XEROX

Xerox Development Environment

Typefounder Reference Manual

610E00270

XEROX

Xerox Development Environment

Typefounder Reference Manual

610E00270

TYPEFOUNDER REFERENCE MANUAL

XEROX

Version 3.6
610E00270
June, 1986

Xerox Corporation
Information Systems Division
2100 Geng Road
Palo Alto, California 94303

Printed in U.S.A.

Table of contents

1. Introduction 1-1

What Typefounder can do 1-3

What you need to run Typefounder 1-5

Installing Typefounder 1-7

How Typefounder is organized 1-9

2. Typefounder tools 2-1

Alias 2-3

Background 2-5

Barcode 2-9

Camera 2-11

Character 2-13

Converter 2-21

Dictionary 2-25

Distributor 2-31

Fitter 2-35

Font 2-37

Font finder 2-41

Help 2-43

Lister 2-45

Little image 2-47

Merger 2-49

Metrics 2-51

Proofer 2-55

Raster image 2-57

Scratchpad 2-59

Segmenter 2-61

Text display 2-63

Transforms 2-65

Typefounder 2-67

Variations 2-69

3. Using Typefounder with Tool Driver 3-1

Appendices:

A.	A few hints on how to do things	A-1
B.	Making fonts by scanning artwork	B-1
C.	Creating a logo	C-1
D.	Customizing with TypefounderUser.cm	D-1
E.	What to do if Typefounder breaks	E-1
F.	Error messages	F-1

1. Introduction

Typefounder is an extensive set of software tools for creating and managing digital fonts and logos. It runs in the Xerox Development Environment (XDE), version 4.0, on Tajo or CoPilot volumes of an 8010 or 6085 workstation.

Typefounder lets you create raster or contour fonts and logos from scanned or synthetic images. An image can then be scaled to arbitrary size and resolution, thickened, thinned, slanted, rotated, letter-spaced, examined, edited and packaged in a variety of formats.

If you're using Typefounder to design a font, be sure to read Appendices A and B before you begin. Also, it's assumed you have experience with XDE and understand the art of type designing.

If you're using Typefounder to create a logo, use the procedures in Appendix C to guide you through the process of making a logo from a ViewPoint Freehand drawing, an XDE brush file, or scanned artwork. It's assumed you've completed the XDE online tutorials to learn the basics of XDE.

Note: Typefounder cannot read or modify Helvetica printer fonts in accordance with licensing agreements for those fonts.

(This page intentionally blank)

What Typefounder can do

The major functions of Typefounder are listed below. Relevant tool names are shown in parentheses.

- Reading and writing of raster, contour, and metrics (widths) font files in all common Xerox font formats at any orientation, with full translation ability. (Font)

- Scan conversion or scaling of contour or raster font masters to produce fonts of any size and resolution, with simultaneous thickening or thinning, and simultaneous production of a template font for use as an editing background. (Converter)

- Display and editing of contour and raster character images, with choice of eight shades of gray, arbitrary magnification, optional display of grid, baseline and sidebearings, and any number of background character images. (Character, Background)

- Automatic letter spacing of raster fonts, including imposition of fixed pitch and incrementing of widths as well as full automatic operation. (Fitter)

- Automatic generation of contour (spline) representations from raster images. (Converter)

- Production of printable interpress proof files for raster fonts. (Proofer)

- Merging of fonts and moving of characters to new character codes. (Merger, Distributor)

- Rotation, mirror imaging, vertical or horizontal thickening or thinning, and slanting of individual character images or entire raster fonts. Automatic production of outline, inline, gray, and shadow fonts. (Variations)

- Complete production of logotypes from scanned images, including use of an editing background. (Raster Image)

- Reading, writing, display and cropping of one-bit-per-pixel raster images in a variety of formats (AIS, compressed AIS, strike fonts, Viewpoint freehand drawing canvases,Doodle brushes, or Doodle bitmaps). (Raster Image)

- Building of font dictionaries and CD files. (Dictionary)

- Building of ViewPoint screen font dictionaries. (Dictionary)

- Segmentation of raster images into individual character images. (Segmenter)

- Production of composite characters by merging raster character images. (Background)

- Display and editing of font parameters. (Font)

- Extraction of widths from fonts, and imposition of widths on fonts. (Font)
- Display of arbitrary text in a desired font, with optional display of baseline and sidebearings, and a movable horizontal rule. (Text Display)
- Listing the characters of a font, displaying metrics of individual characters and of the font as a whole. (Lister, Metrics)
- Easy production of barcode symbols. (Barcode)
- Measurement of distances on displayed character images, using the mouse. (Character)
- Listing and describing all font files on the local volume. (Font Finder)
- Simultaneous display of and operation on any number of fonts and any number of characters.
- Customization of the system for individual users through a TypefounderUser.cm file.
- 'Batch' operation by ToolDriver scripts.
- A camera feature to generate printable (interpress) pictures of tool window contents. (Camera)
- Quick on-line help for tool operation. (Help)
- All the benefits arising from the XDE window package: easy control over window size and position, scrolling, simultaneous operations in different windows, and so on.

What you need to run Typefounder

Typefounder requires a Xerox 8010 or 6085 workstation with at least 768 kilobytes of memory, and the XDE 4.0 Desktop software.

Here is a list of Typefounder related files, available on your release diskettes:

TypefounderTool.bcd	*The program itself.*
TypefounderScreen12.strike	*Suggested system font for the volume on which Typefounder is executed.*
TypefounderTool.symbols	*Symbols file for use by experienced debuggers in reporting errors.*
TypefounderUser.cm	*Auxiliary file which allows you to customize Typefounder window layouts and tool startup sequences.*
TypefounderTutorial.interpress	*Printable version of a tutorial for new users.*
TypefounderTutorial.nsMail	*The tutorial in the style of the XDE tutorials.*
Atol.text	*Text file used in Typefounder scripts.*
Excelsis.cis	*A scanned master alphabet image.*
RGE.ais	*A scanned logo.*
TestModern8x300.ac	*A low resolution font with some defective characters.*
XeroxLogo.sd	*A contour master font.*
Sample.script, SampleLoop.script	*Sample ToolDriver scripts.*
Tool.sws	*A list of tools and their subwindows; used by ToolDriver.*
Xerox.text, Items.list	*ToolDriver example files.*
NextItemTool.bcd	*A tool used with ToolDriver*

See the *XDE Users Guide*, 610E00140, for directions on using FileTool, Print, Floppy, and MailTool to access these files from the release diskettes.

(This page intentionally blank)

Installing Typefounder

Retrieve TypefounderTool.bcd from the release diskettes to a Tajo or CoPilot volume on your workstation. (Sample fonts, logos, and scripts can be retrieved as required.) Tajo volumes are preferred, because they have larger virtual memory available.

Also retrieve TypefounderScreen12.strike to your volume and make it the system font by editing the [volumeName:System] section of the file user.cm to contain the line:

Font:TypefounderScreen12.strike

and booting the volume. This font contains displayable graphics for nearly all 256 character codes. Also, while Typefounder tools adjust their sizes to accomodate to different system fonts, certain compromises are necessary, and tool window layout is optimized for TypefounderScreen12.strike .

If you wish to use the screen layout customization features described in Appendix D (warning: this greatly increases initial loading time of Typefounder), retrieve TypefounderUser.cm and edit it to reflect your needs.

Type "TypefounderTool.bcd" to the Executive to load and start the program.

If you have never used Typefounder before, retrieve TypefounderTutorial.nsMail and read it with MailTool. It leads you on a guided tour to most of the commonly used Typefounder features.

(This page intentionally blank)

How Typefounder is organized

Typefounder is organized as a tree of tools or windows. Each tool represents an object such as a font or a character, or performs an operation on such an object. From each tool, you can use, by means of a menu, subtools which operate in the context of the parent tool. The subtools can similarly spawn additional tools. Subtools are automatically destroyed when their parent is destroyed or deactivated.

The top level tool is named Typefounder. It provides the entry point for the system, as well as posting messages and maintaining a log.

One of Typefounder's subtools, the Font tool, provides a means of accessing a font in the same way that an XDE File Window accesses a document. It provides reading, writing and display functions for the font, and supports a variety of subtools for performing specific operations on the font, such as displaying arbitrary text (Text Display), automatically assigning character widths (Fitter) and listing character metrics (Lister).

A subtool of the Font tool is the Character tool, which allows access to individual characters of its parent font for display and editing.

Another Typefounder subtool is the Raster Image tool, which allows binary raster images (in AIS or several other common formats) to be accessed by Typefounder, supporting generation of font or logo masters through high-resolution scanning. Its Segmenter subtool allows segmentation of the image into individual character images.

The full list of the constituent tools of Typefounder is shown below. Indentation indicates the parent-child relationship of the tools. If in reading the individual tool descriptions you forget the parent of a tool, look it up here.

Typefounder	*entry tool, posts messages, keeps log*
Barcode	*constructs bar codes*
Dictionary	*examines and builds font dictionaries and screen fonts*
Distributor	*moves characters to new character codes*
Font	*gives access to a font, displays parameters*
Alias	*assigns names to characters*
Converter	*scales font to different resolution or size or type*
Character	*displays and edits character images*
Background	*shows other images in background*
Transforms	*rotates, slants, thickens, makes mirror images*
Little Image	*shows character at scale = 1*
Fitter	*performs automatic or manual letter spacing*
Lister	*lists and describes characters in font*
Merger	*merges, moves, or deletes characters*
Metrics	*reports metrics of the current font*
Proofer	*makes printable interpress files of proof text*
Text Display	*displays text in current font*
Variations	*makes design variations (slanting, thickening, etc.)*
Font Finder	*lists font files on volume*
Raster Image	*displays and crops raster images*
Segmenter	*makes characters from portions of a raster image*
Scratchpad	*provides a place to select characters; type file names, text*

(This page intentionally blank)

2. Typefounder tools

In general, there is no limit to the number of instances of each tool. The tools are all ordinary XDE tools and can be moved, grown, scrolled, deactivated and otherwise manipulated in the usual way. Since there is usually one window per instance of a tool, a tool is sometimes referred to as a window.

Each of Typefounder's constituent tools has a Help subtool which describes the tool and its use. Most tools have a Camera subtool for printing the tool's screen image.

Most tools contain a one line message subwindow at the top which is used for prompts and warning messages. If the message subwindow is made larger than one line, messages wrap to the next line automatically, and the last five lines written are viewable by scrolling. These subwindows allow typein. You can use them as scratch space for entering file names or other command arguments.

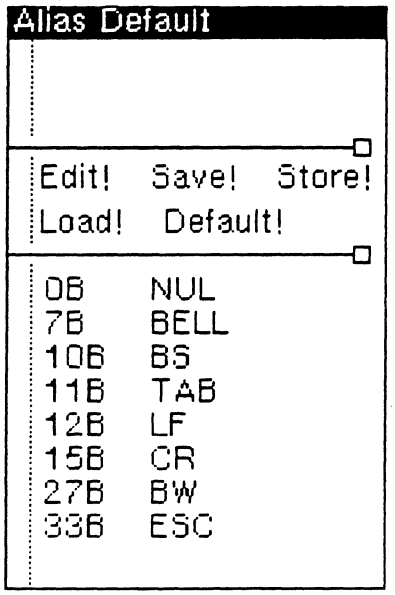
You can change parameters displayed in form subwindows, but some are for information only. These items blink and remain unchanged if you attempt to edit them.

Each of these tools is described in detail on the following pages, in alphabetical order by tool name. Each description includes the purpose of the tool, the commands and parameters that are used to control it, the subtools available through its Typefounder menu, some notes on usage, and a picture of what the tool looks like on the screen.

(This page intentionally blank)

Alias

Alias automatically translates mnemonic character names to character codes.



Commands

Edit! permits editing of the alias list. The format of the list is octal character code/TAB/mnemonic/CR. The alias list is not available for lookup by other tools when you are editing it.

Save! stores the edited list into the file.

Store! stores alias list into the file specified by the current selection.

Load! loads the alias file specified by the current selection. Only as much of the name as necessary to uniquely identify it need be selected.

Default! displays a list of characters and typical aliases.

Notes

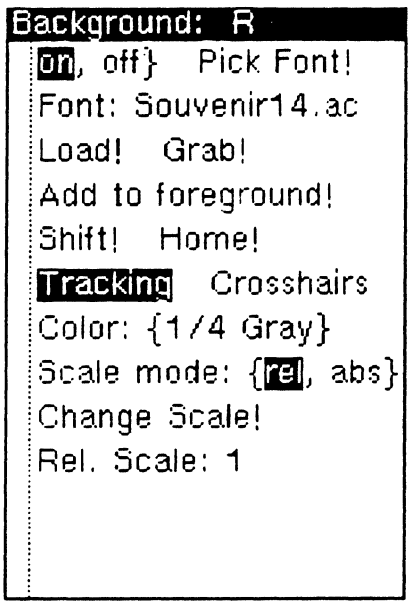
The typical use of Alias is to tag special characters (or those without displayable graphics in the screen font used with Typefounder) with names easier to refer to than their octal character codes.

Whenever this tool is active (and not in Edit mode), the names defined in the alias list may be used in all places in Typefounder at which a character specifier is expected (such as the character range in Variations or Merger or the current selection for commands which load characters, and so on).

(This page intentionally blank)

Background

Background displays secondary raster or contour images in the parent character window. These background images can be moved independently of the foreground, and under the right conditions be added to the foreground character image.



Commands

Pick Font! lets you choose a background font by clicking Point in a font window. The font name displays in the Font: parameter.

Load! displays the character specified by the current selection as a background character. Character symbol and character code are posted in the namestripe as a reminder. If Adjust is used, the current character is loaded.

Grab! displays as background an image from a Character, Raster Image, or Barcode window that you point to. If the window contains a box, only that portion enclosed by the box is grabbed. "Grabbed" appears in the namestripe as the character designator.

Add to foreground! paints the background into the foreground. This command works for raster images only, and both foreground or background must be displayed at the same scale factor. The resulting foreground image contains a pixel wherever there was a pixel in either the foreground or the background image.This can be used to construct a character from pieces of others, or to paint in an initial first character as a first cut for later editing.

Shift! moves the background image the difference in positions of two successive mouse clicks.

Home! moves the background image to home position (so that its origin is at foreground character origin).

Change Scale! changes magnification. Scale factors may range from .1 to 99. X and Y scales may be set independently by turning off the X = Y switch.

Switches

On/Off: background showing or not.

Tracking: when on, the background character tracks the parent character. For example, if an A is loaded into the character window, an A is loaded into the background. Turn this switch off to leave the background unaffected by the Character Load, Next, and Reset commands.

Crosshairs: when on, the left side bearing and width of the background character is marked by gray crosshairs symbols.

