, specially optimized macro procedures for generating font variations.

In the future, you will be able to create your own Stylizers. For details, see "Using the Macro Language."

Font Copyrights

When you use Stylizers or otherwise modify fonts purchased or licensed from other vendors, you should pay special attention to License Agreements. Many agreements on the use of fontware limit your rights to modify the fonts. We shall not be liable for any consequences of violations of copyrights which occur during the use of FontLab, so please be careful!

Using Stylizers

To make a Stylized version of a font:

- ❶ Load it into FLW. You can do this by the **Open...** command from the **File** menu for FontLab fonts (try to open **FLW_DEMO.VFA** font from the FLW directory), by **File / Import / TrueType...** command for TrueType fonts (you can find many TrueType fonts in the **WINDOWS\SYSTEM** directory) or **File / Import / Type 1...** for PostScript Type 1 fonts.

If you don't know where your fonts are, try to use the **FindFont** program (**File / FindFont**) to search your drives and find Type 1, TrueType and FontLab fonts. When you run FindFont, try to press the secondary (usually right) mouse button on the FindFont's options and controls to get a short description of them. After you find some fonts, select them in the **Results List** and press the **Load** button.

You can get a full description of how to open and import fonts later in this book.

- ❷ Select one of the Stylizers in the **File / Stylize** pop-up menu. You can get a full description of the Stylizers later.
- ❸ Wait while FLW stylizes your font. Most Stylizers work very quickly, but several complex ones can take up to 10 minutes. You can preview a sample of the new font with the **Preview...** command of the **File** menu or print a sample with the **Print...** command.
- ❹ Save the Stylized font in the FLW format with the **Save As...** command of the **File** menu, or export it in the TrueType or Type 1 format with the two **Export** commands of the same menu.
- ❺ Install this font in the Windows Control Panel (for TrueType fonts) or in the ATM Control Panel (for Type 1 fonts) to use it in Windows applications.

Stylizers Samples

FontLab

Basic font

FontLab

Obliqued

FontLab

More Obliqued

FontLab

Reverse Obliqued

FontLab

Narrowed

FontLab

Condensed

FontLab

Bold

FontLab

Black

FontLab

Outlined

FontLab

Small Caps

FontLab

Small Obliques

FontLab

Blow Up

FontLab

Outlined + Blow Up

FontLab

Broken Glass

FontLab

Winter Night

FontLab

College (Width = 10, Distance = 20)

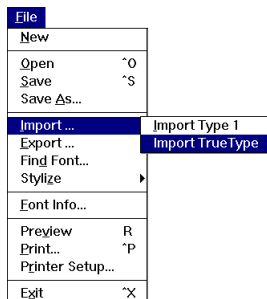
Let's remember how to create such fonts:

- 1 **Open** or **Import** the source font using the appropriate commands from the **File** menu;
- 2 Open a **File** menu, select **Stylize** and choose Stylizer in the **Stylizers list**;
- 3 Save or export new font;
- 4 Install it in the fonts manager (Windows Control Panel or ATM).

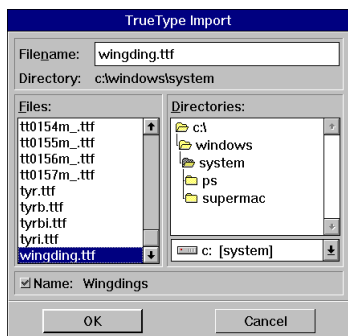
"How To" Samples

Creating a Bold Version of Windows Wingdings Font

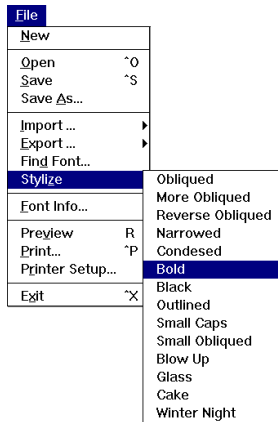
- 1 Run FLW. Double click on the FLW icon in the Program Manager.
- 2 Open **File** menu, select **Import, TrueType**.



- 3 In the **Import** dialog box, select your Windows drive in the **Drive** list, WINDOWS\SYSTEM directory in the **Directories** list and WINGDING.TTF in the end of **Files** list. Press **OK** button to start importing.



- 4 Open the **File** menu, select **Stylize** and choose **Bold** in the Stylizers list.

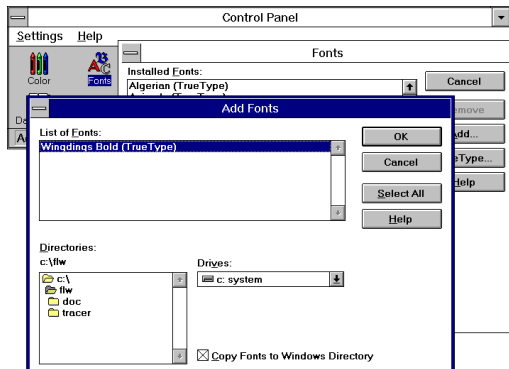


- 5 Wait while FLW stylize your font



- 6 In the **File** menu select **Export, TrueType**. In the **Export** dialog box, select FLW directory in the **Directories** list and enter WINGBOLD.TTF in the **Filename** box. Press **OK**.

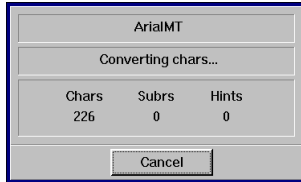
- 7 From the **Main** group of the Program Manager, run Control Panel. In the **Fonts** part of the Control Panel, install Wingdings Bold font.



Now you can use the new font in any Windows application.

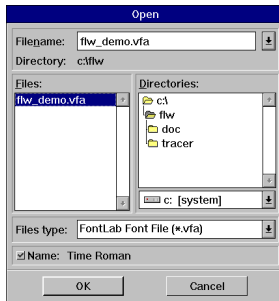
Converting a TrueType Font into Type 1 Format

- 1 Run FLW and import the TrueType font that you want to convert. Refer to the previous example for more information.
- 2 From the **File** menu, select **Export** and **Type 1**.
- 3 Enter a name for the Type 1 font in the **Filename** box of the **Type 1 Export** dialog box. Press OK to export the font.

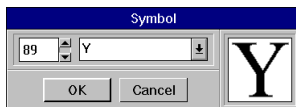


Creating a Yen Character

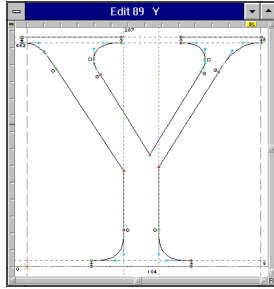
- 1 Run FLW and open or import the font where you want to create ¥ character. For example, open the FLW_DEMO.VFA font from the directory where you install FLW. To do this, select the **Open** command (or press **CTRL+O** on the keyboard) from the **File** menu and double-click on the FLW_DEMO.VFA file in the **Files** List.



- 2 From the **Symbol** menu, choose the **Get Symbol...** command (or press **CTRL-HOME** on the keyboard).

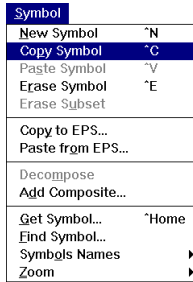


Select the Y symbol in the **Names** list or enter 89. Press **OK**.

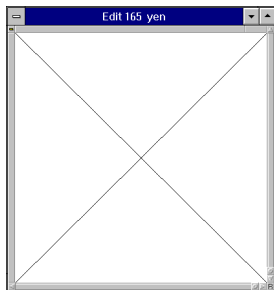


Symbol Y in the Edit window

- 3 Select **Copy Symbol** from the **Symbol** menu (or press **CTRL-C** on the keyboard).



- 4 Press **CTRL-HOME** again, enter 165 and press **OK**. This is the place for the Yen character.

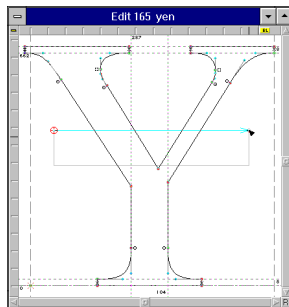


Empty place for Yen in the Edit window

- 5 In the **Symbol** menu, select **Paste Symbol**. A copy of Y will appear in the **Edit** Window.
- 6 Select the **Meter** tool in the **Tools** window.

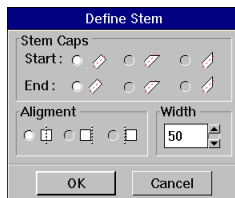


Press the primary (usually left) mouse button and, after that, the **ALT** key. You will see a "rubber" line with gray box on it. Press **SHIFT** key and, by dragging mouse, define position of the Yen horizontal stem. Release primary button.



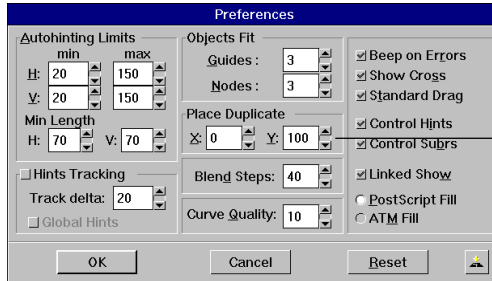
Copy of Y in the Edit window

In the dialog box, enter the width of the stem (for example, 50).

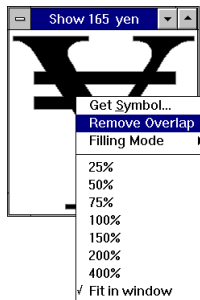


Meter's **Define Stem** dialog box

- 7 If necessary, with the **Move** tool drag the stem to the desired place.
- 8 Select **Preferences...** in the **Options** menu and enter **X: 0, Y: 100** in the **Place Duplicate** part of the dialog box. Press **OK**.



- 9 In the **Edit** menu, select **Duplicate** (or press **D** on the keyboard). You will see a copy of the first horizontal stem above the original.
- 10 In the **Show** window, press the secondary mouse button and select the **Remove Overlap** command in the pop-up menu.




How to Do It More Easily

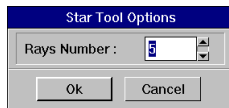
- 1 Instead of selecting symbols through the **Get Symbol** dialog box, you can use the **Table** window. Open this window by clicking on the upper right icon or by double clicking on the **Table's** caption. Click on the image of the symbol you want to select in the **Edit** window. You can get more information about choosing symbols in the "Choosing Symbols for Editing and Previewing" of the "Fonts: Introduction".
- 2 You can use the **Table** window's copying capabilities. Press the secondary mouse button on the image of the symbol you want to copy and drag it to the new place. Release the button (drop the symbol). In the dialog box, press the **Copy** button. For more information, see "Copying and Moving Symbols Using Drag-Drop" of the "Fonts: Introduction".


Creating a Star-like Symbol in the Arial font

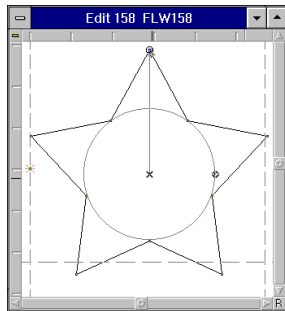
- 1 Run FLW and import the Arial font from the WINDOWS\SYSTEM directory.
- 2 Choose the 158 symbol (FLW158) for editing. Use the **Get Symbol...** command from the **Symbol** menu or **Table** window.
- 3 Open the **Palette** window. To do this, select **Palette** from the **View** list of the **Window** menu.



- 4 Hold the **CTRL** key and click on the  icon of the **Palette** window. Enter the number of the rays in the dialog box.



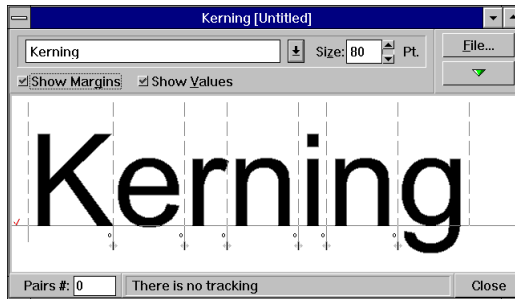
- 5 Select the **New Symbol** command from the **Symbol** menu.
- 6 Press the secondary (usually right) mouse button on the  icon of the **Palette** window and drag the mouse into the **Edit** window. Release the button. A Star will appear in the work place of the **Edit** window.




- ⑦ If necessary, modify the shape of the star by moving the handles. Click the secondary (right) mouse button to fix the star.
- ⑧ Export the font in the TrueType format. After you install it in the Control Panel, you will be able to get the star symbol in any Windows applications by holding ALT and typing 0158.

Generating Kerning

- ① Run FLW and import the font for which you want to create a pair kerning table.
- ② Select the Set Kerning... command from the Expert menu. The Kerning panel appears.



- ③ Press on the Magic Stone () button and select the **Autokern Font** from the drop-down menu.



Wait while FontLab autokerns your font.

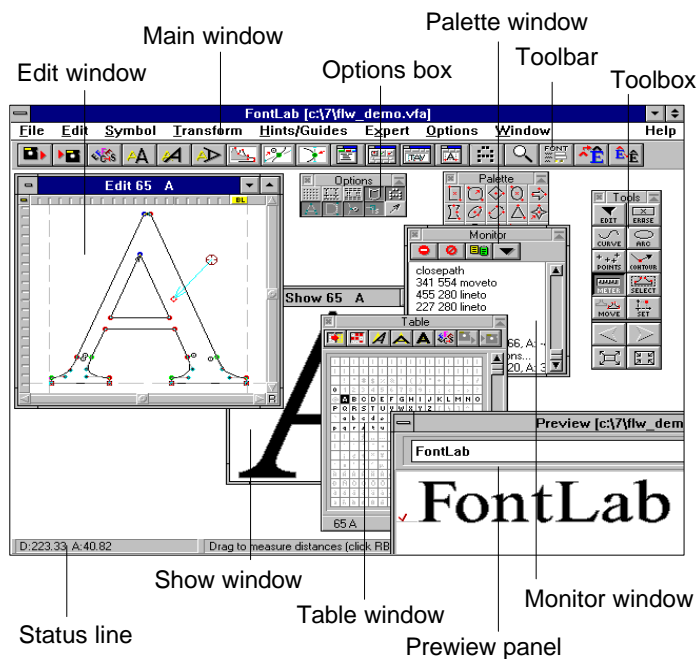
- ④ Export and install font

Screen Layout

The FLW environment consists of the following main components:

- ⌚ Main window
- ⌚ Edit window
- ⌚ Show window
- ⌚ Toolbar
- ⌚ Toolbox
- ⌚ Options box
- ⌚ Table window
- ⌚ Monitor window
- ⌚ Palette window
- ⌚ Status line
- ⌚ Panels

Each of them is shown on this typical screen snapshot.



Main Window

The **Main** window is the standard representation of all Windows applications. Since FLW is such an application, you can manipulate its main window (size, move, minimize/maximize, close) in the same way as that of any other program working in Windows.

The key component of the Main window is the menu bar containing various commands that give you instant access to various functions. You will find detailed descriptions of all these commands in the following sections of this manual.

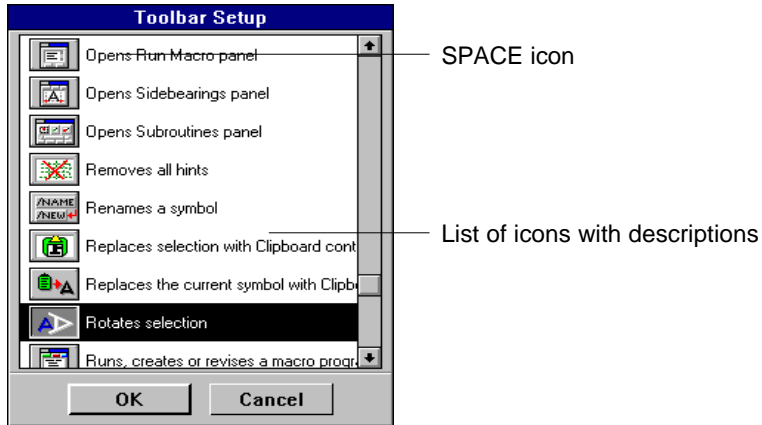
You can easily show and hide all windows by using the **Window** menu of the **Main** window:

Shows and hides Toolbox-like windows	View	✓ T ools
Creates a new Edit window	C reate E dit	✓ O ptions
Creates a new Show window	C reate S how	✓ T able
Hides currently active Edit or Show window	C lose W indow	✓ P alette
Switches on and off Autolink mode	✓ A utolink	✓ M onitor
Arranges Edit and Show icons on screen	A rrange I cons	✓ T oolbar
Tiles Edit and Show windows	T ile	
Cascades Edit and Show windows	C ascade	
Closes all Edit and Show windows	C lose A ll	
Switches between Edit and Show windows	1 Edit 65 A	
	✓ 2 Show 65 A	

Toolbar

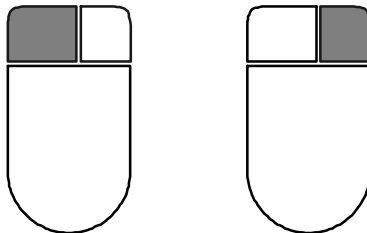


Using the **Toolbar** offers quick access to various menu items which can also be accessed in the normal manner. If you click on an icon from the Toolbar, you'll access that menu or perform the action represented by that icon. If you press and hold the **CTRL** key, then click the primary button on an icon, you'll enter a customizing menu which lets you substitute the icon under the cursor for an icon from the customizing menu.



This will change the action that the previous icon performed. If you press and hold the **SHIFT** key, then press and hold the primary button on an icon, you'll be able to move that icon to the desired position on the **Toolbar**. If while doing this you move the cursor out of the **Toolbar**, you'll be asked if you want to delete the chosen icon from the **Toolbar**. Blank "icons", in other words blank spaces between icons (named in the customizing menu "<SPACE>") are regarded as ordinary icons with all their intrinsic characteristics (e.g., move, delete and substitute).

NOTE: Throughout this manual the mouse button which you press to select a menu command or double-click to run a program is called the *primary* button, and the other button is called *secondary*. These buttons are commonly known as the left and right buttons; however, Windows allows you to swap buttons as an aid to left-handed people, so we use universal names here. Messages appearing in the status line use the common terminology: in them LB stands for the left (primary) button and RB means the right or secondary button. Furthermore, the term "mouse" corresponds to any pointing device supported by Windows and emulating mouse functions.



Primary (left) button

Secondary (right) button

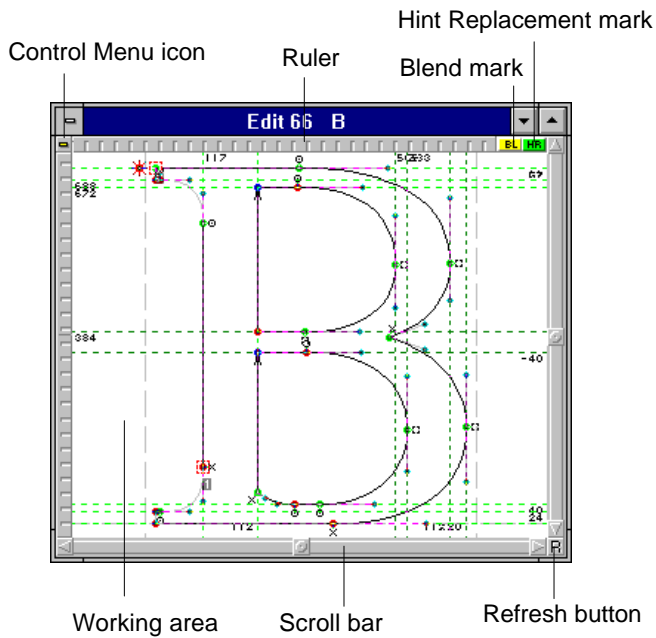
Edit Window

The **Edit** window is the most extensively used window in the system. Almost all operations connected with the design of symbols are carried out in it.

You may create up to ten **Edit** windows. (If the **Autolink** mode, which is controlled from the **Window** menu, is on, all windows will show the same symbol.) This feature lets you easily copy elements of symbols between windows or set different options for different windows and work in several editing environments. Once you've modified a symbol in one of these windows, all other windows are updated.

You may move, resize, and minimize/maximize **Edit** windows in the usual way. When you are working with several windows, it may be useful to turn inactive ones into icons (by clicking on the **Minimize** icon).

In its turn, the **Edit** window consists of the following elements:



⌚ Working Area

Here you create and modify symbols. The appearance of symbols (the types of objects to be displayed) is controlled via the switches set after selecting the **Edit** window... command from the **Options** menu, or clicking on the icons in the **Options** box.

⌚ Control Menu

This menu gives you quick access to some often used commands.


⌚ Rulers

There are two functions of rulers: first, they indicate your current position; second, the rulers are used for placing hints and guides.

⌚ Scroll Bars


The scroll bars have the same function as standard Windows scroll bars: they let you bring into view different portions of the working area. The scroll bars are especially useful when you work at a magnified view.

⌚ *Blend Mark*

The blend mark allows you to activate the blend function that is described below. The blend mark button appears in the right upper corner: 

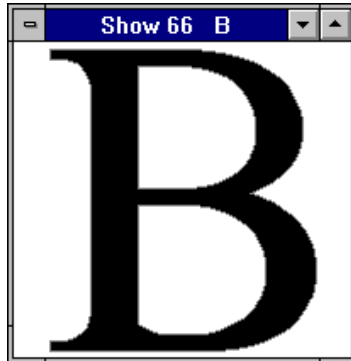
⌚ *Hint Replacement mark*

The hint replacement mark allows you to activate the hint function that is described below.

The blend mark button appears in the right upper corner: 

Show Window

The **Show** window allows you to preview the current symbol as it would be printed. It lets you see the results of your work and detect some errors, e.g., wrong directions of contours. Besides previewing single symbols, you may also examine strings (using the **Preview...** command from the **File** menu).



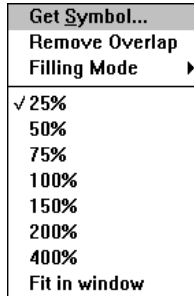
As with the **Edit** window, you may alter the representation of **Show** in the standard way and "tie" windows together by switching on the **Autolink** mode.

If you press the primary button on the **Show** window, you'll see an image of an open palm. While constantly holding the button and moving the mouse, you can drag the whole picture of your symbol within the window.

Mouse Popup Menus

If you press the secondary button, you will enter a special Popup menu. If the cursor was not on the **Show** window or **Edit** window, this menu is the Window menu in the main window. On those two special windows, you can access the **Zoom** menu which allows you to change the chosen symbol and modify the size of this image in the respective window (**Show** or **Edit**).

Popup menu in the **Show** window:



NOTE: To activate the **Popup** menu in the **Edit** window, hold the **CTRL** key when clicking the primary button.

If activated in the **Edit** window, the **Popup** menu contains the following fields:

- Get Symbol** lets you change the actual symbol shown in the **Edit** window.
- Choose Hint** (Horizontal or Vertical) lets you choose global hints. This is equal to the submenu **Choose Global Hint** from the menu **Hints/Guides**.
- 25% - 400%,
Fit in window** represents the visible size of the actual symbol in the **Edit** window.

If activated in the **Show** window the **Popup** menu contains the following fields:

Get Symbol lets you change the actual symbol shown in the **Edit** window.

Remove Overlap lets you remove possible overlap of the actual contour.











Filling Mode (ATM, PostScript) allows you to choose a filling mode for the actual symbol.

**25% - 400%,
Fit in window** represents the visible size of the actual symbol in the **Edit** window.

NOTE: If you want to open a popup menu in the main window but all free space in the main window is overlapped by other FLW windows, press the secondary button on the **Toolbar** or **Status** line.

Toolbox

The **Toolbox** contains icons corresponding to the following instruments:

Icon	Shortcut	Description
	1 or SPACEBAR	The Edit tool performs various manipulations of all objects which appear in the Edit window.
	2	The Erase tool quickly deletes points.
	3	The Curve tool defines curves and approximates parts of contours.
	4	The Arc tool defines ellipses, circles, or their arcs.
	5	The Points tool places guidepoints.
	6	The Contour tool creates new contours or extends existing ones.
	7	The Meter tool measures distances and angles, draws stems and places hints.
	8	The Select tool highlights parts of contours which are to be transformed.
	9	The Move tool changes positions of selected fragments or transforms them.
	0	The Set tool defines positions of nodes or the grid origin.

In the lower part of the box, two pairs of icons appear:





Bring the next or previous symbol into the **Edit** window (shortcuts: **CTRL+←** and **CTRL+→**).



Zoom in or out on the symbol displayed in the **Edit** window (shortcuts: **GRAY +** and **GRAY -**).

Just click on an icon to activate one of the tools, move to the next or previous symbol, or zoom in or out.

In the upper right corner of the **Toolbox** title bar (as well as in three other windows: **Options, Table, Monitor, Palette**), the **Minimize**  icon appears. Clicking on it, you can "hide" the tools and only the title bar will appear on your screen; another click will show the hidden window.

Each of those bars (**Options, Table, Monitor, Palette, Tools**) has a **Close** icon  appearing in the upper left corner.

Options Box



The icons contained in the **Options** box let you hide or display guiding objects and some elements of contours. The following table lists all icons and indicates elements controlled via them.











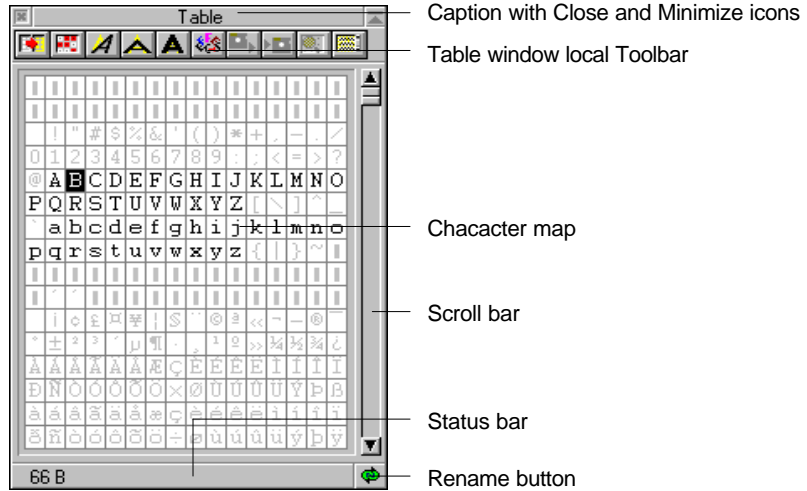
Icon	Object to be shown / hidden
	Grid
	Local guides, global guides, guidepoints
	Local hints, global hints,
	Mask
	Bitmap
	Points
	Control vectors of curves
	Connections
	Subroutines
	Empty vectors

Table Window

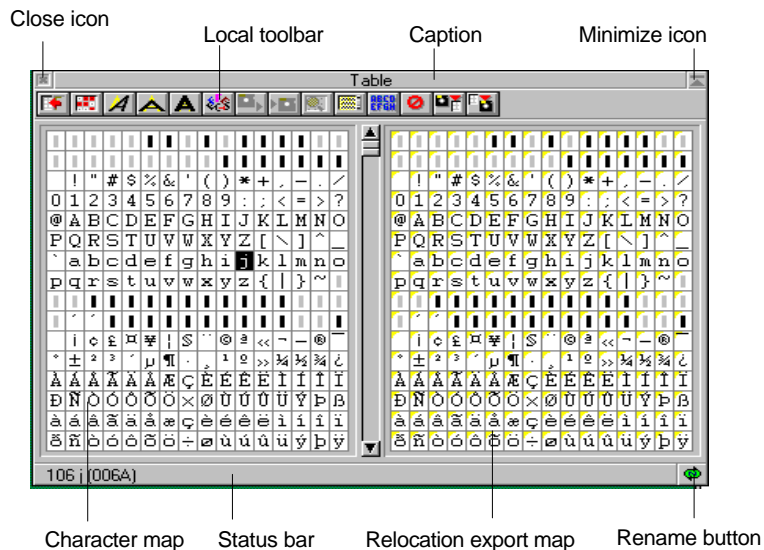
The **Table** window can be represented in two forms, the normal one which consists of one table and an extended one that consists of two tables.

Now let's look at the **normal form** of the **Table** window:



Usually you will use the normal form of the **Table** window. However, in some operations concerned with import and export which will be discussed later, you will use an *extended* form of the **Table** window.

Here is the **extended** form of the **Table** window:




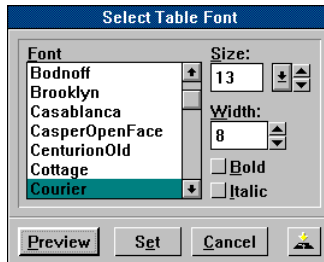
Selecting the current symbol

This window shows you a character map of the current font. Its main purpose is to let you directly select symbols for editing (without clicking on the scroll icons). You can choose symbols in two ways: first, you may place the cursor over the symbol you need and click the primary button. Second, you may click on the status line (in the above picture, it says that the current symbol is "A" and its code is 65) and specify the symbol either by name or by code. This allows you to quickly preview symbols, copy them, and change their names. For more information on these functions, see "Managing a Font in FLW" in the "Fonts: Introduction" section.

Another purpose of this window is copying and transforming of symbols.

If you want the **Table** window to appear on your screen, choose the **Table** command from the **Window** menu. To quickly access this menu, click the secondary button in a free area of the FLW window or on the status line, and the menu will appear right under your cursor.

You can easily modify the appearance of the **Table** window. To do so, select **Table Font...** from the **Options** menu. Or click the  button in the Table's toolbar. The following dialog box appears:














This dialog box lets you select the font that will be used in **Table**, its point size and style, and the width of the **Table** window. If you click on **Preview**, the **Table** window will reflect the settings you've specified, but you'll still be in the dialog box. To quit the dialog box and make the changes permanent, click on **Set**. If you wish to save the current configuration to disk, click on the icon that appears in the lower right corner of the dialog box.

TIP: Using the **Table Font...** command you can get a preview of your font in small sizes for choosing and copying symbols. To do so, export your font in the TrueType format (Select **Export, TrueType** from the **File** menu, for more information refer to the "Managing Fonts" chapter) and select it in the **Select Table Font** dialog box. This technique is especially useful when you edit foreign or symbolic fonts.

Table Toolbar

Now we are going to describe the *local toolbar* of the **Table** window.

Icon	Description
	Switches the Table window between the normal and extended forms.
	Toggles Selection mode. If the Selection mode is on, the image of this icon becomes different:  . See Making a Selection for more information.
	Lets you apply the Slant transformation to the whole font, selected part of a font or a single symbol.
	Lets you apply the Scaling transformation to the whole font, selected part of a font or a single symbol.
	Lets you apply the Bold/Outline transformation to the whole font, selected part of a font or a single symbol.
	Lets you apply one of the set of Special effects to the whole font, selected part of a font or a single symbol.
	This icon is available when the Selection mode is on, and it lets you read a previously saved selection.
	This icon is available when the Selection mode is on, and it lets you write the current selection on the disk.
	Lets you enter the Selection dialog.
	Lets you change the font shown in the left panel of the Table window.

The following icons are available only in the extended form of the Table window.

Icon	Description
-------------	--------------------



Fills relocation map with the default set of symbols.



Erases relocation map.



Reads a relocation map from disk.

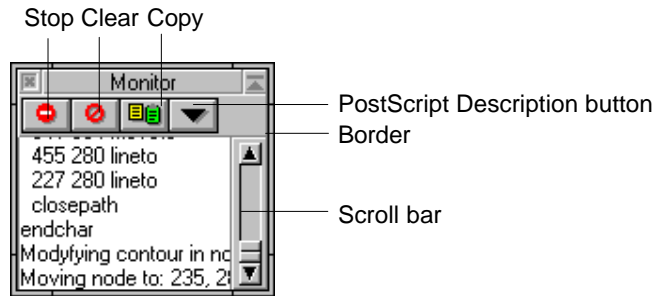


Writes relocation map on the disk.

Monitor Window

The **Monitor** window performs two functions: 1)traces the history of your work in FLW and 2)lets you get a PostScript description of the current symbol, its outline, any selected part, or of any subroutine defined for the font you're editing. The latter feature lets advanced users fully control all objects. You can edit symbols not only with the help of various visual instruments, but by changing PostScript instructions directly. For more details, see the "Using the Macro Language" section.

To display the **Monitor** window on your screen, choose **Monitor** from the **View** popup of the **Window** menu. (Note that a "□" will appear to the left of the command's name). You can move or resize this window the way you would for all Windows applications and scroll through its contents using the scroll bar. By clicking on the **Stop** button, you can suppress the "echo" of all your actions (click on it again to continue tracing), and the **Clear** button lets you erase all information stored in the **Monitor** window.



To close the **Monitor** window, select the **Monitor** command again or double-click on the **Close** icon.

Status Line

The status bar is situated at the bottom of the FLW window and displays information on your current action. It is divided into two parts. The left part indicates coordinates, distances and angles; the right part gives you short hints on what to do next.

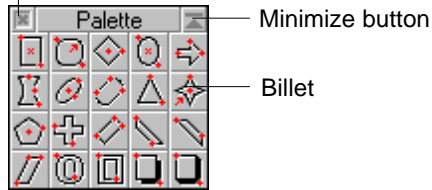


Sample of Status line when Meter instrument is active

Palette Window

FLW provides you with a set of useful shapes (*billets*) that can be added to your symbol.

Close button



For more information about using the Palette window please refer to **Creating Outlines Using the Palette Window** of the next chapter.

Getting Help

FLW comes with an extensive on-line Help system which provides you with instructions on how to use various features of FLW. To access the Help file, press **F1** to get information on the current context, or select one of the commands contained in the **Help** menu. If you need any information on using the Help system, click on the last item of the **Contents** help screen.

Drawing Outlines

This chapter discusses the basic principles of creating vector objects in FLW (since FLW is a drawing program in addition to a font CAD system, it does not matter whether these objects are symbols or any other images). You will learn how to start a new working session, create new objects from scratch, draw lines, curves, ellipses, circles and use billets to modify the outlines you've drawn, view them, and finally save your work as a .VFA or .EPS file.

Creating a New Font

After you double-click on the FLW icon in Windows, its logo will appear and in a few seconds (this delay depends on the speed of your computer), you will see an empty screen with only one menu at the top: **File**. To open a new file for editing, choose **New** from this menu.

You may also use the **New** command at any moment of your FLW session to start working on a new font. If you select this command while editing a font that you have not yet saved (we discuss saving files later in this chapter), the system will prompt you to do so.

Creating New Symbols

After clicking on either of these buttons, you enter the FLW environment. By default, FLW's window is maximized and contains three elements: the **Edit** window, the toolbox with the **Edit** tool selected, and the **Options** box. The symbol offered for editing has the default name "A" and its code is 65. In this chapter, we will forget about the existence of other symbols in the font and discuss all drawing techniques working with the default symbol.

Once a font is initialized, it contains only empty characters because even if they contained nothing and were defined only by their margins, they would be called spaces (empty characters appear as intersecting lines in **Edit** and **Show**). Therefore, you have to initialize a symbol to be able to create its outline. For this purpose, select **New Symbol** from the **Symbol** menu or press **CTRL+N**. This command is similar to **New** for files: it is used whenever you want to start working on a new symbol. You may choose **New Symbol** when you're already editing a symbol; since all results of your work will be lost after initializing a new character, FLW will ask you to confirm the initialization.

After selecting the command, the **Edit** window appears empty except for three dotted lines and an asterisk (we will discuss their purpose later). At this moment, consider the **Edit** window as an area where you can draw any objects. If the default size of this area is not sufficient for you, size the window in the usual way or use the scroll bars to bring into view other portions of the editing field.

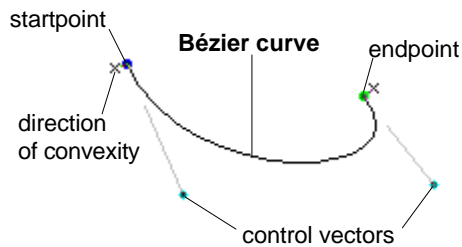
Character Outlines

Before proceeding with drawing symbols, we would like to discuss the way characters are built in any Type 1 font. Since FLW is oriented towards creating PostScript fonts, it uses the same ideology and the same objects for shaping symbols.

TrueType fonts use different mathematical methods to define a symbol's contours, but FLW automatically converts outlines during import and export of TrueType fonts, so you do not have to know how TrueType fonts are "constructed" and may use the same technique as for Type 1 to define symbol's shapes.

Vectors and Bézier Curves

The PostScript language uses two objects for drawing outlines: straight lines and Bézier curves (their important quality is smoothness at any resolution). In fact, every straight line is a vector having a startpoint and an endpoint (also called *nodes*) which are connected, so all lines have definite directions.

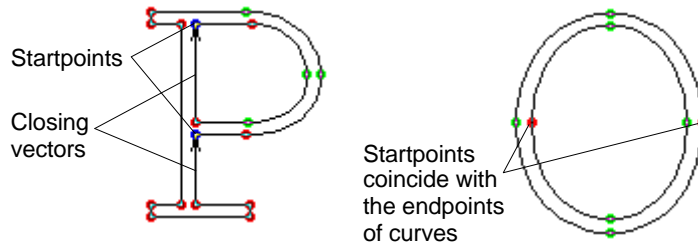


Every Bézier curve is defined by four points: two nodes and a pair of control points which are the endpoints of two vectors starting at the curve's ends and defining its shape (longer control vectors correspond to more convex curves). Another important characteristic of a curve is the direction of its convexity. The above picture shows a Bézier curve with all these elements. (If you draw arrows connecting the curve's nodes with the "flags" situated nearby, they will show the direction of convexity. Another purpose of these flags is discussed later in this chapter under "Creating Smooth Outlines.")

Closing Vectors

Every outline has its *startpoint* (the first point of the first outline's element) which is also its endpoint (i.e., all contours are closed paths; this requirement is necessary since there is no way to fill an open path). For this purpose, all outlines are closed with *closing vectors* which connect the endpoint of the last contour's segment and its startpoint. Closing vectors do not differ from ordinary straight lines except that they are always present in an outline. Therefore, it is better to use closing vectors instead of straight segments (if your outline contains straight lines, it is always possible to place the contour's startpoint at the end of one of them and replace this line with a closing vector). If a symbol must be composed of curves only, place the last curve's endpoint over the contour's startpoint; in this case, the closing vector's length will be equal to zero.

In the picture below, "P" consists of two outlines and each of them uses closing vectors instead of ordinary straight lines. The "O" character has no straight segments, so the startpoints of both outlines and the endpoints of their last curves occupy the same positions; closing vectors are of zero length.

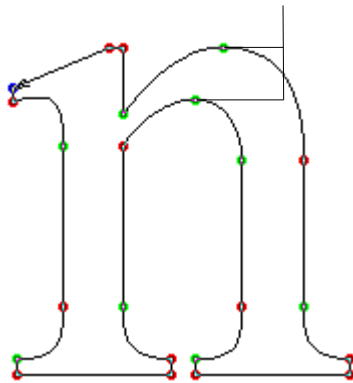


In all editing operations, closing vectors will work as normal vectors. The only difference is its appearance as arrows.

Nodes at Extremes

With straight lines and Bézier curves, you can create outlines of all possible shapes. The problem we must discuss before going on to drawing contours is the optimum number of segments necessary for producing an outline.

Every non-elementary contour has a certain number of points where it makes a turn or S-bend. All such points are called *extremes* and should be marked with nodes. This rule lets you create outlines that will be properly reproduced by all PostScript language devices and will also look good. There are many contours that can be created with this method only. The symbol shown here meets all of Adobe's requirements in the placement of nodes.



FLW allows you to automatically detect extremes and place nodes at them; this feature is described in the "Transformations" section under "Automatic Modification of Contours."

Directions of Paths

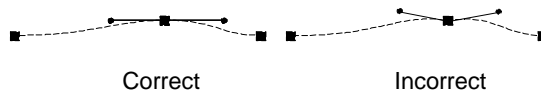
Since many filled images consist of several outlines, the PostScript interpreter needs to distinguish filled subpaths from unfilled ones. The rule is simple: if you imagine walking along a contour in its direction (which is indicated by the closing vector), the filled part must be to the left. The following picture illustrates this convention:



You can easily detect this error by previewing your picture in the window (see "Viewing Your Picture" later in this section) and eliminate it by selecting the `stroke` command discussed under "Transforming Outlines" or using `stroke` for several symbols (or even for the whole font) as shown in the "Transformations" section.

The key to making outlines look better is performing smooth transitions between their segments. The algorithm for smoothly connecting two curves


segments, the control point(s) associated with it, and the second node of the straight line (in the line-to-curve case) should all be collinear (i.e., lie in the




FLW lets you choose among three possible types of connections that are indicated by the *flags* situated near all nodes. We will discuss them in more detail under "Modifying Outlines Using the Edit Tool."

In most cases, FLW can automatically align a curve's control vectors. For details, refer to "FontAudit" of "Advanced Topics".

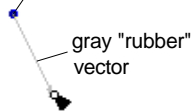
Using the Contour Tool

The **Contour** tool lets you draw straight lines and Bézier curves, so it is the main drawing tool in FLW. To select the tool, click on its icon in the toolbox or press **6**. Note that after moving the cursor into the **Edit** window it appears as .

To draw a new outline with the Contour tool:

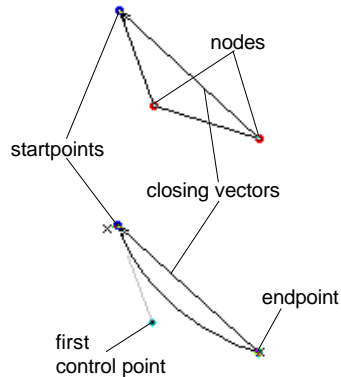
- 1 Move your cursor to the position of the contour's startpoint and click the primary button. A blue point (this color is used for marking startpoints of contours) will appear and the cursor will become a .

startpoint (blue)




- 2 Position your cursor at the point that will be the endpoint of a straight line or the first control point of a curve (depending on the type of object you're defining). You will notice a gray "rubber" vector which follows the cursor and shows that you're actually drawing a vector. Now you may either click the primary button to place the point at the current position, or hold it down, and drag your mouse to manipulate a solid line, then release the button when the line is in place. Holding down the **SHIFT** key while moving the vector will constrain its movements to horizontal or vertical. The status line always indicates the coordinates of your cursor. (We will discuss the coordinate system used in FLW in the "Creating Fonts with Precision" section under "Units of Measurement.")

- ③ If you drew a straight line, continue with the same procedure to define the next object. If you set the first control point of a curve, hold down the **CTRL** key, move your cursor to the curve's endpoint and click the button. A Bézier curve will appear with its second control point coinciding with the endpoint. If you use the dragging technique described above (that is, if you click on the endpoint, hold down the button, and drag your mouse), you will set the position of the second control point.




After drawing two straight lines or a curve, you will notice a closing vector connecting the last point you placed and the contour's startpoint. This operation is performed automatically in order to provide conformity to the Type 1 standard. You may switch off the display of closing vectors by selecting **Edit window...** from the **Options** menu and clicking on the **Closepaths** item, and they will become invisible.

- ④ To finish drawing the outline, click the secondary button. The cursor will appear as a  and you will be able to work on another contour or modify the one you've just created.

Freehand Drawing

Another technique used for drawing Bézier curves is *freehand drawing*. It lets you obtain a curve that is as close as possible to the line that you define by simply moving your mouse. Here is how to use this technique:

- ❶ Once the startpoint of your curve is defined, hold down the ALT-key.
- ❷ Drag your mouse to draw the freehand line. You can define any shape you want.
- ❸ When you've reached the endpoint, release the mouse button. The line you've drawn will be approximated with a series of Bézier curves.

The **Contour** tool also allows you to extend contours. If you want to do so, position your cursor at the outline's last point, click and hold down the primary button, and move your cursor to the position where the next point is intended to appear. Then, follow the steps described above. Don't worry if the cursor is slightly off the last point; by default, it may be within three pixels from it, and you can adjust this sensitivity by choosing **Preferences** from the **Options** menu and setting the **Nodes** value under **Objects Fit**. If you want to use the new value in future, click on the  icon.

Types of Nodes

As you certainly noticed, the points you place with the **Contour** tool are of various colors. Let's discuss them.

As we mentioned above, startpoints are very important elements of contours, hence they differ in color—blue—from all other points. Other nodes have two basic colors: red (endpoints of straight lines) and green (endpoints of curves). Control points finish gray control vectors and are light blue.

Other colors are used when two points of different colors overlap. When a startpoint coincides with a red node, it becomes light green; if the node is green, the resulting point turns light red. If a control point "hides" under a green node, the node becomes yellow.

Modifying Outlines Using the Edit Tool

It is practically impossible to define an outline at one try. Almost always you need to reshape curves, move some nodes, add new points or delete old ones. The tool that performs all these tasks is **Edit**.

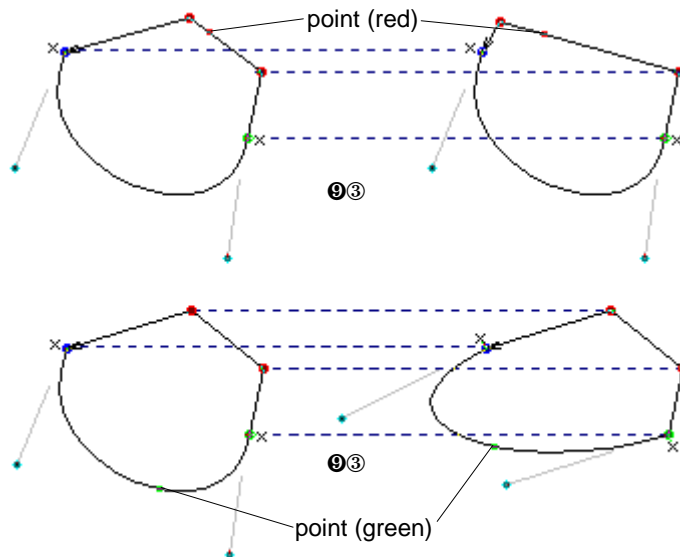
To make the Edit tool active, choose it from the toolbox, or press **1** or **SPACEBAR**. The latter key lets you switch between your current tool (e.g., **Contour**) and **Edit**, that is, if you press the **SPACEBAR** key when using the **Edit** tool, you will select the tool you were last working with. This allows you to draw an outline, quickly reshape it and continue drawing without pressing different keys or moving to the toolbox.

Non-Nodes Editing

The feature that makes FLW one of the most modern programs in today's market of vector graphics software is *non-nodes editing*, which means that you're able to reshape an outline via any of its points (it may be a node, a control point, or an ordinary point belonging to a vector or curve).

To modify a vector or curve, place **Edit**'s cursor on any point, click and hold down the primary button and move your mouse. If the selected point belongs to a line or is close to one of a curve's ends, you will be moving the node nearest to the point, which will become red or blue accordingly. If it belongs to the middle part of a curve, the point will change its color to green and you will be altering the curve's shape without moving its nodes. The status line always shows the current position of the point you're moving (we will discuss the units of measurement used in FLW under "Units of Measurement" in the "Transformations" section). When you're finished, release the button and the outline will be "frozen."

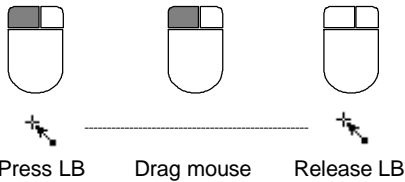
This picture illustrates the two situations discussed above:



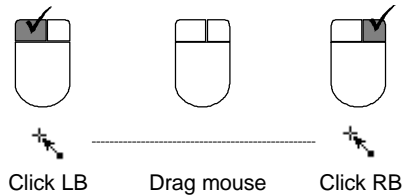
Different Editing Mode

FLW lets you use a new editing mode that differs from the standard drag-and-drop approach. In this mode, you have to click on the point you want to move, position it without holding down the button, and then click the secondary button to finish the operation. From our point of view, this procedure is more comfortable since you do not need to think about holding down the button (which is especially important when you move an object far from the original position). To switch to this mode, select **Preferences** from the **Options** menu and switch off **Standard Drag**.

Standard method of dragging objects:




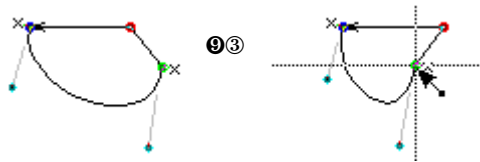
Alternate method of dragging objects:



Moving Nodes and Control Points

The method discussed above is used to visually change the shapes of contours and get results quickly. If you want to create precise, high quality images, you need to determine accurate positions of nodes and the control points of their outlines. Later we will describe the ways to explicitly specify coordinates of points, but now you will learn how to arrange them visually.