Scale Mode: select *abs* to set scale manually, or *rel* to have the background set absolute scale automatically to maintain a constant scale relative to the foregound.

Parameters

Font: shows the name of the file loaded into the font window being used as a source of background images.

Color: chord mouse over this item to select from 8 gray levels for a background raster image. For contour backgrounds, black turns on knots and other colors do not.

Rel. Scale: relative scale factor. This appears when the scale mode is relative. Background images will be displayed at this factor relative to foreground images. Em sizes of foreground and background fonts are taken into account, so at a relative scale of 1.0, full em height characters displayed as background will be the same size as full em height foreground characters. If the requested relative scale cannot be maintained (which happens occasionally with raster backgrounds, because they can be displayed at only integer scale factors), "**" is posted after the scale, the notice *Background off* appears, and the background image is not displayed.

Abs. Scale: absolute scale factor. This appears when the scale mode is absolute. Magnification factor for the background image. Integer values only are allowed for raster characters. Separate X & Y scales may be set if desired.

Notes

Backgrounds are commonly used as editing templates, components of a composite image, or to check inter-character consistency.

Making the background font the same as the foreground can be useful in checking inter-character consistency (such as stroke shapes) by turning Tracking .off and using background to visually overlay the two characters in question. Multiple background windows, with different shades of gray, allow looking at more than two characters at once.

1/4 gray background with 3/4 gray foreground causes overlap to be black.

Bit editing is noticeably slower when a background image is displayed.

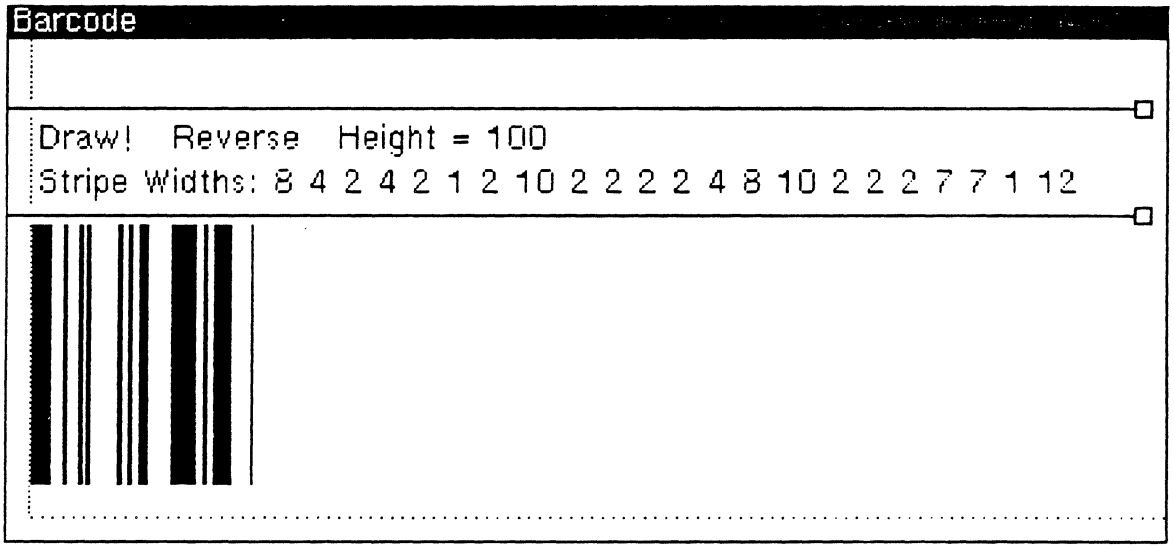
Characters can be constructed from pieces of others (common stroke forms, accents, serifs, small numerals used in fractions, and so on) by Grabbing the components into Background windows, using Shift! to position them properly, and Add To Foregound! to paint them into the character under construction.

If cursor is on a subwindow boundary when selecting a font in Pick Font!, you may get a'wrong kind of window' message. If that happens, try again, clicking mouse when cursor is well within a subwindow.

(This page intentionally blank)

Barcode

Barcode draws barcode symbols from a list of stripe widths.



Command

Draw! paints the image defined by the stripe widths in the window.

Switches

Reverse: video-inverts the image.

Parameters

Height: height in pixels of the image.

Stripe Widths: widths of the stripes in pixels. The first stripe is always black. Enter a sequence of integers separated by spaces.

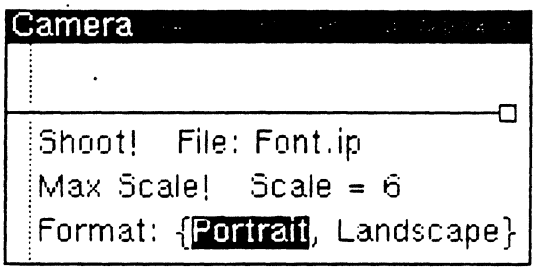
Notes

This tool makes drawing barcode symbols easy. Just specify the height and widths of the stripes and click Draw! The barcode can then be grabbed into a character or raster image window for further editing or storing. Use the scrollbars at left and bottom to move the image around in the window.

(This page intentionally blank)

Camera

Camera makes an interpress file of the parent window's screen image.



Commands

Shoot! makes and stores an interpress file in the file defined by the File: parameter.

Max Scale! sets Scale to the largest integral scale factor that allows the image to fit on an 8½ by 11 inch page.

Parameters

Scale: the number of printer pixels per screen pixel .

File: name for the interpress file. Defaults to the tool name followed by ".ip".

Format: selects portrait (long dimension vertical) or landscape mode.

Notes

The Camera tool appears in the Typefounder menu of all tools which can display images, and many others as well. It can be used to make hardcopy records of character images, tool parameters, transformation results, and so on. The image is centered on the printed page.

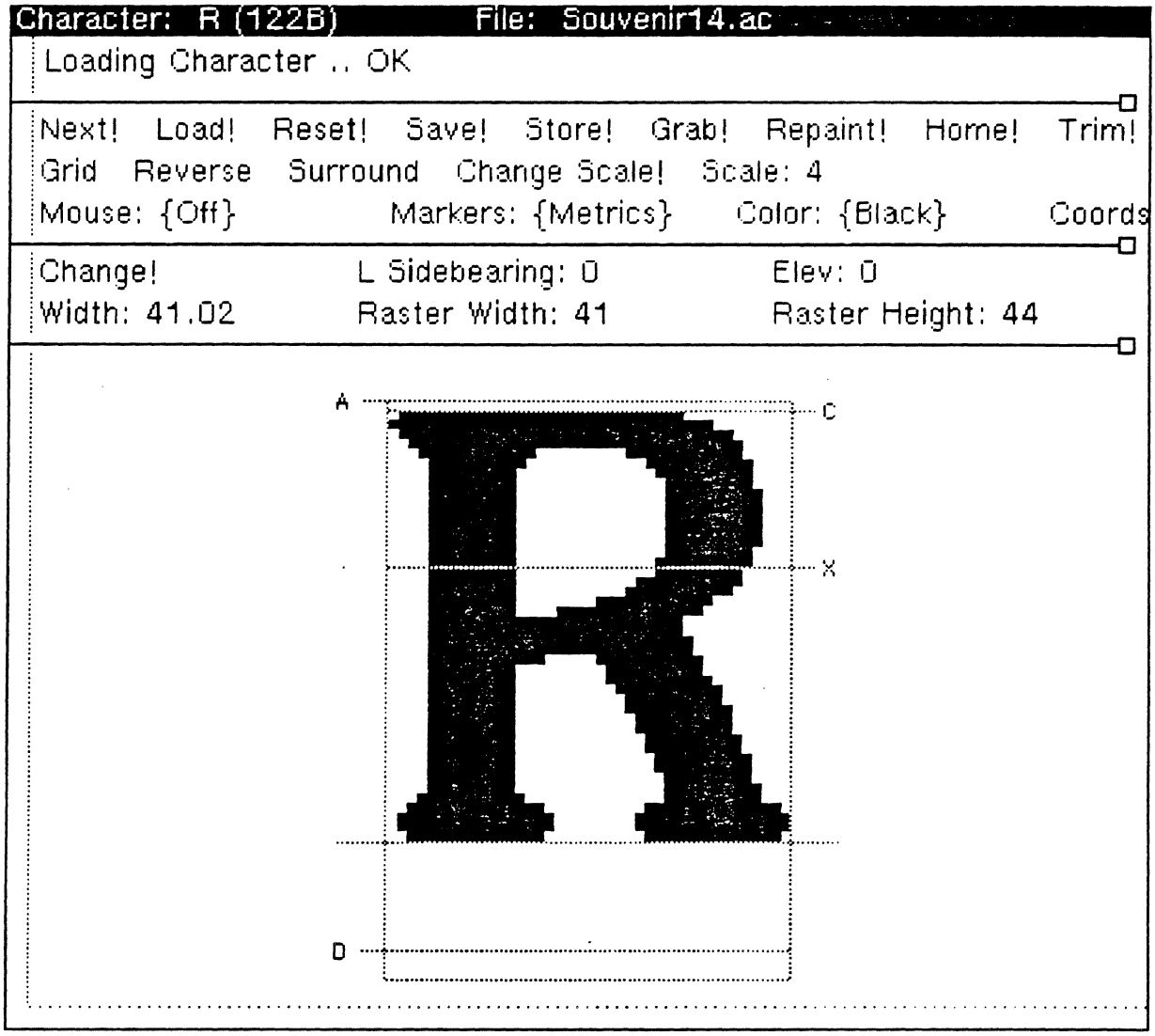
Use the Print utility in the Executive window to print the interpress file on your favorite printer.

The Proofer tool is generally used to make interpress files of font samples.

(This page intentionally blank)

Character

Character displays a character image (raster or contour) and descriptive information. The displayed image can be edited.



Commands

Next! loads the next character from the font if clicked with Point; loads the previous one if clicked with Adjust.

Load! acquires the font character specified by the current selection, either a character, character code, or an alias (see Alias subtool of Font tool). Character and character code are shown in the namestripe.

Reset! during editing, cancels edits; otherwise clears window.

Save! stores the new version of a character into the font. Confirmation is required.

Store! stores a character into the character-code position specified by the current selection. Confirmation is required if that position is already occupied. See the *Character Code Standard*, Xerox System Integration Standard XSIS 058404 for standard character codes.

Grab! loads a character from another character, raster image or barcode window, which you select by pointing with the mouse. Raster images are masked by an active box, if present.

Repaint! refreshes the display.

Home! re-centers a character in the window.

Trim! reduces the image to the smallest possible box which will enclose all the black bits in the image. This is useful after erasing around the edges of raster characters.

Change Scale! changes the display magnification. A scale changing menu pops up; fill in the desired scale factor, and hit Apply (or Abort to cancel the attempt). Scale factors range from .1 to 99. For raster characters, the scale is rounded to the nearest integer. X and Y scales can be set independently by turning off the X = Y switch. This can be used, for example, to display with square pixels images that have been scanned at different horizontal and vertical resolutions.

Change! changes character placement parameters. A parameter changing menu pops up; fill in the new parameter values. Such changes can also be done by dragging markers with the mouse.

Switches

Grid: displays points at pixel boundaries of raster characters.

Knots: shows knots on contour characters. Knots show up as small diamonds, with corners blackened if the contour segment entering that corner is a straight line.

Reverse: reverses the color of a raster character image.

Surround: reverses the color of the area outside a raster character bitmap.

Mouse: selects the action of mouse clicks in the image window.

Edits is for editing the character image and its placement. A menu of edit brushes (different for raster and contour characters) will appear. Choose a brush by clicking over the one you want.

For raster characters, a simple bitmap editor is in effect. Holding down Point blackens the pixels under the edit brush. Holding down Adjust turns the brush gray and erases. The topmost brush (the arrow) affects only the single pixel pointed at, the others blacken or erase all image pixels under the brush. All brushes affect only a single bit at scales over 10. You

can create your own brush by drawing a 15x15 pixel image in the character window and drawing an active box around it. Then chord the mouse inside the brush menu, and display the 'Special Brush' menu. Select 'User drawn' to replace the bottom edit brush by the active box contents. Select 'Default' to put back the default slanted line. An attempt to create a user drawn brush of all white bits will fail.

For contour characters, a simple knot editor is in effect. The arrow moves knots. The diamond changes the flavor of a contour segment from straight to curved or vice versa. The x deletes knots. The + adds new knots, splitting an existing segment if clicked on it, otherwise it adds a new knot and connects it to the previously drawn knot. Segment splitting is disabled while a new contour is being drawn.

If Markers are turned on, buttoning down in the small squares at the ends of baseline and side bearings lets you drag them to new positions with the mouse, thus changing the character's elevation above baseline, offset from left, or width. If you drag sidebearings with Point, the width stays constant; Adjust changes the width.

Draws Box changes the cursor to a crosshairs and lets you specify an active box in a raster image by sweeping out a rectangle with Point held down. Adjust moves the box. Active boxes act as a mask for subsequent Grabs of the image.

Measures reports distance in pixels between successive mouse clicks (delta x and y, and geometric distance). Left button (Point) reports distances rounded to nearest integer; right button (Adjust) reports up to 2 decimal places.

Markers: selects the form of markers superimposed on the character image.

Baseline shows the baseline and side bearings.

Metrics shows cap height (C), ascender height (A), x height (X), descender height (D), as well as baseline and sidebearings.

Boundary shows character bounding box plus baseline and sidebearings

Coords: activates running display of cursor position in character coordinates relative to character origin.

Parameters

Scale: magnification of character. Shown as XScale and YScale if X and Y scales differ.

Color: chord mouse over this item to select from 8 gray patterns for the character.

Width: character width including side bearings.

Elev: elevation of character above baseline.

L Sidebearing: offset of character to the left of the character origin.

Raster/Contour Width: width of the character image (in pixels for raster characters, milliems for contour characters).

Raster/Contour Height: height of the character image.

Subtools

Background: displays images as background.

Transforms: allows rotations, reflections, slanting, and thickening.

Little Image: shows character being edited at scale 1.

Notes

Very large characters and logos can be viewed by using the scrollbars at left and bottom to move the character around in the window. The size of image displayed is limited only by the size of virtual memory of the workstation. A Character window zoomed to fill the entire screen can display an image approximately 700 x 1000 pixels at scale 1.

The Save! and Store! commands affect only a working copy of the font. Changes are not made to the font file itself until the Font tool Save! or Store! is completed. Inadvertent editing mistakes can be reset to the version in the working copy by using the Reset! command, and to the version in the font file by resetting the font window and reloading the character window.

Box drawing is limited to that area of the window occupied by a raster character image.

You can delete a character from a font as follows: reset the Character window so that the name stripe reads "Empty Character", and Store! into the position to be deleted. The Merger is more efficient for deleting large numbers of characters.

Revisions of contours due to knot editing are sometimes painted or erased incorrectly if the revised curve segment lies outside the smallest rectangle enclosing the endpoints. The display can be corrected by clicking Repaint!

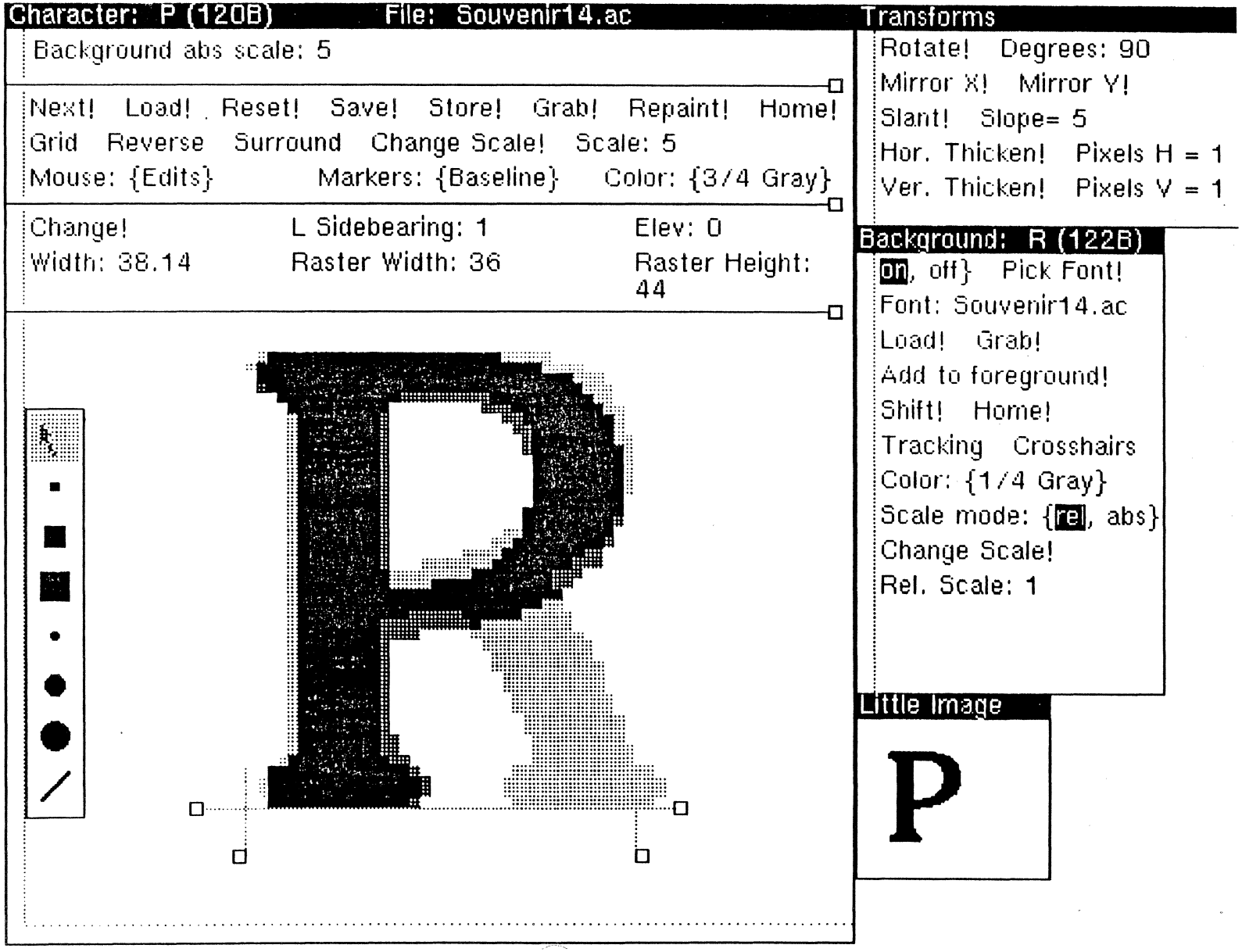
When editing contours, it is sometimes difficult to select a knot if other knots are close to it. If knots are shown on the screen closer than 6 screen pixels (.08 inches), selecting a knot with your cursor sometimes selects a close neighbor instead. To edit closely spaced knots, you should display scale until the knots are shown at least a tenth of an inch apart.

Characters from fonts with different horizontal and vertical resolutions appear distorted (pixels not square) unless the X and Y scales are set independently to compensate.

Edit menus disappear if the window is zoomed or brought to the top. Set Mouse to *off* and back to *edits* to make them visible again.

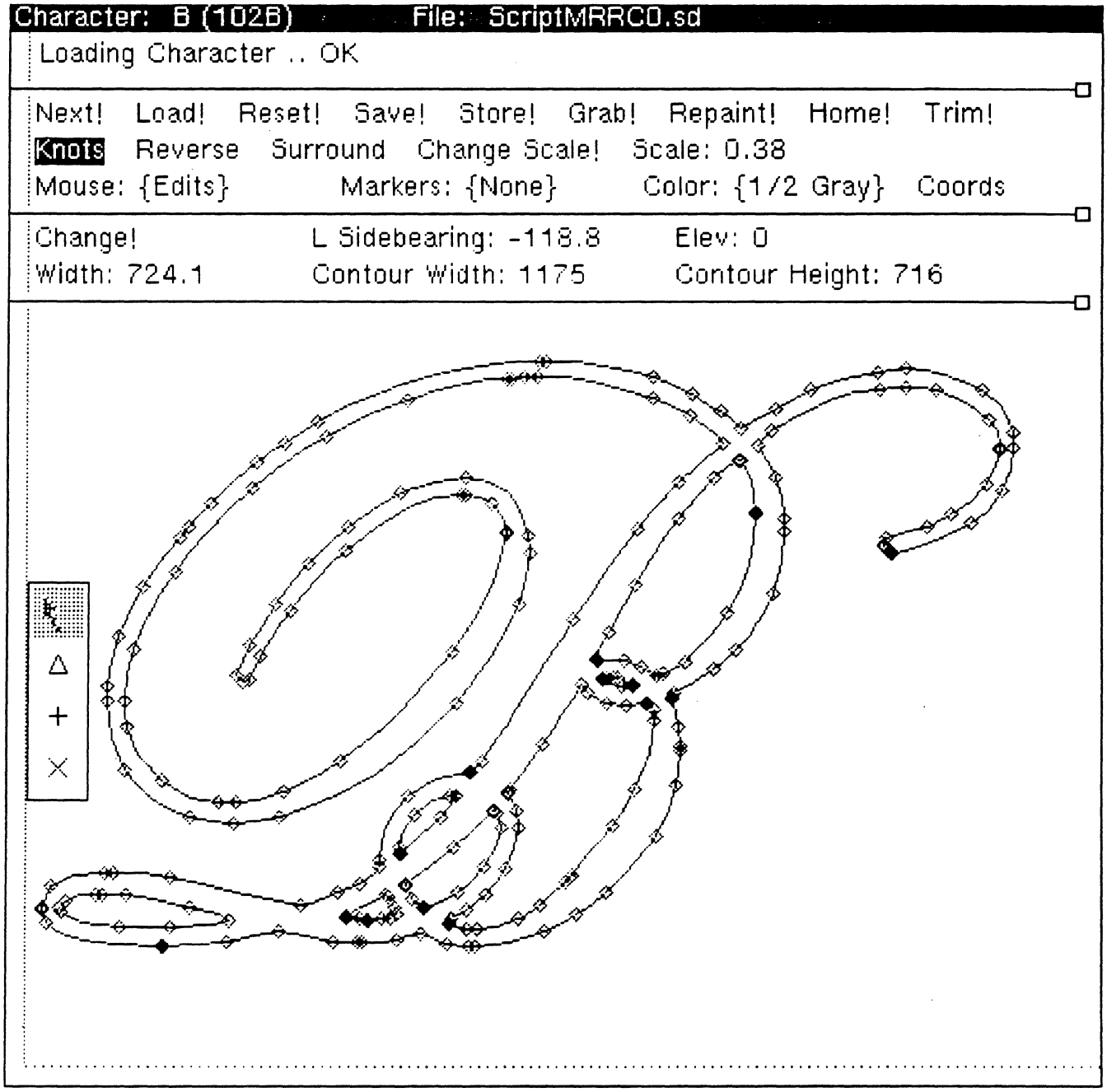
The bit editor in the character window is a very simple one. A much more versatile and sophisticated editor is available in with the Free-Hand Drawing package. This is probably a better choice if you need to draw many characters or logos by hand. Free-Hand Drawing canvases may be transferred into Typefounder by loading them into a Raster Image Window and then stuffing them into a font.

The pictures on the following pages show a Character window ready to edit a raster character with all its subtools active, and also a contour character with knots and the knot editor menu.



The CharacterWindow with its subtools.

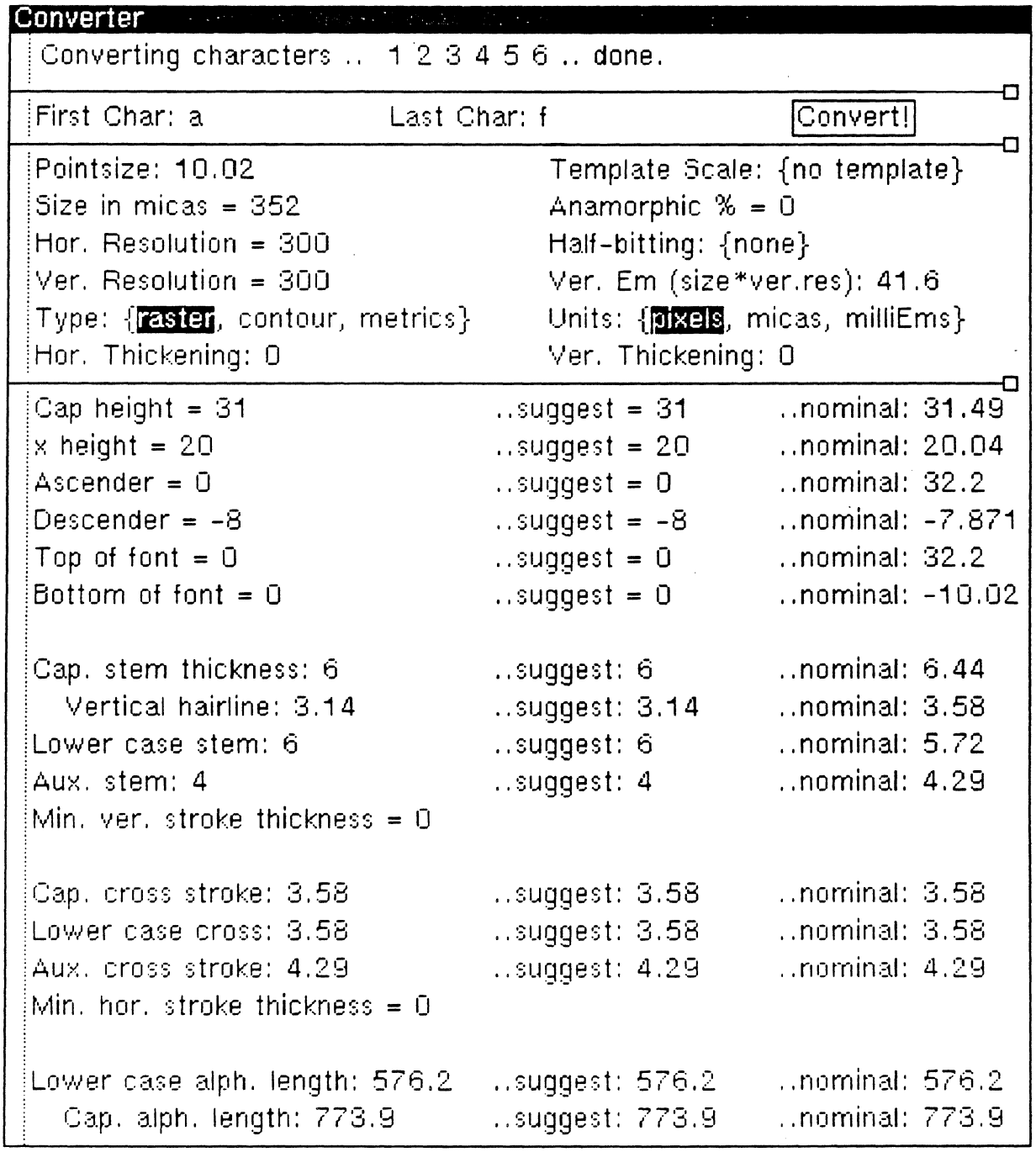
A contour character and the knot editor.



(This page intentionally blank)

Converter

Converter scales a font (raster or contour type) to produce a raster font of any size and resolution, or produces a contour font from a raster font, or produces a metrics-only font from a raster or contour font. Options include stroke thickening and thinning, constraint of font metrics such as x-height, and automatic half-bitting. When producing a raster font, a higher resolution template font may be produced simultaneously for use as a background in later editing.



Command

Convert! starts the scan conversion operation, which is controlled by the following parameters.

Parameters

First Char, Last Char: characters or character codes to specify which characters in the font are to be scaled. Default of 0c..377c indicates entire font.

Point size: size in points of output font (may be fractional, like 10.2).

Size in micas: size in micas of the output font. Size can be set in either points or micas.

Hor./Ver. Resolution: horizontal (along baseline) or vertical pixels per inch of output font.

Type: of output font. Select from *raster*, *contour*, or *metrics*.

Hor./Ver.Thickening: systematic thickening (+) or thinning (-) used to compensate for printer imaging characteristics, in units specified in the Units parameter.

Template Scale: choose from *no template*, or a numeric size multiple for the template font. If the latter, a template font with that magnification is produced along with the output font.

Anamorphic %: percent of horizontal stretching; for example, if + 10, then all horizontal dimensions of characters will be 10% wider than normal. This is useful for expanding small point sizes. If a negative value, the characters are condensed.

Half-bitting: on light, half-bitting is performed on curves and diagonals. On heavy, it's also done on vertical edges.

Ver. Em: (for information only): number of pixels in one em ((point size * resolution) / 72.3).

Units: the units used in the bottom (metrics) subwindow. Pixels for raster output, milliEms for contour or normalized metrics output, or micas for fixed size metrics output.

Constraint metrics

These metrics (shown in the bottom subwindow and defined in the Metrics tool) control scan conversion. The leftmost column shows the values that will constrain the output font, except that zero values are ignored and provide no constraint. Suggested values for the metrics are displayed in the middle column. These values are interdependent in complex ways. The rightmost column (for information only) shows precise values as calculated from master font metrics by applying scaling and anamorphic %.