To reshape a contour by moving its nodes or the control points of its curves, place the  cursor on the point you want to move and click the primary button. The cursor will disappear and you will see two dotted lines intersecting at the point. Now you can move it as in the non-nodes mode and click the secondary button to finish the operation.



Adding Nodes

As we said before, sometimes you cannot make the necessary contour's shape without placing nodes at its extremes. FLW lets you place nodes on both straight lines and curves by positioning your cursor where you want to split the segment and either holding down the secondary button and clicking the primary button, or pressing **CTRL** and clicking the primary button.

Deleting Nodes with the Edit Tool

To achieve the best possible results on any PostScript device and at any resolution, you must take into account the following two factors:

- ⌚ By placing nodes at extremes and thus increasing the number of the outline's segments, you help the rendering algorithms in correctly reproducing its shape;
- ⌚ By using as few elementary segments (straight lines and Bézier curves) as possible, you achieve minimum memory usage and maximum speed in the rendering system.

Well-designed fonts combine both approaches. FLW lets you automatically meet the first requirement by marking all extremes with nodes (for more details, see the "Transforming Symbols" section under "Placing Nodes at Extremes") or, of course, you can place nodes manually as described above. In any case, you may have to minimize the number of used nodes by deleting some of them.

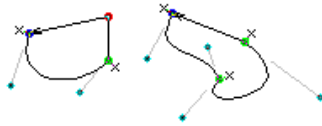
To erase a node with the Edit tool, indicate it with the cursor, hold down the primary button and click the secondary button. If you press this combination when your cursor is on a vector, you will delete the node nearest to the cursor's position. For curves it works differently. For more information, see "Converting Segments Using the Edit Tool" below.

Changing Types of Connections

As mentioned above, FLW uses three types of segments connections. Let's consider them in more detail

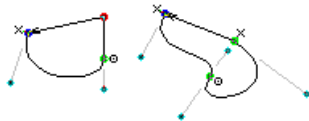
① **Angular** (flag: ●×)

Mutual positions of nodes and control points are not checked:



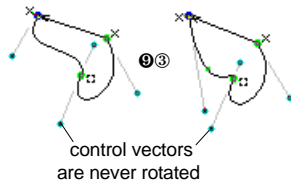
① **Smooth** (flag: ●⊖)

The node joining the two segments, the control point(s) associated with it, and the second node of the straight line (in the line-to-curve case) are always collinear (i.e., lie in the same straight line):



① **Fixed** (flag: ●⊞)

Is used only for curve-to-curve transitions. Equivalent to the Smooth mode when editing contours via nodes or control points; protects control vectors from rotation in the non-nodes editing mode:



To switch between types of connections, press **SHIFT** and click the primary button on the node. Note that there are all three types available for nodes joining two curves: the first two are available for mixed pairs (line to curve or curve to line), and the angular mode is always active for nodes of closing vectors.

NOTE: By changing types of transitions, you do not alter the shape of your contour. Move the node (whose type you just altered) or control vectors and segments associated with it to make changes.

Deleting Points Using the Erase Tool

Erase is the special tool used for erasing points of all types. It works just like an eraser: when you move it, all points within its reach are deleted.

To erase a series of points:

- ❶ Select the **Erase** tool from the toolbox or by pressing **2**.
- ❷ If you double-click on its icon, a dialog box will appear allowing you to specify the width of your eraser in pixels (the default value is 5).
- ❸ Move your cursor into the **Edit** window, position it over the point you want to erase first and press the primary button. You will notice a pair of crossed lines appearing in the cursor's square and, if the eraser's width is more than 5, your cursor will grow in size. The specified point will be erased at once.
- ❹ Continue moving your mouse without releasing the button to delete other points. When you've finished, release the button.

NOTE: **Erase** deletes nodes only while you move the mouse. If more than one node is under the cursor and you want to delete all the nodes, you must slightly move the mouse and the nodes will be erased one by one.

We will also discuss the **Erase** tool in the "Creating Fonts with Precision" section under "Using the Guidepoints".

Converting Segments Using the Edit Tool

FLW lets you easily convert straight lines into Bézier curves and vice versa. You may find this feature useful for making quick sketches of outlines and then improving them by adding curved elements or straightening some lines.

To straighten a curve:

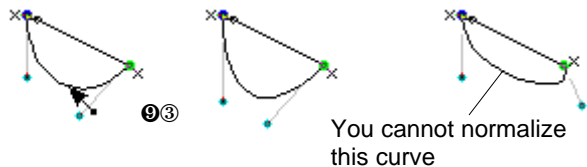
- 1 Select the **Edit** tool (if you're currently using any other tool, press **SPACEBAR** to switch between the current tool and **Edit**).
- 2 Place the cursor on any point of the curve you want to convert into a vector, hold down the primary button and click the secondary button. A straight line will appear connecting the nodes of your curve.

To transform a vector into a Bézier curve, make the **Edit** tool active, press down the **SHIFT** key and click on any point of your vector. You will notice a green dot and two control points appearing on the vector. The green dot means that you're in the non-nodes editing mode, so move your mouse to achieve the desired curve's shape.

Normalizing Curves

To draw good-looking outlines which also conform to the PostScript convention saying that most curves should not include more than 90 degrees of arc, use the **Edit** tool to normalize your curves. This operation transforms your Bézier curve into an arc of an ellipse, the angle of which is determined by two control vectors associated with the curve. If this angle exceeds 90 degrees, FLW will not normalize the curve.

To normalize a Bézier curve, press **SHIFT**, point to the curve, and click the primary button.



Undoing or Redoing an Operation

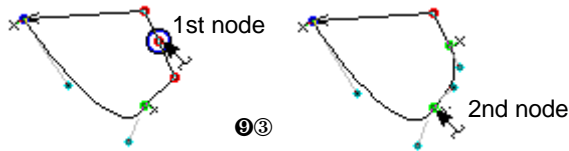
If you're not pleased with the results of the last operation you performed, you can always restore the previous state of your graphic by selecting the **Undo** command from the **Edit** menu or using a keyboard shortcut—**ALT+BACKSPACE**. If you want to return it to the state it was in prior to using **Undo**, choose this command once more.

Using the Curve Tool

The **Edit** tool lets you quickly change types of single contour segments (convert a straight line into a curve or vice versa). However, if you wish to replace several elements with one Bézier curve, using the **Curve** tool is more convenient. Before using the tool, please note that you may transform only segments belonging to the *SAME* contour.

To replace several segments of an outline with a single curve:

- 1 Select the **Curve** tool from the toolbox or by pressing **3**.
- 2 Click on the first node of the future curve (you may also select any point of a vector or curve — the node will be added automatically). You will notice a blue circle surrounding it.
- 3 If you want to reset the startpoint of the curve you're defining, click the secondary button; otherwise, point to the second node or a vector's or curve's inner point and click the primary button.

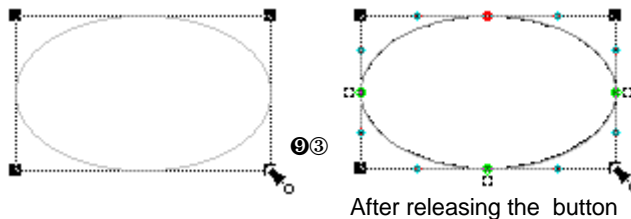


If you press **SHIFT** when pointing to a node, FLW will set the smooth type of connection for it and the control vectors will coincide with the tangent passing through this point. Instead of circles, such nodes are surrounded with blue squares.

Using the Arc Tool

The **Arc** tool has two functions: transforming straight lines into arcs, and drawing ellipses or circles. To make it active, click on its icon in the **Tools** window or press **4**.

- ① **To replace several segments with an arc of an ellipse**, click on beginning and ending nodes as when using the **Curve** tool. You may also transform sequences of vectors and curves if the resulting arc will not contain more than 90 degrees.
- ① **To draw an ellipse**, click the primary button anywhere in the **Edit** window and drag your mouse. The ellipse you are defining fits inside the rectangle that appears, and you alter the ellipse's size by dragging one of the rectangle's corners. Current width and height of your ellipse are constantly shown in the status bar (for a discussion of units of measurement, see "Units of Measurement" in the "Transformations" section). When you are satisfied with the size and shape of your ellipse, release the button. The ellipse will be "frozen."



Now you have several options. If you want to resize your ellipse, point to one of the corner handles, press the primary button, and drag the handle until the ellipse is the desired size. To move it to a different location, place the cursor inside the rectangle and drag the mouse. When you are finished, click the right button.

If you wish to restart defining an ellipse, move the cursor inside the rectangle, press the primary button and then click the secondary button. The old ellipse will be erased.

- ① **To draw a circle**, press **SHIFT** while dragging one of the corner handles. Constrain is only in effect when the **SHIFT** key is held down.

Creating Outlines Using the Palette Window

You can get billets from the Palette window using the **click-click** or **drag-drop** technique.

To Get a Billet With Click-Click Technique

- ➊ Activate an **Edit** window, in which you want to add a billet.
- ➋ Move the cursor to the desired billet in the **Palette** window and click the primary button.
- ➌ Move the cursor to the place in the **Edit** window where you want to place your billet and click the primary button.

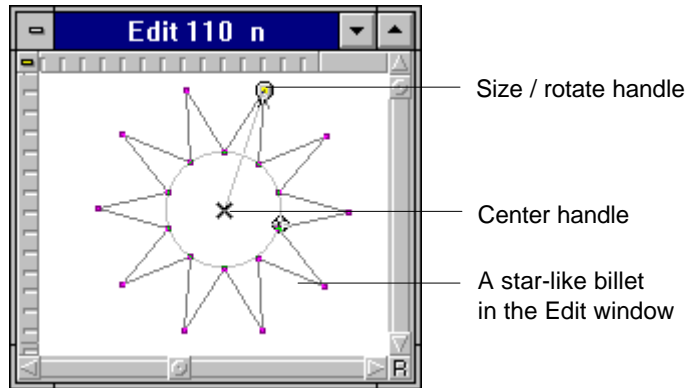
NOTE: Billets can "remember" their sizes between operations. When you define the shape of a billet, it will appear in this size when you place it later again. To reset the initial shape of a billet, hold the **SHIFT** key while you drop it.


To Get a Billet With Drag-Drop Technique

- ➊ Move the cursor to the desired billet in the **Palette** window and press the secondary button. The billet will be highlighted with the green rectangle.
- ➋ Drag your billet to the position where you want to place it and release the secondary button to "drop" it.

Modifying Billets




Your billet will appear in the **Edit** window and will have the following form:



You can modify the shape of a billet by dragging handles (red circles, triangles and boxes and black crosses). If you want to constrain movement of a handle, hold **SHIFT** while dragging it (for example, rotation in this case will be in the step of 15 degrees and scaling will be proportional). In all cases, billets keep their basic shape, and handles determine the parameters of this shape. With some billets, you can change their size in a special way - press and hold the primary button on the  handle, press **CTRL** and move your mouse. You see that some elements of your billet will be tied to each other in a special way. For instance, the two axes of an ellipse will always stay orthogonal while you're moving them.

To fix the billet and convert it to a contour, you must click the secondary button. If you want to delete a billet (to cancel operation) - hold the primary button while clicking the secondary.

To enter precise coordinates of handles, hold **CTRL** when you point to the handle and enter the coordinates in the dialog box.

Handle	Common Characteristic
	This is the center point of a billet. Dragging this handle, you can move the whole figure inside the Edit window.
	These handles let you change the size of a billet round the opposite handle of this kind. The center (marked with a cross) of the billet will move simultaneously with your dragging.
	These handles are additional to the previous one. They also let you change the size and shape of a billet.

Transforming Outlines

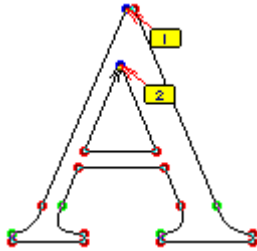
We will finish the discussion of basic drawing features by describing the following four commands contained in the **Transform** menu:

- ① Set Startpoint (shortcut: **T**)
- ① Break Contour (shortcut: **B**)
- ① Combine Contours (shortcut: **F**)
- ① Change Direction (shortcut: **I**)

All these commands work similarly: after selecting one of them, you specify a node (or two for **Combine Contours**); if you wish to cancel the current operation, press the **ESC** key or click on the secondary button **BEFORE** clicking on a point.

Changing a Contour's Startpoint

As we mentioned above, you can make the description of your contour more concise by drawing one of its straight line segments with a closing vector. If your contour's startpoint is the first node of a curve and there are any straight lines following it, it is a good idea to move the startpoint to the beginning of one of them. To do so, select the **Set Startpoint** command. The startpoints of contours will be indicated with yellow boxes, which also contain the numbers of contours:

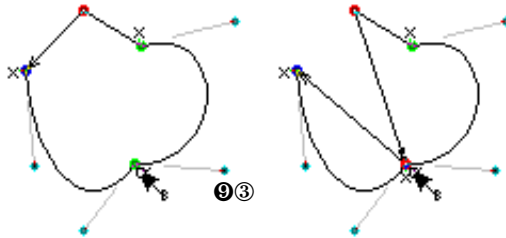


Now click on the node at which you want to place the startpoint. The resulting closing vector will start at the other end of the straight line and terminate at the specified node.

The **Set Startpoint** command lets you renumber the contours of your character, which is useful when you adjust the way it is filled. For this purpose, press **CTRL** and click on the node of the contour which is to be the first in the symbol. Then, finish the operation by clicking the secondary button or proceed with changing the startpoints of contours.

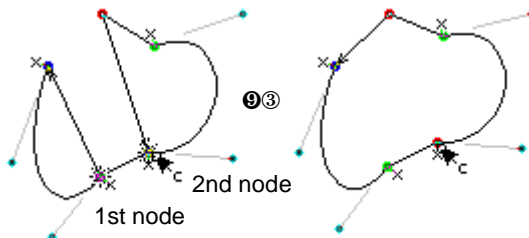
Breaking a Contour Apart

To separate your contour into two subpaths, choose the **Break Contour** command and click on the node which you want to split into two points: the last point of the first contour and the startpoint of the second one. Two new closing vectors will appear, connecting the startpoint and the last node of the original contour with the new nodes. You are now free to move them or otherwise edit the two outlines you've obtained.



Joining Contours

To make a single contour from two separate paths, select **Combine Contours** and click on either of the two nodes belonging to the closing vector of the first outline (this node will become the startpoint of the new outline). The node will become a small 'sun' and you will notice a rubber line following your cursor. Use this line to connect the selected node with one of the nodes belonging to the second outline's control vector, and click the primary button. A straight line will link the selected nodes and a closing vector will connect other straight ends of the original closing vectors.



Changing a Contour's Direction

Directions of contours determine the way paths are filled, so you have to constantly control them. Whenever you see an incorrectly filled picture in the **Show** window, reverse one of its subpaths by selecting the **Change Direction** command and clicking on any of its nodes. FLW will swap the ends of its control vector, thus inverting the subpath.

FLW lets you check for and eliminate this error automatically (for several symbols or the whole font). See the "Transformations" section for more details.

Viewing Your Picture




In this chapter, we will point out the ways to examine pictures using the **Edit** and **Show** windows.

Scrolling the Edit Window


If you wish to use a picture bigger than the editing area, use the **Edit** window's scroll bars to bring its other portions into view. Though the scroll bars are visually different from standard Windows scroll bars, they operate in just the same way.

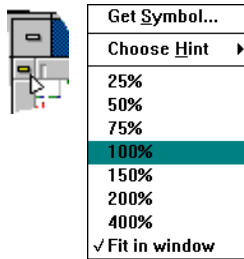
Zooming In and Out

To magnify a portion of the Edit window,

- 1 Click on the  icon. Your cursor will change to a .
- 2 Position the cursor at the upper left corner of the area you wish to zoom in.
- 3 Press the primary button and drag your mouse. A dotted rectangle will appear which is called a "marquee."
- 4 Mark the region to be zoomed with the marquee and release the button. FLW will automatically adjust the zoom ratio to fit the selected area into the **Edit** window.
- 5 At this point, you have three alternatives: you can zoom out by clicking on the  icon, zoom in on some portion of the **Edit** window, or use the scroll bars to see other parts of your picture.

If you click on any point in the **Edit** window after clicking on the **Zoom In** icon, the area surrounding it will be maximized.

To view your picture at a magnified or reduced size, click on the small icon situated in the upper left corner of the **Edit** window (the cursor changes its shape on it to a ) , select **Zoom In** or **Zoom Out**, and choose the size you want from the list.



You can enter either one of the fixed zooms (**25% - 400%**) or the **Fit in window** item. The last one lets you see the whole symbol in the **Edit** window, so that all the parts of your symbol are shown inside the window. Fixed zooms let you choose one of the fixed scalings. To activate this **Popup** menu, hold **CTRL** and click the secondary button.


FLW provides a series of keyboard shortcuts for zooming in and out:

Short-cut	Action
GRAY +	Clicks on the Zoom In icon
- or GRAY -	Zooms out
CTRL+1	100% size (zooms out)
CTRL+2	200% size (magnifies the area around the central point of the picture)
CTRL+4	400% size
CTRL+5	50% size
CTRL+7	75% size
SHIFT+GRAY +	Press successively to magnify by 1.5 times
SHIFT+GRAY -	Press successively to reduce by 1.5 times

Note that there are no equivalents for the **CTRL+5** and **CTRL+7** commands which reduce the view.

NOTE: If you choose any fixed zooming, you'll be able to change the position of the appearance of the base line for the whole font. Holding **CTRL** and **SHIFT** drag the base line with the primary button.

Refreshing the Edit Window

If the **Edit** window does not show the changes you've made, click on the  button which is in the lower left corner of the window, or press **ESC**.

Previewing Pictures

To preview your picture as it would be reproduced on a PostScript device, select **Create Show** from the **Window** menu (if your cursor is in a free area of the FLW window, just click the secondary button and this menu will appear at the cursor position). Resize and move the window which appears so it doesn't interfere with other components of the FLW environment.

By default, the **Show** window reflects all the changes you've made to the current picture while you're doing an operation. However, you can use a special mode that shows changes only after the completion of an operation. To switch on this mode, select Preferences... from the **Options** menu and click on **Linked Show**.

The **Show** window lets you improve your picture's design and detect such errors as incorrect directions of contours.

Saving Results

When you've finished editing an outline or one of its parts, save the results of your work to disk. You have two options.

Saving .VFA files

VFA (Vector Format Advanced) is FLW's own format used for storing any images from single outlines to complete fonts.

To save your file, select **Save...** from the **File** menu or press **CTRL+S**.

If the file is untitled (that is, if you have not yet saved it), you'll have to enter its name in the dialog box. (All files are placed into the directory specified in the **Directory** field; however, you can always save your files to another directory by either specifying their full paths or browsing through the list of directories and drives.) You can just type the filename, and the .VFA extension will be added automatically. When you're done, click on **OK** to save your file. You will notice its name appears in the title bar of FLW's main window.

Once you've given your file a name, simply choose the **Save** command or press **CTRL+S** to update it.

Saving Under a Different Name

If you wish to give your file a new name, choose **Save As...** from the **File** menu. You will enter the same dialog box you used for saving untitled files. As before, specify a new filename and click on **OK**. FLW will preserve the last saved version of your old file and you will continue working under the new name.

Exporting .EPS files

FLW lets you save single images (symbols) in the EPS (Encapsulated PostScript) format for use with many other applications such as Corel Draw!, Adobe Illustrator, Aldus Pagemaker, etc.

To save an .EPS file, choose **Copy to EPS...** from the **Symbol** menu. The **EPS Export** dialog box appears, which is similar to the dialog boxes used for saving files. Enter a filename (the .EPS extension will be added automatically) and click on **OK**.

Since the EPS format is external for FLW, use it only when you want to bring an image from FLW to another application.

NOTES:

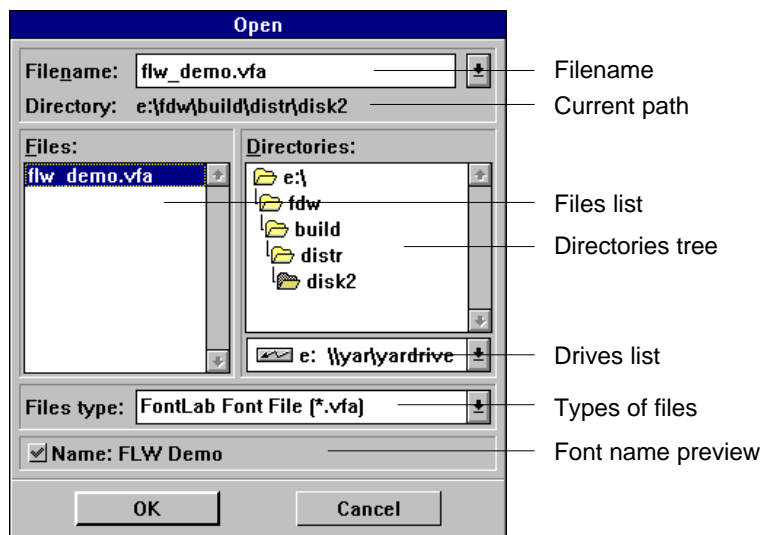
1. Whenever you save a file without changing its name, FLW creates a backup copy of the previous version which has the tilde (~) character in the middle of the extension. Thus, if you want to undo the changes you've made, simply rename the backup copy.
2. Any time you save a .VFA, .EPS or other file, FLW saves the name of its directory in the configuration file (FLW.INI) in the [**directory**] section. (For more information on FLW.INI, see the "Customizing FLW" section under "Examining the Configuration File.") The next time you select **Save As...** or **Copy to EPS...**, the **Directory** field will contain the previous name you used.

Opening Files

Once you have saved a .VFA file, you can continue working on it during your next FLW sessions.

To load a file from disk, select the **Open...** command from the **File** menu (or press **CTRL+O**) either just after starting FLW or at any moment of your working session. If you are editing a file when you choose **Open** and have not saved the latest changes, FLW will prompt you to do so.

The **Open** dialog box appears. Enter a filename manually or click on the appropriate file from the **Files** list. FLW displays only the files that are contained in the directory specified in FLW.INI; to change it, use the **Directories** field. When the filename is entered, click on **OK**.



FontLab common Open dialog box

Importing .EPS files

FLW allows you to modify .EPS files written by other applications (as we mentioned above, we do not recommend using the .EPS format for storing FLW symbols, so there is no need to import .EPS files created by FLW). Since you may import only single images, you need to create or load a file before retrieving a PostScript picture.

To import an .EPS graphic, select **Paste from EPS...** from the **Symbol** menu and use the standard procedure to specify the filename. If you are editing an outline when you run this command, FLW will prompt you to confirm replacing the current symbol's contents with the PostScript image.

Exiting FLW

To end the current FLW session, choose **Exit** from the **File** menu or press **CTRL+X**. If there are any recent changes you haven't saved, FLW will ask you whether you want to do so.

FLW "remembers" the last positions of the toolbox, the **Options** and **Tools** box, and the **Table**, **Monitor** and **Palette** windows, so the next time you start FLW you will find them at exactly the same locations you've left them.

After exiting FLW, you will return to the Windows Program Manager.




Fonts: Introduction

Up to this point, you have used FLW to create single vector images, and we have not mentioned the presence of other symbols in the file you were editing. We discussed only the general methods of drawing outlines without concentrating on specific aspects of creating characters. The main purpose of FLW is to produce high quality PostScript fonts, so in the next chapters we will focus on developing fonts. However, you can use almost all the techniques described in the following sections for creating various pictures and libraries of special symbols.

Managing a Font in FLW

Since fonts are collections of symbols, you need tools to organize them. You need to pick symbols to examine and edit, change their names or order, copy or move symbols to other positions. In this chapter, we will deal with these techniques.

NOTE: We will illustrate our explanations applying the discussed techniques to the sample font (FLW_DEMO.VFA) which comes with FLW and resides in the FLW directory. It contains both completely designed symbols and empty characters, so you will be able to learn the features of FLW by examining the finished characters and supplementing the font with your own symbols.

After opening this font, "hide" marking objects and subroutines (we will not use them before the "Creating Fonts with Precision" section) by clicking on the following icons in the **Options** window: , , and .

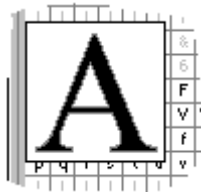
Choosing Symbols for Editing and Previewing

To pick a book in a library or a book store, you can use a catalog or walk along the bookshelves and look at the titles. When a book catches your attention, you take it from the shelf and examine it more closely. Fonts are libraries of characters, so you choose their symbols in the same way.

The **Table** window is similar to a bookstore's catalog. To display it on your screen, select **Table** from the **Window** menu (if you click the secondary button when the cursor is in any free area of the FLW window, this menu will appear right under the cursor).


To choose a symbol using the Table window, use one of the following methods:

- 1 Point to it with your cursor and click the primary button.
- 2 Press the primary button anywhere in the window and move the mouse until the symbol is highlighted (the code and name of the currently highlighted symbol are displayed in the status line of the **Table** window). Release the button.
- 3 Click on the status line. A dialog box appears which lets you specify the symbol's name or code, or choose from the list of available names (it is sorted by codes of symbols).
- 4 If you press **CTRL** or the secondary button before clicking on a symbol, you will see a quick preview of it:



Choosing Symbols Using Drag-And-Drop.



By using the drag-drop technique, you can easily select symbols for various purposes. To do this:

- ❶ Move the cursor to the desired symbol.
- ❷ Press the secondary button (the symbol will get a green background).
- ❸ Drag the symbol into the desired place and release the button. All the places where you can drop your symbol will be highlighted. When you move the cursor to a place where you can't drop any symbol, its shape will change to .

The selected symbol will appear in the current **Edit** or **Show** window (if the **Autolink** mode is on, all windows will display it).


Other Methods to Choose Symbols

You may also enter the dialog box discussed under **3** by selecting **Get Symbol...** from the **Symbol** menu or "small" system menu of the **Edit** window (discussed under "Zooming In and Out" in the previous section). The shortcut for this command is **CTRL+HOME**.

Besides directly selecting a symbol, you can browse through a font by clicking on  or  (which are in the toolbox) to display the next or previous symbol in the current window. If you press **CTRL** with an arrow key, you will move one position in the table in the direction indicated by the arrow.

You can choose one or more symbols using the **Search Symbols...** command from the **Symbol** menu. For details refer to "Symbol's Names" of "Advanced Topics".


Manipulating the Table Window

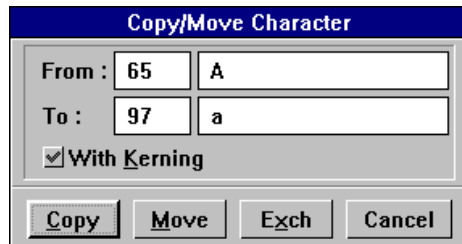
- ① To move the **Table** window, point to its title bar, press the primary button, drag your mouse and release the button to "drop" the window.
- ① The **Table** window may overlap other windows of its kind (see **NOTE** below). To put it beneath them, press **SHIFT** and click on its title bar. By clicking the button again, you will put it on top of all other windows.
- ① You can "hide" the **Edit** window's contents under the title bar by clicking on the **Minimize** () icon that is situated at its right end. To maximize the window, click on this icon again.

NOTE: These techniques may be also applied to all toolbox-like windows.

Copying and Moving Symbols

To copy or move a symbol using the **Table window**,

- 1 Highlight the symbol you want to copy or move.
- 2 Hold down **SHIFT** and press the primary button. Your cursor will change to a . Move the highlight to the desired position and release the **BUTTON** (if you release the **SHIFT** key before the button, you will simply highlight the second symbol).



- 3 In the dialog box, click on **Copy** or **Move**. If you want to interrupt the operation, choose **Cancel**.

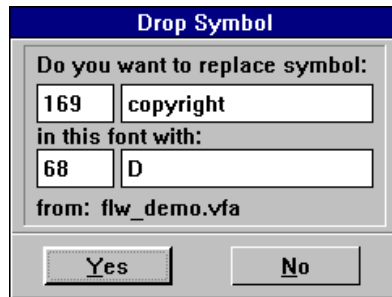
To copy or move a symbol using the commands contained in the **Symbol** menu,

- 1 Display it in the **Edit** window.
- 2 Select **Copy Symbol** from the **Symbol** menu or press **CTRL+C**.
- 3 If you wish to move the symbol, choose **Erase Symbol** or use **CTRL+E** as a shortcut. FLW will ask you to confirm the operation.
- 4 Display the target symbol in the **Edit** window.
- 5 Choose the **Paste Symbol** command or press **CTRL+K**. This command works with "empty" characters, so you do not need to initialize them first. If you've edited the target symbol before choosing this command, you will have to confirm replacing it.


Copying and Moving Symbols Using Drag-Drop

You can use the drag-drop technique to copy, move, and exchange symbols. Simply move the cursor to a source symbol in the **Table** window, press the secondary button and drag this symbol to a new place in the **Table** window. Release the button and choose the appropriate copying options in the dialog box that appears.

Besides copying symbols in one font, you can drag them to another font, which you must open in a different instance of FLW. When you drop a symbol in another **Table** window, you will see the dialog box:



Making a Selection

When the Selection mode is on, (the second icon of **Table Toolbar** looks like: ) you can select some part of the actual font on the left panel.

Drag a Selection

To select a subset of the font, you must move the cursor to the symbol that should become the very first symbol of the desired subset, press the primary button, and move the mouse until the desired part of your font has been highlighted, then release the primary button.

Press the primary button
to begin selection



Here you release the button
to finish selection

Click-Click Selection

Move the cursor to the symbol where you want to start your selection, click the primary button, and press and hold the **SHIFT** key. Then move the cursor to the symbol where your selection should finish, and click the primary button again. If there is a previously made selection then a newly made selection will add to the previous one all the symbols that were between the beginning of the previous selection and the position of the cursor when you last clicked the primary button. If the previously made selection consists of several parts, then the result of the operation will be the same, but all the parts below the final cursor position will become unselected. And all the spaces between the very upper part of the previously made selection and the final cursor position will be marked as selected.

Multiple Selection

You can also make a selection by holding the **CTRL** key and clicking (or holding) the secondary button. If you click the secondary button, the status of the symbol under your cursor will change, so if it was selected it becomes unselected and just the opposite. If you move the mouse holding the **CTRL** key and the secondary button, you'll change the status (selected / unselected) of all the symbols that your cursor runs over.

You can move a whole selection and drag it to another place, perhaps from the left panel of the **Table** window to the right panel. Move the cursor to a selected symbol on the left panel, and press the secondary button without releasing it. You'll see that the color of the selected part of the font has changed.

Now you can move the cursor without releasing the secondary button and drag the selection with the cursor to the desired place on the left panel of the **Table** window or on the right one. Then release the button and the whole selection will be "dropped". FLW copies the symbols or replaces the part of the export map of the font that was under the dragged selection.

Your selection is being dragged as a part of a continuous set of symbols, so all the blank intervals between the selected parts and all the selected intervals between the blank parts can't be shrunk or expanded. Therefore, be absolutely sure that you have properly selected that subset of your font you would like to drag as you cannot undo this operation.

If you drag a selection in the left panel of the **Table** window (belonging to the same FLW or to another FLW running simultaneously working with another font), this means copying symbol(s). See the next chapter for more details.

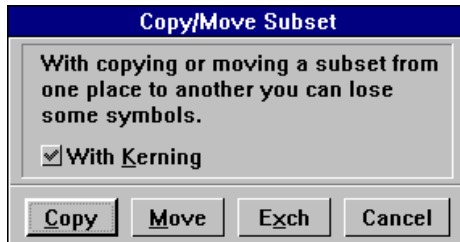
If you drag your selection to a place in the right panel of the **Table** window, you'll set up a subset of a relocation export map that will let you set a specific relocation map for export of Type 1 fonts.

Copying Multiple Symbols

Using the dragging technique described in the section **Dragging a Selection**, you can relocate a selection of your font to another place. If this place is in the same panel of the same table, you're copying the selected subset onto the same font. If this destination is in the right part of the same table, then you're creating part of a relocation map for exporting Type 1 fonts. You can also copy your selection to the **Table** window of another FLW running in the current Windows session as well. To do this, you should change the size of the whole FLW window so that you can see both **Table** windows on your screen at the same time. Then make a selection in the source **Table** window using the technique described above and simply drag your selection to the other **Table** window using the drag-drop technique.

Local Copying

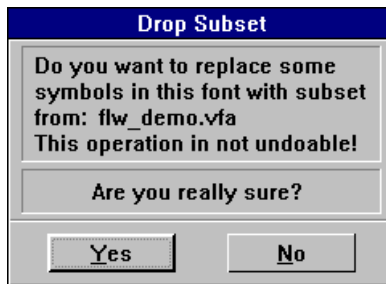
When you drop the selection in the left part of local **Table** window, the following dialog appears:



You choose the appropriate action by clicking the primary mouse button on the corresponding dialog button. This action will be applied to all the symbols that belong to a selection. The source and destination selections may overlap.

Copying to Another Font

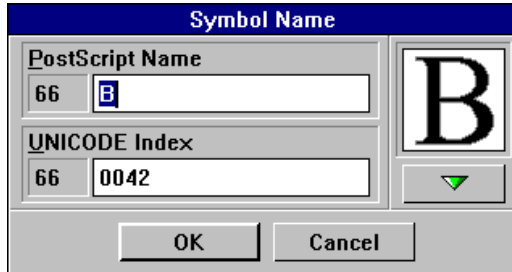
When you drop the selection in the **Table** window of the secondary instance of FLW, you're only able to copy symbols from one font to another. You will see the following dialog box:



If you are absolutely sure you want to complete this operation, press **Yes** and all symbols of the selection will copy to the different font.

Renaming Symbols

Each symbol in a font is identified by its unique code and name. Whereas codes fix positions of symbols in a font, names are descriptive labels of characters and may be changed. To do so, highlight the symbol you want to rename, click on the **Change Name** (🔍) icon and enter the following dialog:



For more info, see the chapter "Symbol's Names" of "Advanced Topics".

Unicode Standard

This is a new character encoding standard designed originally by **The Unicode Consortium** for storage of written characters in computer files or transmission over communication lines. It uses a 16-bit encoding architecture and therefore is completely sufficient for all of the world's written characters, thus surpassing the limitations of **ASCII**. Later, the **Unicode Standard** included some additional features from **ISO 10646**. FLW supports work with the **Unicode Standard** and thus it allows full management with True Type fonts. For more information, see the manual of **The Unicode Consortium** listed in our Bibliography.



ScanFont

When you begin working on a new font, you have three choices:

- ❶ Start from scratch and create all symbols by consulting a font catalog or using your own imagination.
- ❷ Scan a page from the catalog, trace the scanned image, and fine-tune the resulting vector font.
- ❸ Draw individual characters on paper or in any paint program, import them as a raster background, then trace these single images.

We will discuss the third method in the "Working with Bitmaps" chapter of the next section, but now let's deal with the most efficient and time-saving technique, converting scanned images to the vector format and fine-tuning them. This task is performed using a separate program, **ScanFont 2.5 for Windows**, the icon of which resides in the FLW program group in Windows.

Getting Started

To prepare a file:

- 1 Scan the image with symbols of the font with any scanner which has optical resolution of 300 dpi or more (we recommend using 300 or 400 dpi mode and a font sample with at least 30-40 pt. size).

Do not use the automatic resolution enhancement which is available in most desktop scanners. Optical resolution enhancement doesn't really improve the quality of black-white images and can only add more errors.

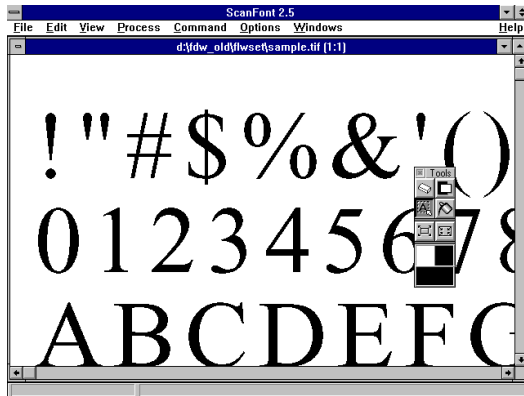
Scan the image as **line-art** (in some manuals this mode is called "black-white"). **Do not scan the image in color or gray scale!**

- 2 Save image as a TIFF file. You can use any standard compression method while saving the image.

To start ScanFont, double-click on its icon in Windows .

To open an image file, select **Open** from the **File** menu (shortcut: **CTRL+O**) and use the standard dialog to load the .TIF, .BMP or .SFI file that stores your image. The BMP format is one of the standard formats supported in Windows. The TIFF format is supported by virtually all scanner programs, so you should have no problem importing your images. The SFI format is an internal ScanFont format. It lets you keep some additional information about the string splitting of your image. If an image file is open, you may simultaneously open many (less than 9) files using **CTRL+O** or from the main menu. You can switch between the open windows with **CTRL+F6** or with the menu item **Window** as usual.

Once an image is loaded, your screen will look like this:



The main purpose of ScanFont is to convert your image into a set of characters available for further transformations using FontLab.

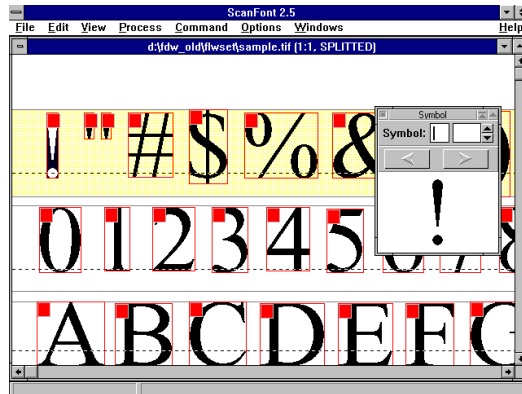
Two Techniques

You can convert your TIFF file into a VFA font file using one of the two techniques.

- ❶ Select symbols one by one and apply the **Copy VFA** command in ScanFont and **Paste Symbol** in FontLab for each symbol. This technique allows you to get several symbols from an image and place them into a font which is open in FontLab.
- ❷ Split strings and characters, Mark characters and Save VFA file. This is an automatic mode, and you can save a lot of time by use of this technique. But in some cases, usually when you try to convert a script or decorative font, this technique doesn't work.

Using the View Menu

The first menu we're going to describe is the **View** menu. There are three choices: **Template Only**, **Template and Split** and **Redraw**. If you load a bitmap file, then the first two choices of this menu are not available. Select **Split All Strings** from the **Process** menu to make them accessible.



Notice that, in comparison with the previous screen image, there are lines, selections and symbol boxes here. Using the **Template Only**, **Template and Split** commands from the **View** menu, try to switch between converted and unconverted modes.

Split All Strings allows you to actually perform an automatic splitting of the image into separate parts that probably represent single symbols. Nevertheless, the system may commit mistakes as you see with the symbol quotation marks ("). In this case, you can fix the situation using the command **Merge Symbols** from the menu item **Command**. **Redraw** lets you simply redraw the whole screen if any "dirt" occurred on the screen for any reason.

NOTE: You can get quick access to the popup menu with most common commands by pressing the secondary button in the image window.

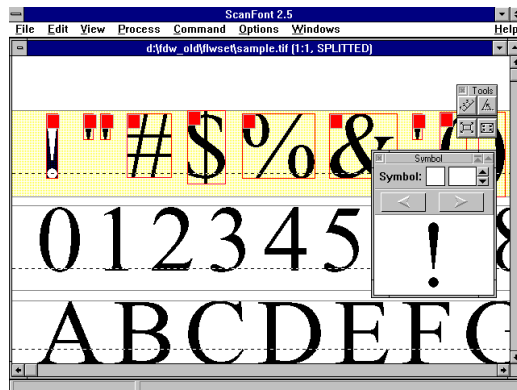
Selecting a Set of Symbols

Before working with the image you've loaded, you must think about selecting some reasonable parts of it. The simplest and most efficient way is to use the automatic selection using the command **Split All Strings** from the menu item **Process**. This lets you have all parts of your image separated. If you're not fully satisfied with the result of this operation, you may do the command **Unconvert**.

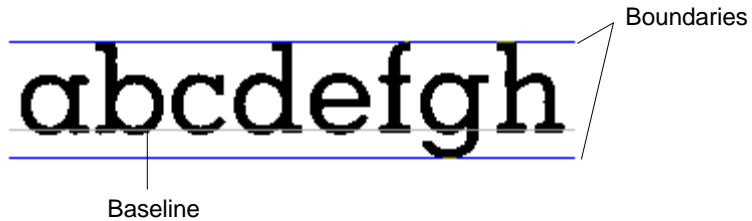
You may prefer to split the image into strings and then split the strings one by one using the commands **Split Current String** and **Desplit Current String**. This will do the same to a single string under your cursor.

- 1 Select **Unconvert** and **Convert to Strings** from the **Process** menu.
- 2 Select the first string (click the primary button on it) and select **Split Current String** from the same menu.

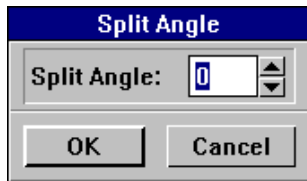
You may see the result as follows:



Only one string, the upper one, is split. As you see, each string has its own two boundaries, one baseline and several boxes which contain proposed symbols.

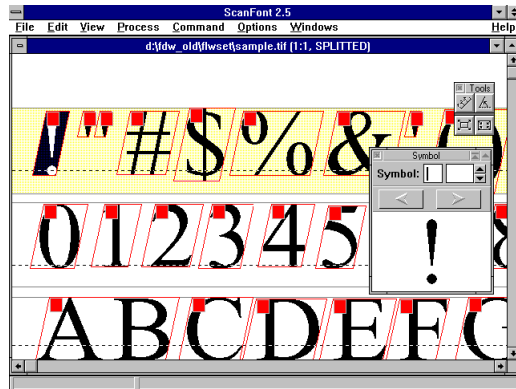



The boundaries of the string, the base line and the boundaries of the boxes can be dragged using the primary button. Perhaps you see that your font is a kind of *Italic* and you want to enter an angle for your splitting that would correspond to this font. Use the command **Enter Split Angle** from the menu **Process** before splitting and you'll see the following dialog box:




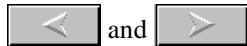
For example, enter 12 in this dialog box. Note that the maximum possible angle is equal to 45° . This system doesn't support greater angles.

Then you can apply the commands described above (**Split All Strings** or **Split Current String**) to your image and see the result.



If you feel uncomfortable setting an exact angle, you may simply draw the line that will present the desired angle. Click on the  tool of the **Tools** panel and you'll see a modified cursor. Then drag a line holding the primary key and this line will set the split angle that will be used. Each string may have its own *unique* angle!

The currently selected string is highlighted and you can move the highlight by clicking on the necessary string or using **CTRL+***, **CTRL+***  **CTRL+*** and **CTRL+ *** will also let you make another symbol current. You can merely point to the desired line or symbol with your mouse. The same is possible with the arrow buttons on the **Symbol** window:



If your scanned image contains accented characters, the accents will appear as a separate string. To group the parts of such characters together, highlight the upper string (the string of accents) and select **Merge Strings** from the **Command** menu (or press **CTRL +**). ScanFont will produce a single string from the former two. To do just the opposite select **Divide String** (**ALT +**). This will divide the current string into two *equal* parts. *Be very careful* doing this as this command will cut the string in two without taking into account the symbols that may be cut by this operation.

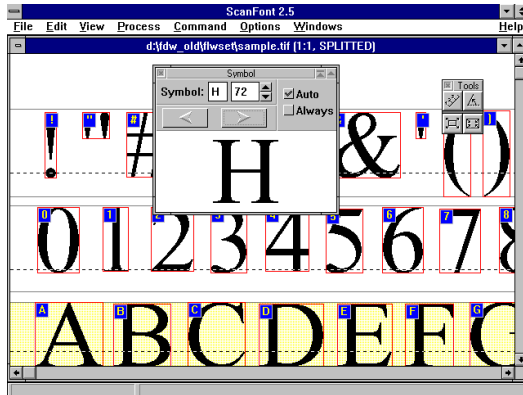
It may happen that some symbols, like quotation marks (") which contain two separate parts, should actually be tied together. To gather those prodigal parts together, click on the leftmost part and select **Merge Symbols** (**ALT +**) from the **Commands** menu. Repeat these operations as many times as necessary to gather all the run away parts together.

In the same menu item as **Commands**, you may notice two other choices - **Delete String (CTRL + Del)** and **Delete Symbol (ALT + Del)**. Those two delete the current string and the current symbol respectively. Nevertheless, if it happens that you accidentally deleted a string, pull the nearest boundary of another string so that the deleted string and the other one become one, then do **Divide String** to them, so they become separate again. Use **Split Current String** to them both and that will correct your error.

In any situation you can manually tune up each boundary by dragging it with your mouse.

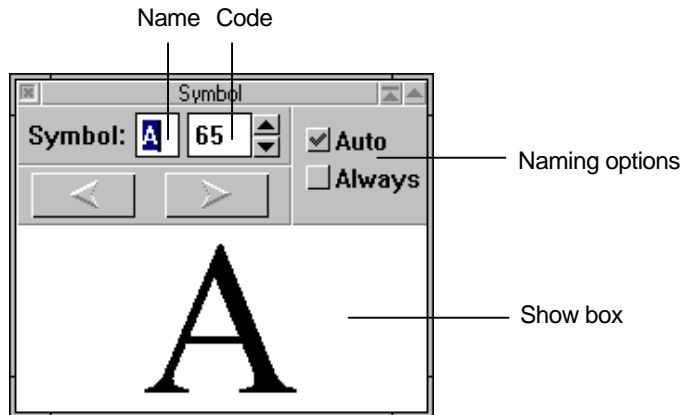
Marking Symbols

Once some symbols in your image are selected, you may have to give names to them. You can do this to all symbols separately, but, if they go sequentially inside your image as they should in a font, you may choose a simpler way. Name the very first (or the very last) symbol in that subset you want to convert later and then turn the option **Auto** on the **Symbol** panel on and then start moving forward (or backward) using the **Arrow** button on the **Symbol** panel or **CTRL+ *** and ***.** Doing this, you will see each sequential symbol in the font named appropriately.



If the option **Always** on the **Symbol** panel is on, all the previously assigned names will be removed beneath your cursor when you move with the **Arrow** buttons. Otherwise, your previously assigned names will be left unchanged. You can assign a name to a symbol using the **Symbol** panel by entering the symbol from the keyboard or by entering its code directly from the **Symbol** panel.

Symbol Panel



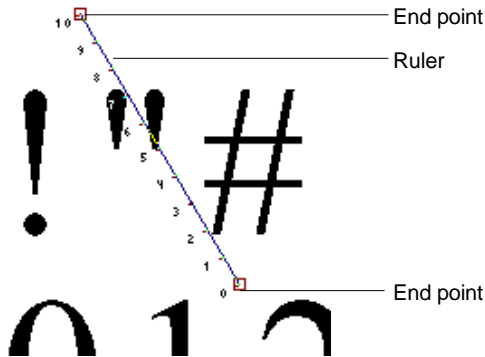
We've already described all the buttons and fields of the **Symbol** panel. There remains only one option of use to you. Click on the magnified image on this panel and you will see a changed cursor. It looks like an open palm of a hand. Holding the primary button, you can drag your symbol inside the show box of the panel.

By pressing the secondary mouse button, you open the popup menu with the most common **Symbol** panel options and commands:

✓ S how Preview
N ext Symbol
P rev Symbol
✓ A uto
A lways

Scaling

Click on the  tool of the **Tools** panel and you get a scaling instrument:



This whole **Ruler** represents 1000 FLW units as they will be seen later in FontLab. You can drag this **Ruler** as a whole by any point that belongs to it excluding both end points. If you start dragging by an end point of the **Ruler**, you can modify its size and rotate it, because the other end point of the ruler will stay fixed in this situation. You can place the **Ruler** in the way that you find most comfortable for your work, i.e., it may be attached to a part of some symbol in any direction.

Saving Fonts

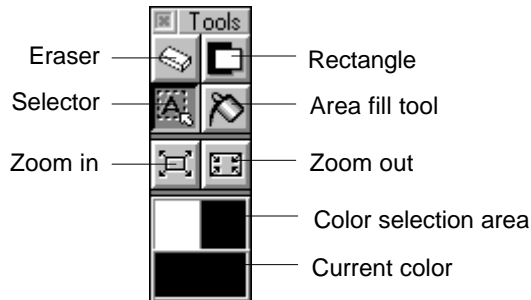
You have two ways to do this. The first one is to save the whole collection of scanned symbols as a VFA file or as a simple image. Choose respectively **Save VFA (CTRL+S)** or **Save Image (CTRL+A)** from the **File** menu and you'll see a standard file dialog box. You can work with .VFA files later using FontLab, though the .SFI format is a ScanFont property and you'll not be able to use .SFI files outside ScanFont.