All the metrics in the leftmost column except the indented ones may be edited. Adjustment of suggested values should be

made only after trying the default metrics parameters first, perhaps on just a few characters. The metrics which affect the scan conversion process the most are cap height, x height, and stroke thicknesses. Alphabet length affects character setwidths, and has a slight effect on character shapes. To change the apparent darkness of the font, adjust stroke thicknesses, but for best uniformity avoid values close to .5, 1.5, 2.5, etc. unless half-bitting is enabled. For very small fonts it is usually desirable to expand the font with the Anamorphic % parameter and increase the the setwidth with the Alphabet length metrics. It is usually not necessary to specify minimum stroke thickness.

Corresponding metrics for the master font may be displayed with the Metrics tool.

Notes

When Convert is invoked, the output and template font windows are made automatically.

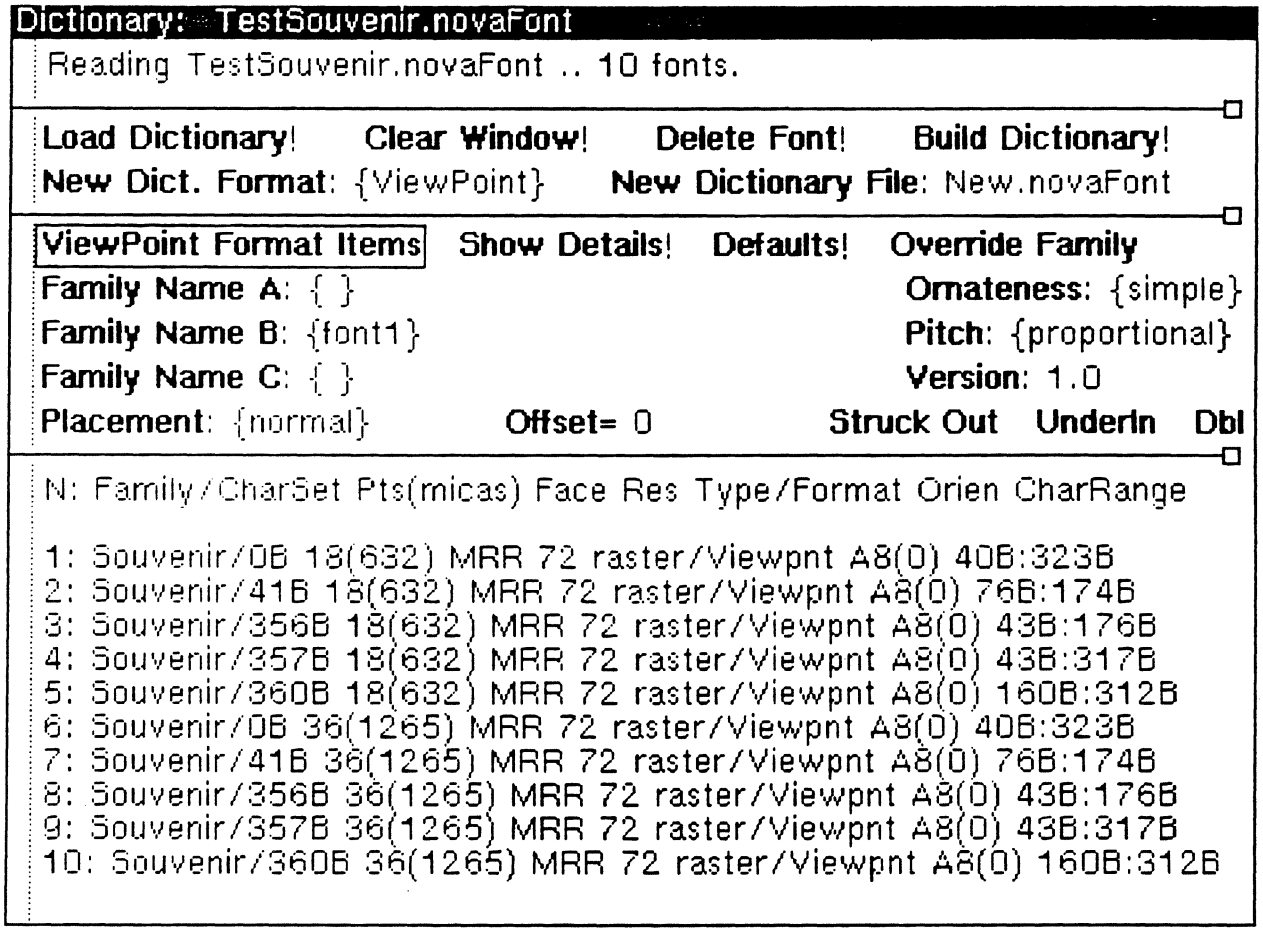
Conversion of an entire master font can be a relatively slow operation. The STOP key cancels the Convert command.

Half-bitting can provide an appearance compromise when ideal stroke widths are close to N.5 bits, when N is som small number. It is used in making the smaller sizes (6 to 14 points) of printer fonts. It smoothes the progression of apparent color as point size changes. Heavy half-bitting is useful when the thickness of the ideal cap stroke is greater than the lower case stroke.

(This page intentionally blank)

Dictionary

Dictionary lists the contents of font dictionaries (including screen font dictionaries), deletes individual fonts from them, or builds them from individual fonts or other dictionaries.



Commands

Load Dictionary! reads the dictionary file specified by the current selection, lists the contained fonts, and closes the file.

Build Dictionary! builds a dictionary from the files specified by the list of file names in the current selection (at most 200 characters when running under ToolDriver). The dictionary will have the name given in the New Dictionary File parameter.

For prepress dictionaries, the files in the list may be of the format AC, SD, WD, OC, and so on, or may be other prepress dictionaries.

For Viewpoint dictionaries, the files in the list must be AC and WD files.The AC files are individual character sets of the screen

fonts (resolution 72), and the WD files contain absolute widths of the corresponding printer fonts. For each AC file there must be a WD file that matches in family name, point size, weight, posture, bc, ec, and setwidth. Order of files does not matter unless fonts with the same family name, point size, weight, and posture have a non-unique beginning - ending character pair, in which case AC and WD file names should be paired (be adjacent in the current selection).

Clear Window! clears the window.

Delete font! (requires confirmation) deletes from the loaded dictionary the font corresponding to the line containing the current selection. Currently implemented only for prepress dictionaries.

If New Dict. Format is set to ViewPoint, the following commands also appear:

Show Details! displays in the bottom subwindow a list of Viewpoint details of the font corresponding to the line containing the current selection. This provides a way to examine parameters not in a font window.

Defaults! sets Viewpoint parameters to default values.

Parameters

New Dict. Format: select *prepress*, *star*, or *viewpoint* as the format for the new dictionary file.

New Dictionary File: name for output of the Build Dictionary command.

If New Dict. Format is set to ViewPoint, the following parameters also appear:

Font Family: name to appear in the Character Properties menu if Override Family is on or if family name in AC file is not a legal ViewPoint family name. If New Dict. Format is Viewpoint, this parameter appears as three items: Font Family Name A, B, & C. Select the one you want by chording the mouse over the item label and selecting from the menu of names that pops up.

Override Family: turn on if you wish the family name to be determined by the Font Family parameter instead of family names in AC files.

Placement: position of the characters relative to the baseline.

Offset: offset in micas above the baseline if placement is user specified.

Ornateness: *ornate* is usually used for seriffed fonts, *simple* for sans serif.

Pitch: *fixed* if all characters are the same width, otherwise *proportional*.

Version: may be used to keep track of changes to the font. Enter a number up to 9.9.

Struck Out: on if the font is struck out.

UnderIn: on if the font is underlined.

Dbl UnderIn: on if the font is double underlined

Subtools

Font: for examining and manipulating individual fonts in the dictionary. Load fonts by selecting the index number of the font in the parent Dictionary and clicking Load!.

Notes

The Font tool Merge command can also be used to add a font to a prepress dictionary.

The Delete font command is fast, but does not cause the dictionary file to shrink. If file size is a concern, use Build Dictionary with a single argument: the name of the file that has had deletions. This condenses the dictionary file as much as possible.

CD and *fonts.widths* files are both prepress dictionaries. Prepress dictionaries are also used for installing fonts on printers.

The horizontal resolution fields in Xerox CD files can contain codes for strikeout and underline positions of the font, not actual resolution of the fonts.

With novaFonts and starFonts, speed of font access decreases the deeper in the dictionary the font is located.

When calculating novaFont metics, ascend is the distance upward from the baseline to the top of the k, and descend is the distance downward from the baseline to the bottom of the p. Height is ascend + descend. All units are in pixels. If K and P are not present, Typefounder tries other characters. It looks left to right through the following arrays (using the value 0 if none of the characters are present):

```
ascenderArray: ARRAY [0..6) OF CHARACTER ← ['k, 'h, 'b, 'd, 'l, f];
descenderArray: ARRAY [0..6) OF CHARACTER ← ['p, 'q, 'g, 'y, 'j, Q];
```

Making screen font dictionaries

Input for ViewPoint screen font dictionaries is a set of pairs of font files. The first of the pair is a screen font file in AC format (resolution is 72 pixels per inch). These are generally made with the Converter from a contour or high resolution raster master, and often require touching up with the Character tool's bit editor. The second of the pair is a WD file made from the printer font corresponding to the screen font. These printer fonts are typically 300 pixel per inch AC files. WD files can be made easily by loading the printer font file into a Font window, enabling editing, changing the file format to WD

absolute, and Storing the font back under an appropriate name.

ViewPoint imposes certain limits on character image height and width of screen fonts. The height limit is 99 pixels; it may be dependent on elevation above baseline. The width limit is 256 pixels. This is especially important to consider when the font contains logos, as it imposes an effective limit of about 1.3 inches on displayed height and 3.5 inches on displayed width.

To use the new screen fonts on a workstation, they must be properly installed:

1. Find a workstation which contains a ViewPoint volume that can edit adf files (small directory files that must be put into a folder along with the novaFont file). You can assure this by creating in Tajo or CoPilot a file named "WorkStationProfile" containing the following:

[Application Loader]
Developer: TRUE
[System]
Developer: TRUE

and copying it to the system catalog with CommandCentral Run and booting the ViewPoint volume.

You will also need the Folder/Application tool Applize.bcd from the ViewPoint Programming Package. When this bcd is loaded in ViewPoint and run, the commands Folder => Application and Application => Folder appear in the desktop auxiliary menu.

2. Move the novaFont file to your ViewPoint desktop. There are several ways to do this:

   A. Use the Executive Floppy command to write the novaFont file onto a floppy disk. Then, with ViewPoint running, insert the floppy disk and Copy from the floppy disk icon.

   B. Store the novaFont file on an NS file server, boot or proceed to ViewPoint, and retrieve the file to your desktop with the Directory icon.

   C. Use the CommandCentral Run command with novaFont file(s) in the Run field, and Client volume the one on which you are running ViewPoint.

3. Make the novaFont file into an application folder:

   A. Copy an existing ViewPoint font application (such as VP Xerox Classic Fonts) from the loader to your desktop. Select it, and invoke the Application => Folder command from the desktop auxiliary menu.

   B. Open the resulting folder, and copy out the .adf file. The contents of the .adf file will be something like:

      [VP Xerox Classic Fonts]
      NovaFontFile: AStarClassic.novaFont.

C. Edit the .adf file, changing the name of the .Novafont file to your new file, and the name in square brackets to whatever you would like for an application name. More than one NovaFontFile line may be used if you would like to bundle several files into one application. With the properties sheet, change the name of the adf file to something appropriate.

D. Copy the edited .adf file and the new novaFont file(s) into a blank folder, change the folder name to the desired application name, and invoke the Folder = > Application command. **Note**: The application name must match the bracketed name in the .adf file.

E. Change the Properties of the application so that Auto Run at System Startup is TRUE.

F. If others are to use the font, copy the resulting application to a convenient file drawer.

4. To make the font appear in the Character Properties sheet font menu, copy the font onto the Loader icon, end the session, and re-boot ViewPoint. If you are replacing an existing font application, first set its properties so that Auto Run is FALSE; after the boot it will be idle and may be deleted.

   To prepare corresponding printer fonts:

   Printer font files that correspond to the new screen fonts must be loaded onto the 8044 print server before documents containing the new fonts may be printed successfully.

   The first step in doing this is to build a prepress dictionary from the same printing resolution fonts used to make the WD files that went into the novaFont file. Use the Dictionary tool Build command to merge all the printing resolution fonts into a single dictionary. For fonts used within Xerox, the dictionary name should follow the pattern Xerox.XC1-1-1.Family.Face, where Family is the Font Family Name used when building the novaFont file, and Face is either missing or one of *Bold*, *Italic*, or *Bold.Italic*. For example, "Xerox.XC1-1-1.Logo1", or "Xerox.XC1-1-1.Bodoni.Bold.Italic". Write the dictionary (or dictionaries) to a floppy disk with an Executive command of the form Floppy Write 4290/t dictionaryFileName. **Note:** Be sure to include the switch 4290/t when writing the printer font to a floppy. An error results if the switch is not used.

   Now take the floppy to the print server and logon as a system administrator (special name and password required, see your support staff). Give the commands Enable, Printing Services, Stop Printing and Install From Floppy. Install From Floppy will ask several questions; carriage return is usually the correct reply. When it is done, issue a Start Printing command. A (sometimes lengthy) font re-cataloging operation occurs, after which the new fonts are available for use.

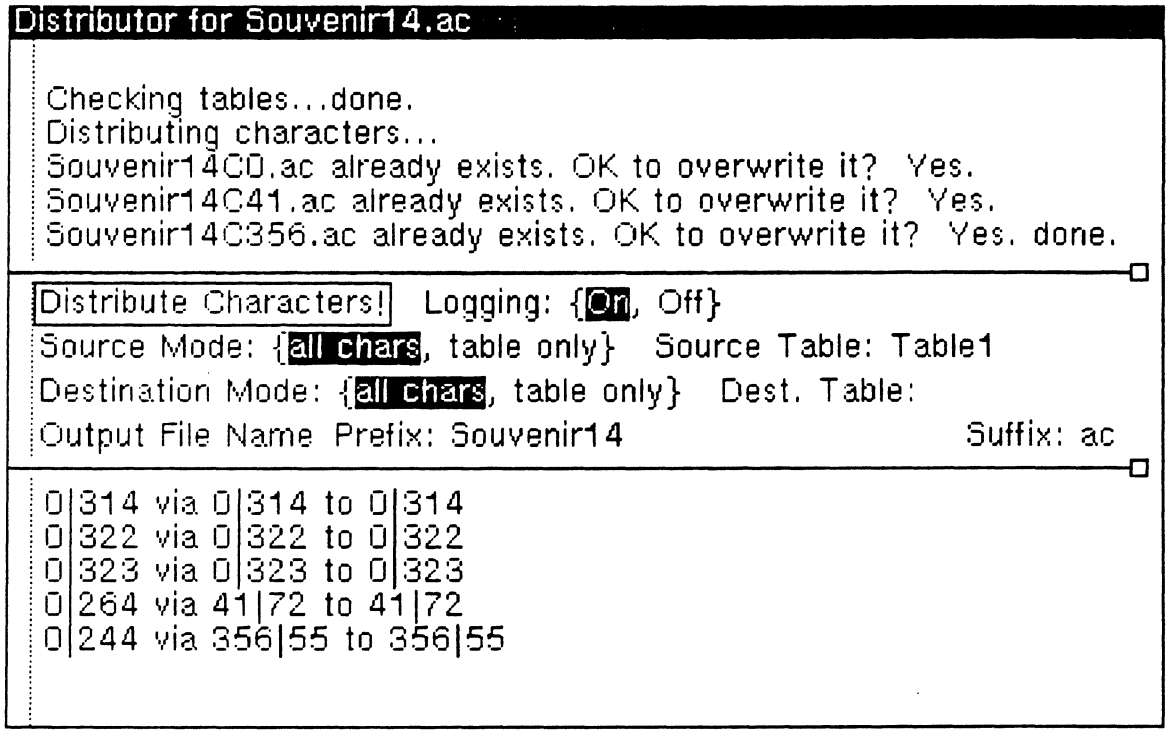
Creating a keyboard picture for your font

Once your new font is moved over to ViewPoint, it's accessed from a document window by bringing up the Properties Sheet and selecting the font's name from the menu of character fonts. If you like, you can create a picture of the keyboard with the new font's characters to remind you which keys to press. This picture displays when the KEYBOARD key is pressed. Unlike the other keyboards displayed by this key, you *can't* use it to set the font or occasionally select a character.

The Keyboard and BitMapEdit tools are used to make a picture of your Typefounder font. These tools are available from your local release directory and are explained in *ViewPoint Programming Tools Guide*, 610E00200.

Distributor

Distributor puts the characters of the parent font file into a set of output font files. Character codes can be changed by supplying tables of character code correspondences.



Commands

Distribute Characters! reads the tables, creates a font window for each destination font (one per character set), distributes the parent font characters among the destination fonts, and SAVEs the destination fonts.

Parameters

Source Mode: *tableOnly* to distribute only those characters listed in the Source Table, or *allChars* to distribute all of the parent font's characters.

Destination Mode: *tableOnly* to distribute only those characters listed in the Dest. Table, or *allChars* to distribute all of the characters read.

Source Table: name of a file containing pairs of parent font character codes and corresponding internal character codes. Required if Source Mode is *tableOnly*; see NOTES for format.

Dest. Table: name of a file containing pairs of internal character codes and corresponding output font character codes. Required if Destination Mode is *tableOnly*.

Output File Name Prefix & Suffix: used to create destination font file names; names begin with the prefix, continue with "C" and the character set number in octal, and end with the suffix.

Logging: turn on to produce in the bottom subwindow a log of each character written to the destination fonts. Each line of the log is of the form <parent font character code> via <internal code> to <destination character code>.

Notes

Distributor was originally written as an aid in the transition from font formats in which all characters reside in a single file to the XC1 format in which separate character sets reside in separate files. It is, however, a general purpose character renaming tool.

The tables used as Source Table or Destination Table specify the character code correspondences. A character code consists of a character set and a position within the character set. The Source Table maps parent font characters to internal character codes (it is suggested that standard XC1 codes [8] be used for internal character codes). The Destination Table maps those internal character codes to output font character codes. Separate tables are allowed on the source and destination sides so that once standard layouts are developed tables can be reused for many fonts rather than building them anew for each pair of input and output formats.

Each line in a table contains four octal numbers representing (in this order) old character set number, position within the old character set, new character set number, and position within the new character set. Numbers may range in value between 0 and 376. They are separated by some string of non-numeric characters (spaces, commas, blanks, helpful labels, etc.). Each line ends in a carriage return. Character set numbers may be omitted; a default of zero is assumed. If the line contains three numbers, the program assumes a value of zero for the new character set. Comments, preceded by -- (double hyphens) or // (double slashes), are allowed.

```
-- Sample Distributor Table

font    0|55        xc 0|30

font    140         xc  252 -- char set zero assumed

font   41|76        xc 0|57
```

Distribution is a two step process. First, characters are read from the parent font into an internal 'basket'. If they are mentioned in a source table, they get a new character code; otherwise they keep the same character code as they had in the parent font. If the source mode is *allChars*, ALL the characters in the font go into the basket, otherwise just the ones mentioned on the left side of the source table.go into the basket.

In step 2, characters are pulled out of the basket and placed in output fonts. Again, they may get a new character code if mentioned in the destination table. If the destination mode is *allChars*, ALL the characters in the basket are placed in an output font, otherwise just the ones mentioned on the left side of the destination table are used. The output font used is determined by the final character set.

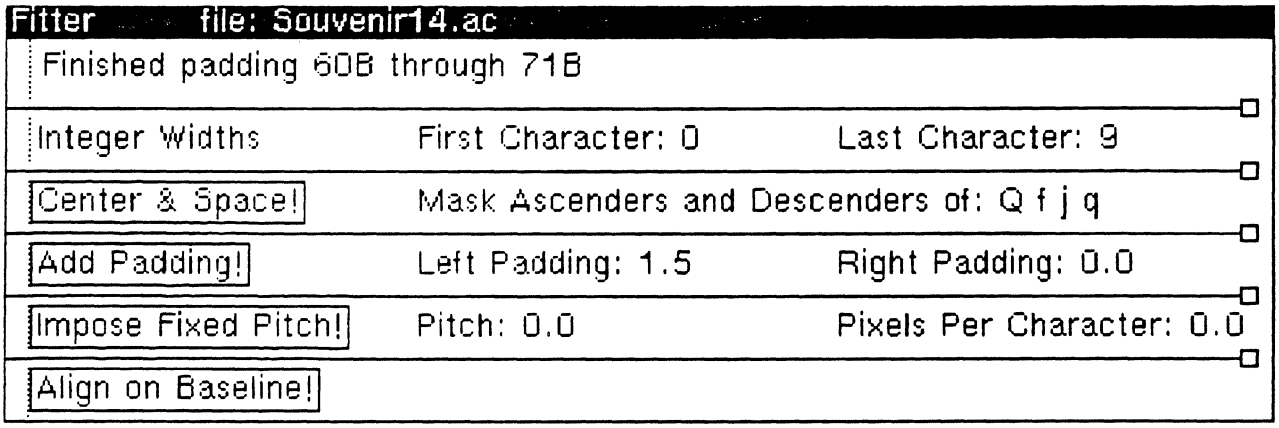
So, for example, if all but a few of the characters in your font have the correct character code, make up a table renaming only those that need to change, supply it as the source table, and set both modes to all chars.

Note: Tables are completely analyzed (and duplicate entries removed) before any characters are moved. So if the parent font contains duplicate characters (same character in more than one position), you cannot remove them by mapping both versions to the same internal code when using *allChars* mode. Instead, remove the duplicates first with Merger.

(This page intentionally blank)

Fitter

Fitter automatically calculates and assigns appropriate widths and sidebearings for the characters of a font, or makes specified changes to a font, or sets elevations so that the baseline alignment is correct.



Commands

Center & Space! automatically finds the optical center and appropriate width for each character. Sets the word spacing to 1/3 of the 'M' width. Press STOP to cancel.

Add Padding! adds Left Padding to left side bearings, and adds the sum of Left Padding and Right Padding to the character widths.

Impose Fixed Pitch! makes the width of each character equal to Pixels Per Character but does not change the center of the character.

Align on Baseline! sets the elevation of each character (using standard typographic conventions) so that baseline alignment is approximately correct.

Switches

Integer Widths: rounds the character widths to nearest integer during Center & Space and Add Padding.

Parameters

First Character, Last Character: character range on which the command operates. It recognizes character, alias, or character code.

Mask Ascenders and Descenders of: characters whose ascenders and descenders will be ignored by Center & Space in determining width. This allows such

characters to be properly kerned. The format is a list of characters (or aliases or character codes) separated by spaces.

Left Padding, Right Padding: positive or negative padding to be applied to the left or right side of character when Add Padding is selected.

Pixels Per Character: width used in Impose Fixed Pitch.

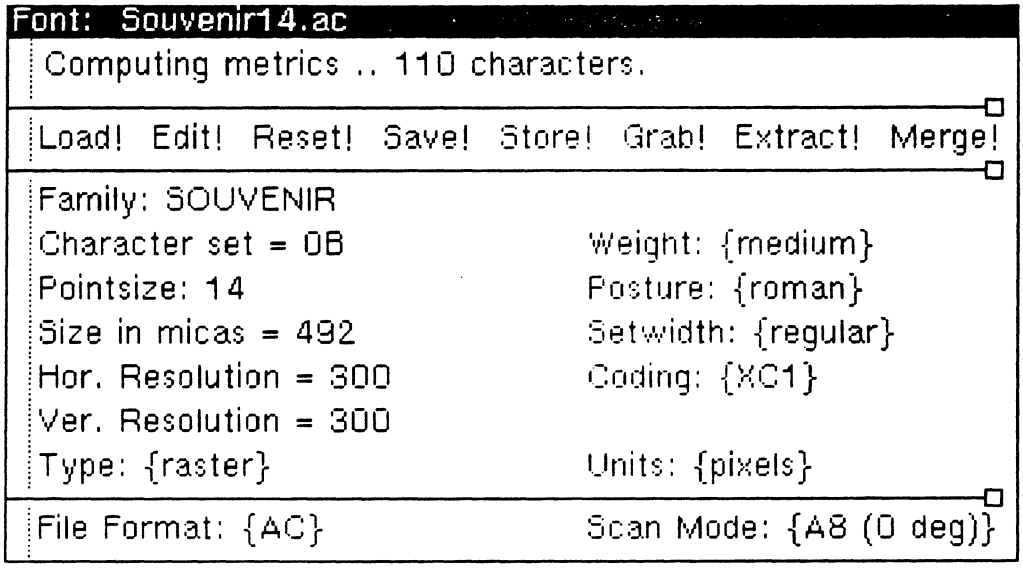
Pitch: characters per inch, an alternate way of specifying width used in Impose Fixed Pitch.

Notes

The Impose Fixed Pitch command is useful in setting the numbers of a font to a given width. Set the character range to be '0 to '9 and specify the width desired in the Pixels Per Character field.

Font

Font displays naming and descriptive information about a raster, contour, or metrics (widths) font and allows it to be edited. It also performs certain format and type conversions.



Commands

Load! acquires the font from the file named by the current selection. The font is displayed in normal reading orientation. If the file is a dictionary, Point loads the next font in the dictionary and Adjust loads the previous one; the ordinal number of the font in the dictionary shows in the namestripe. If the parent is a Dictionary tool and the current selection is a number, say 10, the 10th component of the dictionary is loaded.

Edit! enables editing of the font parameters and storing of edited characters.

Reset! during editing, cancels edits; otherwise it clears the window.

Save! stores font into same file (original saved with appended $). If canceled with the STOP key, the original file is unchanged.

Store! stores the font into the file named by the current selection. Units are converted automatically for the desired format. For example, to extract widths from a raster file, read in the font, select WD format and store the file.

Grab! replaces characters in the font with those from another Font window (selected by pointing to it with the mouse). Characters from the grabbed font replace corresponding characters in this font. Characters not in the grabbed font are unchanged. If the font types are different, only widths are moved, with proper units conversion. Grab is useful for imposing widths.

The following two commands operate on PrePress style dictionaries (such as those used for print servers and *fonts.widths*). See NOTES for information on size comparisons.

Extract! searches through the dictionary file named by the current selection for the font matching the parameters in the Font window. Parameters checked are family name, mica size, x and y resolution, weight, posture, and setwidth. Unspecified parameters are treated as wild cards. Point searches entire file, and Adjust searches forward from the presently loaded font.

Merge! merges the font into the dictionary file named by the current selection.

Parameters

Family: name of the typeface, such as Helvetica.

Character set: 0 through 377B; indicates which set of 256 characters (see Xerox *Character Code Standard*, Xerox System Integration Standard XSIS 058404).

Pointsize: point size of font, such as 10.5.

Size in micas: point size expressed in micas.

Hor. Resolution: bits per inch along the horizontal with character in reading position.

Ver. Resolution: bits per inch along the vertical with character in reading position.

Weight: darkness or stress; values are *bold, medium, light.*

Posture: similar to slant; values are *roman, italic.*

Setwidth: spacing of the font; values are *regular, condensed, expanded.*

Coding: mapping between character codes and character shapes, either *XC1* or *Ascii.*

Type: representation of the characters: *raster* (bitmap), *contour* (outlines) or *metrics* (widths only).

Units: units used in widths, sizes of characters, and in Metrics Tool. A *pixel* is the unit of bitmap resolution, a *mica* is 10 microns, and a *milliEm* is .001 times the nominal height of the font.

File Format: Most common Xerox font formats are supported. Possible values are:

AC	a raster format, preferred for smaller sizes.
OC (orbit)	a raster format often used with press printers.
Compressed OC	OC which has undergone bitmap compression; the preferred raster format for large sizes.

Strike	a compact format designed for easy use with BitBlt, in which the font is stored as one long bitmap. Often used for screen fonts. Maximum character width is 255 and kerned parts of characters are not saved.
Kerned strike	a strike format allowing kerning.
SD	spline representation of contours
Viewpoint	Viewpoint screen font dictionary
WD (abs)	metrics (widths) for a font of a particular size. Units are micas.
WD (norm)	metrics (widths) for a typeface in size-independent form. Units are milliems.
FIS	Font Interchange Standard format, a new 16 bit character code format.
CU	"Carnegie Mellon University" format; simple, but lacks certain useful information, such as resolution and baseline.
MC	master format: OC with added character metrics to assist in scan conversion.
Compressed MC	Compressed OC with character metrics.

Scan Mode: rotation plus an indication of scan direction; possible values are *0, 90, 180, 270* (degrees), and those values mirrored (scanned right to left).

Subtools

Alias: makes a table of alternative names for characters.

Character: creates a window for accessing individual characters.

Converter: scales font to various sizes or resolutions.

Lister: lists the characters in the font with data such as character width and body size for each.

Fitter: adjusts the character widths.

Merger: merges in characters from other fonts, or moves characters to new character codes.

Metrics: creates a window showing various parameters describing the font.

Text Display: displays text from the current font at one sceen pixel per image pixel.

Variations: makes design variations of the font.

Notes

Any changes to font parameters or characters become permanent only when the font is Saved or Stored. Access to individual characters is enabled through Character windows, which may be created from the Typefounder menu.