The second way is to copy some part of your image to the clipboard as a simple bitmap or as a traced bitmap accessible to FontLab for future work. You can do this with the current symbol by choosing **Copy Bitmap (CTRL+Ins)** and **Copy VFA (SHIFT+CTRL+Ins)** from the menu **Edit** respectively. You can save a split box as a single symbol or some uncounted part of the image using the **Tools** panel (see below).

Editing Images

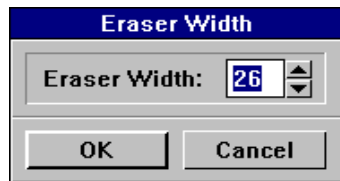
The Tools Panel

As you may have noticed, if you do **Unconvert** from the **Process** menu or if you load an image that has no string information, you'll see a special **Tools** panel on the screen. If it is not seen for any reason, select it from the **Windows** menu.



⌚ **Eraser**

If you select **Eraser**, you'll be able to erase some areas of the image by holding the primary button. If you click on this tool holding **CTRL**, you can enter the eraser width via the following dialog box:



⌚ **Rectangle**

It may happen that you need to clear up some area of your image manually before converting it to VFA format. For this case there is a special tool named **Rectangle** in our toolbox. Select it and draw a rectangle with the primary button. **Rectangle** uses the current color. If you draw rectangles holding **SHIFT**, you'll be able to draw squares only.

🕒 **Selector**

If you hold the primary button pressed on this tool for some time, you'll get the following picture:



These two items represent three variants of the **Selector** you can use to select an area.

The first variant lets you draw rectangles using the primary button.

The second one allows you to draw a curve with the primary button using the usual **freehand** or **polygon** technique. Drag the cursor by holding the primary button or add a line segment by clicking it. Click the secondary button to finish selection.

The **Selector** tool is used for selecting an area on your image that will be used later by the **Copy VFA** or **Copy Bitmap** operations.

🕒 **Area Fill Tool**

With this tool, you can fill black areas with white or white areas with the black. Select this tool and click the primary button anywhere in the area you want to fill.

🕒 **Zoom In and Zoom Out**

If the image is too large or too small for comfortable work, you can reduce or enlarge it with the **Zoom** tools.

To reduce the image, hold **SHIFT** key and click on the **Zoom Out** tool.

To enlarge part of the image, click on the **Zoom In** tool and select a rectangular area you want to enlarge.

To switch to the normal (1:1) mode, click on the **Zoom Out** tool.

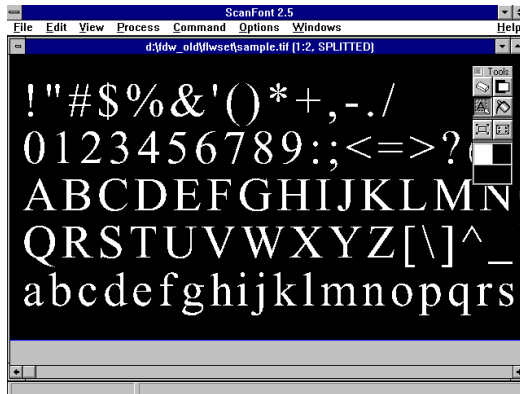
You can modify the zooming modes with the **Zoom In** and **Zoom Out** commands from the **View** menu.

🕒 **Color Selection**

You may wish to change the current color. Click on the desired part of the **Color Selection Area** (black or white) and the current color changes. See the **Current Color** box to be sure.

Reversing an Image

You may wish to reverse the image you have on the screen. To do this select **Reverse Image** from the menu **Process**. You see the whole screen as follows:



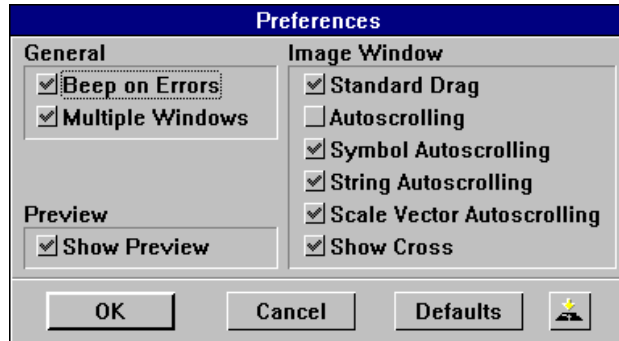
As you may see, the colors of some additional instruments (say, the **Ruler**) have correspondingly changed in accordance with the newly changed color of the image.

Options

Let's describe the items of the **Options** menu.

Preferences

Select **Preferences** from the **Options** menu and you get the following dialog:



□□ General Options

Beep on Errors will cause the computer to beep if an error occurs. **Multiple Windows** lets you open many windows simultaneously. If this option is on and you're making a **Save** command, then all named symbols from all opened windows will be gathered together into one file. *Be careful* selecting this option.

🕒 Preview

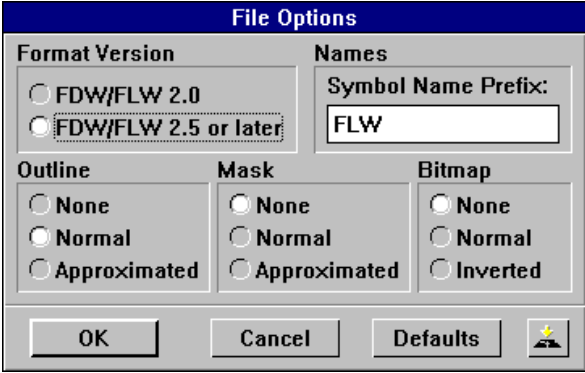
Show Preview lets you see a magnified image of the current symbol in the **Show** box of the **Symbol** panel.

🕒 **Image Window**

If **Standard Drag** is on, you can drag by holding the primary button and stop drawing when you release the button. If it is off, you can drag in this way - click the primary button to start drawing and click then the secondary button to stop it. **Symbol Autoscroll** (if on) lets you follow the current split box when moving using the **Arrow** buttons, fully redrawing the whole screen. **String Autoscroll** (if on) lets you do the same to strings. **Scale Vector Autoscroll** (if on) lets you see the place on the screen when the scaler is positioned when you select **Specify Scale** from the **Process** menu. If **Show Cross** is on, you'll see a cross (like a sight) when you use **Pen** from the **Tools** panel.

File Options

Select **File Options** from the **Options** menu and get the following dialog box:



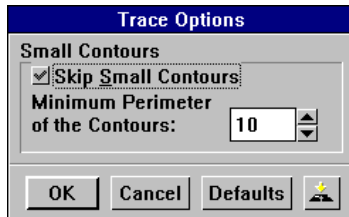
The dialog box is titled "File Options" and contains the following sections and controls:

- Format Version:** Two radio buttons: "FDW/FLW 2.0" (unselected) and "FDW/FLW 2.5 or later" (selected).
- Names:** A text field labeled "Symbol Name Prefix:" containing the text "FLW".
- Outline:** Three radio buttons: "None" (unselected), "Normal" (selected), and "Approximated" (unselected).
- Mask:** Three radio buttons: "None" (selected), "Normal" (unselected), and "Approximated" (unselected).
- Bitmap:** Three radio buttons: "None" (selected), "Normal" (unselected), and "Inverted" (unselected).
- Buttons:** "OK", "Cancel", "Defaults", and a help icon (lightning bolt).

The box **Format Version** lets you switch between two possible versions of the format of SFI files. The box **Symbol Name Prefix** lets you enter the font prefix that will appear in the VFA files created by ScanFont. Each symbol will have this prefix. The **Outline** box allows you to select the type of outline that will be saved into VFA files. If you select **Approximated**, an automatic approximation algorithm will try to represent the outline as a set of Bezier curves and lines. You can choose the same variants in the **Mask** box. The box **Bitmap** lets you choose among three variants of bitmap saving.

Trace Options

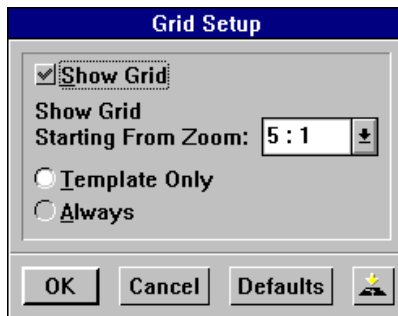
Select **Trace Options** from the **Options** menu and get the following dialog box:



The option **Skip Small Contours** (if on) lets you skip bad contours that have seemingly appeared in the image because of the usual noise of scanning machines. **Minimum Perimeter of the Contours** allows you to set the threshold perimeter of the contour that will be skipped. All those contours that are of a smaller perimeter than this will be skipped by any tracing operations.

Grid Setup

Select Grid Setup from the Options menu and see the following dialog box:



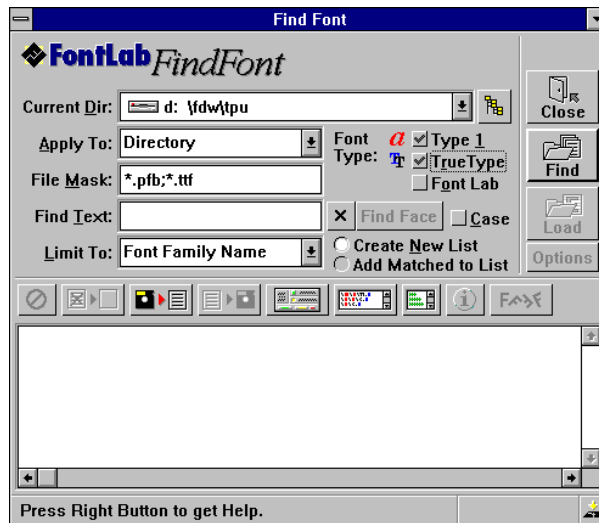
Show Grid toggles the grid lines appearance in the image window. **Show Grid Starting From Zoom** sets the Zoom In mode from which grid lines will appear. With **Template Only** and **Always** options, you can select modes where the grid will appear.

Find Font

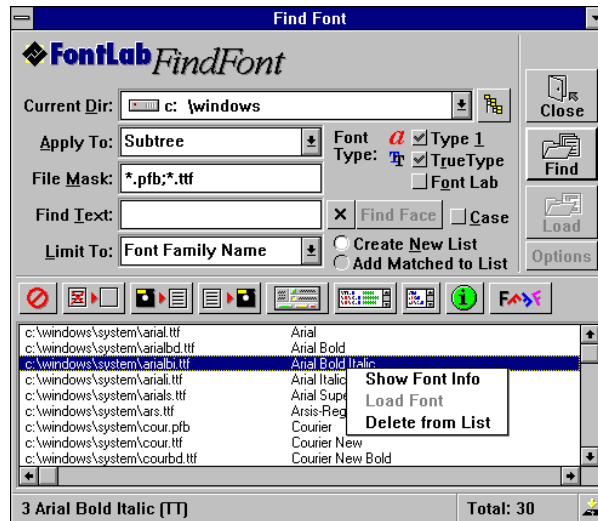
This is a very portable and easy-to-use program for finding fonts which currently exist in your computer. We highly recommend it for a quick search if you want to gather information on the font files installed in your computer.


Getting started

When you run this program for the first time, you enter the following dialog box:



Set the desired options, switch on the types of font files you want to find and click on **Find**. FindFont will perform a search and you'll see the list of all found files in the Result List Window.



You can save this list, edit it and get information about each font file found in your disk. Point to a file in the **Result List Window** and click on the  button. You'll see an information table which may be useful to you in the future. If FindFont is loaded from Fontlab, you can load the desired file immediately into Fontlab.

The FindFont Controls

① **Current Dir**

This lets you select the current drive/directory. The current directory will be used as a starting directory if you set **Apply To** for **Directory** or **Subtree**.

① **Apply To**

Select a type of disk source to find fonts. If you select **Subtree**, all sub-directories of the current directory are searched as well.

① **File Mask**

Specify a filename pattern template that will be used for your search. Only those files that match this template will be scanned. **File Mask** accepts all valid DOS characters, including wildcards (* and ?). You must separate multiple file masks with a semicolon (;)

① **Find Text**

Specify a text string to look for in scanned font files.

① **Limit To**

Select a part of the font file to which string searching is applied. For example, if you want to find all fonts which belong to the *Time* font family, you should enter times in the **Find Text** field, switch off the **Case** option and select **Limit To Font Family**. To find all Adobe fonts, enter **Adobe** in Find Text and **Limit To Notice** or **Copyright**.

① **Create New List**

If this option is on, FindFont will create a new Results List while performing a search.

① **Add Matched to List**

If this option is on, FindFont will add all searched fonts to the current FindFont Results List.

① **Type 1, TrueType, FontLab**

These options let you specify the types of files to be searched.

① **Case**

When the **Case** option is on, FindFont distinguishes uppercase from lowercase when performing a search.

🕒 **Find Face**

If you switch on the Find Face check box, you can open the Select Face dialog box where you can choose typeface and other parameters which describe the fonts you want to find.

🕒 **Close**

This will exit FindFont. If you run FindFont from FontLab, FindFont will be closed when you leave FontLab.










🕒 **Find**

Click here to start scanning for font files.

🕒 **Load**

Click here to retrieve a font file into FontLab. If required, the font will be converted into the internal (VFA) format. You may load a font only if you run FindFont from within FontLab.

The FindFont Toolbar (from left to right)

Icon	Name	Description
	Clear button	Click here to clear the current Results List.
	Delete button	Click here to delete the selected file from the Results List.
	Open button	Click here to open a previously saved FindFont Results List.
	Save button	Click here to save the current FindFont Results List as a single (*.lst) file for future use.
	Options button	Click here to set additional FindFont Options.
	Show button	Shows in the Results List. Allows you to select which kind of information will be stored in the Results List: <ul style="list-style-type: none"> ❶ full file name; ❷ font name; ❸ short file name and font name; ❹ full file and font names.
	Sort button	Sorts the Results List. Enables you to change the Results List sorting order. You may sort the Results List by file or font names. Sorting by font name is available only if the List known formats only option is specified.
	Info button	Click here to see some additional information about the selected font.
	Convert button	Click here to open a Convert dialog box where you can select many options of converting fonts from one format to another.

Results List Window

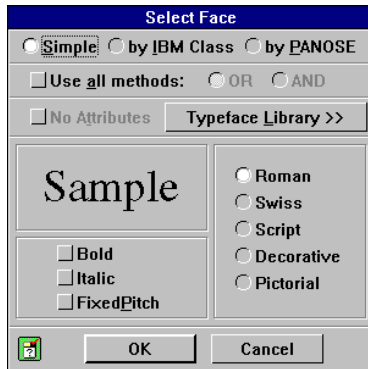
This is the FindFont Results List window. It contains the names of the font files found during the last search. Double-click to begin a quick search in the list. Click the secondary button to display an additional menu. This menu will contain the following fields:



Show Font Info
Load Font
Delete from List

equal to the **Info** button
equal to the **Load** button
equal to the **Delete** button

Find Face Dialog

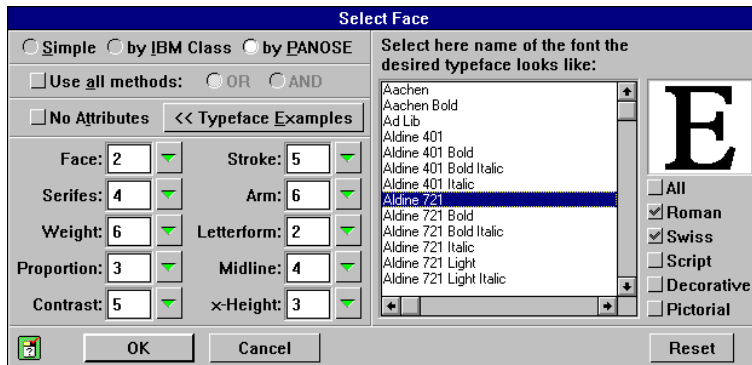


In the Select Face dialog box, you can describe fonts you'd like to find. You can do this by three methods: **Simple**, **by IBM class** and **by PANOSE** or combine **all methods** by **AND** or **OR** boolean operations.

In all the modes, you have to describe the fonts. In the **Simple** mode, you just select a type of font (Roman, Swiss etc.) and style (Bold, Italic and FixedPitch). **Sample** shows you how these fonts look.


In the IBM and PANOSE modes, enter numbers which describe fonts in the IBM or PANOSE standard.

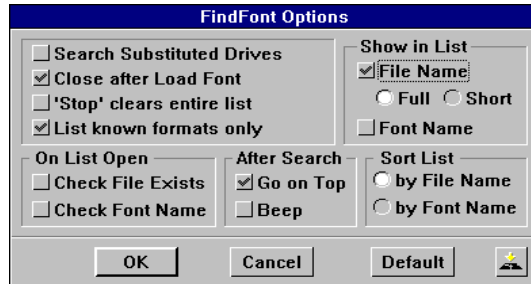
PANOSE Classification with the Examples Library:



In all modes, you can press the **Typeface Examples** button and select an example of the font. While searching, FindFont will select only fonts whose parameters are close to the ones you selected.

Options Dialog

When you press the **Options** () button, you'll see the following dialog box:



Let's describe the fields of this **Options** box.

① **Search Substituted Drives**

This option lets you search throughout those drives that are substituted by Windows.

① **Close After Load Font**

If this is on, FindFont will be closed automatically after loading the font. This is valid only when running from within FontLab.

① **Stop Clears Entire List**

If this is on and you break FindFont during its search by clicking on **Close**, the whole Results List will be cleared.

① **List Known Formats Only**

This option, if on, lets you perform your search throughout all files, no matter what extension they have. FindFont will then peek into those files trying to figure out what type of font they belong to. If it is off, FindFont will search only those files with the proper extension.

① **File name (Full, Short)**

This lets you choose from two various forms of file representation in the Results List.

① **Font Name**

If this is on, FindFont will show all font names of the found files in the Results List.

🕒 **Check File Exists**

If on, FindFont will check after loading a list file whether the font files from that list file really exist at present in your computer.

🕒 **Check font name**

FindFont will check (if this option is on) the internal structure of each font file from the loaded list file to clarify whether the structure corresponds to the declared type of this file.

🕒 **Go on Top**

If this is on, FindFont will pop up to the top of your desktop when the search is completed.

🕒 **Beep**

If this is on, FindFont will beep after the search is completed.

🕒 **By File Name**

If this is on, FindFont will sort all files in the Results List by file names.


🕒 **By Font Name**

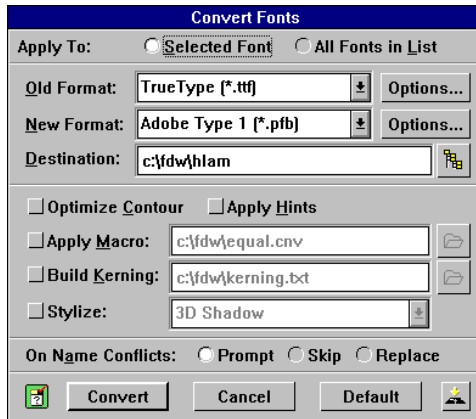
If this is on, FindFont will sort all files in the Results List by their font names.

🕒 **Save button**

As usual in FontLab, this dialog box has a **Save** button. Click on this button to save the current settings on disk.

Converting Fonts

When you find some fonts, you can press the  button and enter the **Convert** dialog box.




In this dialog box, you can select various options for fonts conversion.

First, you must select the **New Format** of the font and its **Destination** directory. You can change conversion options by pressing the corresponding **Options...** button.

If you run FindFont from FontLab, you can apply additional procedures to the font conversion. You can **Optimize Contour** for the all symbols of the font, automatically **Apply Hints** if you convert the font into VFA or Type 1 format, **Apply Macro** program, automatically **Build Kerning** pairs and run a **Stylizer** to transform this font.

You can convert only one font that you select in the Results List or all fonts in the list.

Font Information

With FindFont, you can easily get a short description of a font. These are presented in three formats: *TrueTyp*, *FontLab* (VFA and Type). To do this, select a font in the **results** window, and press the **Info** () button. Or select a font, press the secondary mouse button and select **Show Font Info** in the popup menu.

True Type Font Information



The image shows a dialog box titled "True Type Font Info" with a blue header. It contains several fields and checkboxes for font information. The fields are: PS FontName (CommonBullets), Vendor (empty), Version (empty), FontBBox (16, -11, 1216, 1270), Weight (400, Regular), IBM Class (0, 0), UnitsPerEm (2048), and Panose (2, 11, 6, 3, 5, 3, 2, 2, 2, 4). There are checkboxes for Bold, Italic, and FixedPitch, and a checkbox for Unicode mapping. A Close button is at the bottom.

True Type Font Info									
PS FontName:									
CommonBullets									
Vendor:									
Version:									
FontBBox:									
16	-11	<input type="checkbox"/> Bold		<input type="checkbox"/> Italic					
1216	1270	ItalicAngle:		0.0					
Weight:		400		Regular					
IBM Class:		0 0		<input type="checkbox"/> FixedPitch					
UnitsPerEm:		2048							
Panose:		2 11 6 3 5 3 2 2 2 4							
<input type="checkbox"/> Unicode mapping									
Close									

Let's describe the fields of this box. Get quick help by clicking the secondary button on the desired item.

PS FontName contains the PostScript name of this font.

UnitsPerEm means the font coordinate space. Valid range is from 64 to 16384.

The **Vendor** box contains the four symbol identifier for the vendor of the typeface.

FontBBox contains the rectangle (lower left and upper right corners) which encompasses the symbol imageable area.

If **Unicode Mapping** is on, the symbols in this font have Unicode indexes.

If **Italic** or **Bold** is on, the font has these characteristics.

Weight means the weight of the font. Indicates visual weight (degree of blackness or thickness) of the symbols in the font. The right part of the box **Weight** (separated from the left one with a vertical line) contains the font style name.

IBMCSS classifies a font design as to its appearance.

FixedPitch means that your font is monospaced (e.g. Courier or LetterGothic).

Panose describes the visual characteristics of the typeface. Windows v3.1 uses Family, Serif and Proportion in the font mapper to determine family type.

Version means the font version identification string.

FontLab Font Information

VFA Font Info	
Menu Name:	ZapfHumnst BT
PS Name:	ZapfHumanist601BT-Roman
Vendor:	Bits
Version:	mfgpctt-v1.52 Monday.
UniqueID:	4000000
<input type="checkbox"/> Bold	<input type="checkbox"/> Italic
Weight:	Roman
ItalicAngle:	0.0
IBM Class:	8 2 <input type="checkbox"/> IsFixedPitch
Panose:	2 11 5 2 5 5 8 2 3 4
Close	

Let's describe briefly the fields of this box that *haven't been describe* in the previous case. Get quick help by clicking the secondary button on the desired item.

Menu Name is the Windows menu name (the name the user sees).

PS Name contains the name presented to the PostScript 'findfont' operator.

UniqueID contains the PostScript font ID. Number unique to each Type 1 font.

Type 1 Font Information

Type 1 Font Info			
FontName: Helvetica			
FontBBox: -174 -299 1050 944			
Weight: Regular		ItalicAngle: 0.0	
Version: 001.004	St.Encoding <input checked="" type="checkbox"/>		
UniqueID: 263806	IsFixedPitch <input type="checkbox"/>		
ForceBold <input type="checkbox"/>			
Scale: <input type="text"/>	Shift: <input type="text"/>	Fuzz: <input type="text"/>	
Blues: -23 0	HStem: 77.0		
Other: 269 283	VStem: 84.0		
Close			

Let's describe briefly the fields of this box that *haven't been describe* in the previous cases. Get quick help by clicking the secondary button on the desired item.

Font Name contains the name presented to the PostScript 'findfont' operator.

If **ForcedBold** is switched **ON**, you can force bold appearance at small sizes.

If **St.Encoding** is **ON**, this means that the encoding for this font is Adobe Standard Encoding.

Scale contains the point size at which to deactivate overshoot suppression.

Shift contains the overshoot enforcement.

The value in **Fuzz** extends the range of alignment (Blue) zones.

Blues contains the value of the font vertical alignment zones. This is global hint data.

Other contains the value of the additional bottom alignment zones. This is global hint data. **HStem** contains the value of the array of common horizontal stem width. This is global hint data.







Creating Fonts with Precision

Until now, we have discussed visual drawing techniques without paying much attention to placing objects at exact locations and precisely setting their dimensions. However, if you wish to obtain professional results, your symbols and other graphic images must accurately express the shapes and proportions of their original design. To do this, you must maintain the relative positions of all characteristic points, including lengths of straight segments, positions of endpoints and control points of curves, and alignment of nodes along horizontal and vertical lines, to mention a few.

In fact, you should mark all important proportions of your future symbol or any other image **BEFORE** proceeding with defining its outline. Otherwise, you will spend much time and effort reshaping your contour to maintain the consistency of its design.

This section consists of three parts. In the first part, we will discuss main characteristics of symbols, the units of measurement used in FLW, and three new tools: **Meter**, **Set**, and **Points**. The second part explains general methods that let you draw all sorts of objects with precision. The third part (the "Design Consistency" and "Using Hints" chapters) is more fonts-specific.

Before experimenting with the techniques discussed in this section, make sure that the guiding objects you want to create will be displayed in the **Edit** window. The following table lists the icons contained in the **Options** box that switch on/off guiding objects.

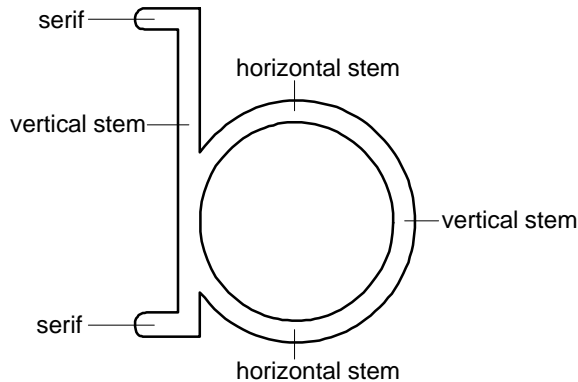
<u>To display</u>	<u>Click on</u>
Grid	
Guidelines, guidepoints	
Hints	
Mask	
Bitmap	
Subroutines	

Examining Characters

We will begin the discussion of precision drawing techniques with an overview of the main characteristics of symbols. By maintaining the consistency of these characteristics throughout your font, you will achieve good results when reproducing your font at any size and resolution.

Although typography has a wide variety of terms defining miscellaneous parts and dimensions of symbols, we will limit our discussion to the features which are pertinent to the Adobe Type 1 font format.

Character Components



This picture shows the main elements of existing symbols (see also the "Vertical Dimensions" paragraph below). Main strokes of characters are called *stems*; there are two types of stems: horizontal and vertical.

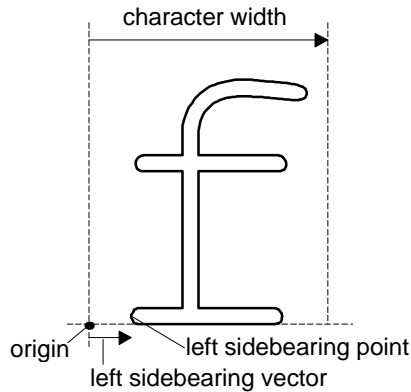
The Type 1 format provides the means for preserving proportions of symbols at any resolution and size. This is achieved by fixing stems using pairs of horizontal and vertical lines called *hints* (for more details, see the "Using Hints" chapter of this section). Since it is necessary to keep the proportions and relative positions of all elements, including curved segments and serifs, they are also considered horizontal and vertical stems (as shown on the above picture).

TrueType Fonts

TrueType fonts, in fact, use only lines and curves of the 2nd degree instead of Bezier curves. Nevertheless, using FLW you will always work with Bezier curves. We have provided you with automatic transformations of the curves from TrueType fonts into Bezier curves and vice versa. Those transformations will occur when you export or import a TrueType font using the FLW system. You can be sure that all conversions will be performed correctly in spite of some negligible loss of accuracy. Frankly, we have incorporated some artificial intelligence into the system of transformation to take into account all possible dubious situations.

Horizontal Dimensions

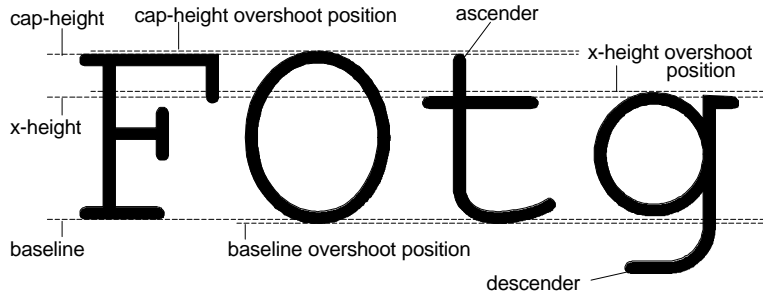
Horizontal measurements of symbols strongly affect the visual appearance of strings printed with your font. In fact, the mutual positions of symbols in a string are determined by three parameters: character widths, left sidebearings, and kerning values. (Kerning is described in the "Advanced Features" section under "Adjusting the Spacing of Characters.")



- ① The initial reference point of each character is its *origin*. All other horizontal dimensions are measured from it; it is the default zero point of the absolute coordinate system used by FLW. When you enter text at a certain position, the first character's origin coincides with the position.
- ① The length of the vector which connects origins of two neighboring symbols in a string is the first character's *width*.
- ① The *left sidebearing* value is the length of the vector which starts at the origin and terminates at the leftmost point of the character.

FLW shows margins and *baselines* (which are discussed below) with light gray dotted lines as shown on this picture.

Vertical Dimensions



When reproducing text with various scaling factors, it is necessary to keep relative proportions between the following dimensions (they are measured from the *baseline* which coincides with bottom lines of most flat characters):

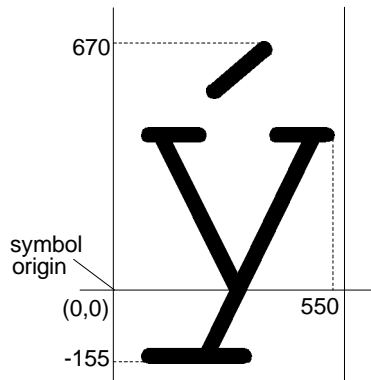
- ① *cap-height*—the top of flat capital letters;
- ① *x-height*—the top of non-ascending, flat lower-case letters;
- ① *overshoot positions*—coordinates of uppermost and lowermost points of round characters (excluding ascenders and descenders of lower-case letters).

You may also fix some other proportions, such as ascender-height and descender-depth. The methods that control the rendering of symbols at limited resolutions are discussed in this section under "Using Hints" and in the "More About Ensuring Proper Reproduction" chapter of "Advanced Features."


Units of Measurement


In order to achieve excellent quality when reproducing symbols at any size and resolution, they are created in their own coordinate system (which is called *character space*) distinct from coordinate systems used by output devices. The typical scaling ratio used by PostScript font programs is 1000 to 1, which means that 1000 character space units will be scaled down to 1 pt (before the output device applies any procedures to scale your font to the desired size). Your symbols must not deviate too far outside the 1000 units limit, and the maximum possible absolute coordinate is 2000.

In FLW, all characters are created in the character space coordinate system; therefore, 1000 FLW units correspond to 1 pt as described above. The (0,0) point always coincide with the symbol's origin.



Determining Coordinates

To determine the coordinates of a point, click the secondary button on it regardless of the tool you are currently using. (Of course, you cannot do so when performing an operation that may be terminated by pressing the secondary button.) Your cursor will change to a  and the coordinates will be displayed both under the cursor and in the status line.

If you hold down the secondary button and drag the  cursor, it will stick to all vectors, curves, nodes, control points, and those guiding objects that are selected in the **Snap To** field of the **Options** window (for more details on available guiding objects, see the following chapters of this section).

Positioning Margins and Baselines

When you start working on a new character, FLW sets its width to the default value of 500. If you want your font to look good when printed, you have to adjust the widths of your characters (unless you design a monospaced font like Courier). Normally, you set approximate positions of margins when you draw the character's outline, and then use the **Set Sidebearings...** command to fine-tune them. (This command is described in the "Advanced Features" section under "Adjusting the Spacing of Characters.")

To change a margin's position, click on it with the **Edit** tool and keep the mouse pointer on the line until the cursor becomes a ↔ (this delay protects you from accidentally moving the margin; for more details, see the "Locking Delays" chapter that follows). Then, drag the margin to the desired location. As you move the line, the status bar will indicate its current X coordinate. If you move the left margin and the coordinate system's origin coincides with the origin of your character, the current horizontal position of the margin is calculated from its former position. Since margins are moved separately, you change only the symbol's width by repositioning the right margin whereas both the left sidebearing value and the width are changed by moving the left margin.

You can also change the sidebearing and width values by pressing **CTRL** and then clicking on either margin with the **Edit** tool. A dialog box will appear in which you will be able to set exact figures in character space units.

If you want to adjust the vertical position of your character about the baseline, it is more convenient to move the baseline rather than the entire symbol. Baselines are moved in the same fashion as margins. If you click on a baseline while holding down the **CTRL** key, a dialog box will appear letting you type or select an offset from the original baseline's position. If you specify a negative number, the baseline will be moved down.

Using the Guidelines

If several nodes of your contour must have equal horizontal or vertical coordinates, it is useful to fix these coordinates with nonprinting lines called guides, and then align nodes to them. You can define guidelines either for the current symbol (local guides) or for all symbols of your font (global guides).

Manipulating Guidelines

- ❶ **To place a guideline**, with any tool active, point to the top or side ruler to get a horizontal or vertical guideline, respectively. Your cursor will change to a \updownarrow . Now hold down the button and drag onto the editing area. As you move the blue dotted line that appears, the status line shows its coordinates (Y for horizontal and X for vertical guides). When you are finished, release the button.

To add a global horizontal or vertical guideline, hold down **SHIFT** while clicking on the ruler. The dotted line will be dark red.

- ❷ **To reposition a guideline**, use the **Edit** tool to drag it to a new location. Holding down **ALT** will constrain your movements to one coordinate at a time. Or, point to it and press **CTRL** and the primary button. Hold them down until the **Guideline** dialog box appears, and then type or select the desired guideline position.
- ❸ **To erase a guideline**, drag it off the editing area using the **Edit** tool. Or, point to it with your cursor, press and hold down the primary button, and click the secondary button.

Since global guides are used by all symbols of your font, moving or erasing them for a specific symbol affects all others as well.

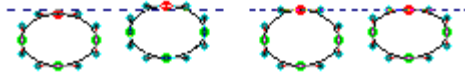
- ⌚ **To erase several local guidelines at once**, select **Remove All Guides** from the **Hints/Guides** menu. A drop-down menu appears offering to erase all horizontal or vertical guides, or all guides set for the current symbol.
- ⌚ **To duplicate global guides with local guides** set at the same positions, select **Paste Global Guides** from the **Hints/Guides** menu and specify whether you want to paste horizontal, vertical, or all global guides.

This command is especially useful when you reposition global guides but still need to fix their previous coordinates for a specific symbol.

Snapping to Guidelines


Once you have defined guidelines, cursors of all available tools will be forced to stay on them when performing any operation. Your cursor snaps to a guideline if it is within a certain distance from it. You may change this distance in pixels by typing or selecting the desired value in the **Guides** entry field of the **Preferences** dialog box, which is available after choosing **Preferences...** from the **Options** menu. Since the value is in pixels, points snap differently in normal and magnified views.

If you define guidelines after placing nodes, these nodes will retain their positions regardless of how close they are to the new guidelines. To align nodes to guidelines, select **Snap to Guides** from the **Transform/More...** menu. As all other **Transform** menu commands, **Snap to Guides** processes a highlighted area or an entire symbol if no selection has been made.



After applying the
Snap to Guides command

Note that this command slightly reshapes contours since FLW moves only the nodes that are close to guides. If you wish to align contours without distorting them, use the **Move** tool to reposition them via nodes. When your cursor is close to a guide, it will stick to it, and the contour will be perfectly aligned. The rule is: use the **Snap to Guides** command to improve shapes of your contours; once a contour is shaped, use the **Move** tool to align it.

If you wish to disable snapping to guidelines but leave them on your screen, select **Edit Window...** from the **Options** menu and switch off **Guides** under **Snap To**. If you click on the  icon in the **Options** box, both local and global guides will disappear from view and no snapping will be performed.

Locking Delays

FLW protects you from accidentally performing "dangerous" operations by the means of locking delays. Whereas all ordinary operations are done right after you press the primary mouse button, to accomplish one of the tasks listed in the **Locking Delays** dialog box shown below, you have to wait for several moments without releasing it (as well as **SHIFT** or **CTRL**, if necessary) or moving your mouse.

Operation	Delay (ms)
Adding Nodes	0
Digital Editing	200
Adding Guides/Hints	0
Editing Global Objects	100
Editing Subroutines	50
Moving Baselines	200

Enable Locking Delays

OK Cancel Default

We have already discussed five of them: **Adding Nodes** (refer to "Modifying Outlines Using the Edit Tool" and "Converting Segments Using the Edit Tool" in the "Drawing Outlines" section), **Moving Baselines** (see the "Positioning Margins and Baselines" chapter of this section), **Digital Editing** (see step 2 under "Placing Guidelines"), **Adding Guides** (see step 1 above), and **Editing Global Objects** (this delay occurs whenever you wish to move a global guide).


- ⌚ **To change the duration of locking delays**, choose **Locking Delays...** from the **Options** menu and type or select desired values. If you wish to disable a delay, enter "0" in the corresponding field. To switch on/off all existing delays, use the **Enable Locking Delays** button.
- ⌚ **To make your changes permanent**, click on the **Save to Disk** icon that appears in the lower right corner of the dialog box.
- ⌚ The **Default** button lets you reset all durations to default values, that is, to the values that are currently active. For example, if you set the **Adding Nodes** duration to 50 after entering the **Locking Delays** dialog box for the first time and then click on **Default**, the value will be reset to 0. However, if you click on **OK** after entering 50 in the **Adding Nodes** field, the new value will become active, so the next time you specify 0 in this field and click on **Default** your 0 will be reset to 50.

We will discuss other operations appearing in this dialog box in the following chapters of this manual.

Using the Grid

Displaying the Grid

The grid is another means of aligning nodes and segments horizontally or vertically. It is useful if you want all distances between nodes and objects to be multiples of two specific numbers called horizontal and vertical grid steps.

To display the grid, click on the  icon contained in the **Options** box. In fact, the grid is a series of nonprinting equally spaced horizontal and vertical lines, but FLW displays only the points of their intersection to avoid cluttering the **Edit** window if the grid steps are relatively small. The marks on the rulers correspond to the positions of the grid's horizontal and vertical lines.

The grid's initial reference point always coincides with the *guides origin* (*), the default position of which is (-13, 700). Besides the grid, the guides origin determines absolute positions of all global guiding objects relative to the baseline and margins of your symbol. Therefore, if you move the guides origin (we will discuss this procedure under "Using the Set Tool" later in this section), all other global guiding objects will be moved accordingly.

By default, the (0, 0) point of each symbol's coordinate system coincides with its origin (see the "Units of Measurement" chapter of "Transformations"). Therefore, the grid lines may have coordinates that are not multiples of grid steps (for example, if the vertical grid step is 18, the first horizontal grid line will be at Y=16). You can avoid this situation by attaching the (0, 0) coordinates to the guides origin.

To change the default grid steps (50), select **Edit window...** from the **Options** menu and specify the desired values in the **Grid W** (horizontal step) and **Grid H** (vertical step) entries.

Snapping to the Grid

Everything we said under "Snapping to Guidelines" is also true for the grid. Just imagine that all the grid's points are connected with guidelines and you will understand how it works. Naturally, you can enable/disable snapping to the grid by choosing the **Edit window...** command from the **Options** menu and clicking on the **Grid** button in the **Snap To** field. We must note that guidelines have priority when both **Snap To Grid** and **Snap To Guides** are on.

Using the Guidepoints

Unlike the guidelines and grid, guidepoints let you mark single positions in the **Edit** window. Once you've marked such a point, you can use it as the (0, 0) point of the local coordinate system in which you will perform measurements and digital editing. Guidepoints are also useful when you wish to experiment with your contour's shape. Before reshaping your contour, fix important positions with guidepoints; then, if you are not satisfied with the results achieved, you will always be able to return it to its original state.

The special tool used for marking positions in the **Edit** window is **Points** (you may also use the **Meter** tool as described in the next chapter). To make it active, click on its icon in the toolbox or press **5**.

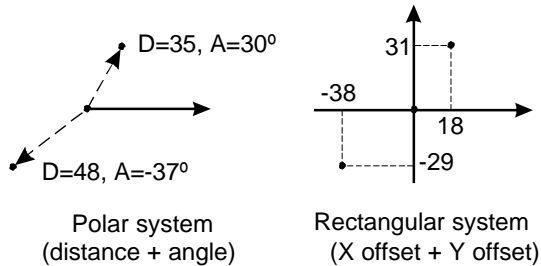
To place a guidepoint, move your cursor over the desired position and click the primary button. Or, press the button anywhere in the **Edit** window and drag your mouse to the necessary location. Then, release the button to "drop" a guidepoint or click the secondary button to cancel the operation. If you hold down the **CTRL** key, the guidepoints will snap to neighboring guiding objects, and **ALT** lets you move the guidepoint one coordinate at a time. The drag-and-drop method is useful when you want to mark a point in a free area since the status line reveals current coordinates of your cursor while you drag it.

To erase a guidepoint, use the **Edit** or **Erase** tool as explained in the "Drawing Outlines" section. If you wish to avoid accidental erasing of other points while using the **Erase** tool, hold down the **CTRL** key.

If you wish to erase all existing guidepoints, select **Remove All Guidepoints** from the **Hints/Guides** menu.

Measuring Distances and Angles

Besides clicking the secondary button on an object to determine its absolute coordinates, you may use the **Meter** tool to measure coordinates of points that are relative to any point in the **Edit** window. This point is the origin of a temporary coordinate system used by the **Meter** tool for determining relative positions of all other objects. You may use either a polar or rectangular coordinate system. The following picture explains how positions of points are represented in both systems:



- ❶ To activate the **Meter** tool, click on its icon in the toolbox or press 7.
- ❷ Click on the point in the **Edit** window that you want to make the origin of a temporary coordinate system and hold down the button. The point will be surrounded with a circle. If you click close to a node, curve, vector, or any guiding object, the circle will be forced to stay on it.

- ③ Drag your cursor towards the object whose coordinates you wish to measure. You will notice a light blue "rubber" vector, which follows the cursor. The red dot situated at the end of this line snaps to all objects listed in step 2, so you will be able to determine their precise coordinates. By default, they are measured in the polar system, so the status line indicates your current position as an offset (**D**) and angle (**A**) from the reference point you have set in step 2.

The **SHIFT** key lets you constrain your movements. If the reference point does not belong to a vector or curve, or is on top of a node, you will be able to move only horizontally or vertically while holding down the constraining key. Otherwise, your movements will be limited to a line that is orthogonal to the contour containing the reference point.

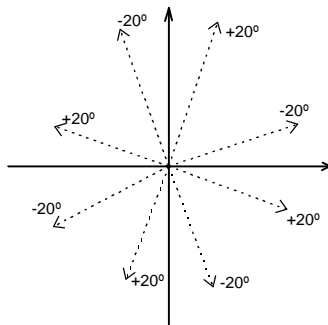
- ④ **To place a guidepoint at your current position**, click the right button.
- ⑤ When you are finished measuring coordinates, release the primary button. Then, proceed with another reference point or switch to another tool.

Customizing the Meter Tool

If you double-click on the **Meter** tool icon in the toolbox, a dialog box appears that lets you control the way this tool operates.

- ① From the upper field of this dialog box, select the coordinate system you want to use. If you click on **Rectangular**, the status line will indicate your current position as horizontal (**DX**) and vertical (**DY**) offsets from the reference point.

If you select the *Angular* coordinate system, all angles will be calculated in a range of $[-45^\circ..+45^\circ]$. That means that all your angles are calculated exactly from the axes of the ordinary rectangular system. Angles will be calculated from the nearest axis as positive if the angle goes clockwise from the axis and as negative otherwise.



FLW angle measurement system

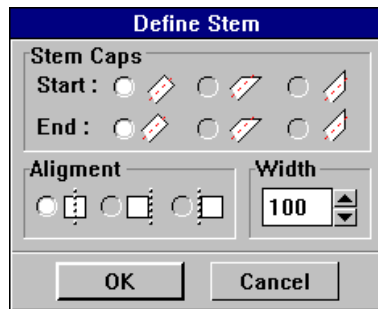
- ① In the **Breakpoint** field, type or select the percentage of line length where you want guidepoints to appear after clicking the secondary button. Guidepoints are placed along the vector's direction at an offset obtained by multiplying its length and the given percentage. By default, the value is equal to 100% and the guidepoints appear at your current position. If you specify greater values, guidepoints will be placed ahead of the "rubber" vector (e.g., the 200% percentage corresponds to placing guidepoints at two vector's lengths from the reference point). If your value is smaller than 100%, guidepoints will be offset to a fraction of your vector's length.

Drawing Quadrangles Using the Meter Tool

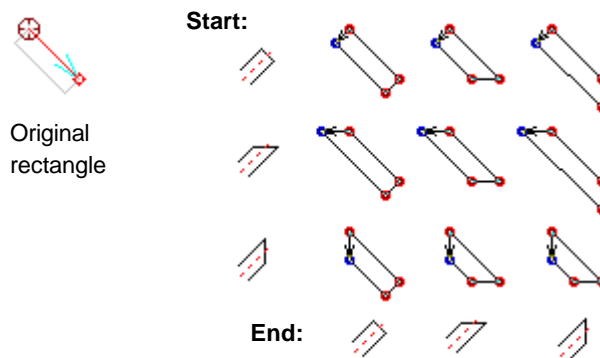
If you hold down **ALT** while dragging the "rubber" vector, a rectangle appears at one of its sides:



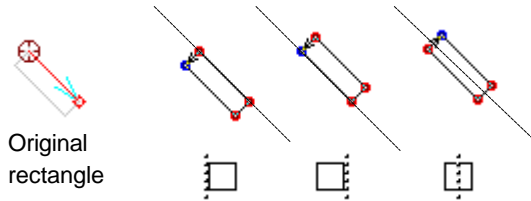
When the rectangle's side is the desired length and orientation, release the mouse button. The following dialog box appears:



In the **Stem Caps** field, select the types of caps your quadrangle will have. You may use different caps for two opposite sides that adjoin the vector. If you choose the first (default) type, the corresponding side will be orthogonal to the sides adjoining to it. If you click on the second or third type, the quadrangle side will be horizontal or vertical, respectively. The following picture illustrates all possible combinations of **Stem Caps**:



In the **Alignment** field, choose the first (default) type if you wish your quadrangle to be placed below the "rubber" vector if you moved it to the right of the reference point, or above the vector if you moved it to the left. To place the quadrangle at the other side of the "rubber" vector, click on the second type. If you select the third option, the quadrangle will be symmetrically aligned to the vector.



In the **Width** field, enter the desired distance between the vector and the quadrangle's side parallel to it.

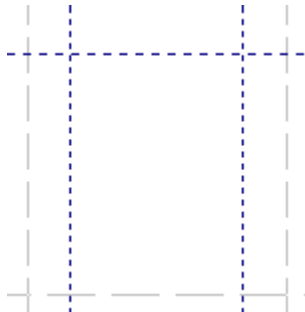
Once you've specified all necessary options, click on **OK**. The next time you define a quadrangle, FLW will use them as default.

Drawing the Letter "I"

We will create the capital "I" of a Courier-like font as an illustration of three techniques: using guidelines, measuring distances and drawing quadrangles with the **Meter** tool, and using the **Overlap** command. Before drawing the letter, make sure that the zero point of your absolute coordinate system for the new letter is at the intersection of its left margin and baseline (coincides with the symbol's origin).

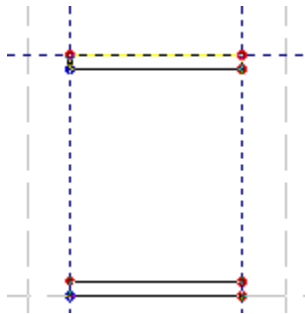
NOTE: When you create a real font, it is more efficient to mark stems with *hints* before drawing them. If you fix stems with guidelines as shown below, you will have to add hints later. For a detailed discussion of hints and situations when you must use them, see the "Using Hints" chapter of this section.

- ➊ Place a vertical guideline at the position where the leftmost line of your "I" will appear. The X coordinate of the guideline will be equal to the left sidebearing value.
- ➋ Place a horizontal guideline where the uppermost lines of your letter will appear (at the cap-height).
- ➌ Now mark the rightmost line of your "I" with a vertical guideline. To determine its X coordinate, add the left sidebearing value and the desired width of the letter. Or, switch to the **Meter** tool, click on the intersection of your guidelines and drag your cursor along the horizontal guideline until the status line shows the necessary width. Then, click the secondary button to place a guidepoint, release the primary button, and place a guideline at the marked position.