Font file format conversion may be performed with the Font tool by loading a font, changing the File Format parameter, and storing it under a new name. Type conversion from raster to contour or vice versa is done with the Converter.

Attempts to edit font parameters that are not currently set to edit mode cause a window to blink. The name stripe must read "Editing font ..." to change any parameters, and even then changes to certain parameters (for example, Hor. Resolution when File Format is WD) do not make sense and will cause a blink.

The character set number is actually stored in the Ver. Resolution field in most font file formats. To make a permanent change to Character Set, you must edit the Ver. Resolution field, not the Character set field.

Font writing can be canceled with the STOP key, and will leave a valid font file if canceled (though it will be missing any characters not yet written). Font reading cannot be canceled.

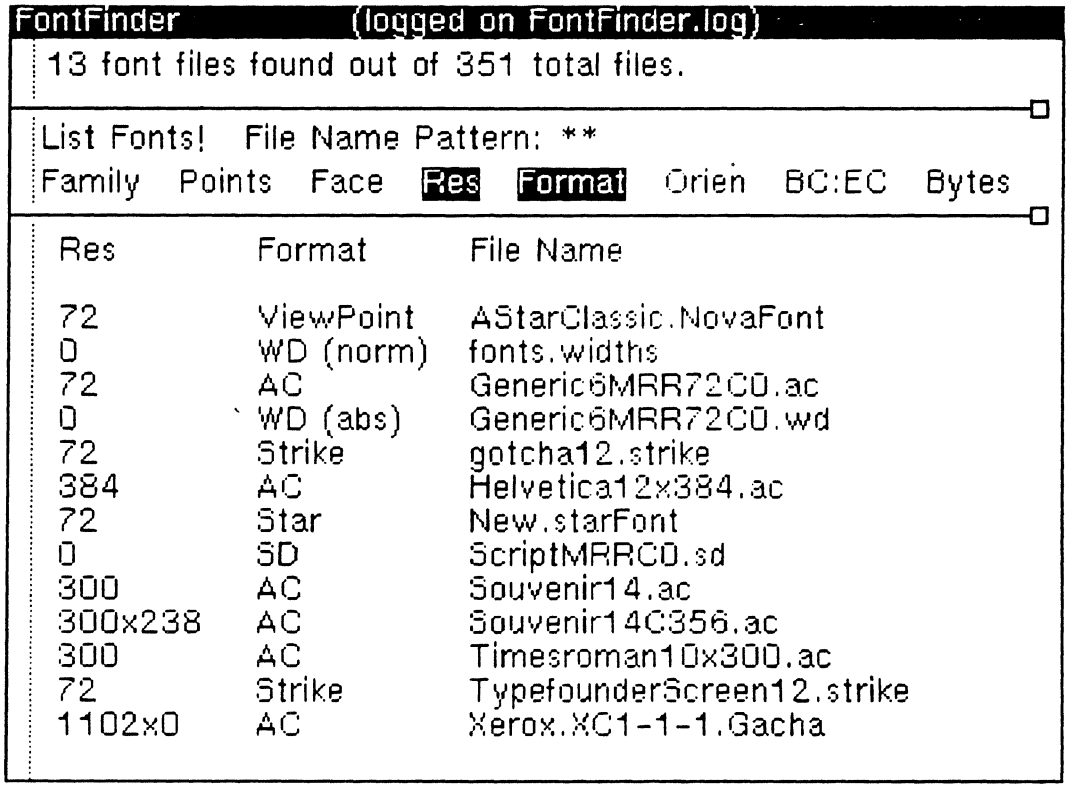
AC, OC, CC, COC, MC, CMC, SD, and WD files all encode size as an integer number of micas. In calculating this value, Typefounder assumes 72.289 points per inch. PrePress assumes 72.0 points per inch, so slight differences from Prepress-created files may occur. In size comparisons when matching fonts in dictionaries, Typefounder allows a tolerance of ±.057 points (2 micas) or 0.5%, whichever is larger.

Most raster font formats contain equivalent information. Use AC for smaller sizes, Compressed OC for larger sizes of printer fonts, and MC or Compressed MC for master fonts on which scan conversion will be performed.

When you specify an integer point size in the Font window then save or store the font file, Typefounder converts that integer point size to a mica size. This mica size is rounded to the nearest integer. The point size is recalculated from this integer mica size, and displayed to the nearest hundredth. Unexpected changes in value can be caused by roundoff in this calculation. Conversion factors used are 72.289156 points per inc and 2540 mica per inch.

FontFinder

Font Finder lists font files on your logical disk volume.



Commands

List Fonts! searches the logical volume and lists all Typefounder readable font files, no matter what the file name. The formats currently recognized include AC, OC, Compressed OC, MC, Compressed MC, CU, SD, ViewPoint, Strike, Kerned Strike, Dictionary, WD (absolute),and WD (normalized).

Parameters

File Name Pattern: specifies the set of files to be searched. Use ** to look at all files on the volume, *.ac for just files with the suffix ".ac", and so on.

Switches

Family: shows family name.

Points: shows point size.

Face: shows the three letter code indicating weight, posture, and set width. First character (weight) is L for light, M for medium, or B for bold. Second character (posture) is I for italic or R for

roman. Third character (set width) is C for condensed, R for regular, or E for extended.

Format: format of the font file. One of { AC, OC, Compressed OC, MC, Compressed MC, CU, SD, Strike, Kerned Strike, Dictionary, ViewPoint, Font Interchange Standard,WD (absolute),and WD (normalized)}.

Orien: shows orientation (rotation). This is given in degrees, followed by "mirror" if a mirror image.

Res: shows resolution. This is listed as HRes x VRes if horizontal and vertical resolutions differ.

BC:EC: shows the character codes of the beginning and ending characters.

Bytes: shows the size in bytes of the font file.

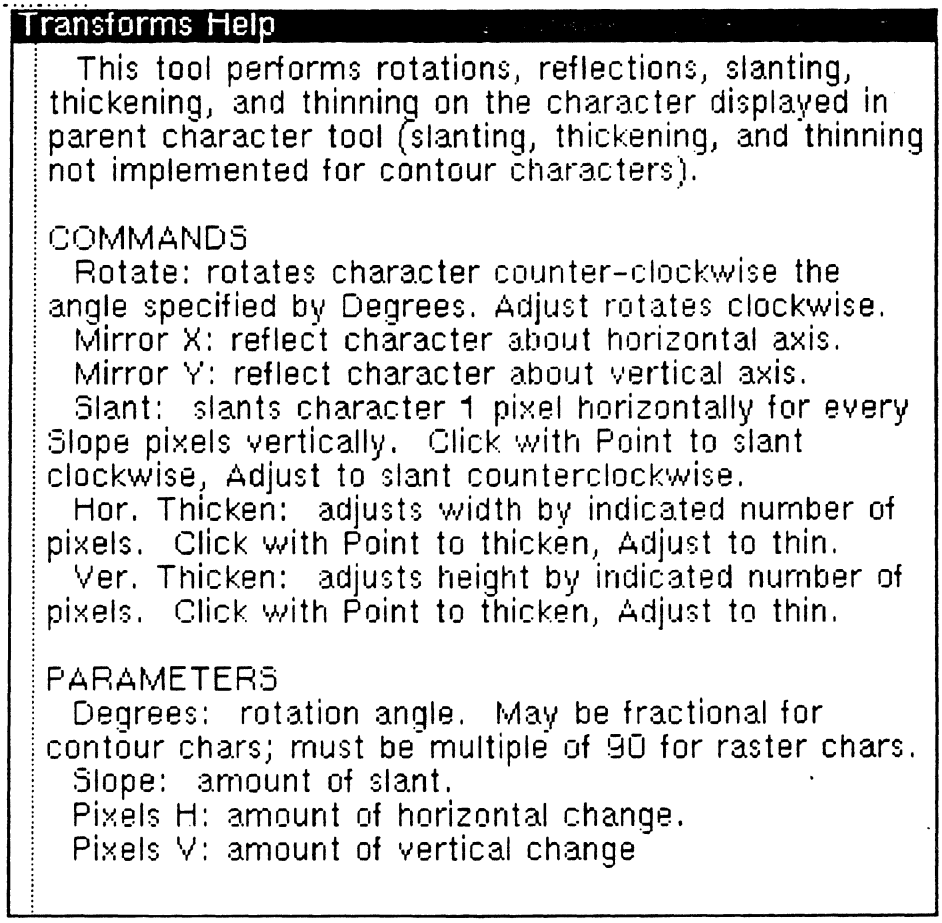
Notes

The STOP key cancels the listing. The results are left in the FontFinder.log when the tool is deactivated.

The search can be restricted to a sub-directory with the file name pattern, or by using the SearchPathTool to set the current search path.

Help

Help provides quick on-line help for using Typefounder tools.



Notes

Information on the purpose and use of any Typefounder tool can be displayed by selecting Help in its Typefounder menu. This information is shown in a pop-up window at the right ot the tool. It is generally a summary from this manual, organized as follows:

COMMANDS: available commands. Commands always end in an exclamation point (!).

SWITCHES: true/false items (switches are TRUE when video-inverted)

PARAMETERS: displayed numbers, enumerated items, and strings, which you can usually change.

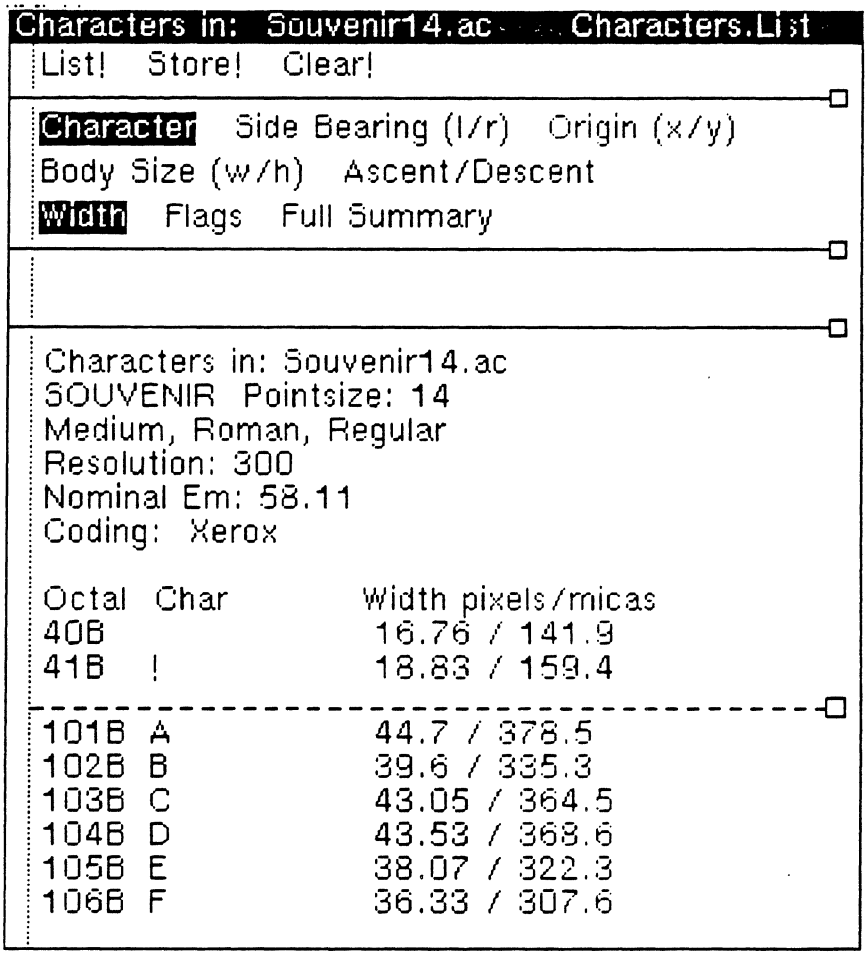
SUBTOOLS: tools which can be run from the Typefounder menu in the window. The Camera tool is generally not mentioned but it is, available in most tools.

In the Help text, 'Point' refers to the left mouse button, and 'Adjust' to the right one. 'Chording the mouse' means to press both buttons at the same time.

It is usually necessary to scroll the text using the scrollbar at the left side of the window to view all the help text. Remove a Help window by using Destroy in its menu.

Lister

Lister lists all the characters in a font and provides a variety of metrics for each.



Commands

List! shows the character code and the parameters defined by the switches for each character in the font.

Store! stores the list in the file indicated by the current selection.

Clear! clears the bottom subwindow.

Switches

Character: shows the character symbol.

Side Bearing (l/r): shows the distances of the left and right side bearings to the left and right edges of the character image.

Origin (x/y): shows the placement of the character image relative to left side bearing and baseline, respectively.

Body Size (w/h): shows the width and height of the character image.

Ascent/Descent: shows the distance from the top of the image to the baseline and the distance from the baseline to the bottom of the image.

Width: shows the character width between side bearings (as opposed to image width). Uses pixels, micas, or milliEms as appropriate for the font.

Flags: notes the characters which are left or right kerned or have no image.

Full Summary: lists the values of the following, some of which are font metrics:

Average character width: sum of the widths of all the characters divided by the number of characters.

Number of space characters: the number of characters which have a non-zero width but no image.

Upper and lower case alphabet lengths: sum of the widths of all the upper or lower case characters.

Font height: the distance from the top of the highest image to the bottom of the lowest image.

Max ascender: distance from the top of the highest image to the baseline.

Max descender: distance from the bottom of the lowest image to the baseline.

Notes

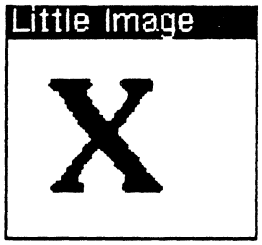
Listing can be canceled by pressing the STOP key. The Store! command can be used to put the list into a file for further editing, or you can deactivate the tool and use the default log file named in the name stripe. This file is named Characters.list for the first Lister activated, Characters.list1 for the next, and so on. All measurements are in pixels unless otherwise stated. For best column alignment in printing use an 8 or 10 point, fixed pitch font with a tab setting equal to 6 spaces.

Characters which have neither an Alias name or a printable character in the screen font display as a black box in the Character column. This black box may be selected and copied to a Text Display window to display the character image for that character code.

If the column headings disappear from the second subwindow, scroll it until they reappear. You may need to make the window wider for all the headings to fit.

Little image

Little Image displays a raster image of the parent chacracter with a scale factor of 1.



Notes

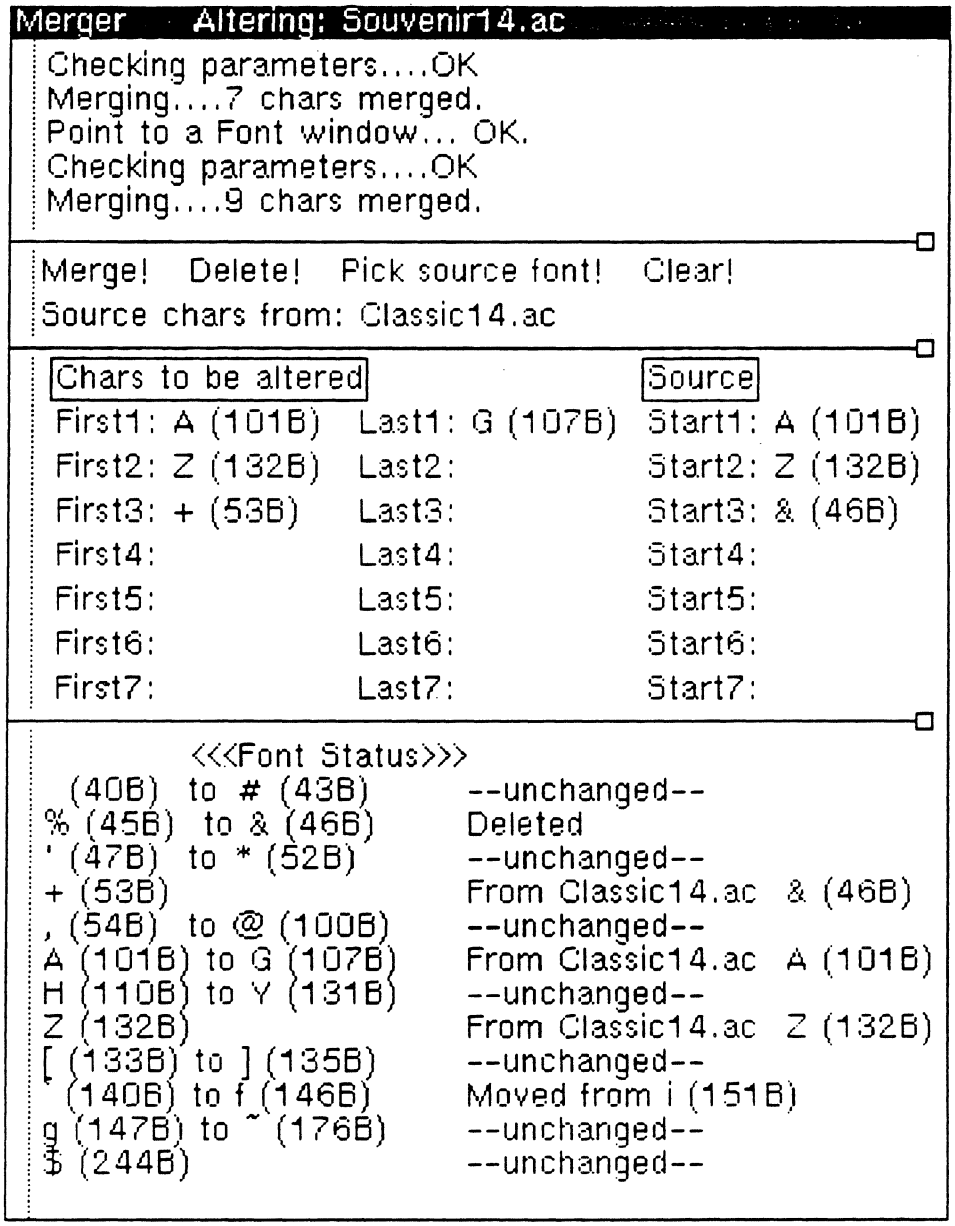
Little Image windows are created large enough to show the currently displayed character with a small margin. They may be made larger or smaller as needed with the Window Mgr Grow command. Click the mouse in the window to invert the black/white sense of the image.

Contour characters do not appear in the Little Image window.

(This page intentionally blank)

Merger

Merger alters the parent font by merging in characters from another font, moving characters to new character codes, or deleting characters.



Commands

Merge! replaces the characters or character ranges specified as 'Chars to be altered' by the characters from the source font, starting at the specified start character and increasing the octal character code by one each time. If a source character is NIL, the corresponding parent font character is not changed. If code 377 is reached, the operation ends. When the source chars are from a parent font, this simply moves the characters to new character codes within the font.

Delete! deletes the indicated characters from the font.

Pick source font! prompts you to point to a Font window which should be used as the source of characters for the Merge! command. Defaults to parent font.

Clear! clears all the character range fields.

Parameters

Seven character ranges are visible when the tool is started, but 40 are available if needed. Scroll the ranges subwindow or make it larger by dragging the small box at the right of subwindow to see the others. Numbers on the end of the labels are for ToolDriver reference and are otherwise insignificant.

Chars to be altered: character ranges to be altered. Specified by typing characters, aliases, or character codes into the fields labelled First & Last. Last may be left blank for single characters.

Source: start of replacement range. If left blank, defaults to whatever is entered in First. Syntax is same as for First and Last.

Source Chars from: file name of font used as source characters for Merge!, or the words "parent font" if the source font is the same as the destination.

Notes

A font status display is presented in the bottom subwindow (and logged in Merger.log). It shows ranges of non-empty characters, indicating whether they are unchanged (by Merger commands), Deleted, Moved from (other positions in the parent font), or From another source font. In the latter two cases, the start of the replacement range is shown. This information is preserved in the Merger.log when the tool is destroyed or deactivated.

Source fonts should usually match the parent font as to point size and resolution. Pick source will warn you if this is not the case, and ask if you wish to proceed.

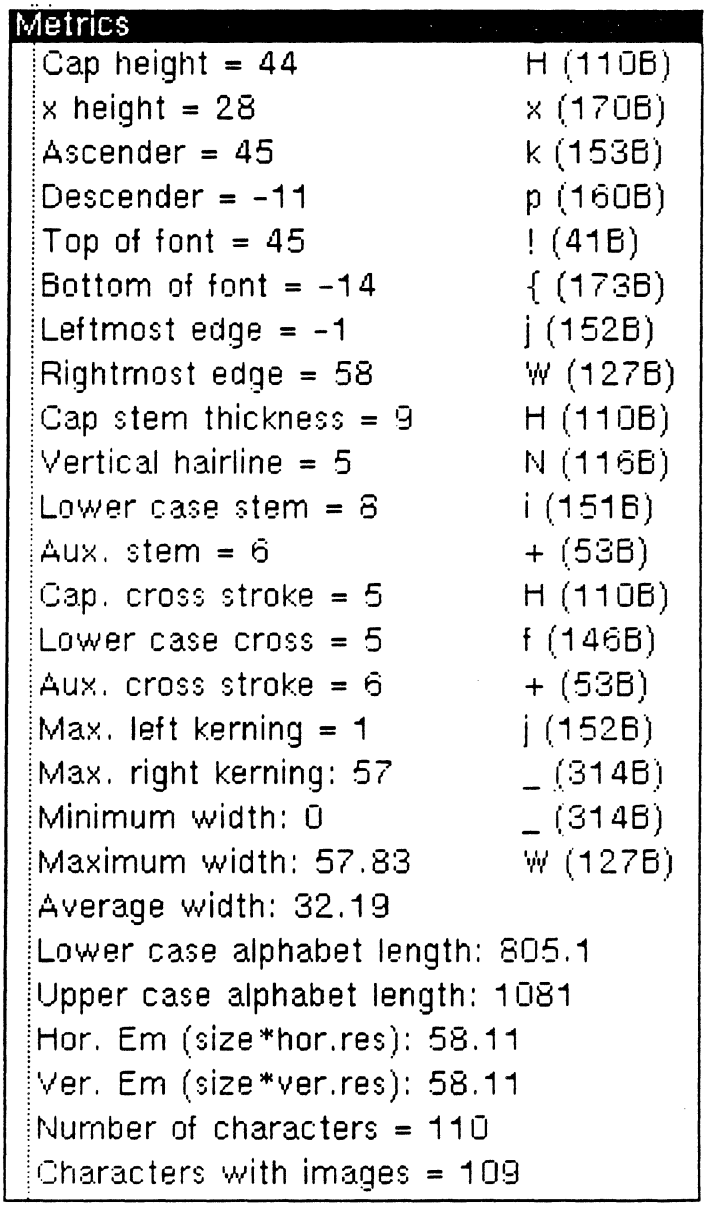
Merger can handle only one source font at a time. If you need to merge in characters from more than one source font, just repeat the Pick source / Move chars sequence, or start multiple Mergers under the parent font. Making the source fonts tiny helps conserve screen space.

Note: Merger alters the parent font in place. Effects of Merger operations can be undone by resetting the font, but if you wish to be extra careful, work on a copy of the font file.

The *Character Code Standard*, Xerox System Integration Standard XSIS 058404, defines the standard Xerox character code assignments.

Metrics

Metrics displays various measurements and parameters of the parent font.



Parameters

Units are those of the parent font: usually pixels for raster fonts and milliEms for contour fonts.

Cap height: height of a representative capital letter, usually H.

x height: height of a representative lower case letter, usually x.

Ascender: distance upward from the baseline to the top of a representative character, usually k.

Descender: distance upward from the baseline to the bottom of a representative character, usually p. **Note:** since this is distance upward, it is typically negative.

Top of font, Bottom of font, Leftmost edge, Rightmost edge: the font bounding box, the smallest box that would completely contain any character in the font (positions are relative to the character origin).

Cap stem thickness: width of a typical vertical stroke of a capital letter, usually H.

Vertical hairline: width of the leading vertical stroke of an M or N.

Lower case stem: width of typical vertical stroke of a lower case letter, usually i.

Aux stem: width of typical vertical stroke of a non-alphabetic letter, usually the plus sign (+).

Cap. cross stroke: height of the cross stoke of a representative capital letter, usually H.

Lower case cross stroke: height of the cross stoke of a representative lower case letter, usually f.

Aux. cross stroke: height of the cross stoke of a representative non-alphabetic letter, usually the plus sign (+).

Max. left kerning: the most a character overhangs the origin to the left.

Max. right kerning: the most a character overhangs its right sidebearing to the right.

Minimum width: width of the narrowest character in the font.

Maximum width: width of the widest character in the font.

Average width: average of the character widths.

Lower case alphabet length: sum of the widths of the lower case letters.

Upper case alphabet length: sum of the widths of the upper case letters.

Hor. Em: horizontal em size (size times horizontal resolution).

Ver. Em: vertical em size (size times vertical resolution).

Number of characters: count of the characters in the font.

Characters with images: count of the non-space characters in the font.

Notes

Each metric is shown with the character from which the measurement was derived, if appropriate.

Displayed metrics are revised automatically when the font is loaded or changed, but they are not recomputed when a font is loaded which has the same style and size parameters (family,

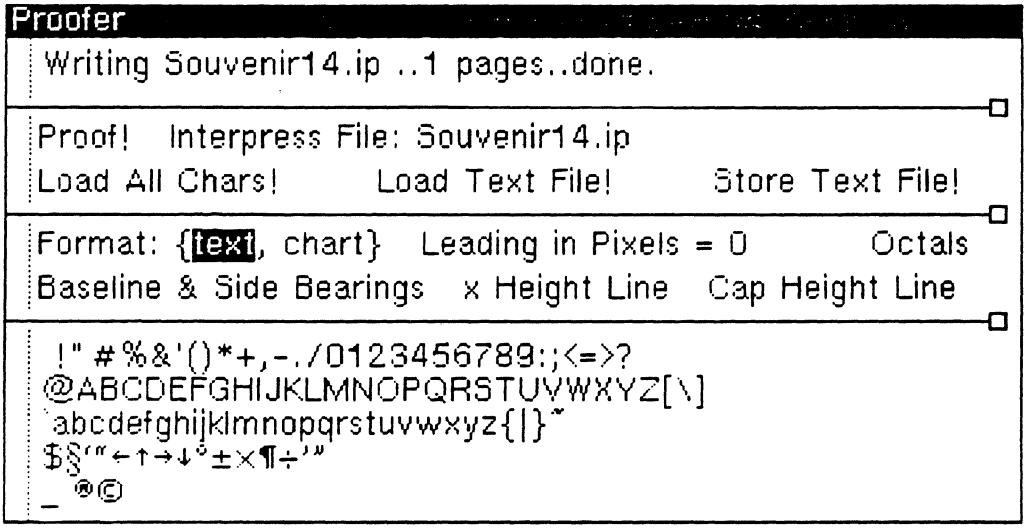
point size, weight, set width, and posture) as the previously loaded font. This enables the metrics for character set 0 to be retained and used for other character sets.

For CU fonts (which contain no baseline information), Typefounder calculates Ascender and Descender by using the baseline at the bottom of the lowest pixel of character A.

(This page intentionally blank)

Proofer

Proofer produces proofs of either selected text or all the characters in the parent raster font. Proofs are printable interpress files.



Commands

Proof! writes an interpress file of the text shown in the bottom subwindow (or all the characters if *chart* is selected), in the font of the parent font window (which must be a raster font). The file is named as defined in the Proof File parameter.

Load All Chars! loads the bottom subwindow with all the characters in the font, in ascending order by character code. For easier reading, a space is inserted after characters whose width is zero.

Load Text File! loads the bottom subwindow from the file named by the current selection. You may also type directly into this subwindow, and edit it as you would an ordinary text subwindow.

Store Text File! writes the contents of the bottom subwindow to the file named by the current selection.

Parameters

Interpress File: name for the interpress file. Defaults to the font name followed by ".ip".

Format: determines which characters are printed and how they are placed on the page.

text: print whatever has been typed or loaded into the bottom subwindow.

chart: print a grid containing all the characters in the font positioned by octal character code.

Leading in Pixels: distance in pixels between text lines in text format.

Switches

Baseline & Side Bearings: shows baseline and side bearing positions with dotted lines.

The following applies only to *text* format:

Octals: shows the octal character code above the character.

x Height Line: shows a horizontal line at the x height of the font.

Cap Height Line: shows a horizontal line at the cap height of the font.

The following applies only to chart format:

Widths: shows the numeric width of each character in the lower right hand corner of its box .

Auto Scale: if the characters are too big to fit in the boxes, scale them to fit. If off, the characters that have a bounding box dimension greater than 180 will generate an error message and the Proof command wil end.