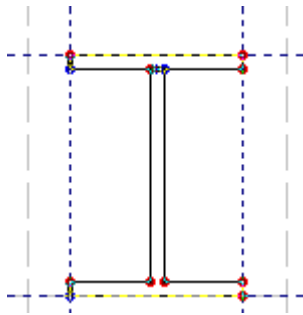


- 4 With the **Meter** tool, click on the intersection of the left vertical and horizontal guidelines. Press the **ALT** key and drag your mouse to the second vertical guideline. Release the mouse button when you are there. In the **Define Stem** dialog box, enter the desired width of stems of your letter, specify the first type of **Stem Caps** to define a rectangle, and the first type of **Alignment** to place it below the guideline.

To define the second horizontal stem, click on the other intersection of the first vertical guideline and the baseline, and follow the steps described above, except that you must specify the second type of alignment to place this stem above the baseline.



- 5 Now we have to define the vertical stem that must be centered between the vertical guidelines. To do so, double-click on the **Meter** icon in the toolbox and set the percentage to 50%. After clicking on **OK**, move your cursor over a node that is not at an intersection of guidelines and drag to the second node of the same horizontal line. Then, click the secondary button. A guidepoint appears right at the midpoint of your line. Now point to it with your cursor, press the primary button together with **SHIFT** and **ALT**, and drag to the opposite horizontal line. Release the mouse button and choose the third type of **Alignment** in the **Define Stem** dialog box. After you click on **OK**, a vertical stem appears that is centered between the margins of your symbol.



- 6 The final step we will make is removing intersecting parts of stems. Select **Overlap** from the **Transform** menu and click on **OK** (with the default settings selected) 890 (which can also be chosen from the **Popup** menu in the **Show** window). FLW will remove the overlapping parts of your rectangles and make a single outline from the original three.

Changing Positions of Points

Using the Edit Tool

The **Edit** tool proves the following two techniques for precision drawing:

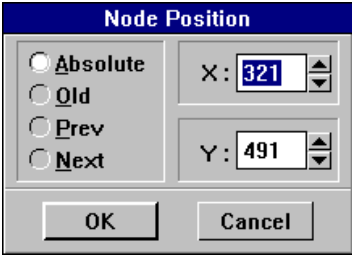
🕒 **Nudging**

When you move a node with the **Edit** tool, you may constrain your movements to one coordinate at a time by holding the **ALT** key down. Press **ALT** when you are close to the desired location to accurately position your point.

If you're working in a magnified view, you can move a point with precision without having to hold **ALT** down. However, the nudging technique lets you smooth and speed up the movements of your cursor.

🕒 **Digital Editing**

If you hold down the **CTRL** key and click on a node, the **Node Position** dialog box appears asking you to enter its new absolute or relative coordinates. You specify them in a rectangular coordinate system the (0,0) point of which coincides with:



The image shows a dialog box titled "Node Position". It has a blue header bar with the title in white. Below the header, there are four radio buttons: "Absolute", "Old", "Prev", and "Next". To the right of the "Absolute" radio button, there are two input fields. The first is labeled "X:" and contains the number "321". The second is labeled "Y:" and contains the number "491". Both input fields have small up and down arrow icons next to them. At the bottom of the dialog box, there are two buttons: "OK" and "Cancel".

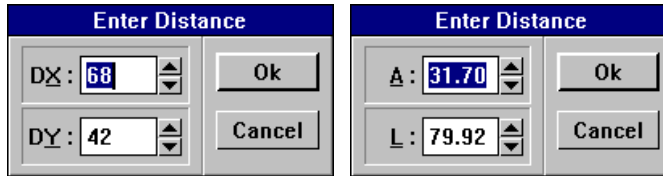
- 🕒 The zero point of your absolute coordinate system if you click on **Absolute**;
- 🕒 The current position of your node if you choose **Old**;
- 🕒 The previous or next node of your contour if you select **Prev** or **Next**, respectively.

Using the Set Tool

The **Set** tool lets you precisely set the positions of the guides origin and nodes in a local coordinate system the zero point of which may be placed anywhere in the **Edit** window.

To activate the Set tool, click on its icon in the toolbox or press **0**. Then you must complete two steps:

- 1 Click on the point you wish to make the reference point of your local coordinate system. As with the **Meter** tool, if you click near a node, vector or curve, or any guiding object, the (0,0) point will be forced to stay on it. Once you've clicked the button, a blue circle appears indicating the location of your reference point. If you wish to indicate a different position of the (0,0) point, press the secondary button, and the circle will disappear. Then, select a new reference point.
- 2 Click on the node you want to move or on the guides origin. A dialog box appears asking you to enter new coordinates of the specified object.



By default, you specify its position in a rectangular system; if you wish to enter polar coordinates, double-click on the **Set** icon before indicating the zero point as described in step 1 and select **Polar** in the dialog box.

NOTE: FLW always preserves the coordinates of guiding objects relative to the guides origin. Therefore, if you move the guides origin, they will be offset by the amount you specify after clicking on it with the **Set** tool.


Using the Mask Layer

The mask layer lets you:

- ① Force your cursor to stay on lines of any shape. For example, you may align your objects to a circle or a sloping straight line.
- ① Experiment with your symbols. Use the mask layer whenever you wish to create two versions of the same symbol and then choose one of them or exchange their elements.

The mask layer provides a storage space for your symbol (every symbol has its own mask that may be empty or not), whereas all manipulations of them are performed in the working layer. However, you may freely swap contents of these layers, so it is always possible to modify contours residing in the mask layer. If you define a guiding object while editing a mask, it will be preserved after you swap the layers again.

To copy a selected part of your symbol to the mask layer, select **Copy to Mask** from the **Hints/Guides** menu or press **M**. If no selection has been made, the entire symbol will be copied. Your contour becomes blue since it lays on top of its copy belonging to the mask layer. If contours do not overlap, the objects located in the working layer are black and the mask layer's contents are gray.

If you wish to "hide" the mask layer, click on the  icon in the **Options** box. Or, select **Edit window...** from the **Options** menu and switch off **Mask** in the **Display** field of the dialog box.

Once the mask layer contains a contour, you can use the following commands from the **Hints/Guides** menu:

- ⌚ **Copy to Mask** (shortcut: **M**) lets you copy additional segments to the mask layer.
- ⌚ **Exchange with Mask** (shortcut: **J**) swaps contents of the working and mask layers.
- ⌚ **Paste Mask** supplements contents of the working layer with contours belonging to the mask. Existing contours are not modified.
- ⌚ **Clear Mask** deletes all objects belonging to the mask layer.

Snapping to the Mask

If your cursor is close to a line belonging to the mask layer, it is forced to stay on this line. This sensitivity in pixels is adjusted via the **Guides** entry of the **Preferences** dialog box, which is activated by selecting **Preferences...** from the **Options** menu.

If you wish to disable snapping to the mask, click on **Edit window...** in the **Options** menu. In the **Options** dialog box, use the **Mask Vectors** and **Mask Curves** switches situated in the **Snap To** field to control snapping to straight and curved mask segments, respectively.

NOTE: We recommend disabling the **Snap To Mask Vectors** mode if you want to speed up the editing operations (especially on slow computers).

Please note that the **Snap to Guides** command discussed before will not align nodes to the objects residing in the mask layer even if the snapping feature is enabled.

Working with Bitmaps

This capability lets you convert raster images into high quality PostScript objects, or simply use a bitmap picture as a background while creating a symbol. Bitmaps lay in a separate layer and appear on your screen as gray paths.

You can work with bitmaps created in any Windows paint program (e.g., Microsoft Windows Paintbrush) and copied to the Windows Clipboard. You have two opportunities: you can draw your own images in a paint program, or import a scanned image (almost all programs that work with scanners are capable of saving PCX files). In both cases, you have to place your image into the Clipboard to make it available to FLW.

Once a bitmap picture is in the Clipboard, the **Paste** command from the **Hints/Guides/Bitmaps...** menu becomes available. If you select it (or press **W**), the bitmap will be placed onto the bitmap layer and displayed in the **Edit** window as a gray path.


There is an alternative way of creating raster backgrounds for your symbols. If you choose the **Generate** command from the same menu, FLW will create a filled raster image of a current symbol and place it onto the bitmap layer without copying it to the Clipboard. To make this raster image available as a background for other symbols or to other applications via the Clipboard, select **Copy** or press **Y**.

Now you have two choices:

- ⌚ If you want to use the bitmap as a raster background for information purposes only (snapping to a raster image is impossible), leave it as it is or use the **Move/Size** command (shortcut: **Z**) to reposition or resize it. After you select this command, a rectangle appears around your bitmap. Drag one of its corner handles to resize it, or place your cursor inside the rectangle and drag it to a new location. When you're finished, click the secondary button.



- ⌚ You may want to convert your bitmap into a vector background (that is, to place it onto the mask layer as a vector object) and then either use it as a usual mask or bring it into the working layer for editing (you might want to do so when you have a scanned image to be transformed into a high quality PostScript symbol). To do so, choose the **Trace** command or press **G**. A dialog box will appear showing the tracing progress, and then a traced outline will be placed into the mask layer. Note that the original bitmap will be preserved.

After tracing your bitmap, you might want to "hide" the bitmap layer. To do so, click on the  icon in the **Options** box. Or, select **Edit Window...** from the **Options** menu and switch off **Bitmap** in the **Display** field of the dialog box.

If you wish to permanently erase the contents of the bitmap layer, choose the **Erase** command.

Design Consistency

Now it's time to outline the two rules for obtaining the best possible results regardless of the device and resolution you use.

- ① All horizontal and vertical dimensions that are intended to be the same must be exactly the same. This rule applies to widths of vertical and horizontal stems and to such main dimensions as sidebearings, cap-height, x-height, overshoot positions, etc.
- ① All parts of symbols (e.g., serifs) that are intended to be the same must be exactly the same.

The first rule is ensured by setting precise coordinates of objects and using hints. The use of subroutines lets you meet the second requirement. Hints will be discussed in the next chapter and in the "Advanced Features" section under "More About Hints." For more details on the use of subroutines, see "Using Subroutines" in the "Advanced Features" section.

Using Hints

When a character is rendered at a limited resolution, it is especially important to preserve correct proportion of its elements. Since all digital devices employ coordinate systems of their own in which the minimum unit is one pixel (its absolute size depends on the device you use), errors are inevitable when a PostScript interpreter transfers symbols from their original coordinate system into the system specific for the current device. These inaccuracies matter much less when a font is reproduced at a high resolution. For example, if you render a stem that is 100 pixels wide, a 1-pixel error may not even be visible; however, when reproducing your stem at 5 pixels wide or less, you will notice an inconsistency at once.

The key to maintaining proportions of symbols is constraining sizes and positions of their features. For this purpose, the Adobe Type 1 format provides the following two means: *character level hints* for describing main stems and stem-like features of individual characters, and *font level hints* that control vertical alignment of character features and state constraints on main widths of horizontal and vertical stems. In this chapter, we will discuss the first type of hints (they are also called *stem hints*); for more information on font level hints, refer to the "Advanced Features" section.

If you're going to create only TrueType fonts, there is no strict need to place any hints. By exporting a font in TrueType format all needed instructions will be set automatically. Nevertheless, if you use hints, you will be able to create more correct and quality fonts, and to export fonts in the format **Adobe** Type 1, if necessary.

TrueType Hinting Methods

The TrueType instruction set provides a large number of commands which allow designers to specify how character features should be rendered. Instructions also provide a mechanism that preserves the shape of characters when they are scaled. Every symbol is presented in the language of TrueType instructions as a chain of commands normally hidden from the user. Working with TrueType fonts, you will need to study neither the interiors of TrueType language, nor the external representation of this language. Both are necessarily only for designers of software for font management.

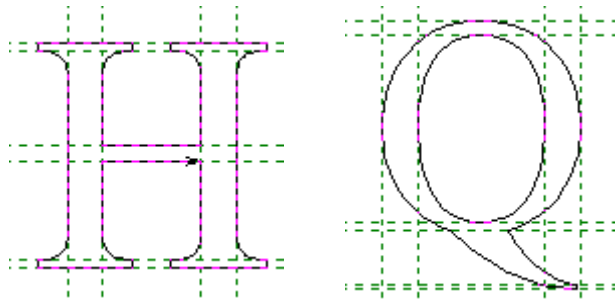
FLW lets you create an additional packet of information to be stored later in the output file. This is done automatically, so you don't have to worry about seeing the output file.

As we have said many times in this manual, FLW is a superb tool for creating and managing PostScript fonts. But you can use it for work with TrueType fonts, too. You can import and export TrueType fonts with FLW without worrying about conversion of 2nd degree curves used in TrueType fonts into Bezier curves or about proper TrueType hinting.

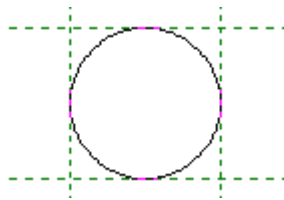
What Are Stem Hints?

Every stem hint is a pair of straight lines that indicate the positions of lines that form the corresponding stem or stem-like feature. A horizontal stem hint is described by two values: the vertical coordinate of its lower line and the range between the lines that is equal to the width of the corresponding stem. Similarly, every vertical hint takes two values: the X coordinate of the left line and the width of the corresponding vertical stem.

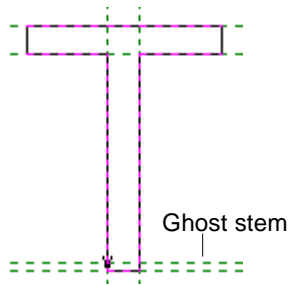
It is very important to mark specific character features with stem hints to ensure their correct reproduction at all sizes. The following picture illustrates different situations when it is necessary to use stem hints:



If your symbol has a single vertical or horizontal stem (like a sans serif "I" or the bullet character), use stem hints that are as wide or tall as your character. However, you may not use stem hints that lay in both a top and bottom zone; this situation is discussed in the next paragraph. The following picture shows a correctly hinted bullet:



In order to perform vertical alignment, PostScript interpreters need hints at characteristic horizontal lines of your symbols (baseline, cap height, x-height, ascender height, descender depth, overshoots, etc.). In some serif fonts this happens naturally (consider the above picture). However, characters of sans serif fonts often have no horizontal stems in such alignment zones to be hinted. In this case, you have to use the so-called *ghost stems*, the upper or lower lines of which fix the uppermost or lowermost coordinates of such symbols. The height of these hints must be equal to 20 or 21 (the latter value is the default height that FLW offers when you define a hint). Please note that you may not use a single horizontal hint spreading from a top zone (e.g., a cap-height zone) to a bottom zone (e.g., a baseline zone). This picture illustrates a sans serif "T" that has a ghost stem at the bottom:



In fact, the Adobe Type 1 font format incorporates a special mechanism for ensuring proper vertical alignment. We will discuss it in the "Advanced Features" section under "More About Hints."

The Adobe Type 1 format requires that stem hints of the same direction must not overlap. However, you will often find situations when it is necessary to hint several features laying in the same vertical or horizontal hint zone. In this case, avoid hint overlaps by using the hints replacement mechanism described in the "Advanced Features" section under "More About Hints."

Placing Stem Hints in FLW

In FLW, you may use stem hints as the guidelines. Since you have to define stem hints to ensure high quality of reproduction, it is far more convenient to use stem hints **INSTEAD OF** pairs of guidelines when you mark a character, rather than define them later.

NOTE: Before working with hints, make sure that they will be displayed in the **Edit** window (check the **Display** section of the **Options** dialog box, which is available via the **Edit Window...** command from the **Options** menu).

To Define a Hint,

hold down **CTRL**, press the primary button on the corresponding ruler, and drag your hint onto the working field. As you move the pair of dark green dotted lines that appear, the status line reveals the current coordinates of its reference line (lower line for horizontal and left line for vertical hints).

When the line is in place, release the button. In the dialog box that appears, enter the desired width of your hint that might correspond to the width of the future stem, and click on **OK**.

If you place hints on stems that are already defined, the method described above is not really convenient since you will need to input an exact width for every new hint. Another technique is as follows:

- 1 Switch to the **Meter** tool by pressing **7** or clicking on its icon in the toolbox.
- 2 Hold down the **CTRL** key, measure the distance between the lines you wish to fix with a hint. If you want to add a horizontal hint, measure the vertical distance between your lines, and vice versa. You may also drag the "rubber" vector diagonally; in this case, you will be able to add two hints at once. When you're finished, release the mouse button.
- 3 In the appearing dialog box, indicate the necessary hint type. You may also edit the offered values. When you're done, click on **OK**.

To Reposition a Hint

click on its line with the **Edit** tool, hold down the button and drag the hint to the desired location. Or, press the **CTRL** key and click on a hint's line. A dialog box appears in which you may type or select the new hint's position and width.

NOTE: Hints are protected from accidental modification by means of locking delays. For more details, refer to the "Locking Delays" chapter of this section.

To Remove a Hint

drag it off the editing area using the **Edit** tool. Or, with the **Edit** tool active, press and hold down the primary button on its line, and click the secondary button.

To Remove All Hints of a Specific Type at Once

select the **Remove All Hints** command from the **Hints/Guides** menu and choose the desired type in its submenu.

Working with Global Hints

FLW lets you define a set of hints that may be used by different symbols. Once such a hint is defined, you may make it active for the symbol you're working on. As with global guidelines, if you change the properties of such a hint while editing a certain symbol, all other symbols that contain it are affected.

These global hints have NO connection with the Adobe Type 1 font level hints which are discussed in the "Advanced Features" section under "More About Hints." Whenever you export a Type 1 font, all FLW global hints are converted to ordinary character level hints. However, it is useful to define global hints while working on a font to ensure the design consistency. As we mentioned above, you may use hints as pairs of guidelines, so it makes sense to use global hints just as pairs of global guides.

To define a global stem hint, hold down the **SHIFT** and **CTRL** keys together and drag your hint from the corresponding ruler. The color of global hints is light green. As with ordinary hints, you have to specify the desired width after releasing the mouse button. Please note that it is impossible to define a global hint using the **Meter** tool.

To change the properties of a global stem hint or remove it, follow the instructions outlined above.

Once a global hint is defined, it works only with the symbol for which it has been set. When you work on other characters of your font, you may activate certain global hints for them. To do so,

- ❶ Select **Choose Global Hint** from the **Hints/Guides** menu or **Choose Hint** from the small menu the icon of which is situated in the upper left corner of the **Edit** window. From the drop-down submenu, choose the appropriate type of hints. (You may also press **F5** to choose a horizontal global hint or **F6** for vertical hints.)
- ❷ Move your cursor into the editing field, click and hold down the primary button, and marquee select the area in which you wish to set a hint.
- ❸ After you release the button, a dialog box appears that lets you preview all global hints available in the chosen area and select one of them. Hints are displayed as thick green lines. If there are several global hints defined for the given area, use the scroll bar to switch between them.
- ❹ Once the desired hint is highlighted, click on **OK**.

As we said above, by editing a global hint you affect all symbols for which it is set. If you wish to avoid this effect, turn all global hints of a given type into local ones by clicking on **Paste Global Hints** in the **Hints/Guides** menu and selecting the desired type from the drop-down submenu.

Hinting Completed Outlines

If you have created a character that lacks some hints, you have two options.

- ❶ You may hint your character manually. The most convenient method here is using the **Meter** tool as described above.
- ❷ Select the **Autohinting** command from the **Hints/Guides** menu or press **F7**. FLW will analyze the character displayed in the **Edit** window and generate a complete set of hints. The algorithm is based on all Adobe's rules and requirements including the hints replacement technique.

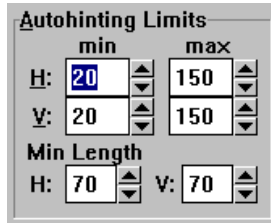
Normally, there is no need for manual hinting of completed characters since the autohinting algorithm usually achieves the same results. However, if you are an experienced font designer, you may find it necessary to fine tune the results obtained by using the algorithm.

Summarizing the above, we recommend using the **Autohinting** command right after defining character outlines to ensure correct placement of all hints. Once this operation is completed, examine your characters and rehint them if you need to.

FLW lets you automatically hint a series of characters by using the **Apply to Range...** command from the **Transform** menu. For more details on the use of this command, see the last chapter of the "Transformations" section.

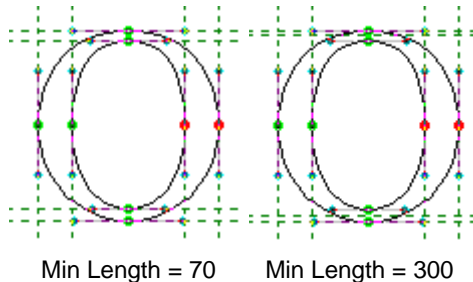
Customizing the Autohinting Feature

The **Preferences** dialog box, which appears after selecting **Preferences...** from the **Options** menu, contains a special field, **Autohinting Limits**, which states constraints on widths of stems and lengths of vectors that are taken into consideration during the autohinting process.



- ⌚ The **Min Length** parameters sets the minimum length of vectors of character features that will be treated as stems. If the length of a straight segment or a curve's control vector exceeds this value, the autohinting algorithm considers the element to which it belongs as a stem whose direction matches that of the vector. If several vectors are collinear, FLW uses their total length.

The following picture illustrates using different **Min Length** values. The capital "O" from FLW_DEMO.VFA has two pairs of control vectors (associated with the upper and lower nodes of the inner contour) that have total lengths slightly lower than 300, while other pairs of control vectors are longer than 400. If we use the default value of **Min Length** equal to 70, all vectors are considered as components of stems and FLW fixes them with hints. However, if we increase the value to 300, shorter vectors are not recognized and two ghost stems are placed at the top and bottom of the "O."



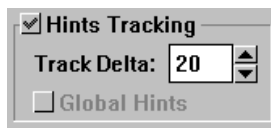
- ⌚ The pairs of **H** and **V** boxes limit horizontal and vertical dimensions of stems and stem-like character features that are hinted by the algorithm. (These distances are equal to distances between vectors that are recognized as belonging to stems.) For example, if the width of a vertical stem falls within the **H** interval (is greater than 20 and less than 200), a vertical hint will be placed at this stem.

If it is necessary to place a stem hint that "covers" an entire character (e.g., if you are hinting a bullet), these constraints do not work.

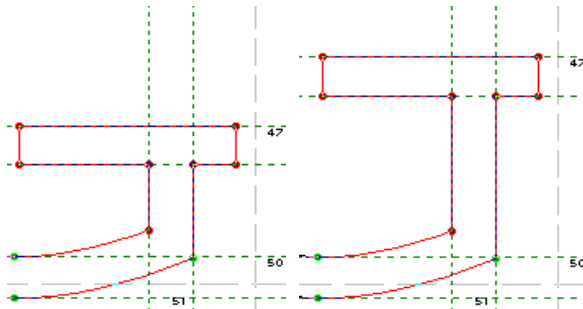
Hints Tracking

Properly placed hints allow you to use a special technique called **Hints Tracking**.

To switch on **Hints Tracking**, open the **Preferences** dialog and switch on this option:



When Hints Tracking is switched on and you move a vertical or horizontal hint a distance lower than the **Track Delta** field, all nodes whose horizontal or vertical coordinates are equal to the coordinates of the hint will be moved, too.



Above are two pictures taken from the **Edit** window. You see some parts moved simultaneously with their hints.

When the **Global Hints** checkbox is switched on and you move a global hint used in other symbols, this operation (hint tracking) will be applied to all those other symbols as if this tracking was applied to them directly.

Transformations


In this chapter, we will discuss:

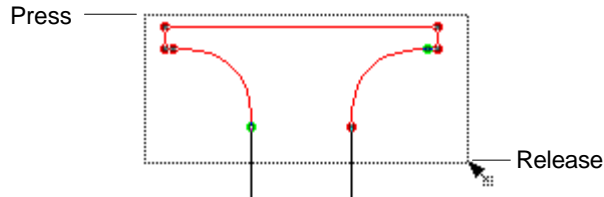
- ⌚ another function of the **Edit** tool;
- ⌚ two new tools: **Select** and **Move**;
- ⌚ copying, duplicating, and erasing parts of symbols using the **Edit** menu commands;
- ⌚ copying symbols and their parts between different fonts;
- ⌚ applying various effects using the commands contained in the **Transform** menu.
- ⌚ using a Table window Toolbar to transform symbols

Selecting Parts of Outlines

Before performing any transformations, you must specify the part of the symbol you want to transform or copy. There are two ways to do this.

Selecting Segments with the Edit Tool

If you wish to quickly select a contour's part when the **Edit** tool is active, move the cursor into a free area of the **Edit** window and press the primary button. The cursor will change to a . Now drag your mouse and surround the segment you want to select with the marquee. When you release the button, the segments entirely enclosed in the marquee become red (i.e., selected).



If you now select other object(s), the previously highlighted segments will be deselected. However, if you do this holding **SHIFT**, the previously selected part of a symbol will not be deselected, but selection will then include both old and new selected segments and all fragments of the contour located between them, therefore making the selections connected.

Using the Select Tool

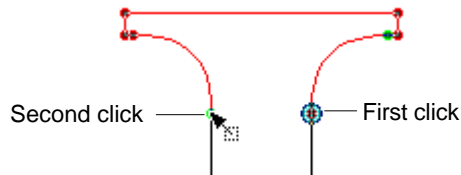
The special tool used for selecting segments of outlines is **Select**. You can make it active by clicking on its icon in the toolbox or pressing **8**. The **Select** tool works in the following two modes:

Marquee select

This mode is equivalent to using the edit tool as discussed above. Just press the primary button on a white space and drag your mouse to enclose the segments in the marquee.

Nodes select

It is faster and easier to select segments by pointing to their nodes. To select a segment or a series of consecutive segments, click on the startpoint of it. You will notice a blue circle surrounding it. Now point to the endpoint of the contour's part you want to highlight and click again. The segments connecting the nodes will become selected. Naturally, if you click twice on the same node, the whole outline will be selected.



You can click the secondary button after selecting the first node to terminate the operation.

Selecting and Deselecting All Segments of a Symbol

The **Select All** and **Deselect** commands contained in the **Edit** menu perform two opposite tasks: the first of them highlights the whole symbol while the second command removes any existing highlight. The shortcuts for these commands are **L** and **U**.

Using the Edit Menu Commands

Copying and Moving Segments

To make a copy of a segment,

- ❶ Highlight it as described above.
- ❷ Select **Copy** from the **Edit** menu or press **CTRL+INS** to place the segment into the copy buffer. If you wish to move the segment to a new location, select **Cut** or press **SHIFT+INS**. In the latter case, the original copy of your segment will be erased.
- ❸ Locate the target symbol. If you wish to replace one of its parts with the segment you have placed into the buffer, select the part and replace it by choosing **Paste** or pressing **SHIFT+INS**. If you want to supplement the target symbol with the copied segment, choose **Add** or press **A**, and the buffer's contents will be placed in the center of the **Edit** window.

If the target symbol belongs to another font, you have two options: 1) you may open this new font, bring the target symbol into the **Edit** window, and select **Paste** or **Add**. 2) However, if you wish to continue working on the current font, switch to Windows Program Manager, run a second copy of FLW, and open the target font there.

In addition to copying or moving parts of symbols, you can manipulate their entire outlines in the same way. As you know, the **Copy Symbol** and **Paste Symbol** commands allow you to copy entire symbols, but they always replace the target symbol with the buffer's contents. Conversely, the **Paste** and **Add** commands let you supplement the target symbol with a new contour, which is useful when you want to compose a new symbol from two. Actually, FLW uses two buffers: the first buffer serves for copying outlines, and the second buffer is capable of storing all information about the symbol. This information includes all objects associated with the symbol which are discussed in the next section.

NOTE: You may also use the **Copy Symbol** and **Paste Symbol** commands to copy symbols between different fonts using the techniques described here.

Duplicating Segments

The **Duplicate** command (keyboard shortcut: **D**) lets you create a copy of a selected segment without placing it in the buffer. The copy will be slightly offset to the lower right of the original and highlighted. You can set the amount of offset by selecting **Preferences** from the **Options** menu, and typing or selecting the **X** and **Y** offsets in the **Place Duplicate** field.

Erasing Segments

To delete a segment, select it and choose **Erase** from the **Edit** menu, or simply press **DEL**.

Simple Transformations

After selecting an outline or its part, you may apply several standard transformations to it:


- ⌚ Stretch and Scale
- ⌚ Mirror (horizontal or vertical, or both)
- ⌚ Rotate and Skew
- ⌚ Move

The first three may be performed either manually (using the **Move** tool) or by choosing commands from the **Transform** menu. There is no direct command for moving objects (though you may use the **Duplicate** command from the **Edit** menu as shown later), but the **Move** tool lets you achieve precise results.

Using the Move Tool

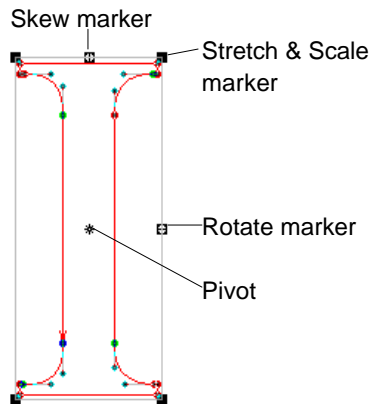
To select the **Move** tool, click on its icon or press **9**. Note that you must highlight an outline or a series of segments before selecting it.

If the current symbol doesn't have any selection you can make it using the same marquee technique described above. Moreover, if you want to change your selection before applying a transformation, you can press **CTRL** and temporarily switch to the **Select** tool in the marquee mode.

- ❶ You can move a highlighted outline using one of the following methods: (The **Move** tool lets you select segments in the same fashion as the **Edit** tool; however, if any segments are already highlighted, you need to hold down **SHIFT** to be able to marquee select other portions of your character.)
- ⌚ Click anywhere in the **Edit** window except the highlighted outline. The selected part will be surrounded with a rectangle and the cursor will change to a . Now place it inside the rectangle, press the primary button, and drag your mouse to move the rectangle. The status line will report the offset from the original position along the OX and OY axes of the coordinate system. When you are finished, release the button.

- ① Click on any point belonging to the selected segment and drag it to the desired location. As you move this point, the entire fragment will be moved with it. This technique is similar to non-nodes editing. For more details, see the "Modifying Outlines Using the Edit Tool" chapter of "Drawing Outlines."

Let us take a closer look at the rectangle that appears after clicking the primary button anywhere in the **Edit** window.



The elements of this rectangle let you perform any of the transformations mentioned above. The rule is: point to the marker of the desired operation, press the primary button, drag the marker to achieve the necessary effect, and release the button when you are finished.

By default, a special marker called pivot appears in the center of the rectangle. When you skew or rotate a fragment, you actually move a vector that connects the pivot and the marker you are moving. Thus, if you move the pivot, your fragment will be rotated about its new position.

- ② To stretch an object, drag one of its Stretch & Scale markers. The status line tells you the stretch factor along the horizontal (SX) and vertical (SY) axes. If you swap the rectangle's sides, you will mirror your fragment with some stretch factor that will be a negative value (horizontal mirror corresponds to swapping vertical sides). A stretch factor of -100% corresponds to reflecting the fragment without stretching it.

- ③ To scale a fragment, hold down **SHIFT** and drag one of the Stretch & Scale markers. The current amount of scaling is displayed in the status line as SX and SY (when scaling an object they are always equal)
- ④ To rotate or skew a fragment, drag the corresponding marker. A vector appears that connects the pivot and the marker you are moving, and the status line displays the angle between this vector and the horizontal axis. Angles of rotation vary from 360 to -360 degrees (positive values produce clockwise rotation) and you may achieve any skew value between -80 and 80 degrees (positive values slant your fragment to the right).

NOTE: Please keep in mind that any values displayed in the status line refer to the current transformation and are not cumulative.

Besides transforming objects by dragging the markers, you can specify a precise factor of transformation. To do so, hold down **CTRL** and click on the corresponding marker. A dialog box will appear letting you specify a scaling factor or an angle of rotation or skew. These dialog boxes are shared by the **Transform** menu commands discussed below.

Using the Transform Menu Commands

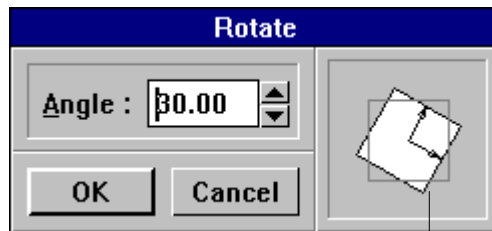
Unlike the **Move** tool, the commands contained in the **Transform** menu work with the whole symbol if no part is selected.

① **H Mirror** and **V Mirror**

To reflect a fragment about an imaginary vertical or horizontal line through its center, select H Mirror or V Mirror, respectively.

② **Rotate...** and **Slant...**

These commands work quite similarly: after selecting one of them, enter the desired amount of rotation or skew (the preview box constantly shows the effect that you will achieve after clicking on OK). You must enter values within the ranges discussed above.



Preview box

③ **Scale...**

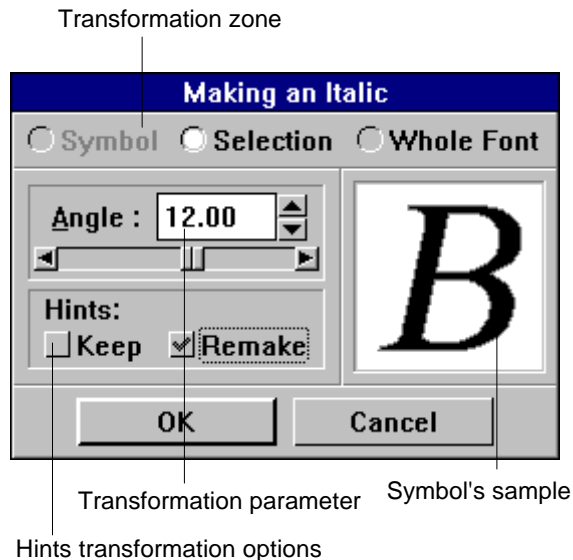
This command lets you perform four operations: Horizontal and Vertical Mirror, Stretch, and Scale. After selecting this command, you enter the Scale dialog box described below.

- ① To scale an object, switch on Scale Proportionally (which is the default mode) and enter the desired percentage in one of the editing fields (the other field will be updated automatically).
- ② To stretch your fragment, switch off Scale Proportionally and specify the amount of stretching along the horizontal (X Scale) and vertical (Y Scale) axes.
- ③ To reflect the fragment, click on H Mirror or V Mirror, or both.

Before clicking the OK button, consult the preview box that shows the effect your values will have on the object.

Transforming Symbols Using Table Toolbar

Using the **Table** window you can transform a single symbol, a selected part of your font or the whole font. You can achieve this by pressing one of the icons from the **Table Toolbar**. If the **Selection** mode is on, when applying a transformation, you'll be able to apply the transformation to the selected part of your font.



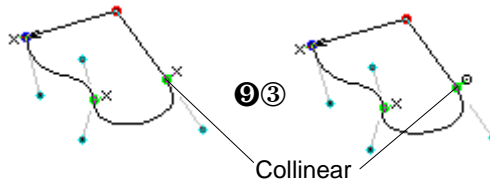
To apply a transformation effect to a symbol (selection, font) you should tune the Transformation parameter to the desired value; choose the area on which to apply the effect; set some combination of the additional options that correspond to the effect suited to your needs; and, finally, press the **OK** button.

Automatic Modification of Contours

In this chapter, we will continue the discussion of FLW's features that let you automatically improve your Postscript symbols. We have mentioned several conventions which allow you to create not only correct but also good-looking outlines; to refresh your memory, please look through the "Character Outlines" chapter of the "Drawing Outlines" section.

Changing Types of Connections


As we mentioned in the "Drawing Outlines" section, smooth transitions between path elements are made by making control vectors collinear (in FLW these are the \bullet^{\ominus} type of nodes). To automatically detect situations when control vectors ARE collinear but the type of connection is angular (\bullet^{\times}), select the fragment to be analyzed and choose **Check Connections** from the **Transform/More...** menu. As with other **Transform** commands, the entire symbol will be processed if no selection has been made.

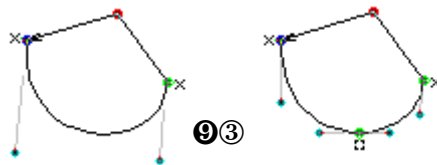


To set the \bullet^{\times} type for all nodes, select **Reset Connections** from the **Transform/More...** menu.

Placing Nodes at Extremes

To help the PostScript rendering algorithms to accurately reproduce your symbols, place nodes at the extremes of your outlines. However, the more segments in your symbol, the slower it will be rendered. So try to use the fewest Bézier curves and vectors that accurately reproduce the desired symbol's shape, while avoiding curves including more than 90 degrees of arc.

The **Nodes at Extremes** command from the **Transform/More...** menu lets you meet the latter convention by automatically placing nodes at extremes. Choose it after highlighting a portion of your outline, and your Bézier curves will be transformed as shown here. The new nodes always have the  (fixed) type.



Contour Optimization

FLW has an automatic optimization feature that can clean up your outlines and remove most common errors. We will discuss this feature in detail in the "FontAudit" part of "Advanced Topics" chapter.

To apply optimization to the selected part of a symbol or to the entire symbol, select **Optimize** command from the **Transform/More...** menu (keyboard shortcut - Shift+F11).

Using the Overlap Command

This command lets you correct directions of contours and deal with intersections of different outlines. To understand how it works, let's discuss the PostScript and ATM (Adobe Type Manager) fill modes before making any further explanations.

The PostScript language has a strict rule saying that if you imagine walking along a contour in its direction, the filled area will always be on your left. Conversely, ATM pays no attention to directions of contours and fills every other path. The inner contour of the symbol shown below has an incorrect direction, but ATM reproduces it as desired. From the PostScript point of view, this contour is unnecessary since it is invisible.



Since you must create symbols that will work properly on any PostScript device, you should follow the PostScript conventions. Correctly defined outlines are always properly reproduced by ATM.

The **Overlap** command from the **Transform** menu allows you to:

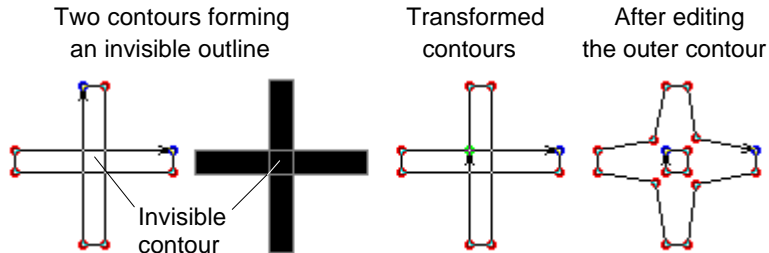
- ⌚ automatically detect contours that will be invisible when reproduced by a PostScript language interpreter;
- ⌚ correct directions of outlines that are properly reproduced by ATM but are not recognized by PostScript rendering algorithms;
- ⌚ transform overlapping or non-overlapping parts of intersecting contours into separate outlines.

In fact, FLW performs the latter operation right after you click on **OK** in the **Contours Checking** dialog box no matter what mode you have chosen, so we will take a look at it before proceeding to explain the dialog box. We will limit our discussion to the case of two overlapping contours having one common part, but the same rules work for several outlines and/or several overlapping fragments.

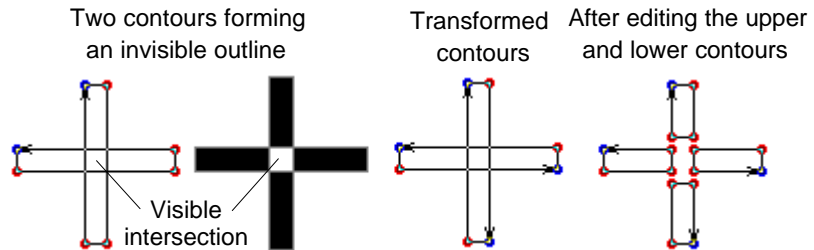
When two contours intersect, their common part may be either filled or not in the PostScript fill mode. Since such parts are invisible when printed, we will call them invisible outlines.

After finding overlapping contours, FLW places nodes at all points of their intersection. If they form an invisible outline, FLW creates a contour from it and transforms all other non-overlapping paths into a second contour. If the common part is not filled (i.e., is visible), several new contours appear surrounding it. Here are two pictures illustrating these situations:

🕒 Invisible outline



🕒 Visible intersection



Now we will describe the commands available from the **Contours Checking** dialog box. (When experimenting with it, create a **Show** window to preview your symbols.)

- ❶ Before choosing any other options, make sure that the desired fill mode is on. If not, click on the appropriate button in the upper field. If you want to delete or select invisible contours, switch on the PostScript mode; if you need to correct directions of contours, click on **ATM**.
- ❷ In the PostScript mode, you may either **Remove Invisible Contours** or **Select Invisible Contours**. If you click on the **Skip** button or FLW finds a visible intersection, the overlapping contours will be processed as described above.

- ③ If you choose the ATM fill mode, all intersections will become visible; therefore, all these options will be equivalent to **Skip**.
- ④ The **Correct Orientation** command lets you change directions of outlines incorrectly filled by PostScript interpreters.

Fill Modes in Overlap... command:

PostScript

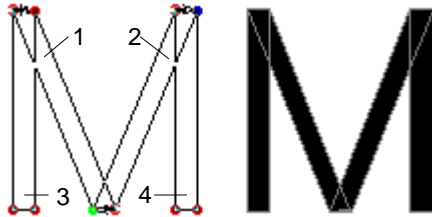
Since PostScript algorithms process incorrectly filled outlines as invisible, this command will invert only single outlines (though they are always properly reproduced).

ATM

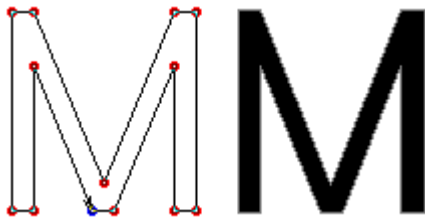
ATM recognizes all outlines, so this command will invert all contours with incorrect directions. It is very useful to apply the **Overlap** command with this option to all your symbols. Later, we will discuss the **Apply to Range...** command which lets you automatically perform this operation.

As a practical example of using the **Overlap** command, we will show how to create an "M" letter with minimal effort.

- 1 Draw four rectangles as shown below. Note that they form three invisible contours:



- 2 Choose **Overlap** from the **Transform** menu, and select **PostScript** and **Remove Invisible Contours**. Click on the **OK** button. Your symbol will be transformed into a single outline:



The easiest and fastest way to apply an **Overlap** command is to use the mouse popup menu in the **Show** window. Simply press the secondary button while the cursor is in the **Show** window. In this menu, select **Remove Overlap** command and all overlapping parts of symbol presented in the **Show** window will be removed. Overlap options will be selected according to the current visualization method of the **Show** window.

Boolean Operations with Overlap Command

Using the Overlap command you can perform several logical or Boolean operations on the contours of the symbol.

Intersecting

To intersect two contours:

- ❶ Apply an **Overlap...** command with the following parameters: **PostScript** filling mode; **Select Invisible Contours**; **Correct Orientation** - on. The intersecting part of the overlapped contours will appear selected.
- ❷ Select the **Move** tool and drag the intersected part to another place.

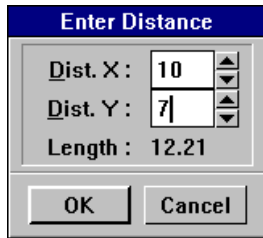
Subtracting

To subtract one contour from another:

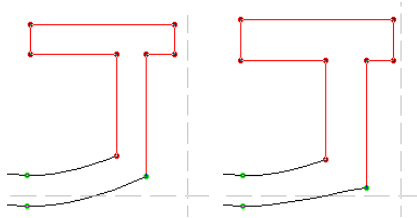
- ❶ Invert one of the contours using the **Change Direction** command from the **Transform** menu.
- ❷ Apply an **Overlap...** command with the following parameters: **PostScript** filling mode; **Remove Invisible Contours**; **Correct Orientation** - on.
- ❸ Using the **Select (8)** tool and **Erase (Del)** command, remove the unnecessary part of the contour.

Changing Weight

Using this, you can change the width of a selected part of a symbol. Select some part of your symbol as described above, enter the menus **Transform, More, Change Weight** and you will get the following dialog box:

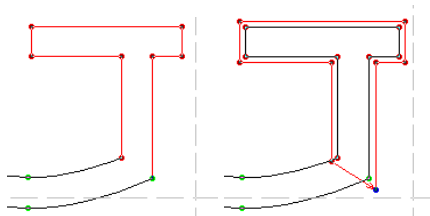


Enter the desired distance and press **OK**. You'll see the result (before and after) as follows:



Creating an Outline

Using this, you can create an outline of a selected part of a symbol. Select some part of your symbol as described above, enter the menus **Transform, More, Create Outline** and you'll get the same dialog box as in **Change Width**. Enter the desired distance and press **OK**. You'll see the result (before and after) as follows:

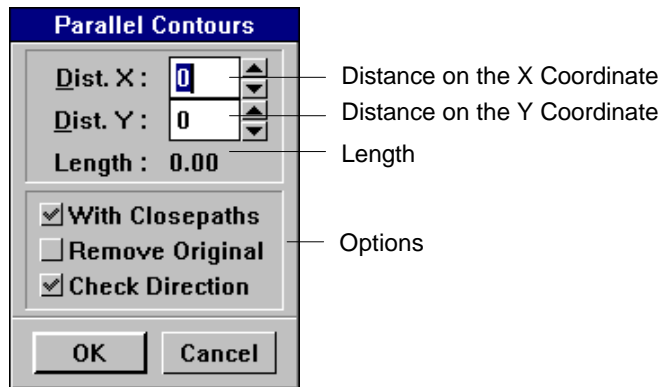


Drawing Parallel Lines

Another useful feature of FLW is its ability to draw parallel contours of any shape.

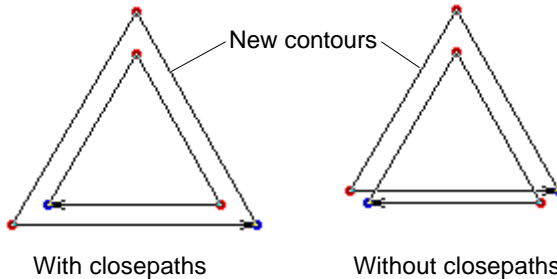
To define a parallel outline,

- 1 Select the original outline. It may contain any number of segments; or you can select several outlines to create parallel contours for all of them. If no selection has been made, FLW will process all contours of the current symbol.
- 2 Choose **Parallel Contours...** from the **Transform/More...** menu (shortcut: **SHIFT+F9**). You'll get the following dialog box.



The **Length** indicator will always show the value calculated as $\sqrt{Dist. X^2 + Dist. Y^2}$

- 3 In the dialog box, enter the desired **X** and **Y** distances between parallel outlines. Note that the new outline will appear outside the existing contour (not in its filled area), so you have to enter a negative value to place the parallel contour inside the existing one.
- 4 Switch on **With Closepaths** (which is the default mode) if you wish to process closing vectors as ordinary straight lines. Otherwise, FLW will handle contours as open paths.

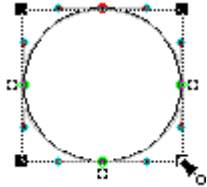


NOTE: *Closepath* is the PostScript command used for drawing closing vectors. Throughout this manual, we use this term as a synonym for "closing vector."

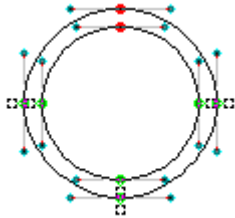
- 5 Click on **Remove Original** if you wish to delete the original contour, then click on **OK**.

As an illustration of this technique, we will create the small "o" of a Courier-like font.

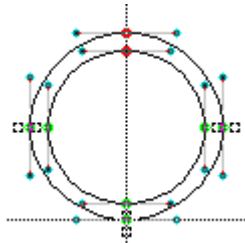
- 1 Switch to the **Arc** tool. Draw an ellipse with the horizontal axis slightly longer than the vertical axis.



- 2 Select **Parallel Contours...** from the **Transform/More...** menu, enter the horizontal width of your letter, and click on **OK** (by default, **With Closepaths** is on and **Remove Original** is off).

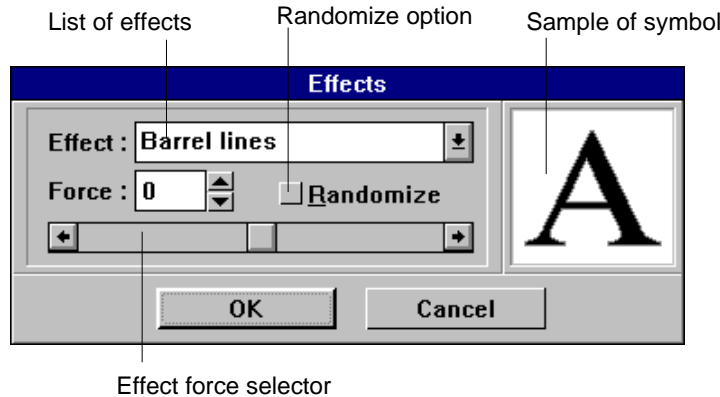


- 3 Slightly flatten the upper and lower parts of the outer circle using the **Edit** tool.



Using Special Effects

The **Effects...** command available from the **Transform** menu lets you apply several decorative effects to your symbols or their highlighted parts.



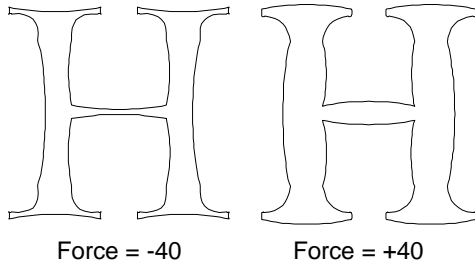
After entering the **Effects** dialog box,

- 1 Use the drop-down **Effect** list to select the effect you want to apply.
- 2 Type or select the **Force** value. The higher absolute value you specify, the more your symbol will be transformed. You may specify values lying within the $[-200,200]$ range; we will point out the difference between positive and negative values in the descriptions that follow. When you check a **Randomize** option, force will randomly vary from 0 to the *Force* ②
- 3 To test the effect and play with various **Flow** values, use the scroll bar. The **Edit** window previews the results of your transformation. The scroll bar lets you enter different **Force** values and smoothly transform the symbol. When you are satisfied, click on **OK**. If you wish to choose another effect, simply select it from the **Effects** list.

Here is an overview of available effects:

🕒 **Barrel lines**

This effect lets you transform all segments of your symbol into barrel-like curves. The higher **Force** value you specify, the more convex your curves will be. Positive values correspond to "inflating" the symbol, which means that new curves will be directed to unfilled areas. (Refer to the "Character Outlines" chapter of "Drawing Outlines" for the description of how directions of convexity are determined.)

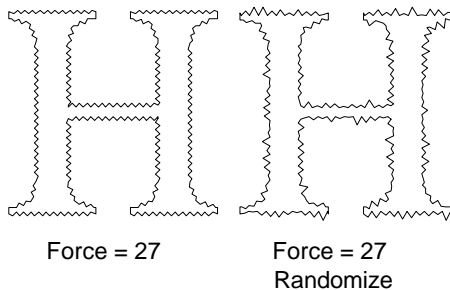


🕒 **Boldness**

This effect is equivalent to the applying the **Change Width...** command where Dist. X and Dist. Y is determined by the **Force** parameter.

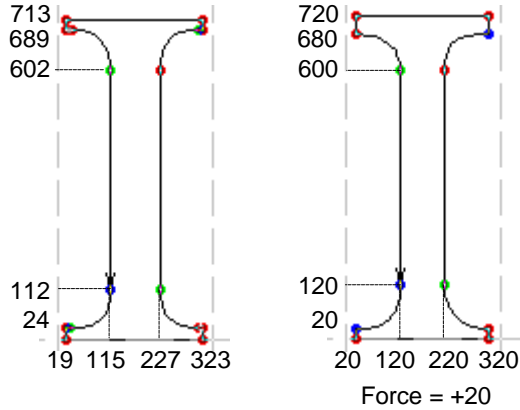
🕒 **Broken lines**

This option lets you transform all segments of your symbol into a series of vectors and control the deviation of positions of their nodes about the original stem. Higher absolute values correspond to longer distances between nodes and the stem. (There is no difference between positive and negative values.)



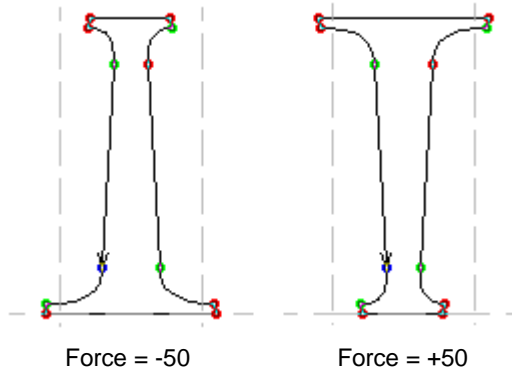
🕒 Grid

By applying this effect, you place all points of your symbol on lines of an imaginary grid, which has an interval equal to the **Force** value you have specified and starts at the baseline and left margin. Points are placed at the nearest lines of the grid. (There is no difference between positive and negative values.)



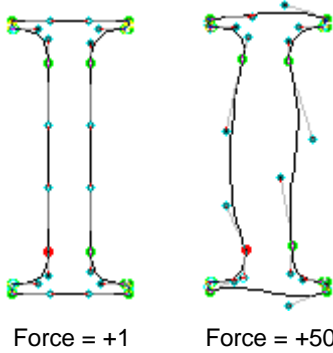
🕒 Perspective

If you imagine standing in front of your symbol made of some hard material and having a horizontal pivot in its middle part, this effect will skew the symbol about the pivot in 3D space. Positive **Force** values correspond to skewing its upper part in your direction.



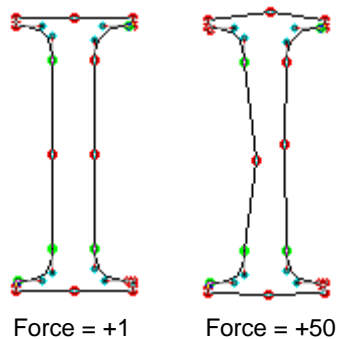
⌚ **Random curved lines**

All straight lines are transformed into curves with their control vectors lying along the segments. The higher values you specify, the more control points deviate about their original positions. Since this effect is random (as well as the next one), only absolute values are taken into account.



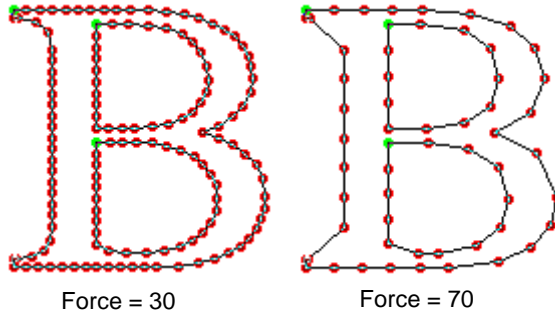
⌚ **Random double lines**

Every straight line is divided into two lines of equal length. Higher **Force** values cause greater deviations of nodes from their original positions. Other nodes are not moved.



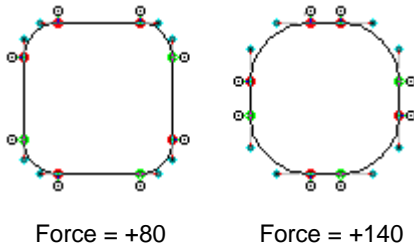
⌚ **Rendered curves**

All curves and vectors are approximated with straight lines. If the absolute **Force** value is low, the number of these lines is greater and the approximation is more accurate. If the value is high, FLW uses fewer straight segments.



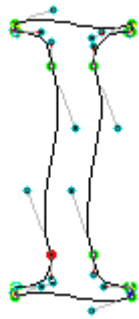
⌚ **Rounded corners**

We will use a simple rectangle to demonstrate this feature. As you see in this picture, FLW replaces angles formed by two straight lines into a curve, the control vectors of which are collinear to corresponding straight lines. Types of curve-to-line transitions are set to smooth. If you increase the **Force** value, curves will become longer.

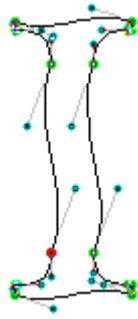


⌚ **Waved curves**

This effect replaces every straight segment with a curve as described under **Random curved lines** and then rotates control vectors to transform them into sinusoid-like lines. The greater absolute values you specify, the more FLW rotates these vectors. The **Force** value's sign determines the direction in which first control vectors of curves are rotated. If you imagine walking along a contour, every first control vector you meet will be rotated to the left if the **Force** value is positive. Second control vectors are always rotated in the opposite direction in order to create a "wave."



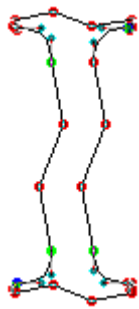
Force = +50



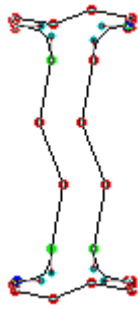
Force = -50

⌚ **Waved lines**

All straight lines are divided into three vectors that are rotated as discussed above. If we connect control points on the picture from the previous paragraph, we get the same results as after applying the **Waved lines** effect.



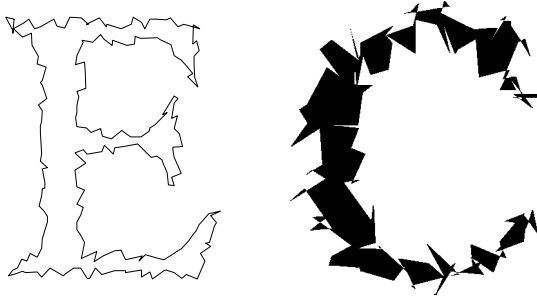
Force = +50



Force = -50

🕒 **Fantasy**

All nodes of the selected part of your contour are randomly moved by the offset, determined by the **Force** parameter. You can get the most interesting results combining this effect with the **Rendered curves** and **Broken Lines** effects.




Combination of Rendered Curves and Fantasy effects

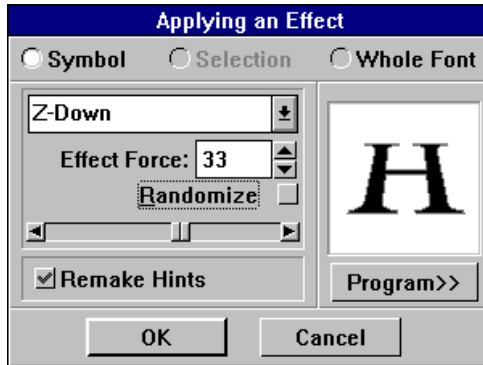
🕒 **Z-Down**

This effect makes a 3d rotating illusion. The absolute value of the **Force** parameter determines the angle of rotation.

Using Table Toolbar to Apply Effects

As with simple transformations, you can apply an effect to a single symbol, group of symbols or to the whole font using the **Table** window local **Toolbar**.

Select a desired symbol or part of a font and press the  button of **Table Toolbar**. The following dialog box appears, letting you select the appropriate options of an effect.

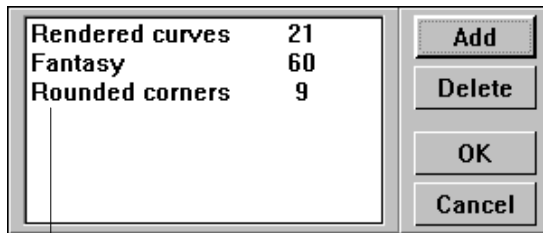


Programming Effects Sequences

Using the Table Toolbar Effects button, you can not only apply one effect at a time, but you can also create sequences of effects.

To create an effects sequence:

- 1 Press on the **Program>>** button of the Effects dialog



List of effects in the sequence

- 2 Using the effects list and the force slider, select the first effect you want to apply.
- 3 Press the **Add** button.
Repeat the last two steps for each effect in the sequence.
- 4 To delete an effect from a sequence, select it in the list and press the **Delete** button.
- 5 Press **OK** to apply to a sequence, or **Cancel** to exit.

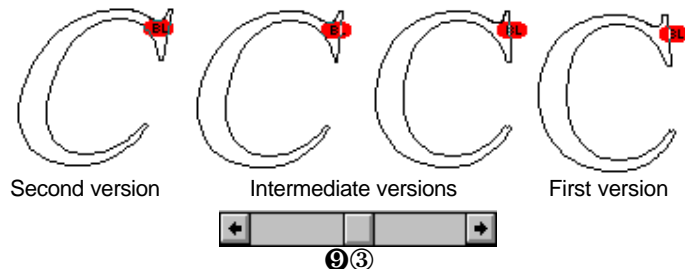
Floating Design (Morphing)

When an artist creates a new font, he/she often experiments with different versions of the same symbol to choose the best looking one. It is a rather time-consuming task since he/she has to draw several symbols on paper and nobody knows which of them will be the best. FLW allows you to draw only two versions of a symbol and then select from all possible intermediate shapes.

Once you have created the first version of your symbol, select **Set Blend** from the **Transform** menu. The current shape is placed into memory and a red blend marker (**BL**) appears in the upper right corner of the **Edit** window. Now edit your symbol to produce the second version of the symbol. When you are done, choose the **Blend...** command that replaces **Set Blend** in the **Transform** menu. Use the scroll bar in the dialog box to preview various intermediate shapes and click on **OK** when you are satisfied. Now you can select **Reset Blend** to exit the blend mode, or create another version of your symbol and choose the **Blend...** command once more.

By default, FLW offers 40 intermediate shapes to choose from. You may alter this value by clicking on **Preferences...** in the **Options** menu, and typing or selecting the **Blend Steps** value.

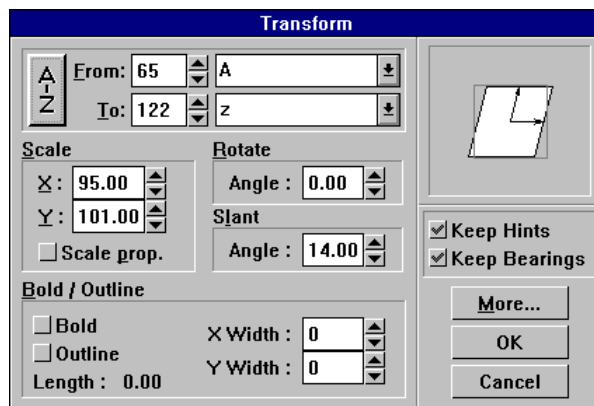
The following picture illustrates how to find the necessary oblique angle using the Blend feature.



Processing Groups of Symbols

Using an Apply to Range Command

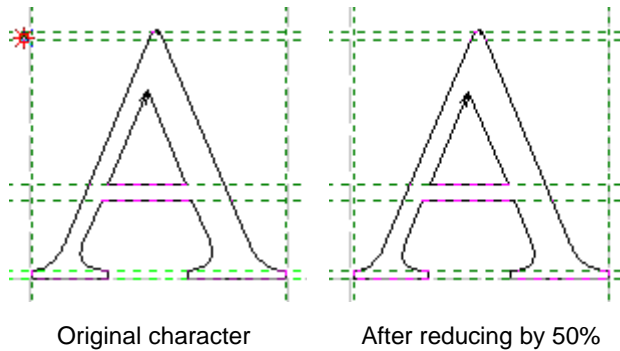
Almost all of the transformations you can apply to single characters can also be applied to several symbols at once. We have already described doing this for transformation effects in the chapter **Transformations**. If you wish to perform a uniform transformation of several symbols, select **Apply to Range...** from the **Transform** menu. The following dialog box appears:



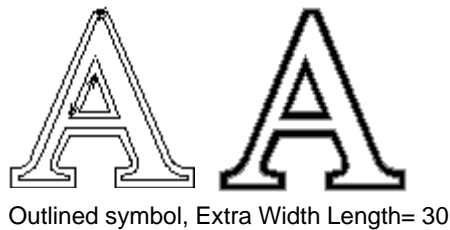
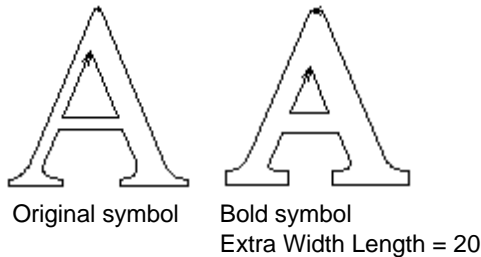
- ① In the upper left field of this dialog box, specify the range of symbols you want to transform. You can select either character codes or names. If both entries contain the same value, only that character will be processed. To select all symbols of the font, press the **A-Z** button on the left side of the symbol choosing controls.
- ① The **Scale**, **Rotate** and **Slant** fields closely resemble the dialog boxes of corresponding commands. The effect that will be achieved after clicking on **OK** is previewed in the upper right field. For example, if you choose the scaling factor equal to 50, and enter 30 in the **Rotate** field and 40 in the **Slant** field, the preview box will look as follows:



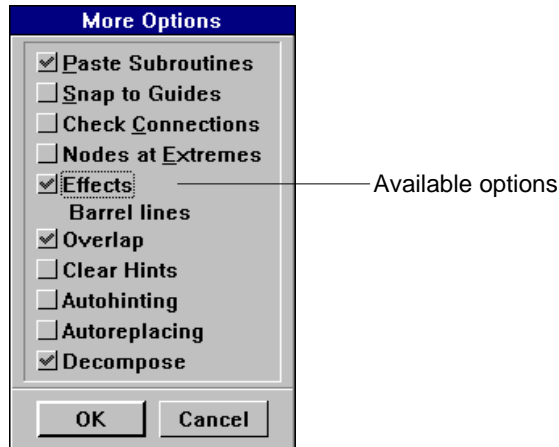
- ① The **Keep Hints** and **Keep Bearings** switches control the way hints and margins of characters are processed. If they are on, FLW tries to maintain their relative positioning. For example, you may notice the difference between the following letters if you look at the guides origin. Since it retains the absolute position about the baseline, it does not fit into the second picture.



- ① The **Apply to Range...** command also allows you to create bold and outlined fonts automatically. To do so, click on the desired option in the **Bold/Outline** field and type or select the amount by which you wish to increase the width of your character(s). With the **Bold** option selected, you may also use negative values to reduce the width of your symbol.



- ⌚ The **More...** button lets you access another dialog box that allows you to apply several other transformations.



Here is the list of options available from this dialog box with equivalent menu commands:

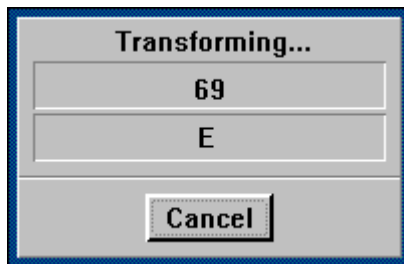
- Paste Subroutines* corresponds to **Paste All** from the **Expert** menu. This command is discussed under "Using Subroutines" in the "Advanced Features" section.
- Snap to Guides*, *Check Connections*, *Reset Connections*, *Nodes at Extremes* correspond to commands of the same names that reside in the **Transform/More...** menu. We've described the first command in the "Creating Fonts with Precision" section; the other three commands are discussed in the current section under "Automatic Modification of Contours."
- If you click on the *Effects* item, a dialog box appears which is almost equivalent to that of the **Effects...** command from the **Transform** menu. It asks you to choose the desired effect and enter its force. However, there is no **Flow...** button in this box, so you have to find the force factor beforehand by experimenting with the **Effects...** command. When you're done, click on **OK**, and the *Effects* item will appear checked with the effect's name shown below:



- (d) After clicking on the *Overlap* item, the dialog box appears that was discussed in the "Automatic Modification of Contours" chapter of this section. Choose the desired values of the overlap parameters and click on **OK**.
- (e) The *Clear Hints* item performs the same action as the **Remove All Hints** command, which is contained in the **Hints/Guides** menu, with the **Both** parameter: it deletes all hints set for the specified character(s). We have discussed this command in the "Creating Fonts with Precision" section under "Using Hints."
- (f) *Autohinting* corresponds to the command of the same name that can be found in the **Hints/Guides** menu. For more details, see "Using Hints" in the "Creating Fonts with Precision" section.
- (g) The *Autoreplacing* item lets you automatically set hints replacement points for a given range of symbols. We will discuss the hint replacing mechanism in the "Advanced Features" section under "More About Hints."
- (h) The *Decompose* item allows you to transform a composite symbol into a set of simple contours. It is discussed in the chapter "Advanced Features" in the section "Composites".

When all necessary items are checked, return to the main dialog box by clicking on **OK**.

Once in the main **Transform** dialog box, click on **OK** to begin processing the group of characters you've specified. FLW will ask you to confirm the operation since you cannot undo it. If you click on **Yes**, the following box will appear indicating the character being processed:



You may terminate the operation by clicking on **Cancel**.

Advanced Features

In this section, we will discuss advanced techniques that let you create fonts of the highest possible quality. These include:

- ⌚ composites, their creation and utilization;
- ⌚ subroutines for ensuring design consistency;
- ⌚ FontAudit, a sophisticated tool for estimation of font quality and accuracy and for improving the representation of contours;
- ⌚ defining symbol names and Unicode indexes
- ⌚ font level hints and the Flex mechanism for eliminating possible distortions of character shapes when rendering at low resolutions and small sizes;
- ⌚ setting margins and sidebearings of characters and kerning fonts for improving the appearance of strings that are typed with them;
- ⌚ checking the quality of the created font by previewing and printing individual symbols and strings of text.

Composites

This is a very powerful tool for creating complex symbols that can be represented as a group of independent subsymbols in addition to a main symbol. Subsymbols are merely other symbols of your font that don't lose their connection with their parents if being used in a composite. Frankly, you can avoid using composites at all, but in some cases it is useful to create composites, especially if you intend to transform the parents of the subsymbols and want to see corresponding subsymbols inside other composites change at the same time.

Base and Accent Symbols

Composite symbols always include two basic components: the **base symbol**, where the symbol really exists, and one or more **subsymbols** or, in Adobe terminology, **accent symbols**.

Base symbols can have contours, hints and other FLW structures or can have only links to the accent symbols. In all cases, FLW treats them as an ordinary symbols.

Accent symbols must follow these rules:

- ❶ The accent symbol and composite symbol must not be the same.
- ❷ The composite symbol cannot be used as an accent symbol in other composites.
- ❸ The accent symbol cannot be used as a composite symbol.

Common Rules

Here we will describe the main restrictions and rules accepted *de facto* in different types of fonts.

- ❶ Adobe Type 1 supports only one base symbol and one subsymbol named accent. They both are tightly connected with their parents, i.e., if you change their parents they will be changed automatically at the same time inside the composite symbol. The base symbol has no offset in the drawing field in comparison with its parent. The accent may have a non-zero offset from its parent, so you are able to move the accent within your composite without moving its parent. Your font containing composites must use only **Standard Encoding**.
- ❷ TrueType format supports many subsymbols. They are all tightly connected with their parents as well, i.e., if you change their parents, they will be changed automatically at the same time inside the composite symbol. Each subsymbol can have a non-zero offset from its parent. You may also apply to any subsymbol (by Import/Export operations) such transformations as : Shifting, Scaling, 2x2-Matrix transformation. If those transformations are applied to a composite, this composite will be pasted (i.e., transformed into an ordinary contour) by Import/Export operations.

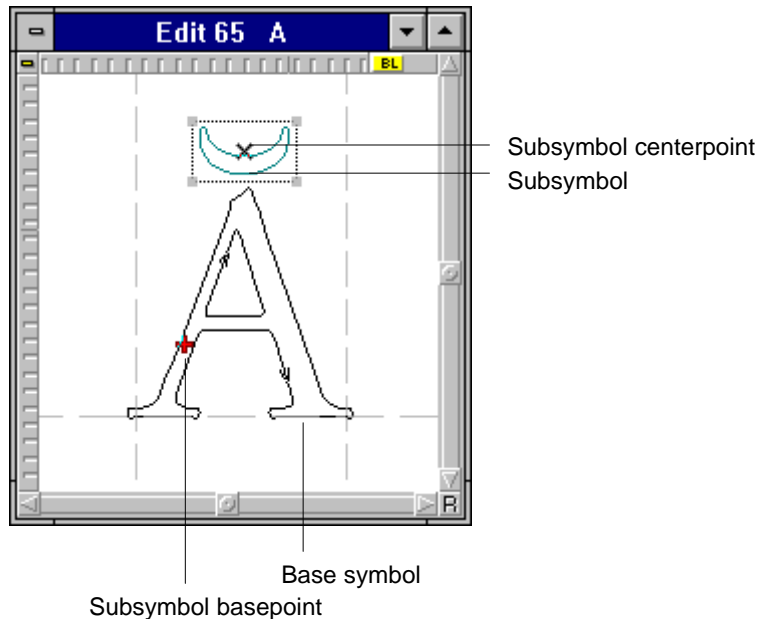
- ⌚ VFA format used in FLW supports as many subsymbols as you wish. They are all tightly connected with their parents as well, i.e., if you change their parents they will be changed automatically at the same time inside the composite symbol. Each subsymbol can have a non-zero offset from its parent. You may also have an ordinary contour as a subsymbol, but in this case, the composite will be pasted (i.e., transformed into an ordinary contour) without fail by Import/Export operations.

As you can see, different font formats set many different rules for using composites. But when you edit your font in FontLab, you need not worry about these rules, because all of them will be applied automatically during import and export of your font files.

In fact, in using composites, you'll be able to flexibly manipulate your font, because you have no need to additionally hint all the composites involved in your font. However, you may lose a little accuracy because of motion of subsymbols during scaling operations using composites.

Creating a Composite

Press the secondary button on the **Table** menu holding **CTRL** simultaneously. Then drag the cursor without releasing either button to the **Edit** window, release the button and **CTRL**. You'll get the symbol you just dragged in the **Edit** window as a subsymbol (accent symbol) of the composite to be edited.

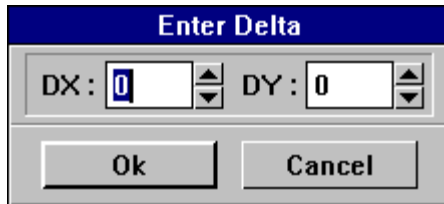


Handling a Composite

If you click the primary button somewhere on the contour of a subsymbol, you'll see a dashed rectangle and two special marks (red plus and black cross).

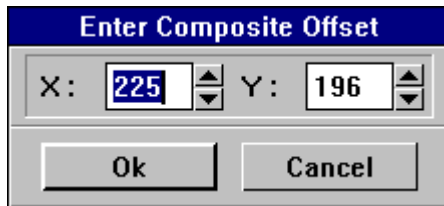
- ⌚ Dragging by the marks or by any point inside the rectangle you can move the whole rectangle until you release the button. When the button is released, you'll see the whole subsymbol move. It will now stay in the rectangle.
- ⌚ Press and hold the primary button on the red plus and then click the secondary button to delete the subsymbol.

- ⌚ Press and hold **SHIFT** on the black cross and then click the secondary button to decompose the subsymbol. You can do the same using the menus **Symbol** and then **Decompose**. This will transform the subsymbol into an ordinary contour. You will not be able to undo this operation - be careful!
- ⌚ Hold **CTRL** and click the primary button on the black cross to enter a dialog box to input the value of delta. Delta lets you move your subsymbol precisely from the current position.



Delta input menu

- ⌚ Hold **CTRL** and click the primary button on the red plus to enter a dialog box to input the value of composite offset. This lets you set an offset that will be used in the future as an offset between the parent and your subsymbol.



Offset input menu

Both special marks (if you drag by them) will normally snap to hint lines, so that lets you easily place your subsymbol in the desired position. Finally, if you press **ESC** or the secondary button, you'll exit the edit mode of your subsymbol.

Dynamic Duplicating

When you want to create a copy of a symbol, which will include all hints, guides and sidebearing information, you can copy it, using the **Symbol** menu commands of the **Table** window. However, when you copy your symbol, it loses all links with its parent. To save this link, you can use a special technique called **Dynamic Duplicating**.

To duplicate a symbol using this technique:

- ❶ Select the destination symbol in the **Edit** window using the **Table** window or **Prev** and **Next** commands.
- ❷ Erase this symbol, if it exists. The **Edit** window must contain only two crossed lines showing that the symbol was erased.
- ❸ Holding the **CTRL** key, drag the source symbol into the empty Edit window, and drop it.

You will see a composite-like copy of the source symbol, with saved sidebearings information. Now, if you edit the source symbol, your destination symbol will be updated automatically.

Using Subroutines

Another key to maintaining design consistency is preserving the identity of those character features that should be exactly the same. Almost all fonts have repetitive elements, such as serifs and bowls. Inaccuracies are inevitable when such elements are created individually for each symbol of a font, especially in the case of complicated decorative typefaces. Another PostScript structure, *subroutines*, lets you define repetitive elements once and then simply refer to them when necessary. In addition, this feature lets you greatly reduce the storage space needed for your font.

A Type 1 subroutine is a contour used by several symbols at once. Therefore, you have to define it only once and then "call" it for each symbol that contains this segment. In fact, Type 1 fonts usually contain libraries of subroutines with various characteristic features. Once such a library is filled, you can easily construct new symbols from ready-made parts.


NOTE: TrueType fonts can't work with subroutines, but FLW lets you work with them even in this case. By export operations, they will be pasted into contours accepted by TrueType fonts.

It is important to keep in mind that subroutines are global elements common to several symbols. Thus, all these symbols are affected when you modify the subroutine to which they refer.

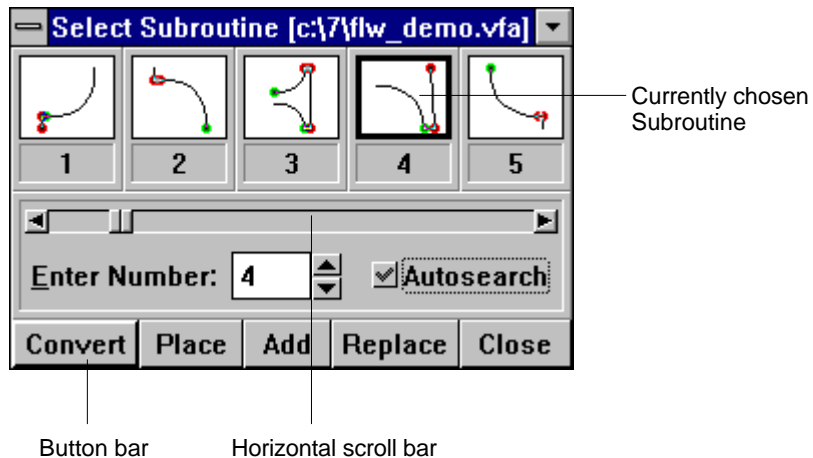
Creating a Serif "F" with the Use of Subroutines

Suppose you're creating a serif font and its capital "E" is ready. Let's create the capital "F" quickly and precisely. Besides showing how to use the subroutines, this example reviews several useful techniques.

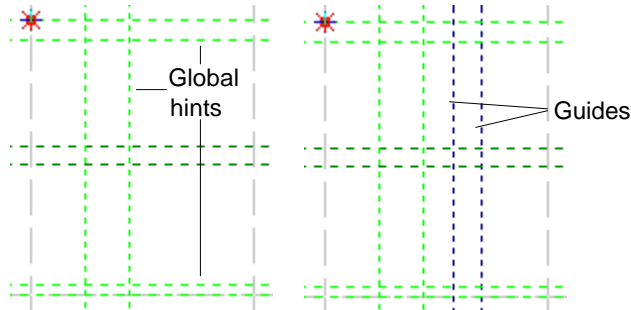
NOTES:

1. Before experimenting with subroutines, make sure that they will be displayed in the **Edit** window. For this purpose, click on the  icon contained in the **Options** box.
2. To repeat the steps we discuss here, open FLW_DEMO.VFA, bring its capital "E" into the **Edit** window, and select **Copy Symbol** from the **Symbol** menu. Then, create a new font, choose its capital "E" (which appears dimmed in the **Table** window), and select **Paste Symbol** from the **Symbol** menu. Now press **CTRL+V** to bring the "F" cell into the **Edit** window and initialize it by clicking on **New Symbol** in the **Symbol** menu. You're ready to complete the steps outlined below.

You can enter the following menu by pressing **F4** or from the **Expert** menu. We'll call this panel the **Subroutine** panel.

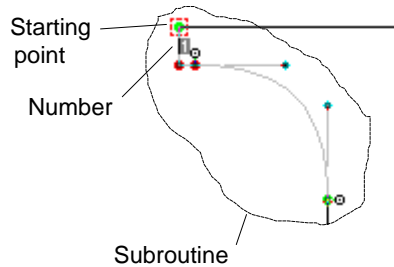


- 1 Mark the main stems of your "F" with hints. Since some hints of the "E" occupy the same positions as the hints you must place for the "F," it would be useful to define them as global while creating the "E" and just choose them for your "F." (The lower horizontal hint will fix the pair of serifs associated with the main vertical stem.) You also need to mark the lengths of straight parts of the upper and middle stems; for this purpose, use two guidelines.

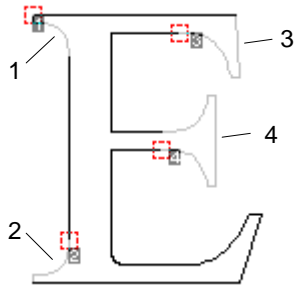


- 2 Bring the letter "E" into the **Edit** window. Select the upper left serif and choose **Add** from the **Subroutine** panel. The selected segment will be converted to a subroutine.

Several visual changes will occur to your segment: it will become gray, its starting point will be surrounded with a red square, and a number will appear below the square. This number is the identifier of your subroutine.



- ③ In the same fashion, convert three other serifs to subroutines.

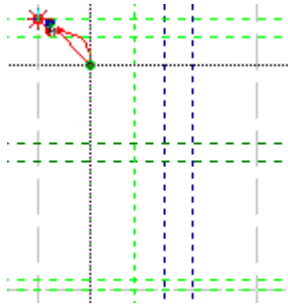


Note that the **Subroutine** panel contains five buttons: **Convert**, **Place**, **Add**, **Replace**, and **Close**. The button **Add** is already familiar to us; the fourth one (**Replace**) lets you replace an existing subroutine with the new subroutine you're defining. Now you have to specify the subroutine you want to replace. Choose the subroutine you want to work with using the **Enter Number** box, the scroll bar, or finally, by clicking on the desired subroutine if it is currently seen in one of the five small boxes in the upper part of this panel. You can also see a magnified image of a subroutine by clicking the primary button on its picture while simultaneously holding down the secondary button. The **Subroutine** panel shown above will stay on the desktop until you have closed it. However, you can set the item **Minimize On Use** from the system menu. This will hide the **Subroutine** panel into an icon until you have activated it again.

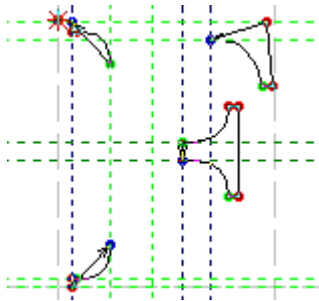
- ④ Once all subroutines are defined, return to editing your "F." You can go either of the two ways shown below.

🕒 **Alternative #1**

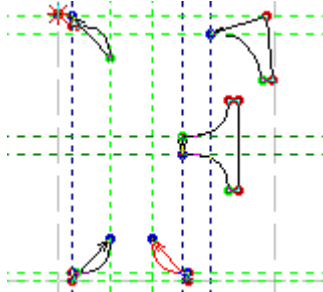
- 5 Choose the subroutine you want to paste into a contour. Select **Place** from the **Subroutine** panel. This command pastes subroutines as ordinary contours and is available only when no segments are selected. Its contour will appear highlighted in the center of the **Edit** window.
- 6 Activate the **Move** tool and place your segment at the desired position by moving it via its start- or endpoint. Make sure that the **Snap To Guides** and **Snap To Hints** modes are on to enable "sticking" to marking objects. (To check the snapping modes you're working in, select **Edit Window...** from the **Options** menu.)



- 7 Repeat the **Place** command for all other subroutines. The **Edit** window will look like this (we've added one more guideline to simplify the placement of serifs):



- 8 Next, we have to create a segment symmetrical to the lowermost serif on the above picture. To do so, highlight the latter serif, press **D** to duplicate it, and select **H Mirror** from the **Transform** menu to create its horizontal reflection. Then, use the **Move** tool to position the serif.

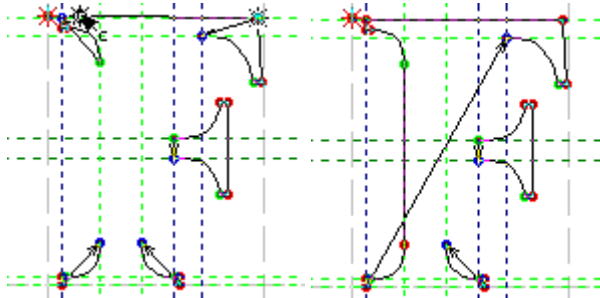


- 9 Now we must connect the serifs to define a single outline. We will use a combination of the following techniques:

- ⌚ Joining segments with the **Combine Contours** command from the **Transform** menu (shortcut: **F**);
- ⌚ Resetting directions of segments with the **Change Direction** command from the same menu (shortcut: **D**);
- ⌚ Extending outlines with the **Contour** tool.

For more details on how to apply these techniques, consult the "Using the Contour Tool" and "Transforming Outlines" chapters of the "Drawing Outlines" section. Here we will just point out the sequence of steps you're to complete.

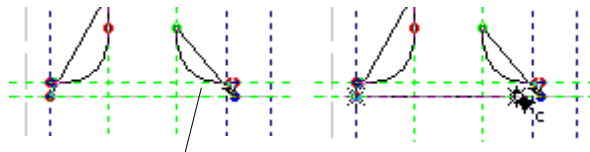
- (a) Join the first three serifs in the counter-clockwise direction beginning from the upper right serif using the **Combine Contours** command.



Combining the first pair of serifs...

After joining three serifs

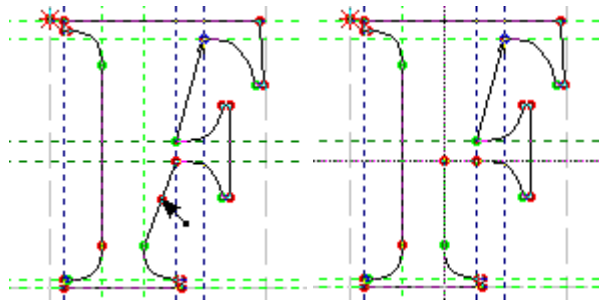
- (b) Since the direction of the mirrored serif is opposite the direction of all other segments, invert it by using the **Change Direction** command. Then, join it to the previous serif.



Inverted contour

Joining serifs...

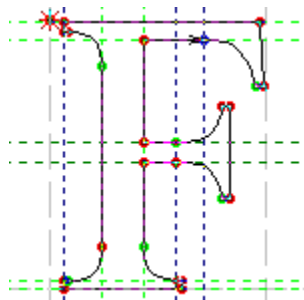
- (c) To join the fourth and fifth serifs, use the **Edit** tool to add a point on the vector which connects them by clicking the primary button while holding down **CTRL**, and move this point to the necessary position.



Add a node and...

move it to the desired location

- (d) Activate the **Contour** tool, click on the endpoint of the next to last segment and draw two straight lines. Congratulations! You've just completed drawing the outline of your "F."



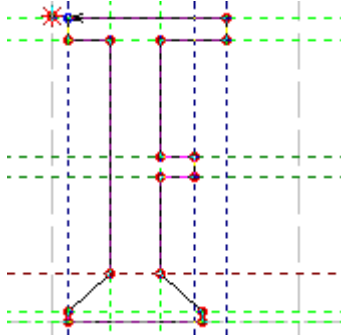
⑩ Now we'll do two last steps.

First, we will associate the serifs of the "F" with the subroutines from which they were obtained. To do so, select them successively and, each time, choose the **Add** or the **Replace** command. **Add** will convert the selected part of your symbol into a subroutine and, additionally, place it on the list of your subroutines as the end subroutine. You can find it if you move the scroll bar in the **Subroutine** panel to the very right edge. Correspondingly, **Replace** will replace the current subroutine from the **Subroutine** panel for the selected part of your symbol after converting it into a subroutine. Later, FLW will use exactly the same serifs for the capital "E" and "F" of your font. You may also select the serif obtained by mirroring the existing contour and then declare it as a subroutine by choosing the **Add** command as described in step 2. If you now modify any serif of your "F," the corresponding serif of the letter "E" will be changed accordingly.

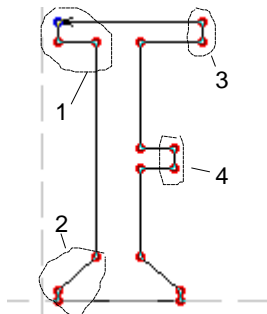
Secondly, select **Autohinting** from the **Hints/Guides** menu to add stem hints not yet defined.

🕒 **Alternative #2**

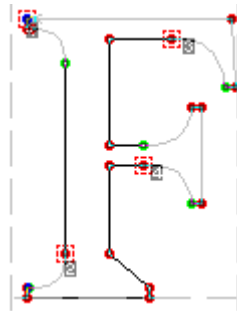
- 5 Create a sketch of your letter with the **Contour** tool. Please note, however, that startpoints of serifs must be placed exactly where they are intended to be. This will eliminate the need to adjust positions of serifs later. Since you've marked all important lines with hints and guides, you will draw the sketch in no time. Define your contour in the counter-clockwise direction as shown below. (We've added an additional global guideline to mark the horizontal coordinate of startpoints of the lower pair of serifs.)



- 6 Successively select the segment that must be replaced with a subroutine, choose the subroutine that should be placed instead of that segment and use the **Convert** command to substitute it for the currently chosen subroutine as discussed in step 10 above.

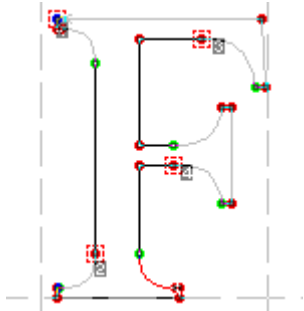


Successively highlight these segments...



and convert them to subroutines.

- Now we must define the last serif. Use the **Place** command to paste the symmetrical serif, the **Change Direction** command from the **Transform** menu to invert it, and the **H Mirror** command from the same menu to obtain its horizontal reflection. Then, press **SHIFT+DEL** to move this segment to the Clipboard, highlight the part of your sketch that must be replaced with the serif, and press **SHIFT+INS**. The serif will be pasted exactly where it is intended to be.



- The last touch we recommend is adding hints with the **Autohinting** command and declaring the last serif as a subroutine.

Pasting Subroutines

Like all other global objects, subroutines may be pasted so that they will become ordinary contours. To paste a single subroutine, choose the **Paste Subroutine** command from the **Expert** menu (or press **F2**) and click on any node belonging to this subroutine. If you wish to paste all subroutines used by the current symbol, select **Paste All**, or press **CTRL+F2**.

Using Autosearch

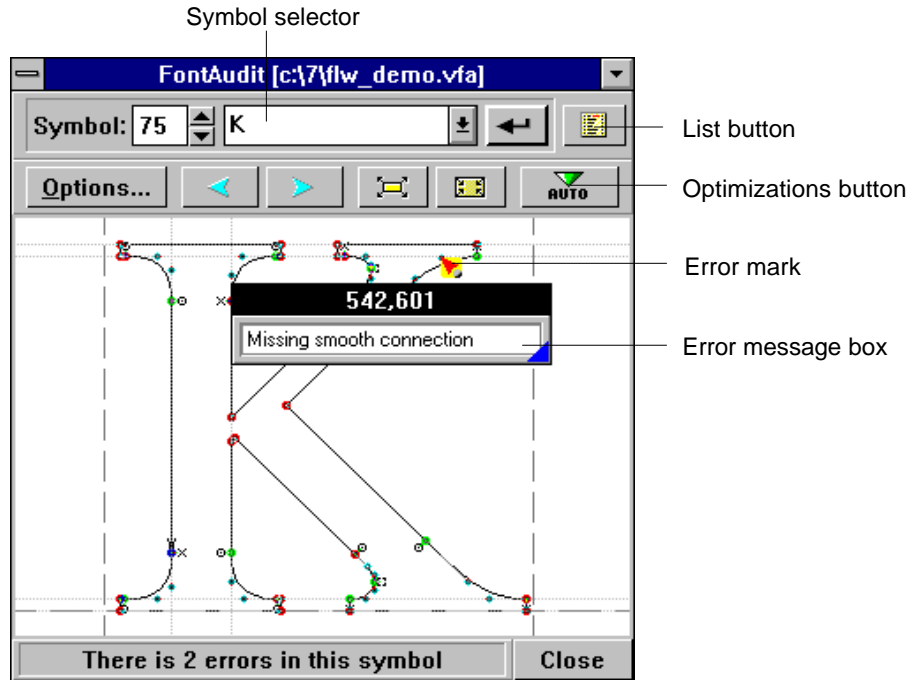
As you may have noticed, there is a special **Autosearch** button in the **Subroutine** panel. If it is on, and you select some part of your symbol and this part is *absolutely equivalent* to a subroutine from your list of subroutines, you'll see a quick green flash over the **Convert** button and that subroutine (that is equal to the selected part) will become the current one. So you can press **Convert** then and this part becomes a subroutine.

Drag-Drop Technique






Using the drag-drop technique already described several times (see: **Table** window), you can drag the desired subroutine with the secondary button to the **Edit** window. If there is a selected part in your symbol when you're dragging a subroutine, you'll be asked to convert that part to the dragged subroutine. This is equal to the **Convert** command. If you reject this offer or there is currently no selected part, you'll be asked to place the subroutine. This is equal to the **Place** command. You will be able to decline this last offer, too.

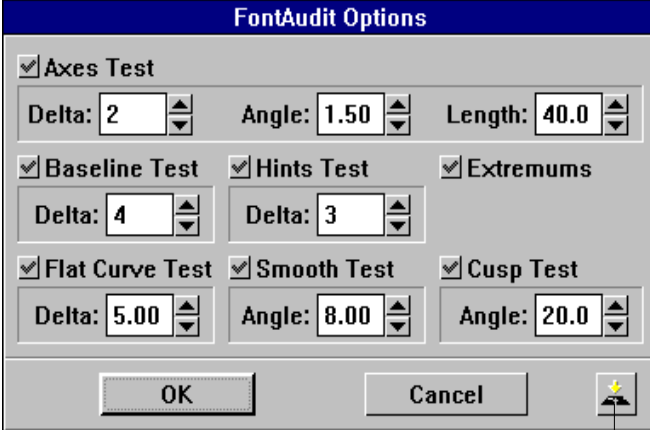
FontAudit

FLW has an advanced set of tools called **FontAudit** that lets you apply various tests and optimizations to your font or to a single symbol. You can enter the following panel from the **Expert** menu.



You'll see the current symbol inside the window and possibly several yellow-red arrows that show probable errors. Clicking the primary button on one of those marks, you'll get a message explaining the reason for marking this point as erroneous. You may either ignore this mark or agree with it and correct it. Without leaving this panel, you may change the current symbol by either entering a new symbol code or using the symbol scroller.

The  (**List**) button lets you transfer all error information appearing in **FontAudit** to the **Monitor** window and save it on disk. The second row of buttons contains, correspondingly, the **Options**, the two **Shift**, the two **Zoom** and the **Optimization** buttons. Using the  and  buttons, you can change the current symbol in **FontAudit**. Clicking the  button, you'll be asked to select an area to magnify. Clicking the  button, you can undo this magnification. Clicking on the **Options** button, you enter the following dialog box:



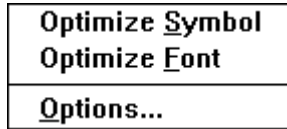
The dialog box is titled "FontAudit Options" and contains several sections of controls. The first section is "Axes Test" with a checked checkbox and three input fields: "Delta" (2), "Angle" (1.50), and "Length" (40.0). The second section contains three checked checkboxes: "Baseline Test", "Hints Test", and "Extremums", with "Delta" input fields of 4 and 3. The third section contains three checked checkboxes: "Flat Curve Test", "Smooth Test", and "Cusp Test", with "Delta" (5.00) and "Angle" (8.00) input fields for the first two, and an "Angle" (20.0) input field for the third. At the bottom are "OK" and "Cancel" buttons, and a "Save options button" icon (a floppy disk) on the right.

Save options button

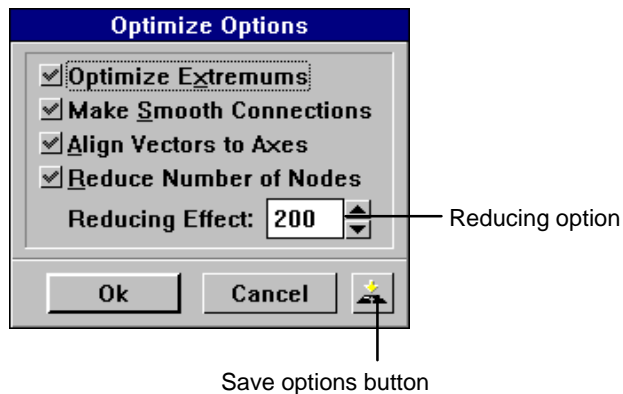
You see a number of various options that can be tuned experimentally. Pressing the save options button, you can save the desired options on disk. We *strongly advise* that only sophisticated users change those options.

Automatic Optimization

Pressing the  button of the FontAudit panel, you enter the following menu:



Using this menu, you can apply several optimizing transformations to either the whole font or the current symbol. Choose the desired options from the **Options** dialog as follows:



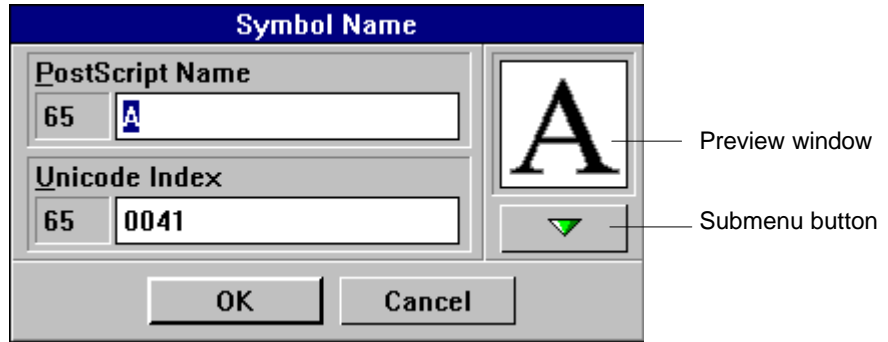
and then apply the chosen set of optimizations to either the whole font or the current symbol. We **strongly advise** that only sophisticated users change those options.

You can tune the **Reducing Effect** experimentally: the more value you enter - the more nodes will be erased.

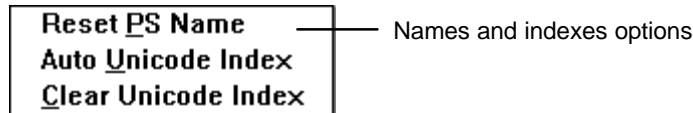
Pressing the save options button, you can save the desired options to disk. Sometimes you may notice that the quality of your font decreases after using those optimizations. This tool is designed for *real optimization*, nevertheless, it is not a universal remedy. In some cases, you're better off using your own hand to provide the quality needed rather than relying on any automatic gadgets. (This advice is also very useful in many other areas of life.) We mainly recommend using **FontAudit** for a better understanding of possible mistakes made while designing by hand.

Symbol's Names

Each symbol in a font is identified by its unique code and name. Whereas codes fix positions of symbols in a font, names are descriptive labels of characters and may be changed. To do so, highlight the symbol you want to rename, click on the **Change Name** (🔍) icon or go to the menus **Symbol**, **Symbols Names**, **Rename Symbol**, respectively, and you'll enter the following dialog box:



You can change the name of the current symbol and its **Unicode** index using the **PostScript Name** and **Unicode Index** boxes. Pressing the **Submenu** button as shown in the picture, you enter the following menu.

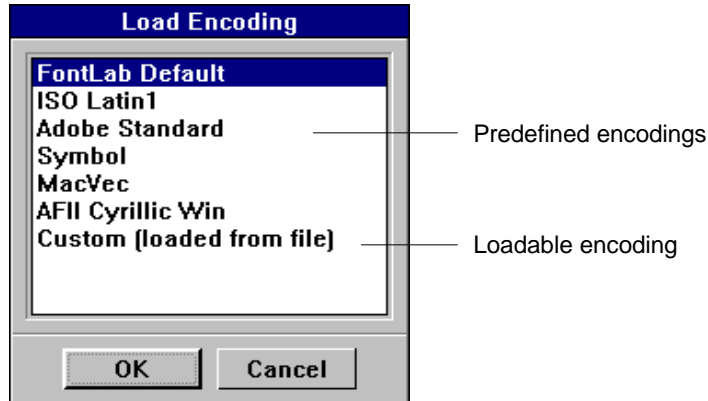


If you enter a new PostScript name but want to come back to the previous one, press **Reset PS Name**. This will bring you to the initial state with the name. **Auto Unicode Index** lets you set a **Unicode** index created automatically for this symbol. **Clear Unicode Index** clears the **Unicode** index, making it empty.

Additional names-related commands are in the **Symbols Names** part of the **Symbol** menu.

Assign Names to Font

If you click on the Assign Names to Font, you'll be asked to choose encoding for your font.



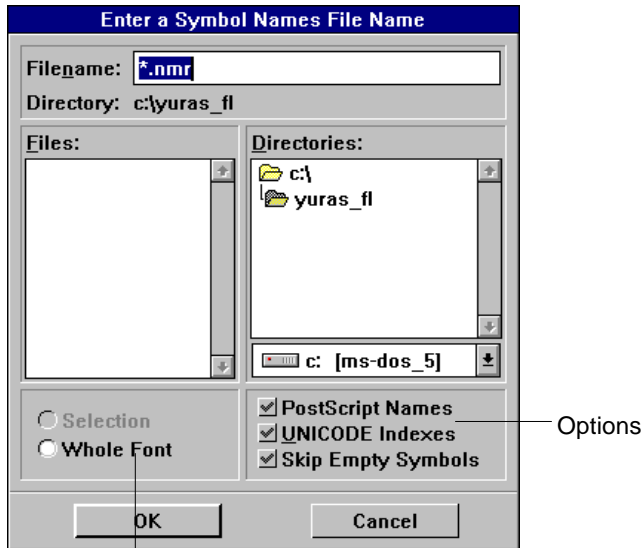
You can choose from a number of prescribed encodings or load a customized encoding (**Custom (loaded from file)**). If you prefer the latter, you'll get a standard file dialog box for loading a file with your encoding.

Predefined Encodings

Name	Description
FontLab Default	This encoding is an optimized WinANSI encoding with the added ability to "understand" some symbols, which are not included in WinANSI
ISO Latin 1	This is one of the standard encodings, which is used in many printers as default. ISO Latin 1 defines names for nearly all European characters and symbols.
Adobe Standard	This is the most standard encoding for Type 1 fonts. However, Adobe Standard Encoding has places for only 150 symbols.
Symbol	This is a version of Adobe Encoding specially for symbols, included into Symbol fonts.
MacVec	Defines standard names for symbols in Mac-compatible fonts
AFII Cyrillic Win	Based on the ISO standard, defines names for all Cyrillic symbols which are supported by Windows 3.1.

Loading and Saving Names

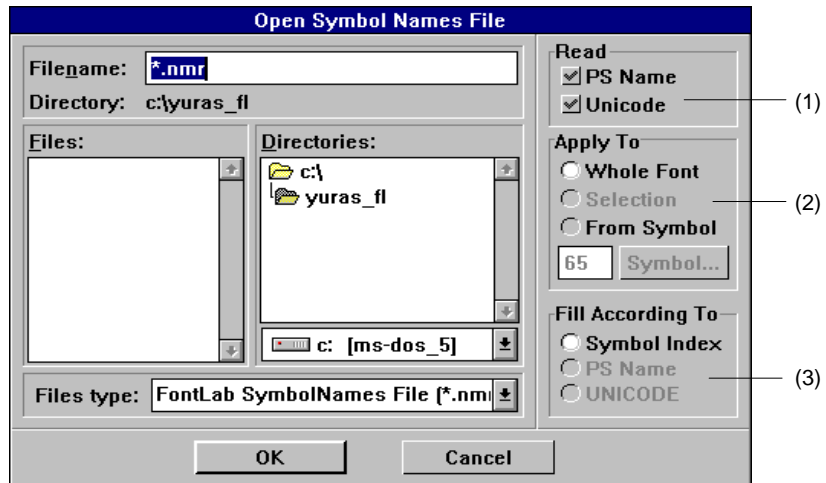
After selecting the **Save Names** command, you'll see the following:



Application area

Using this file dialog box, you can save the symbol names of either the whole font or the currently selected part of the font. As additional options, you can save **PostScript** names and/or **Unicode** indexes within your file. All empty symbols may be skipped as indicated on the third option button.

If you select **Load Names**, you'll enter the following dialog:



Here you're allowed to read the symbol names from a file. Additionally, using the options in this dialog box, you can read the **PostScript** and/or **Unicode** names if they are presented in the file to be read from. You can also apply the names to the whole font, selection or to all symbols starting from the number entered in the box **Symbol**. The subdialog **Fill According To** lets you set the source of the fill of your font. You may tie your symbols to either **Symbol** indexes or **PostScript** names or **Unicode**. This subdialog is available only if you selected only one position in the subdialog **Read**.

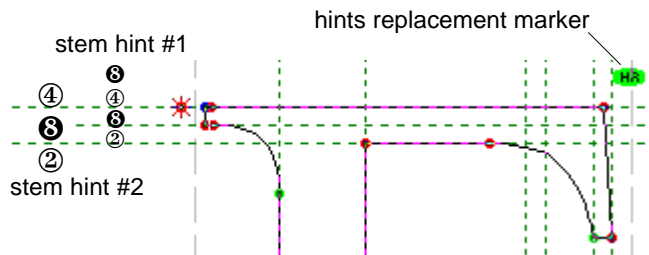
More About Ensuring Proper Reproduction

The Adobe Type 1 font format is loaded with structures that help to maintain the original design of characters when rendering them at limited resolutions and small sizes. We have discussed one of these structures, stem hints, in the "Using Hints" chapter of "Creating Fonts with Precision." Now we will deal with the following techniques:

- ① hint replacement, which allows you to avoid stem hint overlaps and, at the same time, hint all important features of your characters;
- ① font level hints that control vertical alignment of character features and perform overshoot suppression when rendering at a small number of pixels;
- ① defining standard stem widths;
- ① controlling the appearance of bold characters at small sizes with the **ForceBold** parameter;
- ① using triple stem hints to ensure better rendering of some symbols;
- ① the Flex mechanism that deals with the rendering of shallow curves.

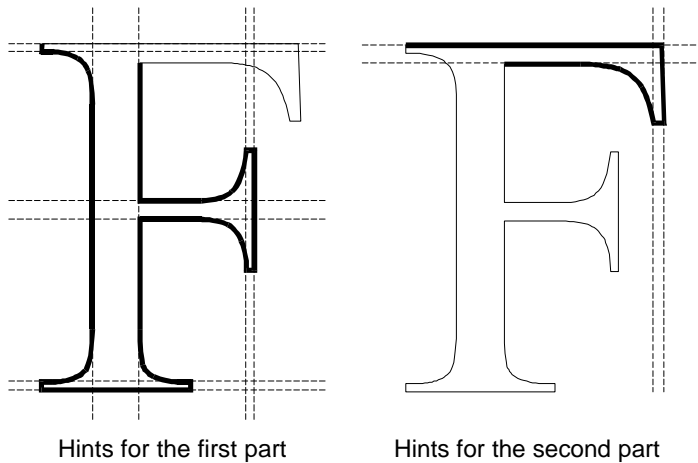
Using the Hints Replacement Technique

If you've created the capital "F" as discussed in the "Using Subroutines" chapter, you have noticed that a green oval-shaped marker appears in the upper right corner of the **Edit** window after performing the **Autohinting** command. FLW also places two horizontal stem hints in the upper part of the "F":



As we have already mentioned in the "Using Hints" chapter of "Creating Fonts with Precision," stem hints must not overlap. However, in the case of our capital "F," we need to hint both the upper horizontal stem and the left serif. The solution: we split the outline into two parts and define a separate set of hints for each of them, or *replace* one set of hints with the other. FLW indicates that it used this technique to hint your "F" by showing the "HR" marker.

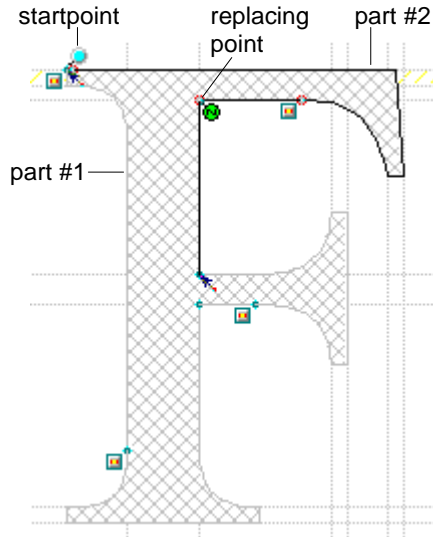
Now let's discuss which parts of the outline must have different sets of hints. Obviously, the upper left serif and the upper horizontal stem must belong to different parts of the outline. For example, we can divide the outline like this:



The hints replacement mechanism is not supported by old versions of PostScript interpreters. The key to properly reproducing your fonts on such devices is defining a special set of non-overlapping hints for them (this set is said to be associated with the invisible *zero startpoint* of your contour) and then creating the necessary number of sets associated with the usual startpoint.

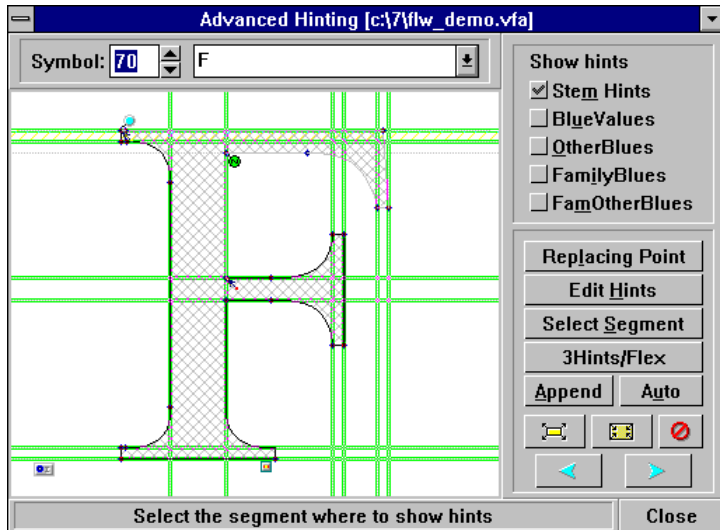
The points where a PostScript interpreter inserts a new set of hints are called *replacing points*. Since all stem hints are associated with the endpoint of a vector or a curve, the replacing points always coincide with the endpoints of segments preceding every new character's portion. In the picture above, replacing points are situated at the connections of the bold and thin outlines.

The following picture illustrates the placement of a replacing point that corresponds to the example discussed above:



The autohinting algorithm applies the hint replacement technique automatically, so usually you need not worry about overlapping hints. However, you may want to examine how the hints are assigned to different portions of your symbol. Then, if there is a better and more efficient way, you can perform this procedure manually. To do so, follow these steps:

- ① Place all necessary hints either manually or using the **Autohinting** command as discussed in the "Creating Fonts with Precision" section. Don't be worried if some of them overlap. We will avoid these overlaps by changing hints for the different portions of your symbol as shown below.
- ② With all hints in place, select **Advanced Hinting...** from the **Expert** menu, or press **CTRL+F7**. The following dialog box appears containing the current symbol:



FLW shows the following elements of your character:

- ① nodes that do not belong to subroutines;
 - ① startpoints of contours are marked with light blue filled circles;
 - ① startpoints of subroutines are marked with subroutine markers (☐);
 - ① stem hints are shown as pairs of gray or green lines. Green is used if there are no overlapping hints defined for your symbol, or if you used the **Autohinting** command and it applied the hint replacement technique.
- If an area of your symbol is controlled by two hints, it is filled with yellow diagonal lines as on the screen snapshot shown above;
- ① replacing points are marked with green filled circles.

In the upper part of the dialog box, two familiar boxes are situated that let you select the symbol you want. Once the desired symbol's name appears highlighted in the drop-down list, click on **Refresh** to display it in the working window. You may also use the arrow buttons to scroll through your font.

The standard **Zoom In** and **Zoom Out** buttons let you magnify a portion of your character. Use them in the same fashion as the corresponding buttons contained in the **Tools** box.


- ③ If there are no conflicting hints, all buttons situated in the right part of the dialog box are dimmed (except the **3Hints/Flex** button that is discussed later in this chapter). In this case, bring another character into the working window as discussed above or quit the dialog box by clicking on **Cancel**.

From now on, we will assume that the character displayed in the working window has some zones where its hints overlap.

- ④ If you have not used the **Autohinting** command, all hints will appear as thin gray lines. It is a good idea to see whether the desired results may be achieved automatically. Therefore, click on the **Auto** button that will perform the hint replacement just as the **Autohinting** command does.

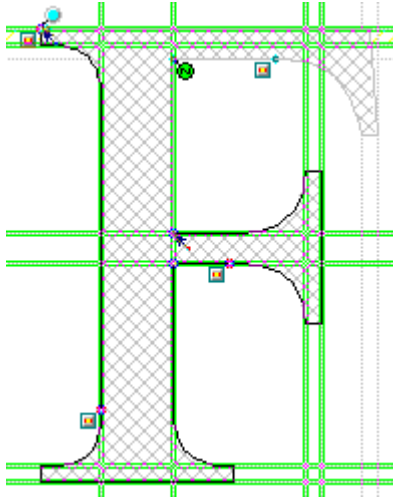
To examine which hints are set for a certain part of your symbol, click on the **Select Segment** button. Then, click on any segment belonging to this part of the outline. In fact, only the overlapping hints are of interest to you since this is the only difference between the sets of hints that were chosen automatically.

- ⑤ To rearrange the hints:

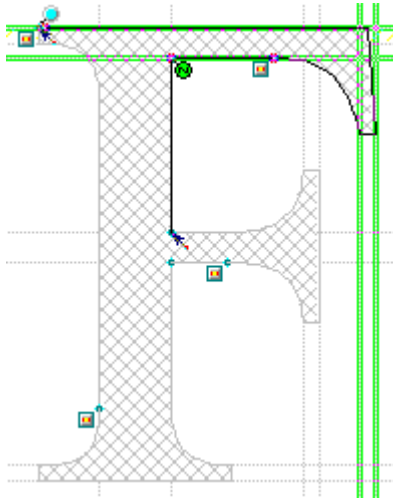
- ① Click on the  button situated in the lower right corner of the dialog box. The existing hints replacement settings will be cleared.
- ② Click on the **Set Replacing Point** button. Then, if you click on a node, a replacing point marker will appear at it. An arrow indicating the corresponding joining point will appear at the startpoint of the segment. If you click on a subroutine marker, the replacing point and the joining point will coincide.


By default, a replacing point is associated with the contour's startpoint. Therefore, if you need to divide your contour into two portions, indicate only one more replacing point. The rule is: to define N different sets of hints, place N-1 replacing points.

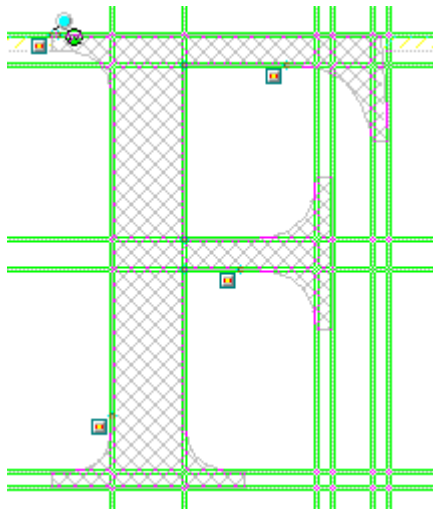
In our example, we choose the following hints for the first part of the capital "F":



When you're finished with choosing hints for the first part of your character, use the same procedure for all other parts. For example, the second part of our "F" would be hinted like this:



To define a special set of hints for older PostScript devices which do not support the hint replacement capability, explicitly associate a replacing point with the startpoint of your contour when no other replacing points are set. You will notice that the whole contour is surrounded with a bold line. Then, press the **Select Segment** button and click on the zero startpoint marker () situated in the lower left corner of the working window. The outline will become light gray and all nodes will change their color to light blue. Now use the **Edit Hints** and **Append** buttons as described above. When choosing between the overlapping hints, select those that are placed on more important stems of your character. For example, the special set of hints for our "F" might look like this (we preferred to hint the upper horizontal stem rather than the serif):



When you're finished with the special set, press the **Select Segment** button and click on the contour's startpoint. Then, define the replacing points in the usual manner.


Font Level Hints

Besides stem hints that define individual properties of characters, the Adobe Type 1 format uses font level hints to perform the vertical alignment of character features. As we discussed under "Examining Characters" in the "Creating Fonts with Precision" section, all characters in a font are aligned with specific lines, namely the baseline, the cap-height line, the x-height, the ascender-height and descender-depth lines. Each of these lines may have a neighboring overshoot position line. Your font also may have some other alignment zones like the superior baseline zone. The font level hints let you maintain the alignment of character features falling within these zones. For some reason (a historical reason, as Adobe says), these zones are called *blue*.

When your font is reproduced at a limited resolution or small size, the character overshoots may appear too prominent. In such situations, the font level hinting system lets you escape this unwanted effect by suppressing all overshooting features (those that are aligned to the overshoot lines).

A special category of font level hints lets you coordinate the character features of different styles of the same typeface when they are mixed in text. When such *family font level hints* are set, all members of a family will be aligned to the same vertical positions.

We must emphasize the fact that the font level hints, as the character level hints, are not at all mandatory. However, your font will be reproduced far better, especially at small sizes, when these hints are in place. You may define only some types of font level hints for your font; for example, if you do not intend to uniformly align all styles of a family, omit the family hints.

Every array has the  button that lets you access the following commands:

- ① **Restore** allows you to undo the changes you've made;
- ① **Clear** removes all existing values from the array;
- ① **Duplicate** >> and << **Duplicate** copy values to the array situated to the right or left of the array you're in. The purpose of this operation is discussed under "FamilyBlues" and "FamilyOtherBlues" later in this chapter.
- ① **Advisor** generates all these values automatically. In principle, you can always use this command and simply fine-tune the values, if necessary.

Now we will concentrate on where to specify the properties of the alignment zones of your font.

① **BlueValues**

The first pair of this array is the baseline alignment zone. (This is a so-called *bottom zone* since the bottom parts of characters are aligned to it.) The first value of the pair is the baseline overshoot position which is typically negative.

All other pairs define alignment zones for the tops of your characters, hence they are called *top zones*. These include: the x-height and the x-height overshoot position, the ascender-height and the ascender-height overshoot position, the cap-height and the cap-height overshoot position, etc. The total number of pairs may count up to 7.

The distances between different blue zones and their widths are controlled by the **BlueFuzz** and **BlueScale** parameters, which are described below.

① **OtherBlues**

You may specify up to 5 bottom zones in this array. The typical pair that is usually contained in this array is the descender-depth overshoot position and the descender-depth.

① **FamilyBlues**

This array defines the alignment zones that will be used when several styles of the same typeface are mixed in text. The rules of filling the **FamilyBlues** array are identical to those of **BlueValues**, so the contents of the latter array defined for the standard style of the family are usually copied into the **FamilyBlues** arrays of all other fonts belonging to the same family. The **Duplicate** buttons discussed above help you perform this operation.

⌚ **FamilyOtherBlues**

Everything we said above for the **FamilyBlues** array is true for **FamilyOtherBlues**, except it corresponds to the **OtherBlues** array of the standard face.

The following parameters affect the alignment control and overshoot suppression:

⌚ **BlueScale**

This value sets the point size at which the overshoot suppression will be turned off. If the **Scale Translation** stands in **pt.** mode, you can enter the point sizes that correspond to a 300-dpi device. If the **Scale Translation** stands in **1 pixel** mode, you can enter the pixel sizes that correspond to a 300-dpi device. If you output your font at a different resolution, the point size will be recalculated by the PostScript interpreter.

In principle, you may disable the **BlueScale in pt.** mode and enter absolute values (these values will appear in the resulting Type 1 program). However, the point size values are clearer.

The **BlueScale** value is directly related to the maximum alignment zone height defined for your font. The formula is:

$$(\text{pt. size} - 0.49) \diamond (\text{max height}) < 240$$

For example, if the maximum height is 10, the **BlueScale** in pt. can be 25 but not 26.

⌚ **BlueShift**

The **BlueShift** value also affects the way overshoots are suppressed. If your character occupies fewer pixels than the figure based on **BlueScale**, the overshoots will always be suppressed. However, if its size exceeds this value, the overshoots will be reproduced if their size in character space units is greater than **BlueShift**, or if their height is more than one-half pixel. The default value of **BlueShift** is 7.

As we will discuss later, the Flex mechanism relies on the **BlueShift** value.

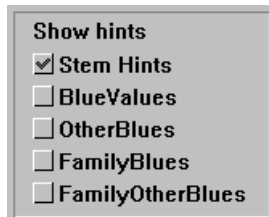
🕒 **BlueFuzz**

The **BlueFuzz** value extends the heights of alignment zones in the case of horizontal stems. If a horizontal stem is outside a blue zone by less than **BlueFuzz** character units, it will be treated as belonging to it. Adobe recommends setting this value to zero and affect the alignment of horizontal stems by adjusting the heights of blue zones.

In order to avoid overlaps of blue zones, you must define them at least (2 \diamond **BlueFuzz** + 1) units apart from each other. For example, if you use the default value equal to 1, your blue zones must be at least 3 units apart from each other.

Viewing Font Level Hints

The **Advanced Hinting** dialog box discussed under "Using the Hints Replacement Technique" earlier in this chapter lets you preview the blue zones defined for your font. To do so, use the buttons situated in the upper right field of the dialog:



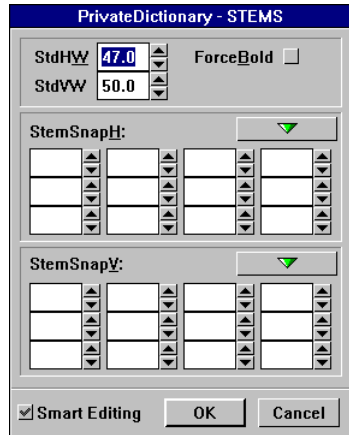
By default, only the stem hints are shown for your symbol. To display font level hints, click on the corresponding button and then on **Refresh**. If a particular zone is only a few pixels tall, you will be able to see it only at a magnified view.

Font level hints may be also previewed in the **Edit** window. You need to select **Edit Window...** from the **Options** menu and choose whether you want to display all blue zones as they are defined (**BlueValues**) or taking into account the BlueShift value (**BlueShift Only**).

Standard Stem Widths and ForceBold

The PostScript rendering algorithms are capable of normalizing stem widths to uniform values. In fact, this normalization is necessary when you have not set enough stem hints to ensure the proper reproduction of your font at small sizes.

To specify the standard widths of stems, select the **Font Info...** command from the **File** menu and then click on **Stems**. The following dialog box appears:



The dialog contains the familiar **Smart Editing** button that forces your cursor to stay in the first empty cell of every field. The arrow buttons open drop-down menus with the **Restore**, **Clear** and **Advisor** commands that act just like the corresponding commands in the **Blues** dialog.

NOTE: To get quick help for any control in the Stems dialog box, move a cursor to it and press a secondary button.

The purpose of the four arrays and the **ForceBold** button is as follows:

⌚ **StdHW**

This array contains a single value in character space units that declares the dominant width of horizontal stems. If the width of a particular stem differs from this standard value, at small sizes, the stem will be rendered as if it had the standard width. However, at larger sizes this suppression will not occur.

⌚ **StdVW**

This entry is the dominant width of vertical stems. If you create an italic font, measure it as a distance along a line perpendicular to the vectors forming the main stem.

⌚ **StemSnapH** and **StemSnapV**

These arrays of 12 entries each contain the most common widths of horizontal and vertical stems, respectively. Please note that the **StemSnapV** array must be left empty if you create an italic font. The values must be sorted in increasing order; however, you need not worry about this since FLW sorts them automatically.

⌚ **ForceBold**

At small sizes, both bold and normal characters may appear to have the same width, so the bold effect will disappear. To force the PostScript interpreters to use a special technique that allows you to make bold characters appear more prominent at all sizes, enable the **ForceBold** mode.

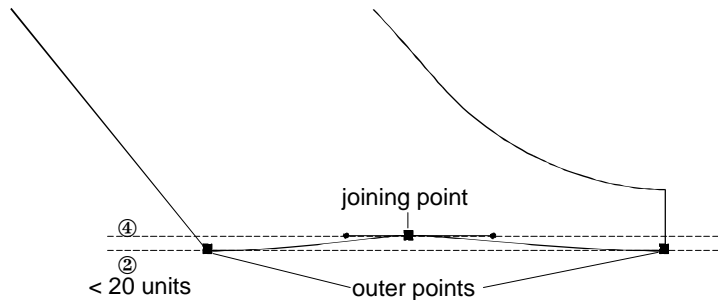
Flex

The Flex mechanism deals with the rendering of nearly horizontal and vertical curves. As with the overshooting features, these shallow curves must disappear at small sizes when it is impossible to reproduce them correctly.

If a part of your symbol meets the following requirements, the Flex mechanism may be applied to it:

- ① this portion of your symbol must be formed by exactly two Bézier curves;
- ② the joining point and the control points of vectors associated with it should all be collinear and either precisely horizontal or vertical;
- ③ the outer points of the segment (the other ends of the Bézier curves) should be either precisely horizontal or vertical. The control vectors associated with them may be oriented as required to produce the necessary shape;
- ④ the distance between the control vectors associated with the joining point and the line passing through the outer points, which is commonly referred to as the *Flex height*, should not exceed 20 character space units.

The following picture illustrates all these conditions:



Whenever a shallow curve (e.g., a cupped serif like the one shown above) is situated within an alignment zone, the best results are achieved when the joining point stays on the leading line of this zone (not on the overshoot position line). For example, the joining point on our illustration should be positioned on the baseline.

The 20 units value is an absolute limit; however, the maximum Flex height that occurs in your font should be less than **BlueSHIFT**. If you use the default **BlueSHIFT** equal to 7, all Flex heights must be less or equal to 6.

By default, FLW checks all symbols to apply the Flex mechanism wherever possible. Therefore, if your symbols meet the requirements discussed above, they will be controlled by Flex.

You can explicitly disable the Flex feature for a specific symbol. To do so, select **Advanced Hinting...** from the **Expert** menu, bring the necessary symbol into the working window, and click on the **3Hints/Flex** button. Then, disable the Flex mechanism by clicking on **Flex**.

When you export a Type 1 font, you can also disable the generation of Flex or set several Flex options. For details, see the corresponding chapter of "Managing Fonts."

Triple Hints

To ensure better control over some symbols which have three horizontal or vertical hints that are equally spaced, the Type 1 format incorporates a special kind of stem hint, which is called *triple hints*. If your symbol meets the following conditions, a triple hint may be generated instead of three usual stem hints:

- ① the widths of two extreme stems should be the same;
- ② the distances between the middle of the center stem and the middles of two extreme stems must be equal.

The following characters are candidates for triple hints: "m," "...," and "÷."

As with the Flex mechanism, FLW will try to replace stem hints with triple hints wherever possible unless you disable this feature by using the **Advanced Hinting...** command (the control buttons become available after clicking on **3Hints/Flex**).

Adjusting the Spacing of Characters

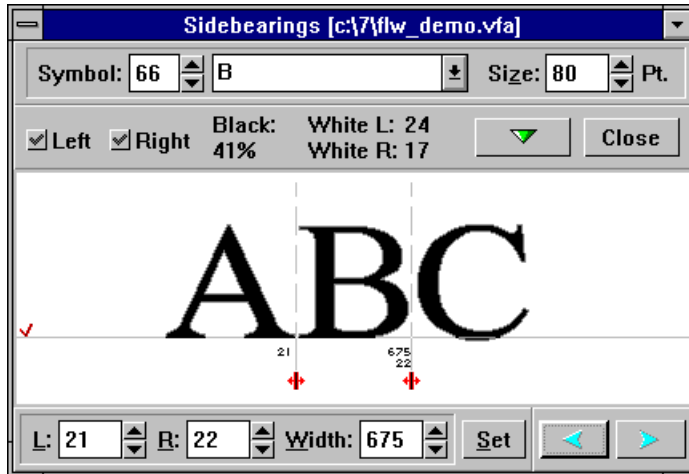
In order to improve the appearance of strings typed with your font, you need to adjust the mutual positioning of characters. With the exception of monospaced fonts like Courier, all typefaces are fine-tuned using the following two techniques:

- ⌚ adjusting widths and sidebearings of characters to improve their appearance in a string regardless of the neighboring symbols;
- ⌚ setting individual spacing attributes, or *Kerning values*, for specific pairs of characters.

Setting Widths and Sidebearings

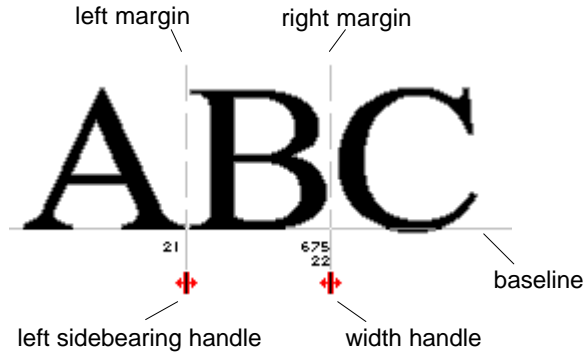
As we have already discussed in the "Moving Margins and Baselines" chapter of "Creating Fonts with Precision," you can set the width and left sidebearing of your character by positioning margins with the **Edit** tool. However, when the character's outline is defined, you need to examine the filled character in a string to determine whether it is necessary to reset these values. The only criteria used here is the visual appearance of your character; therefore, if you're not a professional typographer, you may need to consult with one on some design aspects of the font you're creating. This is also true for the kerning process discussed below.

The **Set Sidebearings...** command from the **Expert** menu provides a full cycle for setting widths and sidebearings. It lets you view a filled character alone or surrounded with one or two neighboring symbols and interactively change its spacing properties. Once you choose this command, the following dialog box appears:



- 1 To choose the symbol for which you want to set the spacing attributes, specify its name in the usual boxes situated in the upper portion of the dialog box, or use the arrow buttons that reside in the lower right corner.
- 2 If you wish your symbol to have a "neighbor" at the left, right, or both sides, click on the corresponding button in the upper left part of the dialog box. Then, choose a symbol by its code or name (you may specify any symbol you want including the one already displayed). These symbols are added for preview purposes only and are not affected when you modify the properties of your character. If you click again on one of these buttons, the corresponding symbol will disappear.

- ③ The **Size** box situated in the upper right corner of the dialog box lists the previewed character's size in points (1 pt is equal to approximately 1/72 inch). You may customize the preview feature so that the dimensions of symbols displayed on your screen will match those of these characters printed on paper at the same point size. For more details on this capability, see the "Customizing FLW" section under "Tuning Display Properties."
- ④ Let's take a closer look at the central part of the dialog box:



The character "B", the properties of which we are examining, is shown with its margins. Each margin has a handle with a number indicated above. Naturally, the figure indicated above the left handle is the left sidebearing value and the character's width and right sidebearing value are shown above the second handle. These values are also contained in the **L**, **R** and **Width** entry boxes at the bottom field of the dialog box.

- ⑤ To change the left sidebearing value or the width, drag the corresponding handle. If you move the left handle, the right margin will be moved accordingly since the width will be kept the same. The figure indicated above the handle will reveal the current offset relative to the previous position.

In addition to the interactive method, you can change the values by typing or selecting them in the **L**, **R** and **Width** boxes. Once a value is entered, click on **Set** to display your character with a new setting.

NOTE: Though you may position the margins anywhere about the character, they must be **OUTSIDE** the character's outline to ensure its proper appearance with any neighboring symbols. Always keep in mind that the left sidebearing and width properties are general characteristics of characters that are used by output routines to reproduce them in any possible string. Therefore, you must preview your symbol with various "neighbors" to determine the optimum spacing attributes. Both too much white space and too dense spacing lead to poor visual appearance.

Kerning Your Font

As you may have noticed, there is too much white space between certain symbols that is impossible to eliminate by positioning the margins. To adjust the spacing attributes of such pairs, the *kerning* technique is used.

The main idea behind kerning is to allow symbols belonging to specific pairs to extend beyond each other's margins. (These pairs are commonly referred to as *kerning pairs*.) For example, consider the following two words:

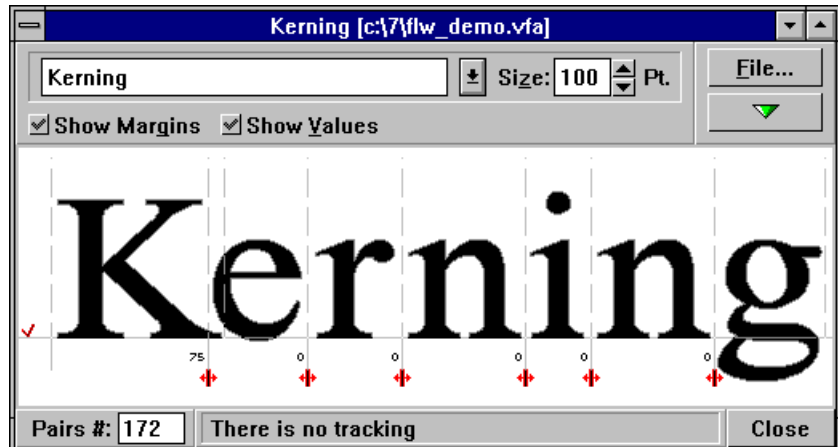
Kern Kern

With kerning

Without kerning

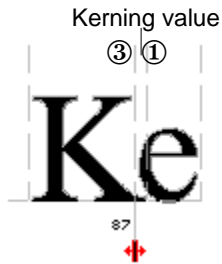
The unnecessary white space between "e" and its neighbors was eliminated in the first word by using the specific properties of their shapes. For example, the left curved part of the lowercase "e" may be placed above the lower right serif of the capital "K." This curved feature is beyond the right margin of the "K." This effect may be achieved only for combinations of "e" with similar characters. For example, you can project it into the capital "A," but with "S" it will not work.

To kern your font, use the **Set Kerning...** command from the **Expert** menu that displays the following dialog box:



By default, all characters are shown with three associated elements:

- ① their margins, which appear as light gray dotted lines;
- ② handles for changing mutual positions of characters in a string;
- ③ kerning values in character space units which are indicated above the handles. These values correspond to the offset between the right margin of the left symbol in a pair and the left margin of the right character of the same pair that appears due to elimination of a certain amount of spacing. You can also loosen your string; then, the kerning values will be negative. When you enter this dialog box for the first time, all these values are zeros.




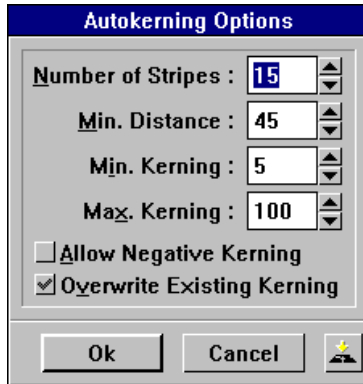
If you wish to "hide" the margins or kerning values, use the switches situated in the upper part of the dialog box.

To set the necessary amount of kerning for a string of characters:

- ① Enter this string in the text entry box that resides in the upper left part of the dialog box. Or, click on the **File...** button and use the standard dialog box to open a text file with kerning pairs. It is useful to create a file containing all typical kerning pairs and then simply load it whenever you need to adjust the spacing attributes for a given font. Your file may consist of several lines of text. In this case, the first line is displayed in the text box and you can choose a string in the standard way.
- ② To set the point size at which your string will be displayed, type or select the necessary number in the **Size** box.

③ The typical kerning definition procedure is as follows:


- 3.1 Press on the  button and select **Autokern String** command in the menu. All kerning values for the current string will be set automatically. The algorithm divides all symbols into a number of layers and tries to move symbols together. By selecting an **Options...** command in the menu, you can adjust autokerning parameters:



Number of stripes	defines the number of layers into which symbols are divided.
Min. Distance	defines the minimum amount of spacing between these layers.
Min. Kerning	defines the minimum absolute value of kerning.
Max. Kerning	defines the maximum absolute value of kerning.
Allow Negative Kerning	if switched off, negative kerning values (which increase spacing between symbols) are never generated
Overwrite Existing Kerning	if switched off, kerning will be generated only for symbols which do not have it.

- 3.2 To fine-tune the automatically set kerning values, position symbols by dragging the handles. If you wish to enter an exact amount of kerning, hold down the **CTRL** key and click on the corresponding handle. In the dialog box, type or select the desired figure or click on the **Auto** button to determine this value automatically.

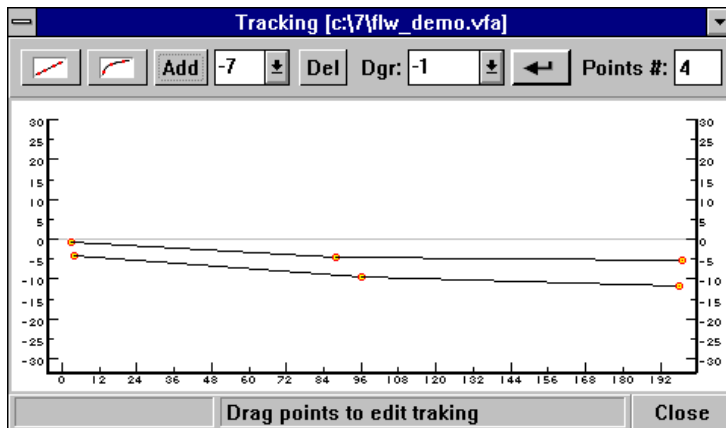
NOTE: If you hold **SHIFT** and click on the kerning mark, you'll enter a Sidebearings panel with the symbol, on which mark you click.

- ④ The **Kerning** dialog box lets you reset all kerning values to zero for the current string or for your entire font. To do so, select the **Clear Kerning in String** or **Clear Kerning in Font** command in the  menu.

If you select the **Clear Kerning in Font** command, FLW will ask you to confirm erasure of all kerning information for the current font.

Defining Tracking Information

By choosing the menu **Expert** and the submenu **Set Tracking** you can enter the following dialog box:



Some programs that can read .AFM files provide users with the possibility of setting different character widths depending on the point size of your symbols. This feature is recommended for professional users only.

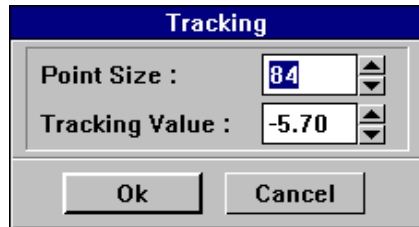
Let's describe the buttons and boxes in the upper row. They are - **Linearize**, **Logarithmize**, **Add**, **Degree Set**, **Delete**, **Degree Select**, **Enter** and **Points**.

If you want to add a new line which will describe the desired tracking, select the *degree* of your tracking by using **Degree Set**. This number will characterize this line in the future.

Then click on **Add** and you'll see a new line in the window. You can drag any point on this line by the primary button.

To add a new point into an existing line, move the cursor to the desired position in the line, hold **CTRL** and press the primary button. You can achieve the same effect by holding the secondary button, then clicking the primary one.

You can also refine the position of a point by holding **CTRL** and pressing the primary button. The following dialog box will appear:



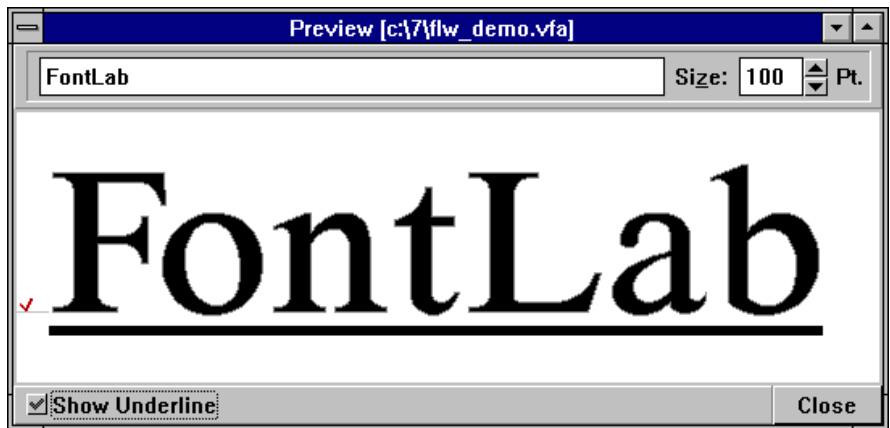
Set the desired values and click on **OK** then. To delete a line, click on **Delete** and you'll see a modified cursor, move it to the desired line and press the primary button. The line disappears. We recommend that you have the **Preview** window on the screen simultaneously. You can then see all the changes in widths you just made using the **Tracking** window if you press on the **Enter** button either in the window or on the keyboard. Because you've possibly made several lines (not just one), you may wish to see the changes that correspond to one of them. Using the **Degree Select** box, select the desired line or **None** (the latter one means you want to see the original setting) and press **Enter**. You'll see a changed string in the **Preview** window. The **Linearize** button lets you make a broken line straight. Click on this button and you'll see a modified cursor, move it to any point on a desired line and click the primary button. You'll see the line straighten. You may use the same technique if you want to transform a line with the **Logarithmize** button. **BE VERY CAREFUL** as there is no undo for these operations. The box **Points** shows the number of points that currently belong to all your lines.

Producing Samples

The final criteria used to detect possible errors and improve the appearance of your font is the examination of text printed with your font. FLW offers two options: you can either preview a string of text on your screen, or print several types of samples.

Previewing a Text String

For this purpose, select **Preview...** from the **File** menu or press **R**. The following dialog box appears:



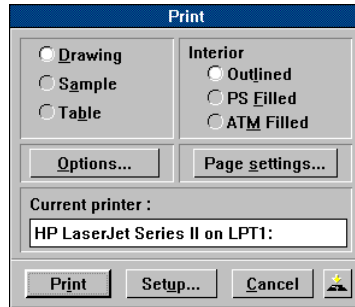
As in the **Kerning** dialog box discussed above, you can specify any string you want in the entry box and type or select its point size. You can make **Preview** show the underline of your string by clicking on **Show Underline**.

To preview the string exactly as it would be printed, tune your display by using the **Display...** command from the **Options** menu. For details, see the "Adjusting the Display Resolution" chapter of "Customizing FLW."

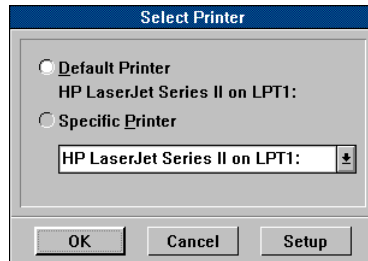
Unlike the **Show** window, the **Preview** feature takes into account the contents of **FontMatrix**. For details, see the "Managing Fonts" section under "Inside the Font Header."

Printing Samples

To access the printing capabilities of FLW, select **Print...** from the **File** menu or press **CTRL+P**. The following dialog box appears:



- 1 Before you proceed with selecting printing options, please take the time to check the setup of your printer (the active printer name is displayed in the **Current printer** field). To change the printer or set it up, click on the **Setup...** button (you can also use the **Printer Setup...** command before choosing **Print...**). You will enter the following dialog box:

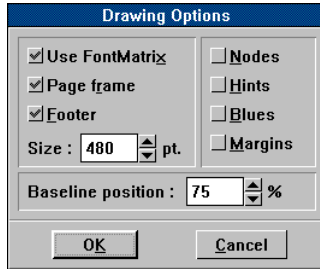


Here you can either select the default Windows printer (for details, see your "Microsoft Windows User's Guide"), or any installed printer from the drop-down list. If you click on **Setup**, a printer-specific dialog box will appear letting you specify such options as paper size and source, page orientation, printer memory, number of copies, etc.

- 2 FLW allows you to print the following three types of samples (you select the desired type by clicking on the appropriate title in the **Print** dialog box):

🕒 **Drawing**

This option prints the symbol that appears in the **Edit** window. You can customize the print settings by clicking on the **Options...** button and selecting the necessary modes in the dialog box:



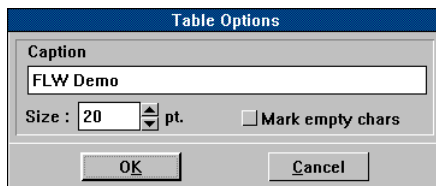
- If the **Use FontMatrix** item is checked, your symbol will be using the contents of FontMatrix. For details, see the "Managing Fonts" section under "Inside the Font Header."
- **Page Frame** surrounds the printed page with a border.
- If **Footer** is on, an information box will be printed in the lower right corner of the page.
- The **Size** entry box lets you specify the point size in which your character will be printed.
- The **Baseline position** entry lists the percentage of the baseline offset from the upper border line. By default, this value is equal to 75%, which means that the baseline of your character will be positioned at an offset equal to 3/4 of the page height.
- You can choose the elements you want to print along with the character from the right field of the **Drawing Options** dialog.

🕒 **Sample**

This mode is equivalent to previewing a text string on your screen, only the string you enter after clicking on the **Options** button will be printed.

🕒 **Table**

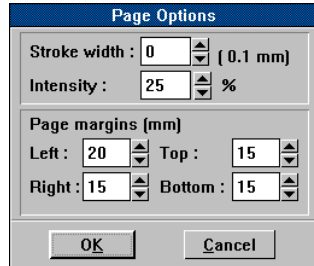
This option lets you print the character map of your font. The **Options** dialog will look like this:



You can enter the heading string (by default, the name of your font is used), its point size, and specify whether you want to print crossed lines in the cells that correspond to empty characters.

- When the sample type is set, choose the fill mode in the upper right field. We have discussed the PostScript and ATM fill modes under "Automatic Modification of Contours" in the "Transformations" section.

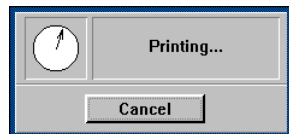
The **Page settings...** button invokes the following dialog box:



In its upper part, set the **Stroke width** (the width of the outline) and the **Intensity** of black fill if you want to print solid letters. Please note that 1 unit of **Stroke width** corresponds to 0.1 mm. For example, to specify the outline width equal to 1 mm, enter 10 in the **Stroke width** box.

The lower field of this dialog allows you to set the page margins in millimeters.

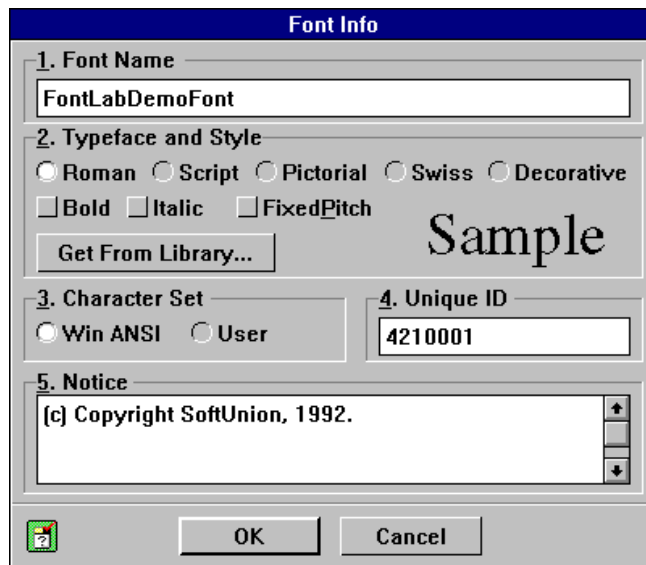
- If you want to make permanent the settings you've specified, click on the **Save to Disk** icon situated in the lower right corner of the **Print** dialog.
- When you're finished, click on **Print**. The following box will appear indicating the current state of the print process:



Managing Fonts

From the menu **File** and submenu **Font Info**, you can enter the font management dialog box. If you set the **Beginner** mode in the **Options** menu, you'll get the reduced form of this dialog box. Let's start there:

Beginner Mode



This dialog box lets you edit a very small set of font parameters; all the rest will be calculated automatically. We *highly* recommend that you use the reduced form of the dialog box only for the very first steps of your study. Later, whether you like it or not, you'll come to the more sophisticated **Advanced Mode** described below.

To define a Font Info in this dialog box:

- ❶ Describe your font by setting up visual correlation between your font and the **Sample** preview. You can do this by manipulating the controls in the **Typeface and Style** group. To select all options from the Typeface Library, press the **Get From Library...** button.
- ❷ Select the character set of your font by selecting one of the options: **Win ANSI** and **User**. Choose Win ANSI if your font is created with the Windows standard character encoding and User - if not.
- ❸ Enter the **UniqueID** number. To make your font fully compatible with all PostScript printers and programs, enter the unique number in this box. Adobe recommends numbers > 4000000. If you plan to create only TrueType fonts or you will use Type 1 fonts only for your own purposes, enter 0.
- ❹ In the **Notice** box, enter any additional information of the font. It may be your name or copyright or something of the kind.

NOTE: By clicking the secondary button on various fields of this dialog box, you get quick help. This is a snap to use!

Advanced Mode

Now we're going to introduce you to the full (advanced) form of the **Font Info** dialog. Don't be afraid of this bulky looking dialog box. It appears if the beginner mode is off.

Font Info

Metrics Typeface Other

FamilyName: FLW Demo
FontName: FLW_Demo
FullName: FLW Demo
Menu Name: FLW Demo

FontMatrix: ScaleX 100.0 Slant 0.0
 translate ScaleY 100.0

Weight: Normal
ItalicAngle: 0.0 FixedPitch

Blues Stems Custom

Ei

Based upon the Latin printing style of the 15th to 17th century, with a mild diagonal contrast in stroke emphasis (lighter in upper left to lower right, heavier in upper right to lower left) and bracketed serifs.

MS Family: Roman UniqueID: 4210001
PCL ID: 5 Times
VP ID: 14 Dutch
PANOSE Family: Any
IBMClass: OldstyleSerifs Library...
SubClass: Dutch Modern

OK Cancel

Let's describe the many fields of this huge dialog box.

⌚ **FamilyName** and **FullName**

The first of these names is the same for all members of the font family you create. The second name is a unique characteristic of your font which usually consists of the family name, the font style and weight. For example, you can use the following full name: *Thames* (family name) *Condensed* (style) *Light* (weight).

⌚ **FontName**

This name of your font will be used by the ATM Control Panel and your printer. This name is used in the PostScript 'findfont' operator.



Menu Name

Windows menu name (the name of the font you'll see in Windows applications).



FontMatrix

This matrix affects the reproduction of your font. If we represent it as follows,

$$\begin{array}{|c|c|c|} \hline a & b & D_x \\ \hline c & d & D_y \\ \hline \end{array}$$

the equations that connect the (x, y) coordinates of a point in your character and the (x', y') coordinates of the resulting point (the point actually printed or displayed) are:

$$x' = a \cdot x + c \cdot y + D_x$$

$$y' = b \cdot x + d \cdot y + D_y$$

Therefore, "a" and "d" are the scaling factors applied to your font. As we mentioned in the "Units of Measurement" chapter of "Creating Fonts with Precision," a typical Type 1 font uses the 1000 to 1 scaling ratio; hence, the default "a" and "d" factors are both equal to 0.001. Adobe highly recommends using exactly these values.

The "b" and "c" parameters specify the amount of vertical rotation and horizontal skew, respectively. Consider the following example:

FontLab

FontLab

$$b = 0.000230$$

$$c = 0.000230$$

The "Dx" and "Dy" parameters set the additional offset of characters and are typically zero.

The "c" factor lets you quickly create oblique fonts. However, to develop a really good italic font, you need to totally reshape your characters.

Usually, you can enter FontMatrix values in a more natural form than difficult numbers. If the **Translate** box is marked, you can enter two scaling parameters and the degree of skewing.

① **Weight**

The drop-down list lets you choose from a collection of standard weight names. However, you can specify any other name as well.

① **ItalicAngle**

Angle (in degrees counter-clockwise from the vertical) of the dominant vertical strokes of the font. Many programs determine font styles (normal or italic) using this value. Enter it carefully. For example, ItalicAngle = -12 means italic or oblique font.

① **FixedPitch**

Check this box if your font is monospaced (e.g. Courier or LetterGothic).

① **Blues**

Displays the dialog box which lets you specify the dimensions of blue zones (global hints which the PostScript interpreter uses to control the overshoot suppression) and associated parameters.


① **Stems**

Activates the dialog box which allows you to set the standard width of symbol stems used in your font.

① **UniqueID**

This number helps the PostScript interpreter uniquely identify your font and perform the caching of produced bitmaps between printing jobs. If your font has the same **UniqueID** as a font from some other vendor and both fonts are loaded into a printer, a conflict will occur. You have three choices:

- ① If you're going to use the font on a limited number of printers, specify a random **UniqueID** from 4,000,000 to 4,999,999. The caching feature will be enabled for your font, so the printing will be performed faster.
- ② If you send your font to a service bureau or any other non-local user, set this value to 0. The caching between jobs will be disabled, so the risk of conflicts will be eliminated.
- ③ If you intend to distribute your font widely, contact Adobe Systems Inc. to obtain a specially registered **UniqueID**.

The button marked with a green **emerald**  lets you access the following menu commands:

- | | |
|----------------|---|
| Restore | Allows you to undo the changes you've made. |
| Clear | Removes all existing values from the array. |
| Advisor | Tries to generate all values automatically. |

Typeface options



PCL ID

Contains the fields used (e.g. by **Adobe Type Foundry**) for generating raster (**HP PCL**) font header.



VP ID

Contains the fields used by **Ventura Publisher** for font mapping.



PANOSE

Describes the visual characteristics of the typeface. Windows v3.1 uses Family, Serif and Proportion in the font mapper to determine family type. Applications use PANOSE numbers for precision mapping of missing fonts.

These parameters are used only in TrueType (and VFA, of course) fonts.



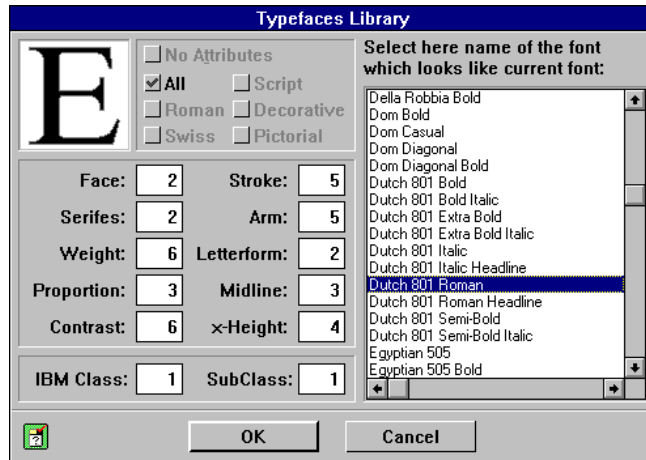
IBMClass and SubClass

These entries are used in TrueType fonts for fonts classification and substitution.



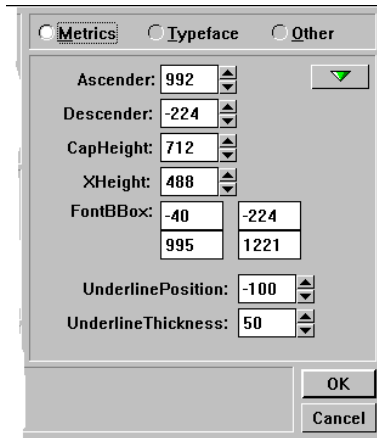
Library...

You can press this button and open a Typeface Library dialog box, where you can select a font sample that looks close to your font.



Metrics Options

There are three special buttons as you may have seen in the right upper corner of the **Font Info** dialog box. If you switch buttons, the right part of the dialog box is changed. Let's describe the two other possible parts of the dialog box. If **Metrics** is on, you get the following:



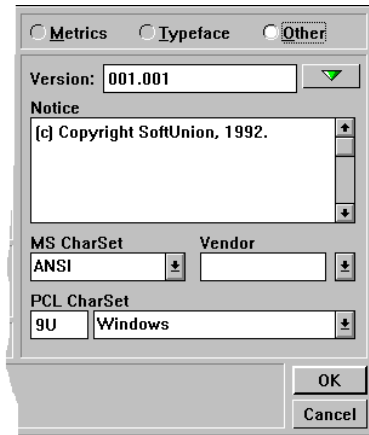
- ⌚ **Ascender**
The top of lower case 'd', measured in symbol coordinate system.
- ⌚ **Descender**
The bottom of lower case 'p', measured in symbol coordinate system.
- ⌚ **CapHeight**
The top of capital 'H', measured in symbol coordinate system.
- ⌚ **Ascender**
The top of lower case 'x', measured in symbol coordinate system.
- ⌚ **FontBBox**
The minimal rectangle that is guaranteed to contain each symbol of this font.

□ Underline Position and Underline Thickness

If an application that uses your font wants to underline the text typed with it, the PostScript interpreter needs to know where to position the underline and which thickness to apply. These entries let you specify the desired offset from the baseline and the line width in symbol space units. You can see the underlined sample of your font in **Preview** panel if you check the **Show Underline** option in the lower part of the panel.

Other Options

Now let's describe briefly the last variant of the right part of the **Font Info** dialog box.



Version

Contains the font version identification string.

Notice

Contains the font name trademark or copyright notice.

MS CharSet lets you choose between:

- ANSI** the standard Windows encoding
- Symbol** used for fonts which contain special decorative symbols
- OEM** for **DOS**-based applications
- Bitstream** this type is equivalent to **ANSI**, but Windows does not recode these fonts when outputting them to a **PostScript** printer
- ShiftJIS** the Japanese encoding standard

Vendor

Contains the four symbol identifier for the vendor of the typeface.




PCL CharSet

Contains the fields used (e.g. by **Adobe Type Foundry**) for generating raster (**HP PCL**) font header.

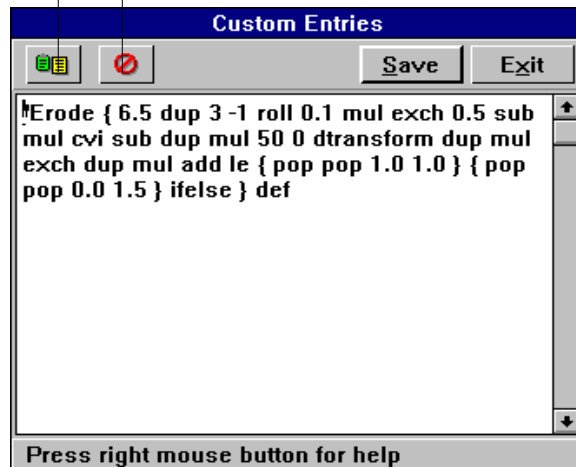
Custom Entries

If an imported **PostScript** font has undocumented entries, you can examine them after clicking on **Custom**.

The dialog box lets you modify these entries, insert them from the Clipboard (by clicking on the left icon), or delete them by selecting the  icon. When you're done, click on **Save** or **Exit**. We recommend preserving all custom entries found in a Type 1 program. For example, Adobe fonts usually contain an undocumented **/Erode** procedure, which can highly improve rendering quality. If you click on **Save**, FLW will include all found undocumented procedures into the exported font, so you will avoid possible information loss. Entering **Custom** lets you see this box:

Clipboard paste button

Clear button

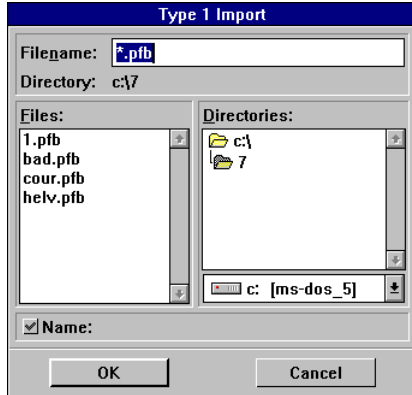


The dialog box which appears after clicking on **Blues** was described in the chapter **Font Level Hints**. When you click on **Stems**, you enter the dialog box described in the chapter **Standard Stem Widths and ForceBold**. For more information or review, (we'll not pester you with those dialog boxes anymore), refer to those chapters.

Importing PostScript Fonts

Beginner Mode

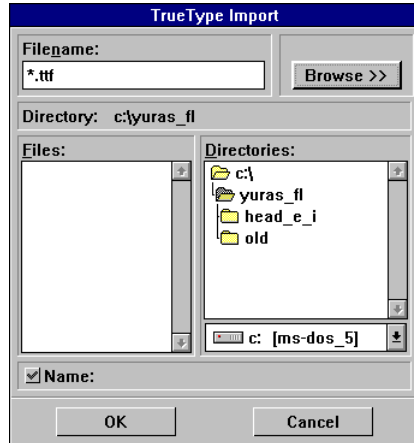
To import a PostScript Type 1 font (with the .PFB extension), select **Import...** from the **File** menu. If the **Beginner** mode is on, the following dialog box appears:



It closely resembles the **Open File** dialog, but there is one special field in it: **Name**. If you click on a filename in the **Files** field, the **FullName** of the selected font will appear in this field. For more information on **FullName**, see "FontInfo" in the previous chapter.

Advanced Mode

To import a PostScript Type 1 font (with the .PFB extension), select **Import...** from the **File** menu. The following dialog box appears:



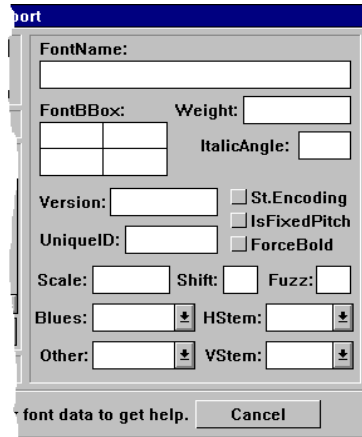
It closely resembles the **Open File** dialog, but there are two special fields in it: **Name** and the field with the **Browse** and **Options** buttons.

⌚ **Name**

If you click on a filename in the **Files** field, the **FullName** of the selected font will appear in this field. For more information on **FullName**, see "FontInfo" in the previous chapter.

🕒 **Browse>>**

This button lets you examine the header of the selected font without importing it (please note that you cannot edit the header before the font is imported). If you click on this button, the whole dialog box will be expanded and you'll see the right part of it as follows:



Let's briefly describe the fields of this box. Get quick help by clicking the secondary button on the desired item.

FontName contains the name presented to the PostScript 'findfont' operator. **Weight** means the weight of the font and indicates the visual weight (degree of blackness or thickness) of the symbols in the font. **ItalicAngle** means the angle (in degrees counter-clockwise from the vertical) of the dominant vertical strokes of the font. **FontBox** contains the rectangle (lower left and upper right corners) which encompasses the symbol's imageable area. **Version** means the font version identification string. **UniqueID** contains the PostScript font ID or the number unique to each Type 1 font - *strongly recommended!* If **St.Encoding** is **ON**, the encoding for this font is Adobe Standard Encoding.

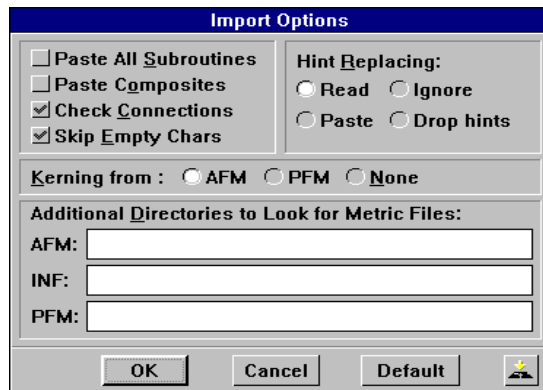
If **IsFixedPitch** is **ON**, this font is monospaced (e.g. Courier or LetterGothic). If **ForcedBold** is switched **ON**, the font will force a bold appearance at small sizes. **Scale** contains the point size at which to deactivate overshoot suppression. **Shift** contains the overshoot enforcement. If the **Flex** feature is used, then this value is equal to the maximum Flex feature height plus 1. The value in **Fuzz** extends the range of alignment (Blue) zones. **Blues** contains the value of the vertical alignment zones. This is global hint data. **Other** contains the value of the additional bottom alignment zones. This is global hint data. **HStem** contains the value of the array of common horizontal stem widths. This is global hint data. **VStem** contains the value of the array of common vertical stem widths. This is also global hint data.

For details on **FONT Dictionary**, **Blues**, **Stems** and **FontInfo**, refer to the "Inside the Font Header" chapter of this section.

⌚ Options...

This button lets you choose the settings which affect the way your font will be imported. Please read the following explanations carefully to achieve the desired results.

The **Import Options** dialog box looks like this:



□ Paste All Subroutines

This option is equivalent to the **Paste All** command applied to all characters of your font. When this mode is on, FLW will replace all subroutines with ordinary segments.

NOTE: If you have any problems with an imported font, try to import it with **Paste All Subroutines** on. Then, define the necessary subroutines manually.

- **Paste Composites**

The Adobe Type 1 format provides a means for creating a standard accent as a separate symbol and then obtain accented symbols by simply adding the accent to non-accented symbols. This is a sort of subroutine that reduces storage space. However, this method may cause inaccurate positioning of symbol features, so recent Adobe fonts do not employ it. If the **Paste Composites** option is on, FontLab will replace the **seac** command with an ordinary symbol contour and will automatically generate the necessary hint replacement commands for this symbol.

- **Check Connections**

An equivalent of the **Check Connections** command from the **Transform/More...** menu. For details, refer to the "Automatic Modification of Contours" chapter of "Transformations."

- If **Skip Empty Chars** is on, FLW imports all characters that have no contour (except the character "space") as empty symbols. (If you bring an empty symbol into the **Edit** window, two crossed lines will appear in it.)

- **Hint Replacing**

FLW offers you the following four choices:

Read	The hint replacement information will be imported from the font
Paste	FLW will import all hints, including overlapping ones, without changing the hints within your characters
Ignore	Only the hints for first portions of your characters (if they are divided into several portions) will be imported
Drop Hints	All hints will be ignored

For more details on the hints replacement mechanism, see the "More About Ensuring Proper Reproduction" chapter of "Advanced Features."

- **Kerning From**

This field specifies the source of kerning information for your font. You can tell FLW to import kerning pairs from the associated .PFM or .AFM file, or ignore the kerning information.

- **Additional Directories to Look for Metrics Files**

If you don't place font metric and font files in the same directory, you may specify **Additional Directories to Look for Metric Files** and FontLab will search metrics files there.



To use the new settings during the next FLW sessions, click on this icon. The changes you've made will be included into the configuration file.

Exporting PostScript Fonts

Beginner Mode

To export a PostScript Type 1 font (with the .PFB extension), select **Export...** from the **File** menu. If the **Beginner** mode is on, the following dialog box appears:

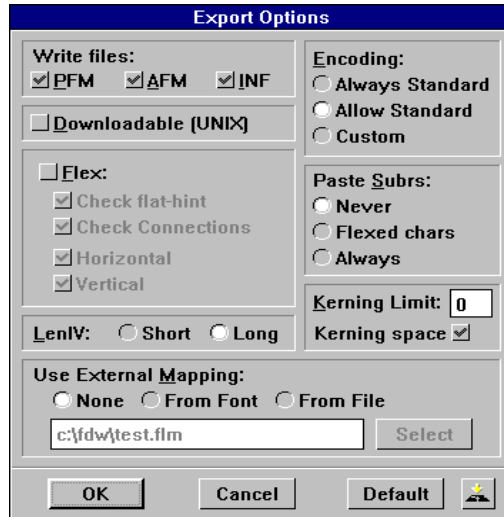


It closely resembles the **Open File** dialog box, but there is one special field in it: **Name**. If you click on a filename in the **Files** field, the **FullName** of the selected font will appear in this field. For more information on **FullName**, see "FontInfo" in the previous chapter.

Advanced Mode

Once a font is created, you need to convert it to the PostScript format to use it on various PostScript devices. For this purpose, choose the **Export...** command from the **File** menu.

The **Export** dialog box is directly analogous to the **Import** dialog box except the **Options...** button that lets you control some key aspects of the export procedure. If you click on this button, the following dialog box will appear:



▫ Write Files

This field lets you select additional files to be exported. For details, refer to the "Inside the Font Header" chapter of this section.

- **Allow StandardEncoding**

Adobe has defined a standard encoding table (in their terminology, the *StandardEncoding vector*) which may be used as a source of information about symbol names and codes. If the encoding table of your font coincides with this vector, you can reduce the size of your font file by replacing the encoding information with a reference to the StandardEncoding vector. If you select the **AllowStandard** option, FLW will perform this replacement if your font conforms to the Adobe standard encoding table. If **Always Standard** options is selected, the font will be exported with standard encoding regardless of any possible symbol names' conflicts. **Custom** encoding will always write encoding data to your font file. Note that composite (**seac**) symbols will be pasted for fonts with encoding other than Standard.

- **Flex**

These options place constraints on the characters which may be imported with the Flex mechanism enabled. Only those characters that meet certain requirements (besides other requirements checked by default) will be rendered using Flex. For more information on the Flex feature, see the "More About Ensuring Proper Reproduction" chapter of "Advanced Features."

- **Check Alignment**

To get better results, the joining point must "sit" on the flat edge of an alignment zone (not on the line of overshoots). If this option is enabled, FLW will check the positioning of joining points.

- **Check Connections**

If you enable this option, FLW will check whether the joining points and the control vectors associated with them are collinear.

- **Horizontal and Vertical**

You can select the orientation of shallow curves for which the Flex mechanism will be applied.

- **LenIV**

The **LenIV** entry deals with the encryption of Type 1 fonts. You can specify whether you want to use 4 bytes per character outline string (**long LenIV**) or only 1 (**short LenIV**). The latter value is used to save storage space; however, **long LenIV** ensures compatibility with some PostScript interpreters. We recommend using the default setting **long**.

- **Downloadable (UNIX)**

This option lets you generate PostScript fonts for UNIX-based operating systems (SUN OS, NeXTSTEP OS). These fonts may also be directly loaded into a PostScript printer.

- **Paste Subrs**

If you want to replace all subroutines in the exported font with the ordinary outline segments, you should set this option to **Always**. If you prefer **Never**, FontLab will export subroutines. The **Flexed chars** option means that FontLab will paste subroutines only if the **Flex** feature is used in the exported font, because flex cannot be placed inside a subroutine.

- **Kerning Limit, Kerning Space**

Kerning Limit sets the minimum amount of kerning for which the kerning record will be exported. If **Kerning Space** is on, FontLab will export kern pairs that include space symbol.

- **Use External Mapping**

External Map allows you to reencode your font in the phase of exporting, without changing the font itself. Because the *Standard Encoding* is a special case of mapping, you can use **External Map** only if you set font encoding to **Custom**. If you set **External Map** to **From File**, you'll have to press **Select** in this box and be asked to enter the name of your mapping file from a standard file dialog box that appears after clicking on **Select**.

- **Default**

By pressing on this button, all the export options are set to the default values. *Be careful* before doing this!



As in other dialog boxes, this icon lets you save the current preferences to disk.

Importing TrueType Fonts

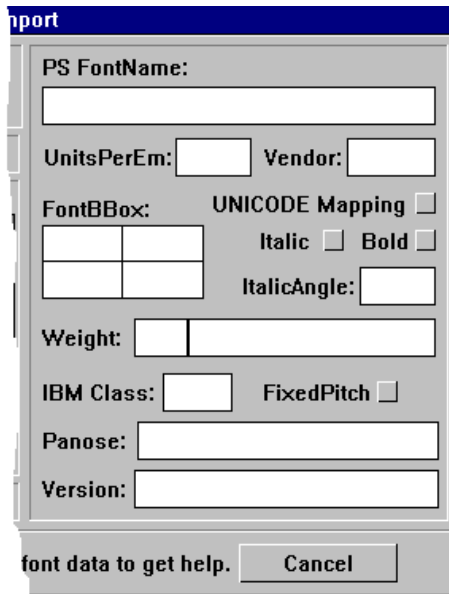
Beginner Mode

To import a TrueType font (with the .TTF extension), select **Import...** from the **File** menu. If the **Beginner** mode is on, the input dialog box appears. It is always that box in PostScript import. Refer to the chapter **Importing PostScript Fonts, Beginner Mode**.

Advanced Mode

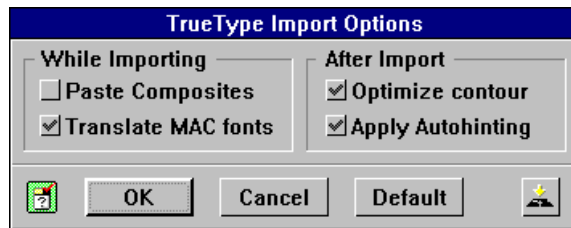
To import a TrueType font (with the .TTF extension), select **Import...** from the **File** menu. The following dialog box appears:

It is quite similar to the one in PostScript import, so we won't describe it fully again. Nevertheless, if you press **Browse**, you'll see a completely different right part. It will look like this:



Let's describe the fields of this box. You can get quick help by clicking the secondary button on the desired item. **PS FontName** contains the PostScript name of this font. **UnitsPerEm** is the font coordinate space. The valid range is from 64 to 16384. The **Vendor** box contains the four symbol identifier for the vendor of the typeface. **FontBBox** contains the rectangle (lower left and upper right corners) that encompasses the symbol imageable area. If **Unicode Mapping** is on, the symbols in this font have Unicode indexes. If **Italic** or **Bold** are on, the font has these characteristics. **Weight** means the weight of the font. This indicates visual weight (degree of blackness or thickness) of the symbols in the font. The right part of the box **Weight** (separated from the left one with a vertical line) contains the font style name. **IBMClass** classifies a font design as to its appearance. **FixedPitch** means that your font is monospaced (e.g. Courier or LetterGothic). **Panose** describes the visual characteristics of the typeface. Windows v3.1 uses Family, Serif and Proportion in the font mapper to determine family type. **Version** means the font version identification string.

To select importing options, press the Options... button and you'll see the following dialog box:



- **Paste Composites**

The equivalent of the Type 1 import **Paste Composites** command.

- **Translate MAC fonts**

Switch on this option if you want to reencode MAC True Type fonts to be compatible with MS Windows encoding.

- **Optimize Contour**

After exporting, FLW can apply its optimization procedure (see the "FontAudit chapter of "Advanced Topics") for all characters of the font. In most cases, this operation improves the quality of the font, but slightly modifies the symbols' shapes.

- **Apply Autohinting**

If you switch this option ON after importing, FLW will automatically generate stem hints and font level hints for this font.

Exporting TrueType Fonts

Beginner Mode

To export a TrueType font (with the .TTF extension), select **Export...** from the **File** menu. If the **Beginner** mode is on, the input dialog box appears. It is always this box in PostScript export. Refer to the chapter **Exporting PostScript Fonts, Beginner Mode**.

Advanced Mode

Once a font is created, you need to convert it to the True Type format to use it on various TrueType devices. For this purpose, choose the **Export...** command from the **File** menu.

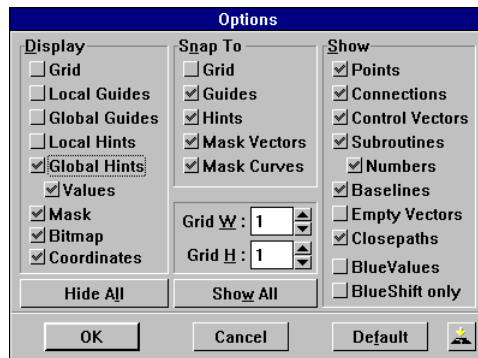
The **Export** dialog box is completely analogous to the **Import** dialog box.

Customizing FLW

The commands that let you tailor FLW to your specific needs reside in the **Options** menu. We have already discussed most of these commands, so we will often refer to previous sections of this manual.

Setting Edit Window Preferences

To change the way your character is presented in the **Edit** window, set the grid frequency, and enable or disable snapping to marking objects, select the **Edit Window...** command or press **F9**. The following dialog box appears:








⌚ Display

This field lists all available marking objects, so you can specify which of them you want to display. We will make three comments:

- 1 FLW can show hints with figures indicating their positions in the character space. To turn on/off these figures, click on **Values**.
- 2 The guidepoints will be displayed if the **Local Guides** item is checked.
- 3 If the **Coordinates** box is checked, the status line will indicate the current coordinates of a point while it is being moved.

You can also use the **Options** box to hide or show objects. The icons corresponding to this field occupy the first row of **Options**. Here is a list showing the relationship between the icons and items contained in the **Display** field:






Icon	Item
	Grid
	Local Guides, Global Guides, Guidepoints
	Local Hints, Global Hints, Values
	Mask
	Bitmap

🕒 **Snap To**

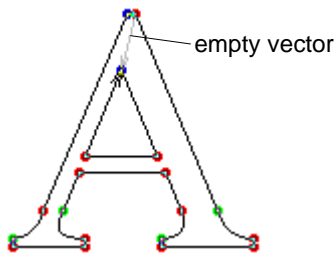
This section of the **Options** dialog lets you control the snapping feature. For more details, refer to the "Creating Fonts with Precision" section.

🕒 Show

Use this field to specify which elements of your characters you want to display.

Item in <i>Show</i> field	Corresponding objects	Icon
Points	Nodes and control points of Bézier curves	
Connections	Flags indicating types of connections	
Control Vectors	Control vectors of Bézier curves	
Subroutines	If this item is on, subroutines are "grayed."	
Numbers	If subroutines are displayed, this item controls whether their numbers and startpoints are indicated.	
Baselines	Hides or displays gray dotted lines, which correspond to baselines and margins.	
Empty Vectors	Invisible vectors that connect contours (see note below)	
Closepaths	Closing vectors	
BlueValues	All alignment zones as defined	
BlueShift Only	Only BlueShift-portions of alignment zones	

NOTE: If your character consists of several contours (for example, the capital "A" is composed of two contours), invisible vectors always connect the endpoint of the first contour with the startpoint of the second contour, and so on. These vectors are necessary for the PostScript interpreter to draw the outline. For example, consider the capital "A":



You can change the order of contours in your character with the **Set Startpoint** command, which is discussed in the "Drawing Outlines" section under "Transforming Outlines."

🕒 **Grid W and Grid H**

These options control the grid interval along the horizontal and vertical axis, respectively. For details, see the "Using the Grid" chapter of "Creating Fonts with Precision."

🕒 **Hide All and Show All, Default,** 

The first two buttons let you switch off or on all options available in this dialog box.

If you wish to restore the default values, click on the **Default** button.

The **Save to Disk** icon allows you to include the current settings into the configuration file, so the next time you start FLW, you will work with the new settings enabled.

Locking Delays


FLW lets you protect important objects from accidental modification by means of *locking delays*. This means that you have to hold down the mouse button and keep the cursor pointing to the object for the duration of the corresponding delay. Please note that you do not need to hold down a key (if one is required) along with the mouse button.

The **Locking Delays...** command is described in more detail in the "Creating Fonts with Precision" section under "Locking Delays."

Adjusting the Display Resolution

The **Display...** command lets you tune the horizontal and vertical resolutions so that the proportions of characters on your screen will match those of printed characters.

To achieve this effect, measure the rectangle situated in the right part of the **Display Options** dialog box. If its sides are not one inch long, adjust it by changing the **H res.** and **V res.** values.

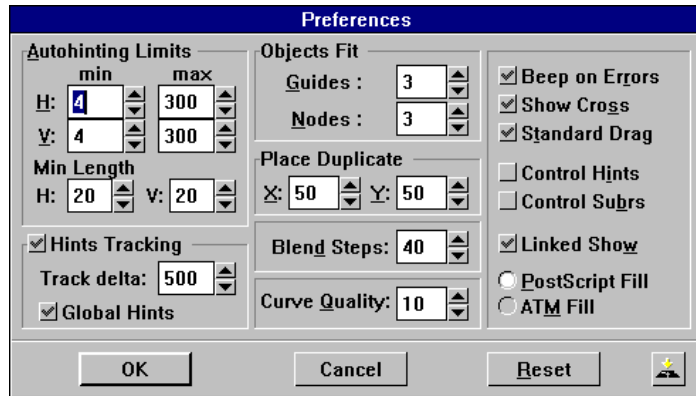
To save the adjusted values to disk, click on the  icon.

Setting Table Options

The **Table Font...** command allows you to change the size of the **Table** window and the font which will be used in it. For more details, see the "Table window" chapter of "Screen Layout."

Setting Other Preferences

The **Preferences...** command (shortcut: **F10**) gives you access to various options that affect the way FLW operates. The **Preferences** dialog box looks like this:



ⓘ Autohinting Limits

These options affect the way that autohinting is performed. For more information on this feature, refer to the "Using Hints" chapter of the "Creating Fonts with Precision" section.

ⓘ Objects Fit

The **Guides** value measured in pixels controls how close a point must be to a guiding object to be forced to stay on it. Since the value is in pixels, a point may be snapped onto a neighboring guide in the normal view yet stay off the guide if you've zoomed into higher magnification. The maximum possible value is 15 pixels.

The second option sets the similar sensitivity for nodes. It defines a circle within which you must place the cursor to extend a contour with the **Contour** tool. As in the previous case, the maximum possible radius of this circle is 15 pixels.

ⓘ Place Duplicate

These values set the horizontal and vertical offset of the copy of the original object produced by the **Duplicate** command. Positive values correspond to shifting the duplicate up and/or to the right of the original.

⌚ **Blend Steps**

This option controls how smoothly the Blend feature will transform one object into the other. In fact, you will be able to choose from as many intermediate shapes as specified in this entry.

The Blend feature is discussed in the "Transformations" section under "Floating Design."

⌚ **Curve Quality**

The higher the value you specify in this entry, the smoother Bézier curves will appear on your screen. You can enter any integer value between 1 and 10, and the default smoothness factor is 4. If your computer is slow, a low value may speed up the redraw process.

Please note that this value affects only the way your curves appear on the screen. The real quality of your font depends on the positioning of nodes and control vectors of Bézier curves.

⌚ **Beep on Errors**

If this switch is on, incorrect actions will cause a sound signal.

⌚ **Show Cross**

By default, a pair of intersecting lines appears when you move a point or an object with the **Edit** or **Move** tool. If you do not want these lines to be displayed, click on this option.

⌚ **3D Dialogs**

This option lets you "flatten" the dialog boxes of FLW. To see how it works, click on it and quit the dialog by clicking on **OK**. Then, reenter the **Preferences** dialog box and decide whether you like its appearance more now.

⌚ **Standard Drag**

This button allows you to switch between the standard drag-and-drop editing mode and the click-move-click FLW mode. For more details, see the "Modifying Outlines Using the Edit Tool" chapter of "Drawing Outlines."

🕒 **Control Hints and Control Subrs**

If your font contains a global object, namely a global hint or a subroutine, that appears nowhere except the memory (this situation may occur when you paste global objects), you can reduce the storage space required for your font by clicking on these buttons. Please note that these options affect only the process of export.

🕒 **Linked Show**

If this mode is on, the **Show** window will immediately reflect all the changes you make in the **Edit** window. Normally, **Show** is redrawn only after the current operation is finished.


🕒 **PostScript Fill and ATM Fill**

This field lets you choose the type of fill which will be used to preview your characters. We recommend using the standard option, **PostScript Fill**, since it is used by all PostScript devices.

For more details on fill types, refer to the "Character Outlines" chapter of "Drawing Outlines."

🕒 **Reset** and 

If you wish to discard recent changes and return to the original settings, click on **Reset**.

The  icon lets you save the current preferences to disk to use during the next FLW session.

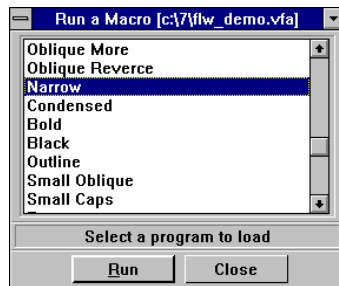
Using the Macro Language

The macro language lets you perform all the same tasks as the tools and menu commands of FLW. The macro language is useful when you want to speed up the execution of a series of commands, or draw a shape that is almost impossible to reproduce using traditional tools (e.g., a star having more than five points, a sinusoid, or a spiral). Those who are comfortable with the PostScript language may find it useful to modify the text descriptions of characters.

The best way to learn the macro language is to examine the examples provided in this section and to experiment with them. Modify our programs to achieve a slightly different effect; try to draw Bézier curves and vectors with the macro language commands; and soon you'll be comfortable writing macro programs.

You can use FontLab macro language in 3 ways:

- 1 When you stylize your fonts with the **Stylize** commands from the **File** menu, you really use macro programs in which transformations are defined.
- 2 You can run the registered macro programs with one-click from the **Macro** panel:



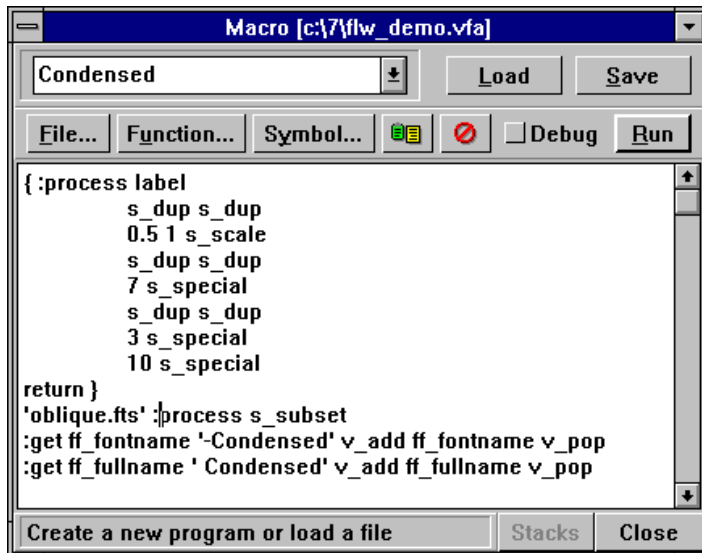
- 3 You can write, debug and run macro programs with the large macro panel, which appears on the **Edit Macro Program...** command from **Expert** menu.

Introduction

Getting Started

Before we proceed with explaining the structures and functions of macro language, let's discuss how to enter and execute a program so you can experiment with the examples discussed in this section.

The program editor is accessed via the **Edit Macro Program...** command from the **Expert** menu. When you click on this command, the following dialog box appears:



You enter new programs and modify existing ones in the large working field of this dialog box. If a portion of your program is not visible in the window, use the scroll bar to bring it into view.

To load an existing macro program file (try loading the sample .CNV file that came with FLW), click on the **File** button, then open the file in the standard fashion. The macro program contained in the file will appear in the **Edit** window. Now you can execute it by clicking on **Run**. To see the results, click on **Exit** and view the results in the **Edit** window.

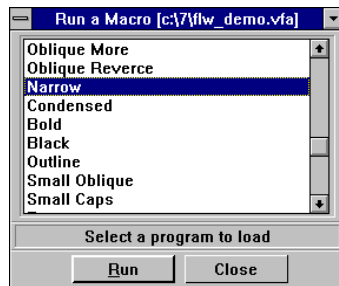
Using the Macro Language

The macro language lets you perform all the same tasks as the tools and menu commands of FLW. The macro language is useful when you want to speed up the execution of a series of commands, or draw a shape that is almost impossible to reproduce using traditional tools (e.g., a star having more than five points, a sinusoid, or a spiral). Those who are comfortable with the PostScript language may find it useful to modify the text descriptions of characters.

The best way to learn the macro language is to examine the examples provided in this section and to experiment with them. Modify our programs to achieve a slightly different effect; try to draw Bézier curves and vectors with the macro language commands; and soon you'll be comfortable writing macro programs.

You can use FontLab macro language in 3 ways:

- 1 When you stylize your fonts with the **Stylize** commands from the **File** menu, you really use macro programs in which transformations are defined.
- 2 You can run the registered macro programs with one-click from the **Macro** panel:



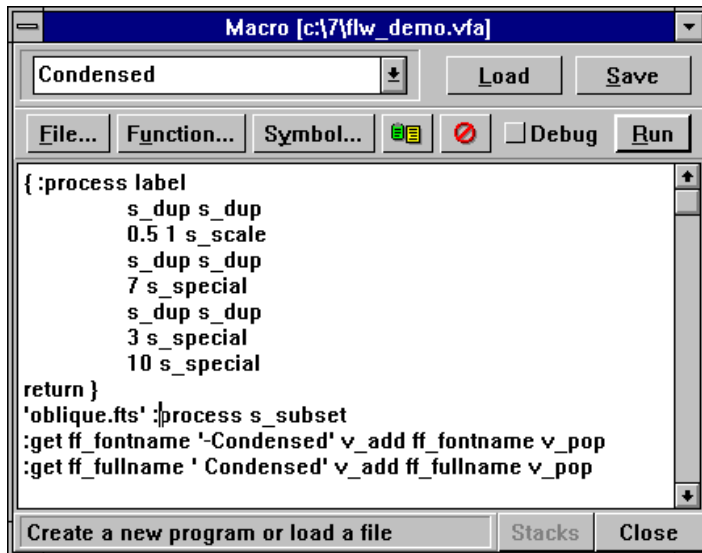
- 3 You can write, debug and run macro programs with the large macro panel, which appears on the **Edit Macro Program...** command from **Expert** menu.

Introduction

Getting Started


Before we proceed with explaining the structures and functions of macro language, let's discuss how to enter and execute a program so you can experiment with the examples discussed in this section.

The program editor is accessed via the **Edit Macro Program...** command from the **Expert** menu. When you click on this command, the following dialog box appears:




You enter new programs and modify existing ones in the large working field of this dialog box. If a portion of your program is not visible in the window, use the scroll bar to bring it into view.

To load an existing macro program file (try loading the sample .CNV file that came with FLW), click on the **File** button, then open the file in the standard fashion. The macro program contained in the file will appear in the **Edit** window. Now you can execute it by clicking on **Run**. To see the results, click on **Exit** and view the results in the **Edit** window.

To enter a new program, click on the  icon to clear the editing window, then type the text of your program. The **Function** button lets you choose from a list of available commands, and the **Symbol** button helps to include a character name using one of three available naming conventions. For details, see the following chapters of this section.


To write a new program to disk, click on the **Save** button.





The image shows a dialog box titled "Macro Registration". It has a blue header bar with the title in white. Below the header, there are two text input fields. The first field is labeled "Registration Name" and contains the text "My First Macro". The second field is labeled "Associated File" and contains the text "mymacro.cnv". At the bottom of the dialog box, there are two buttons: "OK" and "Cancel".

In the upper string of the dialog box, specify the name under which you want to register your macro in FLW. Enter any name you want, like "My first macro." The second string must contain a filename, so it must conform to the DOS naming conventions. After you click on **OK**, the macro program will be saved to disk and its name will appear in the drop-down list. The same procedure is used to register an existing macro program. Registration is necessary if you intend to use several macro programs during the current working session or in future.

To run a registered macro program, choose it from the drop-down list and click on **Load**. Then, click on the **Run** button.

The  icon lets you paste the Clipboard contents into the editing window. It is primarily used when you want to edit the PostScript description of your character. To do so, follow these steps:

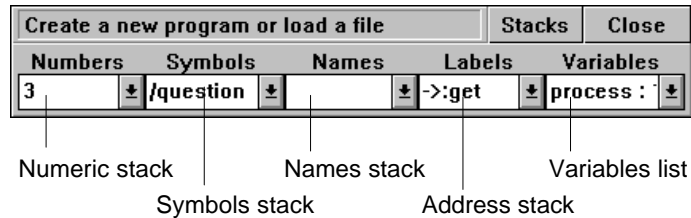
- 1 Bring the character into the **Edit** window.
- 2 Open the **Monitor** window, click on the **PostScript Description** () button and choose **Contour** from the drop-down list. The **Monitor** window is discussed in the "Screen Layout" section.
- 3 Click on **Copy**. Highlight the series of commands that describes your character and click on **OK** to place this description into the Clipboard.
- 4 Enter the macro program editor by selecting the **Macro** command. Then, click on  to paste the PostScript description of your character and edit it as a typical macro program.
- 5 Click on the **Run** button to bring the changes into effect.

Debugging Programs

When you have problems with your macro programs, you may use the debugging facilities of the macro editor. In the debug mode, execute your program in a step-by-step mode and inspect all stacks and variable values.

To switch to the debug mode, mark the **Debug** check box by clicking the primary button on it.

After that, in the lower part of the **Macro** panel, you'll see the stacks and variables combo boxes:




To start execution of the program, press the **Run** button.

To open all stacks lists and keep them open while you execute the program, press the **Stacks** button.

To execute one command of the program, click on the **Step** button, which replaces the **Run** button when you execute the program.

To animate execution, press the secondary button while your mouse cursor is on the dialog box.

To stop execution, press the  button, which replaces the **Close** button in the lower right part of the panel.

Main Structures

② Alphabet

The alphabet used in the macro language consists of letters, numbers and underscores. The slash '/', opening bracket '(', dot '.', quote “, colon ':' and numbersign '#' have a special meaning if they occupy the first position in a name. Except for this special case, a name may contain any valid characters.

All records of the language are separated by spaces, carriage returns or tabulations. You can type the command names in any case you want; however, the character names are case-sensitive.

② Stack

The key structure of the macro language is the stack, a data array organized according to the LIFO (Last In First Out) principle. Thus, the values are sequentially placed on the stack's top, and may be received from the stack in the reverse order. For example, if you've put the values onto your stack in the following order: 10, 20, 30, you will receive 30 back first.

The macro language uses the following four independent stacks:

- The *numeric stack* is used for all operations that involve numeric data. This is the main stack used by the macro language, so the majority of stack operations are defined for it. To place a real number on this stack, include it in the text of your program. For example, the program "10 40 66.4" places 10, 40 and 66.4 onto the stack.
- The *character stack* contains individual characters. You can specify a symbol by its name, code, or the ASCII representation.

Examples:

a) Name

 /F defines the character "F" (code: 70)

b) Code

 .71 defines the character "G" (code: 71)

c) ASCII representation

 (H) defines the character "H" (code: 72)

2.5

d) Unicode representation

 #0041 defines the character "A" (code: 65)

- The *string stack* is used to hold up to 10 text strings of 40 characters, each of which define names of symbols, labels and procedures. Every string must be started with a colon. For example, ":lineto" declares the name "lineto" to be placed onto the string stack. Please note that there is no space between the colon and the name.

2.5

Another method to represent strings is single quotes. This is the only way to define strings containing spaces.

Example:

'Kerning here is :'

- The *address stack* is an internal stack used for calling procedures.

② **Variables**

Variables are the second key structure used for storing individual numeric values (you can define up to 50 variables). A variable is defined by the `def` command. The following program assigns 100 to the new variable "X":

```
100 :X def
```

where "100" places 100 onto the numeric stack, ":X" places the name "X" onto the string stack, and "def" defines the variable "X" equal to 100.

② **Labels**

The labels are used to transfer control to different parts of your program. They are defined with the "label" command, which takes a name from the string stack and treats it as a label.

Example:

```
:lineto label      defines the label "lineto"
```

② Commands

Commands perform all operations in a program. The macro language has about 120 standard commands, which may be classified as follows: stack commands, arithmetic commands, transformation commands, path construction commands, control commands and redefinition commands. Each command gets and returns data from/to the stack.

② Procedures

You can define your own commands in the body of a program. To make a distinction between user-defined and standard commands, these new commands will be referred to as procedures. Commands and procedures act similarly; however, if a command and a procedure have the same name, the procedure will be called, so you have to precede the command with the (@) sign to tell FLW that you wish to call it instead of the procedure.

For example, if you define the procedure "lineto," the name of which coincides with the standard command name, "lineto" will call this procedure, and "@lineto" will perform the command.

First Programs

① Arithmetic Division

Program:

```
100 3 div ns
```

where "100" and "3" place the corresponding numbers onto the stack, "div" divides 100 by 3, and "ns" displays the top stack value in the Monitor window.

Result:

Monitor shows "33.333333."

② Copying Symbols

Program #1:

```
/A /Z /a /z s_copy
```

Here the series "/A /Z /a /z" places the characters "A," "Z," "a" and "z" onto the character stack, and "s_copy" replaces the characters from "a" to "z" with the ("A", "Z") range.

Program #2:

```
/A /Z 20 s_rotate
```

This program rotates all symbols from "A" to "Z" by 20 degrees. The following commands achieve the same result:

```
/A 20 /Z s_rotate
15 20 /D /A /Z s_rotate
.65 (Z) 20 s_rotate
30 10 sub (A) .90 s_rotate
```

3 Renaming a Character**Program:**

```
.65 :NewA setname
```

This program assigns the name "NewA" to the character with decimal code 65 (the letter "A").

4 Using Variables**Program:**

```
20 :R def
/A /Z R s_rotate
```

This macro achieves the same result as "/A /Z 20 s_rotate."

5 Using Procedures**Program:**

```
{ :s_slant12 label          ①
12 s_slant                  ②
return }                    ③
/A /Z s_slant12            ④
```

This example shows how to define the procedure "s_slant12," which skews all symbols in a given range by 12 degrees.

Comments:

- ① starts the procedure definition and assigns a name to it;
- ② the procedure's body—places "12" onto the stack and calls the "s_slant" command;
- ③ returns control to the main program and finishes the definition;
- ④ a sample call to "s_slant12."







Language Reference

Conventions

Throughout the following chapters of this section, the strings that explain the syntax of commands follow these conventions:

- ❶ Contents of a stack before the execution of a command are shown BEFORE the name of this command.
- ❷ The resulting state of the stack is shown AFTER the name of the command.
- ❸ Types of values stored on the stack are indicated using the standard naming conventions:

<i>symbol</i>	symbols on symbols stack
<i>name</i>	names on names stack or strings
<i>num, value, Dx, X, scale</i> etc.	numbers and values on the numeric stack

- ❹ Names of described commands are in bold type: **message**
- ❺ Stacks components are in italic: *symbol₁, scale_x*
- ❻ All commands are described in alphabetical order.
- ❼ Description of each command includes:
 - **name of the command**
 - Commands group:
 -  stack commands
 -  drawing commands
 -  font and symbols transforming commands
 -  program organization commands
 -  dialog and information commands
 -  arithmetic commands
 - Short description of the command
 - *Formal stack description of the **command***
 - *Sample program which includes the **described command***
 - Notes

abs



Converts the top element of the numerical stack into a positive number.

num **abs** *-num*

-10 **abs** ns
MONITOR: 10

add



Adds the two top elements of the numerical stack and leaves the result on the stack.

num₁ num₂ **add** *num₁+num₂*

2 3 **add** ns
MONITOR: 5

addbuffer



Adds the contents of the buffer to the current symbol.

scale_x scale_y rotate slant offs_x offs_y **addbuffer** -

startbuffer 0 0 30 circle endbuffer
/A s_append
2 1 30 0 300 300 **addbuffer**
endchar

allfont



Pushes the first and the last existing symbols in the font onto the symbols stack.

allfont *first last*

allfont 12 s_slant

atan



Leaves on stack the value of Arctan of the top element of the stack.

num **atan** *angle*

0.5 **atan** ns
MONITOR: 45

circle



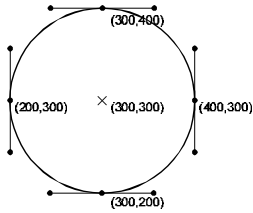
Adds a circle to the current symbol or the buffer.

$pos_x pos_y r$ **circle** -

/A s_append

300 300 100 **circle**

endchar



Negative values of the radius correspond to clockwise drawing and positive ones correspond to counter-clockwise drawing.

clear



Clears the numeric stack fully.

-- $any_1 \dots any_n$ **clear** --

1 2 3 **clear**

NUMERIC STACK: EMPTY

closepath



Adds a 'closepath' command to the current contour.

- **closepath** -

copy



Copies the given number of stack elements starting from its top.

$any_1 \dots any_n n$ **copy** $any_1 \dots any_n any_1 \dots any_n$

100 200 2 **copy** expandpoint

lineto

cos



Leaves on the stack the value of Cos of the top element of the stack.

angle **cos** *Cos(angle)*

60 **cos** ns
MONITOR: 0.5

count



Returns the number of elements on stack.

-- *any₁ ... any_n* **count** -- *any₁ ... any_n* *n*

1 2 3 4 7 **count**
NUMERIC STACK: 1 2 3 4 7 5

currentsymbol



Puts on the symbol stack the symbol from the Edit window and, if this misses, then the current symbol.

- *currentsymbol sym*

currentsymbol s_dup 20 s_slant

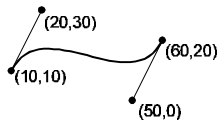
curveto



Adds a curve to the current contour.

x₁ y₁ x₂ y₂ x₃ y₃ **curveto** -

10 10 moveto
20 30 50 0 60 20 **curveto**



def



Defines a new variable or assigns a new value.

num **def** -

100 :stepsnum **def**
VARIABLE: STEPSNUM=100

div



Returns the result of division of the two top elements of the numerical stack.

*num*₁ *num*₂ **div** *num*₁÷*num*₂

30 6 **div** ns
MONITOR: 5

dup



Duplicates the top element of the numeric stack.

num **dup** *num* *num*

7 **dup**
NUMERIC STACK: 7 7

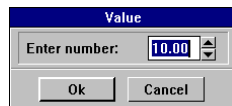
edit1



Opens a dialog for entering one number.

default *min* *max* *caption* *prompt* **edit1** -

10 0 50 'Value' 'Enter number:' **edit1**



NUMERIC STACK: NUMBER FROM 0 TO 50 (10 - DEFAULT)

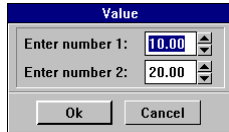
edit2



Opens a dialog for entering two numbers.

default₁ default₂ min max caption prompt₁ prompt₂ edit2 -

10 20 0 50 'Value' 'Enter number 1:' 'Enter number 2:' **edit2**



NUMERIC STACK: NUMBER FROM 0 TO 50 (10 - DEFAULT),
NUMBER FROM 0 TO 50 (20 - DEFAULT)

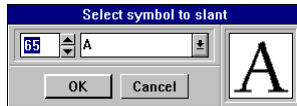
edits1



Opens a dialog for selecting one symbol.

default caption edits1 -

.65 'Select symbol to slant' **edits1** s_dup 12 s_slant



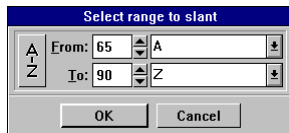
edits2



Opens a dialog for selecting two symbols.

default₁ default₂ caption edits2 -

.65 .90 'Select range to slant' **edits2** 12 s_slant



endbuffer



Completes the definition of a contour buffer.

- **endbuffer** -

```
startbuffer  
0 0 40 circle  
endbuffer
```

endchar



Completes the definition of a symbol.

- **endchar** -

```
/A s_append  
300 300 100 circle  
endchar
```

exch



Swaps the two top elements of the numeric stack.

```
 $num_1$   $num_2$  exch  $num_2$   $num_1$   
10 20 exch  
NUMERIC STACK: 20 10
```

exist



Checks the presence of a symbol and leaves the result on the numeric stack:
0 - no symbol, 1 - there is a symbol.

symbol **exist** *num*

```
/A s_dup exist :process ifcall s_pop
```

Calls *process* subroutine if symbol /A exist

existcontour



Pushes 1 on the numeric stack if a symbol exists and has contour. In other cases, pushes 0.

symbol **existcontour** *num*

```
/A s_dup existcontour :process ifcall s_pop
```

ff_ascender



Lets you redefine the field 'Ascender'.

value **ff_ascender** *oldvalue*

```
0 ff_ascender dup ff_ascender ns
```

Keeps previous value of ascender and pushes it on numeric stack

ff_capheight



Lets you redefine the field 'Cap Height'.

value **ff_capheight** *oldvalue*

```
/H getbb ff_capheight clear
```

Defines cap height as a top node of H symbol

ff_descender



Lets you redefine the field 'Descender'.

value **ff_descender** *oldvalue*

```
/g getbb pop pop ff_descender pop pop
```

Automatically determines descender parameter

ff_familyname



Leaves 'Family Name' on the names stack and lets you redefine it.

newname **ff_familyname** *oldname*

```
:get ff_familyname oldname
```

```
:get ff_familyname '-Super' v_add ff_familyname v_pop
```

Adds '-Super' to the family name of font

ff_fontname



Leaves 'Font Name' on the names stack and lets you redefine it.

newname **ff_fontname** *oldname*

:get ff_fontname oldname

:get ff_fontname :get ff_weight v_add ff_fontname v_pop

Creates a font name based on previous font name and weight of the font

ff_fullname



Leaves 'Full Name' on the names stack and lets you redefine it.

newname **ff_fullname** *oldname*

:get ff_fullname oldname

:get ff_fullname ' ' v_add :get ff_version ff_fullname v_pop

Creates a full font name based on previous full name and version of the font

ff_italicangle



Leaves the value of 'Italic Angle' on the numeric stack and lets you redefine it.

value **ff_italicangle** *oldvalue*

-12 ff_italicangle pop

Sets italic angle parameter for oblique fonts

ff_menuname



Leaves 'Menu Name' on the names stack and lets you redefine it.

newname **ff_menuname** *oldname*

:get ff_menuname oldname

:get ff_menuname '-FLW' v_add ff_menuname v_pop

ff_notice



Leaves 'Notice' on the names stack and lets you redefine it.

```
newtext ff_notice oldtext  
:get ff_notice oldtext
```

'Created by me, 1994. All rights reserved.' **ff_notice** v_pop

ff_scalex



Leaves the value of 'Scale X' of 'FontMatrix' on the numeric stack and lets you redefine it.

```
value ff_scalex oldvalue  
0 ff_scalex dup ff_scalex ns
```

ff_scaley



Leaves the value of 'Scale Y' of 'FontMatrix' on the numeric stack and lets you redefine it.

```
value ff_scaley oldvalue  
0 ff_scaley dup ff_scaley ns
```

ff_slant



Leaves the value of 'Slant' of 'FontMatrix' on the numeric stack and lets you redefine it.

```
value ff_slant oldvalue  
0 ff_slant dup ff_slant ns
```

ff_underlinepos



Lets you redefine the field 'Underline Position'.

```
value ff_underlinepos oldvalue  
0 ff_underlinepos dup ff_underlinepos ns
```

ff_underlinethick



Lets you redefine the field 'Underline Thickness'.

value **ff_underlinethick** *oldvalue*

0 **ff_underlinethick** dup **ff_underlinethick** ns

ff_vendor



Leaves the field 'Vendor' on the names stack and lets you redefine it.

newname **ff_vendor** *oldname*

:get **ff_vendor** *oldname*

'ABCD' **ff_vendor** v_pop

This command understands only the first four characters of the vendor's code name.

ff_version



Leaves the field 'Version' on the names stack and lets you redefine it.

newname **ff_version** *oldname*

:get **ff_version** *oldname*

'001.027' **ff_version** v_pop

ff_weight



Leaves the field 'Weight' on the names stack and lets you redefine it.

newname **ff_weight** *oldname*

:get **ff_weight** *oldname*

'ExtraBlack' **ff_weight** v_pop

ff_xheight



Lets you redefine the field 'x Descender'.

value **ff_descender** *oldvalue*

0 **ff_descender** dup **ff_descender** ns

getbb



Leaves the minimum containing rectangle for the given symbol on the numeric stack.

symbol **getbb** *min_x min_y max_x max_y*

```
/A s_dup getbb
:hiY def :hiX def
:loY def :loX def
```

getkerning



Leaves the value of layer kerning for the given pair on the numeric stack.

symbol **getkerning** *num*

```
/A /N 'Get Kerning' edits2
getkerning
str 'Kerning is :' v_exch v_add write
```

getname



Leaves the PostScript name of the given symbol on the names stack.

symbol **getname** *name*

```
/A 'Get Name' edits1
getname
'PostScript name :' v_exch v_add write
```

goto



Unconditionally goes to the given label.

label **goto** -

```
:infinity label
'For a long time...' write
infinity goto
```

grad



Transforms the top number on the numeric stack from radians to degrees.

num **grad** *num*

3.1415926 **grad** round ns

MONITOR: 180

hintrepace



Adds a hint replacing command to the current symbol.

- **hintrepace** -

10 30 vstem 400 30 vstem

...

hintrepace

10 40 vstem 380 50 vstem

hlineto



Adds a horizontal line to the current symbol. The horizontal offset is set respectively to the previous point.

Dx **hlineto** -

100 100 moveto

100 **hlineto**

CURRENT POINT: (200 100)

hmoveto



Adds a relative horizontal hidden vector to the current symbol.

Dx **hmoveto** -

100 100 moveto

100 vlineto

250 **hmoveto**

CURRENT POINT: (350 200)

hsbw



Begins the definition of a new symbol and defines its metrics.

symbol left width **hsbw** -

```
/A 10 500 hsbw  
200 150 100 circle  
endchar
```

hstem



Adds a horizontal hint to the current symbol.

V DV hstem -

```
0 40 hstem
```

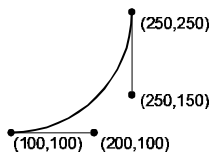
hvcurveto



Adds a relative horizontal-vertical curve to the current symbol.

Dx₁ Dx₂ Dy₂ Dy₃ **hvcurveto** -

```
100 100 moveto  
100 50 50 100 hvcurveto
```



ifcall



Calls the given subroutine if there is a positive number on the numeric stack.

label num ifcall -

```
/A s_dup exist :process ifcall s_pop
```

Calls process subroutine only if symbol A exists

ifgoto



Jumps to the given label if there is a positive number on the numeric stack.

label num ifgoto -

```
40  
:loop label  
'Sometimes...' write  
1 sub dup loop ifgoto pop
```

inc



Increases the given variable.

varname num inc -

```
:x 100 def  
:x 10 inc x ns  
MONITOR: 110
```

index



Selects the given element of the numeric stack and puts it on top.

any_n ... any₀ n index any_n ... any₀ any_n

```
10 20 30 40 2 index  
NUMERIC STACK: 10 20 30 40 20
```

label



Defines a label or a subroutine.

name **label** -

:loop **label**

lineto



Adds an absolute line to the current symbol or buffer.

x y **lineto** -

100 100 moveto

200 200 **lineto**

CURRENT POINT: (200 200)

max



Selects the greater among both top elements of the numeric stack and puts it on the stack.

num₁ num₂ **max** *Max(num₁,num₂)*

{ :minrange label

MinLimit **max**

return }

Provides that the number on stack is not less than 'MinLimit'.

message



Shows a dialog box with the message taken from the names stack.

string **message** -

'Hello, World !' **message**



min



Selects the smaller among both top elements of the numerical stack and puts it on the stack.

num₁ num₂ min Min(num₁,num₂)

```
{ :maxrange label  
MaxLimit min  
return }
```

Provides that the number on the stack is not greater 'MaxLimit'.

moveto



Adds an absolute hidden vector to the current symbol. Defines a new contour.

*x y **moveto** -*
*100 100 **moveto***

mul



Multiplies both top elements of the numerical stack and leaves the result on the stack.

*num₁ num₂ **mul** num₁ x num₂*

```
3 5 mul ns  
MONITOR: 15
```

m_left



Defines the left field of a symbol.

*num symbol **m_left** -*
*/A 30 **m_left***

m_right



Defines the right field of a symbol.

num symbol m_right -

```
/A 30 m_right
```

m_width



Defines the width of a symbol.

num symbol m_width -

```
:fixwidth 700 def
{ :Fixedwidth label
  s_dup s_dup
  getbb
  pop :hix def
  pop :lox def
  fixwidth hix lox sub sub 2 div
  sdup m_left
  fixwidth m_width
  return }
/A /Z :fixedwidth s_range
```

Defines the width equal to 'fixwidth' for all symbols.

If you want to do all the character in the font, then change the last line to:

```
.0 .640 :fixedwidth s_range
```

neg



Inverts the sign of the top element of the numerical stack.

num **neg** *-num*

XC YC R **neg** circle

ns



Takes away the top element of the numerical stack and writes it into the Monitor window.

num **ns** -

10 ns

MONITOR: 15

pop



Takes away the top element of the numeric stack.

num **pop** -

30 40 20 **pop pop**

NUMERIC STACK: 30

pushlast



Puts the coordinates of the last point of the given symbol on the numeric stack.

symbol **pushlast** *x y mode*

/A s_dup **pushlast** s_append pop 100 hlineto

pushmetrics



Puts the left field and the width of the given symbol on the numeric stack.

symbol **pushmetrics** *left width*

/B **pushmetrics**

.195 hsbw

pushsymbol



Puts the code of the top element of the symbols stack on the numeric stack.

symbol **pushsymbol** *x y mod*

/B **pushsymbol** str write

MONITOR: 66

rad



Converts the top number on the numeric stack from degrees to radians.

num **rad** *num*

180 **rad** ns

MONITOR: 3.142

rand



Puts a random number within the range 0..1 on the numeric stack.

- **rand** *num*

100 **rand** mul

A random number from the range 0..100.

rendersymbol



Converts the given symbol running the predefined subroutines 'lineto', 'curveto' and 'moveto'.

symbol **rendersymbol** -

```
{ :lineto label
  2 copy str ' ' v_exch v_add
  str v_exch v_add
  'lineto : ' v_exch v_add write
  @lineto
return }
/A rendersymbol
```

return



Provides the return from a subroutine.

- **return** -

```
{ :subr label
...
return }
```

rlineto



Adds a relative line to the current symbol.

Dx Dy **rlineto** -

100 100 moveto

300 120 **rlineto**

CURRENT POINT: (400 220)

rmoveto



Adds a relative hidden line to the current symbol.

$Dx Dy$ **rmoveto** -

100 100 moveto

300 100 lineto

50 50 rmoveto

CURRENT POINT: (450 250)

roll



Performs a cyclic rotation of the elements of the numeric stack.

$num_{n-1} \dots num_0 n j$ **roll** $num_{(j-1) \bmod n} \dots num_0 num_{n-1} \dots num_{j \bmod n}$

1 2 3 3 -1roll

NUMERIC STACK: 2 3 1

1 2 3 3 1roll

NUMERIC STACK: 3 1 2

1 2 3 3 0 roll

NUMERIC STACK: 1 2 3

round



Rounds the top element of the numeric stack to the nearest integer value.

num **round** num

180 rad **round** ns

MONITOR: 3

rrcurveto

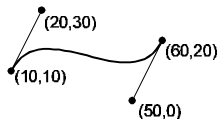


Adds a relative curve to the current symbol.

$Dx_1 Dy_1 Dx_2 Dy_2 Dy_3 Dx_3$ **rrcurveto** -

10 10 moveto

10 20 30 -30 10 20 **rrcurveto**



setcurrentpoint



Changes the position of the current point for relative operations of adding elements to the current symbol.

x y **setcurrentpoint** -

10 10 moveto

100 hlineto

CURRENT POINT: (110 100)

333 500 setcurrentpoint

CURRENT POINT: (333 500)

This command doesn't immediately influence the contour, it just changes the position of the current point.

setkerning



Sets the kerning value for the given pair of symbols.

symbol₁ symbol₂ num **setkerning** -

/A /V 100 **setkerning**

setname



Sets the PostScript name for the given symbol.

symbol name **setname** -

123 dup 'FLW' str v_add symbol s_dup **setname**

getname write

MONITOR: FLW123

setstartx



Sets the horizontal position of the global hints and guides startpoint.

symbol pos_x **setstartx** -

/A 0 **setstartx**

setstarty



Sets the vertical position of the global hints and guides startpoint.

symbol pos_y **setstarty** -

/A 700 **setstarty**

showstacks



Writes the contents of all stacks into the Monitor window.

- **showstacks** -

10 'Hello' /A

200 :X def

showstacks

MONITOR:

Values stack 10.000
Symbols stack 65
Names stack Hello
Address stack Variables
X 200

sin



Returns the value of Sin of the top element of the numeric stack.

angle **sin** *Sin(angle)*

30 **sin** ns

MONITOR: 0.5

sqrt



Returns the value of Sqrt of the top element of the numeric stack.

num **sqrt** ↔ *num*

25 **sqrt** ns

MONITOR: 5

status



Writes the message taken from the names stack in the status line of the Macro or Stylize dialog box

string **status** -

'Processing...' **status**

startbuffer



Begins the definition of a contour buffer.

- **startbuffer** -

startbuffer

0 0 200 circle

endbuffer

str

Converts the top element of the numeric stack to a string and puts it on the names stack.

num **str** *name*

123 **str** write

MONITOR: 123

sub



Subtracts both top elements of the numeric stack and leaves the result on stack.

*num*₁ *num*₂ **sub** *num*₁-*num*₂

5 3 **sub** ns

MONITOR: 2

symbol



Converts the top element of the numeric stack to a symbol's code and puts it on the symbol stack.

num **symbol** *symbol*

65 **symbol** getname write

MONITOR: A

s_append



Defines a symbol for adding elements of a contour.

symbol **s_append**

/A s_append

300 300 100 **circle**

endchar

s_bcopy



Copies the given symbol to the copy buffer.

symbol **s_bcopy**

/A **s_bcopy** /a /z s_bpaste

s_bpaste



Substitutes the given range of symbols for the contents of the symbols' copy buffer.

symbol₁ *symbol₂* **s_bpaste** -

/A s_bcopy /a /z **s_bpaste**

s_compo



Molds a composite symbol of both given symbols.

symbol₁ *symbol₂* *offs_x* *offs_y* **s_compo** -

/A /tilde 30 700 **s_compo**

s_copy



Copies the range of symbols to the place of another range.

fromsymbol₁ fromsymbol₂ tosymbol₁ tosymbol₂ s_copy -

/A /Z /a /z s_copy

s_deltamove



Shifts each symbol within the given range.

symbol₁ symbol₂ d_x d_y s_deltamove -

/A /Z 0 -100 s_deltamove

s_dup



Duplicates the top element of the symbol stack.

symbol s_dup symbol symbol

currentsymbol s_dup 12 s_slant

s_duplicate



Duplicates each symbol within the given range.

symbol₁ symbol₂ d_x d_y s_duplicate -

/A /Z 0 -100 s_duplicate

s_effect



Applies the given effect to a range of symbols.

symbol₁ symbol₂ rand force effect# s_effect -

0 /A /z 80 3 **s_effect**

- 1 Barrel lines
- 2 Boldness
- 3 Broken lines
- 4 Grid
- 5 Perspective
- 6 Random curved lines
- 7 Random double lines
- 8 Rendered curves
- 9 Rounded corners
- 10 Waved curves
- 11 Waved lines
- 12 Fantasy
- 13 Z-Down
- 14 Furs

RAND parameter means the state of the **Randomize** option of the **Effects** procedure. If *RAND* = 0 Randomize is **ON**, in other cases - **OFF**

s_erase



Removes the given range of symbols.

symbol₁ symbol₂ s_erase -

.0 .255 **s_erase**

s_exch



Swaps both top elements of the symbol stack.

symbol₁ symbol₂ s_exch symbol₂ symbol₁

/Z /A **s_exch** 12 s_slant

s_hmirror



Applies the symmetrical transformation with respect to the horizontal axis to the given range of symbols.

```
symbol1 symbol2 s_hmirror  
/a /z s_hmirror
```

s_merge



Merges the two given symbols.

```
symbol1 symbol2 offsx offsy s_merge  
currentsymbol s_dup .1 s_dup s_copy  
.1 s_dup 20 20 4 s_special  
.1 s_dup 5 5 5 s_special  
.1 currentsymbol 0 0 s_merge  
.1 .1 s_erase
```

s_move



Moves the given range of symbols to another place.

```
fromsymbol1 fromsymbol2 tosymbol1 tosymbol2 s_move -  
/A /Z /a /z s_move
```

s_paste



Fills the given range of symbols with another symbol.

```
fromsymbol tosymbol1 tosymbol2 s_paste -  
(.) /a /z s_paste
```

s_pop



Removes the top element of the symbol stack.

```
symbol1 symbol2 s_pop symbol1 -  
/A /Z 'Symbols' edits2  
...  
s_pop s_append
```

s_range



Applies the given subroutine to a range of symbols.

symbol₁ symbol₂ name **s_range** -

```
{ :makeoblique label  
12 s_slant  
return }  
/A /Z :makeoblique s_range
```

s_rotate



Rotates each symbol from the given range by the specified angle.

symbol₁ symbol₂ angle **s_rotate** -

```
/A /Z 30 s_rotate
```

s_scale



Scales each symbol from the given range by the specified angle.

symbol₁ symbol₂ scale_x scale_y **s_scale** -

```
/A /Z 0.7 1 s_scale
```

s_slant



Slants each symbol from the given range by the specified angle.

symbol₁ symbol₂ angle **s_slant** -

```
/A /Z 12 s_slant
```

s_special



Applies one of the transformation operations to a range of symbols.

symbol₁ symbol₂ opcode s_special -
symbol₁ symbol₂ width_x width_x opcode s_special -

/A /Z 6 s_special

/A /Z 10 10 4 s_special

- 1 Snap to guides
- 2 Make connections
- 3 Detect extremal nodes
- 4 Change width
- 5 Create outline
- 6 Remove overlap
- 7 Remove all hints
- 8 Remove all guides
- 9 Paste global hints
- 10 Autohinting
- 11 Autoreplacing
- 12 Paste all subroutines
- 13 Invert all contours
- 14 Creates an intersection of the all overlapped contours of the symbol
- 15 Optimizes the symbol
- 16 Decomposes symbol

s_subset



Applies the given subroutine to the set of symbols defined by the file 'Table subset'.

filename procname s_subset -

```
{ :process label
12 s_slant
return }
'oblique.fts' :process s_subset
```

s_swap



Swaps the two given symbols.

symbol₁ symbol₂ s_swap -

/A /B s_swap

s_transform



Transforms symbols from the given range.

symbol₁ symbol₂ scale_x scale_y rotate slant offs_x offs_y s_transform

/A /Z 0.8 1 0 12 0 0 s_transform

s_transformc



Transforms symbols from the given range while setting the center of transformation.

*symbol₁ symbol₂ scale_x scale_y rotate slant
offs_x offs_y center_x center_y s_transform*

/A /Z 0.8 1 0 12 0 0 300 0 s_transformc

s_vmirror



Applies the symmetrical transformation with respect to the vertical axis to the given range of symbols.

symbol₁ symbol₂ s_vmirror

/a /z s_vmirror

vhcurveto

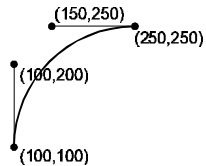


Adds a relative vertical-horizontal curve to the current symbol.

Dy₁ Dx₂ Dy₂ Dx₃ vhcurveto -

100 100 moveto

100 50 50 100 vhcurveto



vlineto



Adds a relative vertical vector to the current symbol.

Dy **vlineto** -

100 100 moveto

100 **vlineto**

CURRENT POINT: (100 200)

vmoveto



Adds a relative vertical hidden vector to the current symbol.

Dy **vmoveto** -

100 100 moveto

100 hlineto

250 **vmoveto**

CURRENT POINT: (200 350)

vstem



Adds a vertical hint to the current symbol.

V DV **vstem** -

200 30 **vstem**

v_add



Merges both top elements of the names stack.

name₁ name₂ **v_add** *name₁+name₂*

'Hello,' 'World !' **v_add** write

MONITOR: Hello, World !

v_dup



Duplicates the top element of the names stack.

```
name1 v_dup name1 name1
```

```
'Bang ! ' v_dup v_add write
```

```
MONITOR: Bang ! Bang !
```

v_exch



Swaps both top elements of the names stack.

```
name1 name2 v_exch name2 name1
```

```
'One ' 'Two ' v_exch v_add write
```

```
MONITOR: Two One
```

v_pop



Takes away the top element of the names stack (strings stack).

```
name1 name2 v_pop name1
```

```
'MyFont' ff_fontname v_pop
```

write



Writes the top string from names stack to the Monitor window.

```
name write -
```

```
'Kerning : ' /A /V getkerning str v_add write
```

```
Writes kerning value in Monitor window
```

{



Begins the definition of a subroutine.

```
{ <text of the subroutine>
```

```
  { :process label  
  <text of the subroutine>  
  return }
```

}



Completes the definition of a subroutine.

<*text of the subroutine*> }

```
{ :process label  
<text of the subroutine>  
return }
```

Examples of Using the Macro Language

Recoding a Font

- ② Moving all lowercase letters on top of capital letters

<code>/a /z /A /Z s_copy</code>	copies the lowercase letters
<code>/a /z s_erase</code>	deletes the original set of lowercase letters

- ② Swapping lowercase and capital letters

<code>26</code>	①
<code>:loop label</code>	②
<code>1 sub</code>	③
<code>dup 65 add dup 32 add</code>	④
<code>symbol symbol s_swap</code>	
<code>dup loop ifgoto</code>	

- ① forms the loop counter
- ② starts the loop
- ③ decrements the counter
- ④ forms two next codes on the numeric stack
moves the codes onto the character stack
closes the loop

Changing Character Names

To rename a symbol, use the following command:

```
.<code> :<newname> setname
```

Example:

```
.65 :NewA setname  
.66 :NewB setname  
.....  
.90 :NewZ setname
```

Creating New Symbols

② Sinusoid

```
/A 20 700 hsbw  
300  
:loop label  
dup  
dup rad cos 200 mul  
lineto  
dup loop ifgoto  
endchar
```

② Polygon

```
:XC 300 def  
:YC 300 def  
:R 100 def  
:V 20 def  
/A 20 700 hsbw  
360 V div :S def  
V  
:loop label  
dup S mul rad dup  
cos R mul XC add exch  
sin R mul YC add lineto  
1 sub  
dup loop ifgoto  
endchar
```

X coordinate of the center
Y coordinate of the center
radius
number of sides
character to be replaced
counts the step to be used in the loop

X coordinate
Y coordinate

Scaling and Slanting

Version #1:

```
/A /Z 0.5 1 s_scale  
/A /Z 12 s_slant
```

Version #2:

```
{ :scaleslant label  
s_dup  
0.5 1 s_scale  
12 s_slant  
return }  
/A /Z :scaleslant s_range
```

Working with Stylizers

With FLW, you get a number of special macro programs, called *stylizers*, which allow you to modify and transform fonts using only one menu command. Using the macro language, you can create new stylizing programs or modify existing ones.

To add a new stylizing program:

- ① Using the **Edit Macro** panel, create the macro program which you want to use as stylizer.
- ① With Windows Notepad, open file **FLW.INI** from the directory where you install FLW.
- ① Find the **[Stylize]** part of this file (**Search / Find / Stylize**)
- ① In the string, beginning from "stylizenum" increase the number at the end of the string by 1. For example, if the string looked like this: **stylizenum=21**, modify it to **stylizenum=22**
- ① Append a string at the end of the stylize section of the initialization file. The string must have the following format:

stylize<number>=<Stylizer Menu Name>;<File with the stylizing program>.


For example, if you want to add a **Cubism** stylizer, and the program is in the file **\flw\cubism.cnv**, you add the following string:

stylize22=Cubism;\flw\cubism.cnv


General Structure of Stylizing Program

All stylizing programs are based on the following structure:

- ① Interface and initialization
- ② Supplemental procedures
- ③ {Main transforming subroutine}
- ④ {Testing subroutine}
Symbol loop
Font Info modification

To enter a new program, click on the  icon to clear the editing window, then type the text of your program. The **Function** button lets you choose from a list of available commands, and the **Symbol** button helps to include a character name using one of three available naming conventions. For details, see the following chapters of this section.


To write a new program to disk, click on the **Save** button.





The image shows a dialog box titled "Macro Registration". It has a blue header bar with the title in white. Below the header, there are two sections. The first section is labeled "Registration Name" and contains a text input field with the text "My First Macro". The second section is labeled "Associated File" and contains a text input field with the text "mymacro.cnv". At the bottom of the dialog box, there are two buttons: "OK" and "Cancel".

In the upper string of the dialog box, specify the name under which you want to register your macro in FLW. Enter any name you want, like "My first macro." The second string must contain a filename, so it must conform to the DOS naming conventions. After you click on **OK**, the macro program will be saved to disk and its name will appear in the drop-down list. The same procedure is used to register an existing macro program. Registration is necessary if you intend to use several macro programs during the current working session or in future.

To run a registered macro program, choose it from the drop-down list and click on **Load**. Then, click on the **Run** button.

The  icon lets you paste the Clipboard contents into the editing window. It is primarily used when you want to edit the PostScript description of your character. To do so, follow these steps:

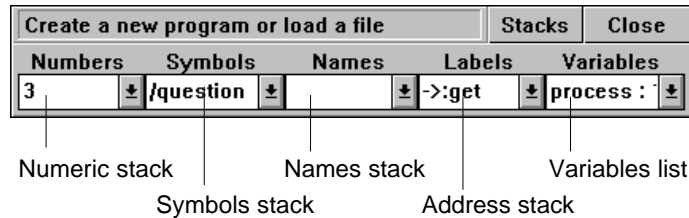
- 1 Bring the character into the **Edit** window.
- 2 Open the **Monitor** window, click on the **PostScript Description** () button and choose **Contour** from the drop-down list. The **Monitor** window is discussed in the "Screen Layout" section.
- 3 Click on **Copy**. Highlight the series of commands that describes your character and click on **OK** to place this description into the Clipboard.
- 4 Enter the macro program editor by selecting the **Macro** command. Then, click on  to paste the PostScript description of your character and edit it as a typical macro program.
- 5 Click on the **Run** button to bring the changes into effect.

Debugging Programs

When you have problems with your macro programs, you may use the debugging facilities of the macro editor. In the debug mode, execute your program in a step-by-step mode and inspect all stacks and variable values.

To switch to the debug mode, mark the **Debug** check box by clicking the primary button on it.

After that, in the lower part of the **Macro** panel, you'll see the stacks and variables combo boxes:




To start execution of the program, press the **Run** button.

To open all stacks lists and keep them open while you execute the program, press the **Stacks** button.

To execute one command of the program, click on the **Step** button, which replaces the **Run** button when you execute the program.

To animate execution, press the secondary button while your mouse cursor is on the dialog box.

To stop execution, press the  button, which replaces the **Close** button in the lower right part of the panel.

Main Structures

⌚ Alphabet

The alphabet used in the macro language consists of letters, numbers and underscores. The slash '/', opening bracket '(', dot '.', quote "", colon ':' and numbersign '#' have a special meaning if they occupy the first position in a name. Except for this special case, a name may contain any valid characters.

All records of the language are separated by spaces, carriage returns or tabulations. You can type the command names in any case you want; however, the character names are case-sensitive.

⌚ Stack

The key structure of the macro language is the stack, a data array organized according to the LIFO (Last In First Out) principle. Thus, the values are sequentially placed on the stack's top, and may be received from the stack in the reverse order. For example, if you've put the values onto your stack in the following order: 10, 20, 30, you will receive 30 back first.

The macro language uses the following four independent stacks:

- The *numeric stack* is used for all operations that involve numeric data. This is the main stack used by the macro language, so the majority of stack operations are defined for it. To place a real number on this stack, include it in the text of your program. For example, the program "10 40 66.4" places 10, 40 and 66.4 onto the stack.
- The *character stack* contains individual characters. You can specify a symbol by its name, code, or the ASCII representation.

Examples:

a) Name

/F defines the character "F" (code: 70)

b) Code

.71 defines the character "G" (code: 71)

c) ASCII representation

(H) defines the character "H" (code: 72)

2.5

d) Unicode representation

#0041 defines the character "A" (code: 65)

- The *string stack* is used to hold up to 10 text strings of 40 characters, each of which define names of symbols, labels and procedures. Every string must be started with a colon. For example, ":lineto" declares the name "lineto" to be placed onto the string stack. Please note that there is no space between the colon and the name.

2.5

Another method to represent strings is single quotes. This is the only way to define strings containing spaces.

Example:

'Kerning here is :'

- The *address stack* is an internal stack used for calling procedures.

🕒 **Variables**

Variables are the second key structure used for storing individual numeric values (you can define up to 50 variables). A variable is defined by the `def` command. The following program assigns 100 to the new variable "X":

```
100 :X def
```

where "100" places 100 onto the numeric stack, ":X" places the name "X" onto the string stack, and "def" defines the variable "X" equal to 100.

🕒 **Labels**

The labels are used to transfer control to different parts of your program. They are defined with the "label" command, which takes a name from the string stack and treats it as a label.

Example:

```
:lineto label      defines the label "lineto"
```

🕒 **Commands**

Commands perform all operations in a program. The macro language has about 120 standard commands, which may be classified as follows: stack commands, arithmetic commands, transformation commands, path construction commands, control commands and redefinition commands. Each command gets and returns data from/to the stack.

🕒 **Procedures**

You can define your own commands in the body of a program. To make a distinction between user-defined and standard commands, these new commands will be referred to as procedures. Commands and procedures act similarly; however, if a command and a procedure have the same name, the procedure will be called, so you have to precede the command with the (@) sign to tell FLW that you wish to call it instead of the procedure.

For example, if you define the procedure "lineto," the name of which coincides with the standard command name, "lineto" will call this procedure, and "@lineto" will perform the command.

First Programs

❶ Arithmetic Division

Program:

```
100 3 div ns
```

where "100" and "3" place the corresponding numbers onto the stack, "div" divides 100 by 3, and "ns" displays the top stack value in the Monitor window.

Result:

Monitor shows "33.333333."

❷ Copying Symbols

Program #1:

```
/A /Z /a /z s_copy
```

Here the series "/A /Z /a /z" places the characters "A," "Z," "a" and "z" onto the character stack, and "s_copy" replaces the characters from "a" to "z" with the ("A", "Z") range.

Program #2:

```
/A /Z 20 s_rotate
```

This program rotates all symbols from "A" to "Z" by 20 degrees. The following commands achieve the same result:

```
/A 20 /Z s_rotate
15 20 /D /A /Z s_rotate
.65 (Z) 20 s_rotate
30 10 sub (A) .90 s_rotate
```

3 Renaming a Character**Program:**

```
.65 :NewA setname
```

This program assigns the name "NewA" to the character with decimal code 65 (the letter "A").

4 Using Variables**Program:**

```
20 :R def
/A /Z R s_rotate
```

This macro achieves the same result as "/A /Z 20 s_rotate."

5 Using Procedures**Program:**

```
{ :s_slant12 label          ①
12 s_slant                 ②
return }                   ③
/A /Z s_slant12           ④
```

This example shows how to define the procedure "s_slant12," which skews all symbols in a given range by 12 degrees.

Comments:

- ① starts the procedure definition and assigns a name to it;
- ② the procedure's body—places "12" onto the stack and calls the "s_slant" command;
- ③ returns control to the main program and finishes the definition;
- ④ a sample call to "s_slant12."

Examples of Using the Macro Language

Recoding a Font

- ① Moving all lowercase letters on top of capital letters

<code>/a /z /A /Z s_copy</code>	copies the lowercase letters
<code>/a /z s_erase</code>	deletes the original set of lowercase letters

- ② Swapping lowercase and capital letters

<code>26</code>	①
<code>:loop label</code>	②
<code>1 sub</code>	③
<code>dup 65 add dup 32 add</code>	④
<code>symbol symbol s_swap</code>	
<code>dup loop ifgoto</code>	

- ① forms the loop counter
- ② starts the loop
- ③ decrements the counter
- ④ forms two next codes on the numeric stack
moves the codes onto the character stack
closes the loop

Changing Character Names

To rename a symbol, use the following command:

```
.<code> :<newname> setname
```

Example:

```
.65 :NewA setname  
.66 :NewB setname  
.....  
.90 :NewZ setname
```

Creating New Symbols

⌚ Sinusoid

```
/A 20 700 hsbw  
300  
:loop label  
dup  
dup rad cos 200 mul  
lineto  
dup loop ifgoto  
endchar
```

⌚ Polygon

```
:XC 300 def  
:YC 300 def  
:R 100 def  
:V 20 def  
/A 20 700 hsbw  
360 V div :S def  
V  
:loop label  
dup S mul rad dup  
cos R mul XC add exch  
sin R mul YC add lineto  
1 sub  
dup loop ifgoto  
endchar
```

X coordinate of the center
Y coordinate of the center
radius
number of sides
character to be replaced
counts the step to be used in the loop

X coordinate
Y coordinate

Scaling and Slanting

Version #1:

```
/A /Z 0.5 1 s_scale  
/A /Z 12 s_slant
```

Version #2:

```
{ :scaleslant label  
s_dup  
0.5 1 s_scale  
12 s_slant  
return }  
/A /Z :scaleslant s_range
```

Working with Stylizers

With FLW, you get a number of special macro programs, called *stylizers*, which allow you to modify and transform fonts using only one menu command. Using the macro language, you can create new stylizing programs or modify existing ones.

To add a new stylizing program:

- ① Using the **Edit Macro** panel, create the macro program which you want to use as stylizer.
- ① With Windows Notepad, open file **FLW.INI** from the directory where you install FLW.
- ① Find the **[Stylize]** part of this file (**Search / Find / Stylize**)
- ① In the string, beginning from "stylizenum" increase the number at the end of the string by 1. For example, if the string looked like this: **stylizenum=21**, modify it to **stylizenum=22**
- ① Append a string at the end of the stylize section of the initialization file. The string must have the following format:

stylize<number>=<Stylizer Menu Name>;<File with the stylizing program>.

For example, if you want to add a **Cubism** stylizer, and the program is in the file **\flw\cubism.cnv**, you add the following string:























stylize22=Cubism;\flw\cubism.cnv

























General Structure of Stylizing Program


















All stylizing programs are based on the following structure:

- ① Interface and initialization
- ② Supplemental procedures
- ③ {Main transforming subroutine}
- ④ {Testing subroutine}
Symbol loop
Font Info modification

Appendix A *Toolbar Buttons*

Icon	Menu Item	Alternative
	Symbol / Add Composite	Ctrl-drag from Table window
	Transform / More / Change Width	Table Toolbar - 
	Transform / Overlap	[Shift]+[F10]; Mouse popup menu in Show window
	Transform / Apply to Range	
	Hints/Guides / Bitmap	
	Transform / Set Blend	Blend button () in the Edit window
	Transform / Break Contour	[B]
	Transform / Change Direction	[I]
	Transform / Combine Contours	[F]
	Symbol / Decompose	Shift+secondary button click on the X mark of the move composite routine
	Expert / Paste Subroutine	[F2]
	Edit / Copy	[Ctrl]+[Ins]
	Hints/Guides / Copy to Mask	[M]
	Symbol / Copy Symbol	[Ctrl]+[C]
	Symbol / New Symbol	[Ctrl]+[N]
	Transform / More / Create Outline	Table Toolbar - 
	Transform / More / Parallel Contours	[Shift]+[F9]
	Transform / More / Nodes at Extremes	[Shift]+[F8]
	Transform / More / Check Connections	[Shift]+[F7]

	Edit / Duplicate	[D]
	File / Font Info	
	Hints/Guides / Clear Mask	
	Symbol / Erase Subset	
	Hints/Guides / Exchange with Mask	[J]
	File / Export	
	Expert / FontAudit	
	Hints/Guides / Autohinting	[F7]
	File / Import	
	Symbol / Get Symbol	[Ctrl]+[Home]; Status line of the Table window
	Transform / H Mirror	
	Transform / V Mirror	
	Options / Edit Window	[F9]
	Options / Preferences	[F10]
	File / Preview	[R]
	Expert / Advanced Hinting	[Ctrl]+[F7]
	File / Open	[Ctrl]+[O]
	Transform / Effects	Table Toolbar - 
	Expert / Set Kerning	
	Expert / Macro	
	Expert / Set Sidebearings	
	Expert / Subroutines	[F4]
	Hints/Guides / Remove All Hints / Both	

	Symbol / Rename Symbol	Right part of Table's toolbar ()
	Edit / Paste	[Shift]+[Ins]
	Symbol / Paste Symbol	[Ctrl]+[V]
	Transform / Rotate	[Shift]+[F2]
	Expert / Edit Macro Program	
	File / Save	[Ctrl]+[S]
	Transform / Scale	[Shift]+[F4]; Table Toolbar - 
	Symbol / Zoom	Mouse Popup in Edit and Show windows and  mark of Edit window
	Expert / Set Tracking	
	Transform / Set Startpoint	[T]
	Transform / Slant	[Shift]+[F3]; Table Toolbar - 
	Transform / More / Snap to Guides	[Shift]+[F5]
	Edit / Undo	[Alt]+[Backspace]

Bibliography

For those interested in an in-depth study of the Adobe Type 1 format and the principles of producing high quality fonts, we recommend the following books:

Adobe Systems Incorporated, *PostScript Language Reference Manual*, Addison-Wesley, 1990. ISBN 0-201-18127-4. The complete guide to the PostScript language.

Adobe Systems Incorporated, *Adobe Type 1 Font Format*, Addison-Wesley, 1990. ISBN 0-201-57044-0. Whereas the previous volume is a general-purpose guide to the language, this book deals with the organization of Type 1 programs.

Adobe Systems Incorporated, *PostScript Language Tutorial and Cookbook*, Addison-Wesley, 1985. ISBN 0-201-10179-3. Illustrates the use of the PostScript language with many examples.

Peter Karow, *Digital Formats for Typefaces*, Himmelheber, Hamburg, 1987. ISBN 3-926515-01-5. Contains many useful tips for creating high quality digital fonts.

The Unicode Consortium, *The Unicode Standard. Volumes 1 and 2*, Addison-Wesley, 1992. ISBN 0-201-60845-6. Official document describes the Unicode Standard in detail.

Technical Support

Technical support for FontLab is available by mail, email or phone. If you have questions or problems, please write to:

FontLab Technical Support
Box 465
Millersville, MD 21108

or send us a message via:

CompuServe - 70220,344

Internet - 70220.344@compuserve.com

or call us at:

(410) 987-5616
between 9 a.m. and 5 p.m., Eastern Time
Monday through Friday



Create new typefaces - Adobe Type 1 and TrueType
Design your own keystroke logos
Add special characters and ligatures to your fonts
Adjust hinting and kerning
Make a typeface of your own handwriting
Construct foreign language characters
Scan in new fonts, logos, symbols

Features:

Automatic kerning, tracking and hinting
No-node freehand drawing
Automatic font scanning
Letterform parts library
Font Auditor
Easy font transformations
Font blending/morphing
Unicode compatibility
... and much, much more



Санкт Петербург, Россия



*distributed by Pyrus North America
Box 465, Millersville, MD 21108 U.S.A.*