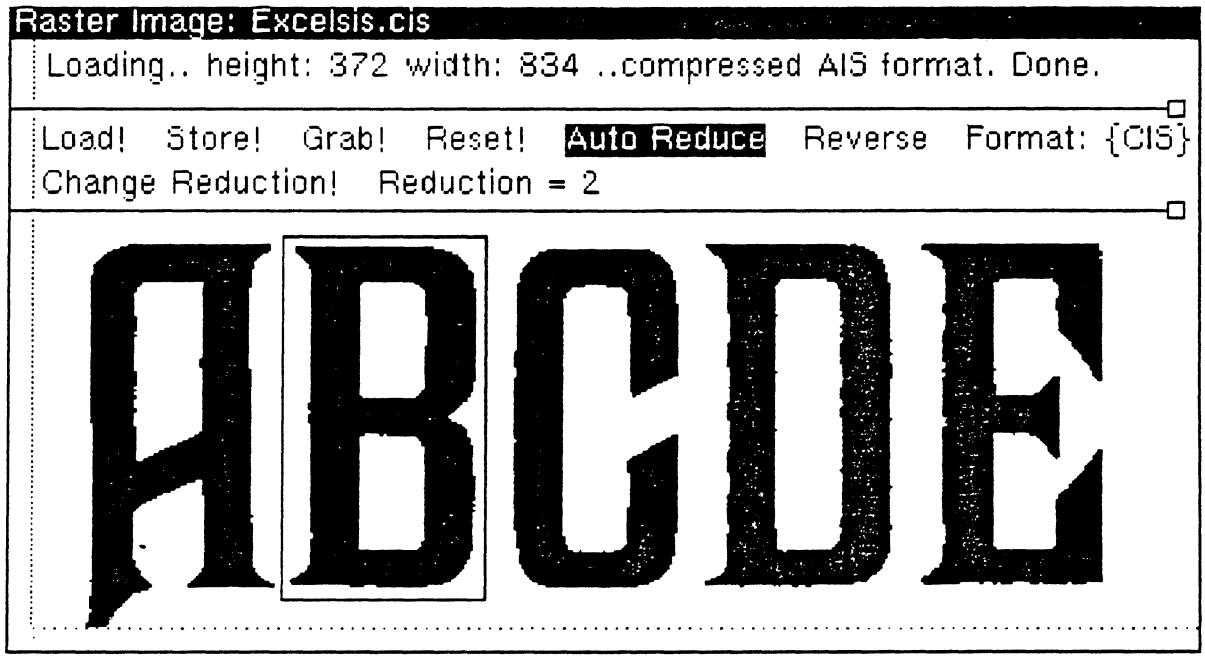
Notes

Use the Print utility described in the *XDE User's Guide* to produce hardcopies from the interpress files.

Certain text format proofs can tie up the printer for long periods of time. In general, expect long waits (or in rare cases, print server crashes) on more than a page or two or very large characters (an inch or more high), or more than a few hundred very small characters.

Raster Image

Raster Image displays AIS or compressed AIS files, strike font bitmaps, or Viewpoint Free-Hand Drawing canvases, and acts as a source of images for the Segmenter subtool. Raster images may also be cropped and stored in a new file, with optional format conversion.



Commands

Load! displays the raster image from the file named in the current selection. Load! can be canceled by pressing the STOP key. The file name is posted in the namestripe.

Store! saves the current image (masked to the active box, if present) in a file named by the current selection. The current selection must be a name different from the file loaded. Store! can be canceled by pressing the STOP key. The output format is controlled by the value of the Format parameter, except that strike format cannot be written.

Grab! loads the image (masked by its active box, if present) from a Character window, Barcode window, or another Raster Image window. Active box is described in NOTES at the end of this section.

Reset! clears the window.

Change Reduction! lets you change the reduction factor. Type new value, and click Apply! Turn the x = y switch off to make the x and y reduction different.

Switches

Auto Reduce: if on, the Load! or Grab! commands automatically set a reduction factor that allows the entire image to be displayed in the window. If off, the value in the Reduction parameter is used.

Reverse: video-inverts the image.

Parameters

Reduction: the factor by which the image is made smaller. If the x and y reduction factors are not equal, each will be shown.

Format: indicates the type of displayed image, and controls output format for the Store! command. Possible values are AIS, CIS (compressed AIS format), strike, Interpress (also used for Viewpoint Free-Hand Drawing canvases), Doodle bitmap, or Doodle brush. Interpress output is an interpress file which is printable, but is not a legal Viewpoint Free-Hand Drawing canvas.

Subtools

Segmenter: makes characters from rectangular sub-areas of the raster image.

Notes

Loading time depends on the image size and can be long. The process also requires significant amounts of disk space for large images, especially if the file being loaded is compressed.

The cursor changes to a cross when inside the bottom portion of the window: left button down on the mouse outlines a rectangular area called the "active box", and the right button moves the box. This active box is used in partitioning the image with Segmenter, or to mask off an area for use as the target of a Store! command in this tool or a Grab! command from another tool.

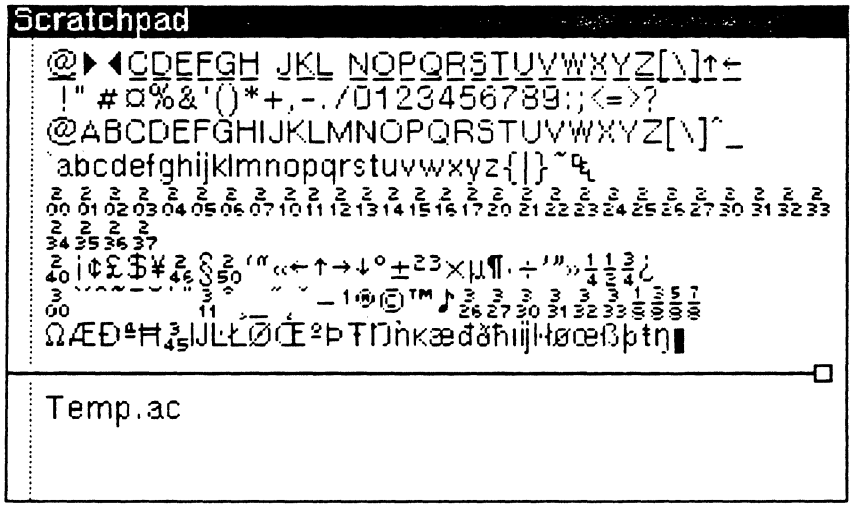
Characters, logos, or bar codes may be converted to the various raster formats by using Grab! and then Store!. AIS files may be compressed by Loading them, setting Format to CIS, and clicking Store!

Use the scrollbars to the left and bottom of the window to move the image around.

Use of Viewpoint Free-Hand Drawing is described in the *ViewPoint Reference Library*, 610E01030.

Scratchpad

Scratchpad displays a sample character set, and provides a place to type font names, file names, characters, or any other text. Both of its subwindows may be scrolled, split, and edited. Use it as a handy place to select arguments needed for Typefounder commands.



Notes

To type 'upper octal' characters (those with character codes of 200 octal and above) into this window, enter upper octal mode by clicking in any Typefounder window and pressing the "DO IT" key (usually labelled "MARGINS"). This causes 200 octal to be added to the value of characters subsequently typed into any Typefounder window. Hit DO IT again to exit upper octal mode. Example: to type octal 341, hit DO IT and type a lower case "a" (octal 141).

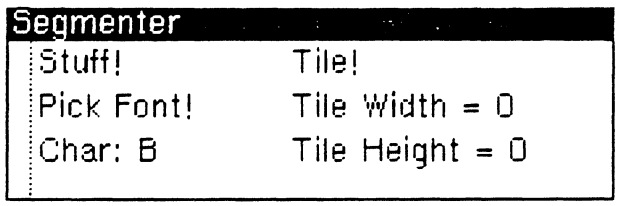
Characters not in the system font display as black rectangles.

Most message subwindows in Typefounder tools also accept type in. It is sometimes more convenient to type file names and other information in the message subwindow of the tool you are using than to move to the Scratchpad or Executive.

(This page intentionally blank)

Segmenter

Segmenter makes raster characters out of rectangular portions of an image displayed in the parent Raster Image window.



Commands

Stuff! makes a character using the image surrounded by the active box of the associated Raster Image Window, and stuffs it into the character and font indicated in the parameters. Confirmation is required to write over any previous version of the character. Char is automatically advanced to the next higher character code.

Tile! stuffs successive fixed size rectangular areas of the image into successive characters in the font. Starting at the upper left corner, a box of dimensions Tile Width x Tile Height is drawn, its contents stuffed into Char, Char is incremented by one, and the box is moved one position to the right (moving down to the next row as needed).

Pick Font! lets you select the font into which the characters will be stuffed, by pointing to it with the mouse. Font file name displays in name stripe.

Parameters

Char: the character into which you want to Stuff. Char recognizes a character, character code, or an alias string.

Tile Width & Height: dimensions of the box used for the Tile command.

Notes

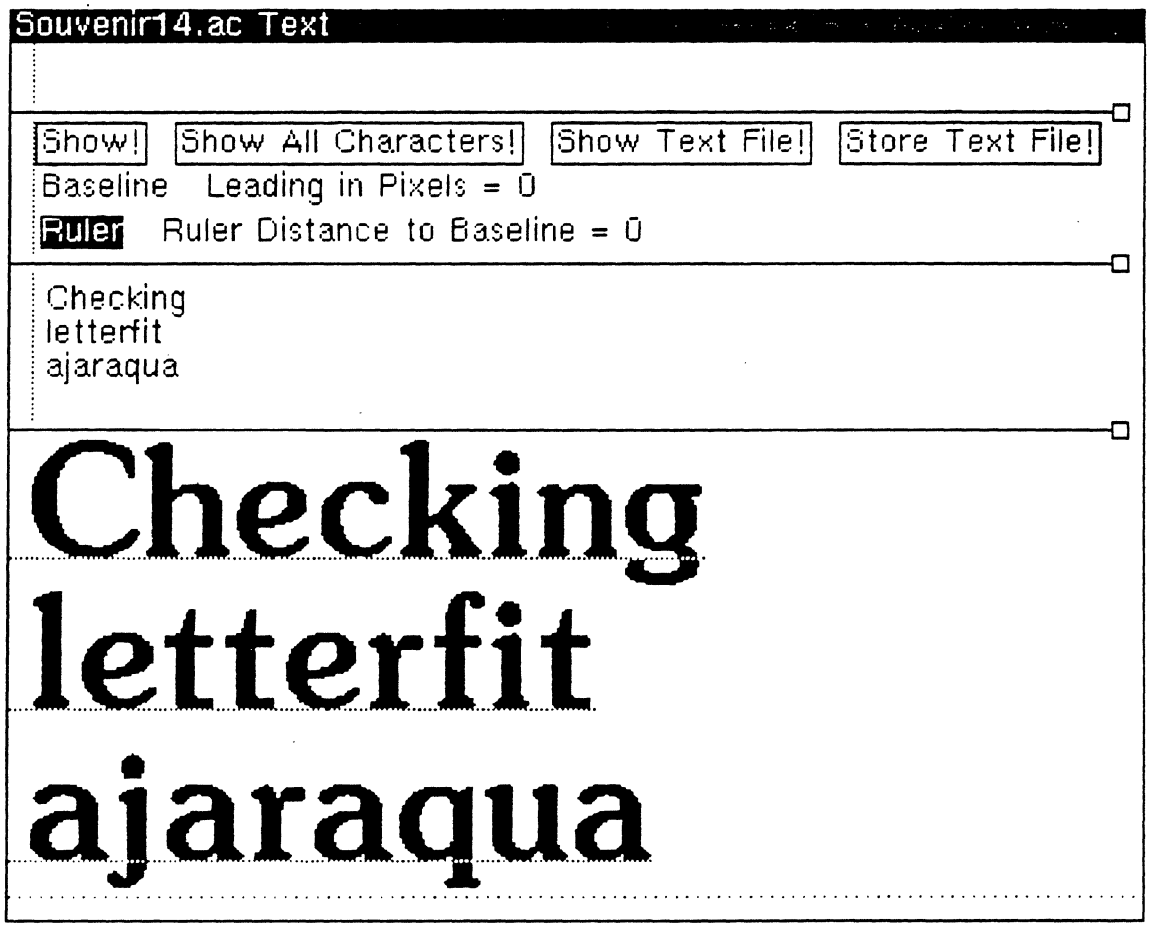
Images are automatically trimmed of excess white margins as part of the stuffing process. Default placement parameters are assigned: left side bearing 1, elevation above baseline 0, width = raster width + 2.

Editing of image and placement parameters may be performed by starting a Character window under the associated font, and Loading the stuffed characters.

(This page intentionally blank)

Text Display

Text Display displays text in the font of the parent window. This is generally used to check a font's appearance.



Commands

Show! displays the text that has been typed or loaded into the third subwindow. The font used is that of the parent font, which must be a raster font. Display begins with the first character visible in the third subwindow. May be canceled with the STOP key.

Show All Characters! displays all the characters in the font. A space is inserted after characters of zero width for easier reading.

Show Text File! displays the contents of the file named by the current selection.

Store Text File! writes the contents of the third subwindow to the file named by the current selection.

Switches

Baseline: when on, baseline and side bearings are displayed as dotted lines.

Ruler: when on, a horizontal line, movable with the horizontal scrollbar, is displayed. This may be used for checking height alignment of characters.

Parameters

Leading inPixels: distance in pixels between text lines. The display is not updated until a Show! command is given.

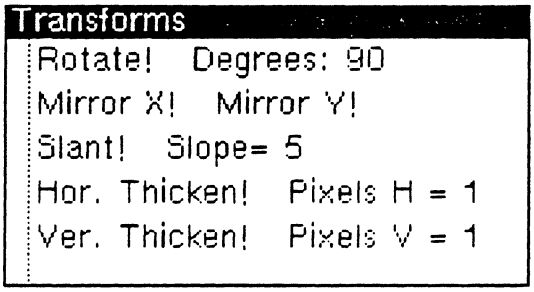
Ruler Distance to Baseline: a display-only parameter indicating the number of pixels above the baseline of the ruler. The ruler is moved by clicking the mouse in the horizontal scrollbar at bottom.

Notes

The third subwindow may be scrolled or made larger to accommodate more text.

Transforms

Transforms performs rotations, reflections, slanting, thickening, and thinning on individual character images in the parent tool.



Commands

Rotate! rotates a character the amount indicated in the Degrees parameter. Click with Point to rotate counter-clockwise, Adjust for clockwise.

Mirror X! reflects a character about the horizontal axis.

Mirror Y! reflects a character about the vertical axis.

Slant! slants a character 1 bit horizontally for every Slope bits vertically.

Hor. Thicken! thickens the image horizontally by the indicated number of pixels. Click with Point to thicken, Adjust to thin.

Ver. Thicken! thickens the image vertically by the indicated number of pixels. Click with Point to thicken, Adjust to thin.

Parameters

Degrees: amount to rotate. This can be positive or negative, and fractional for contour characters. The value is restricted to multiples of 90 for raster characters.

Slope: amount of slant.

Pixels H: amount of horizontal change.

Pixels V: amount of vertical change.

Notes

A single transform made with one mouse button is undone by the other button (except for thinning). This is handy for trying things out.

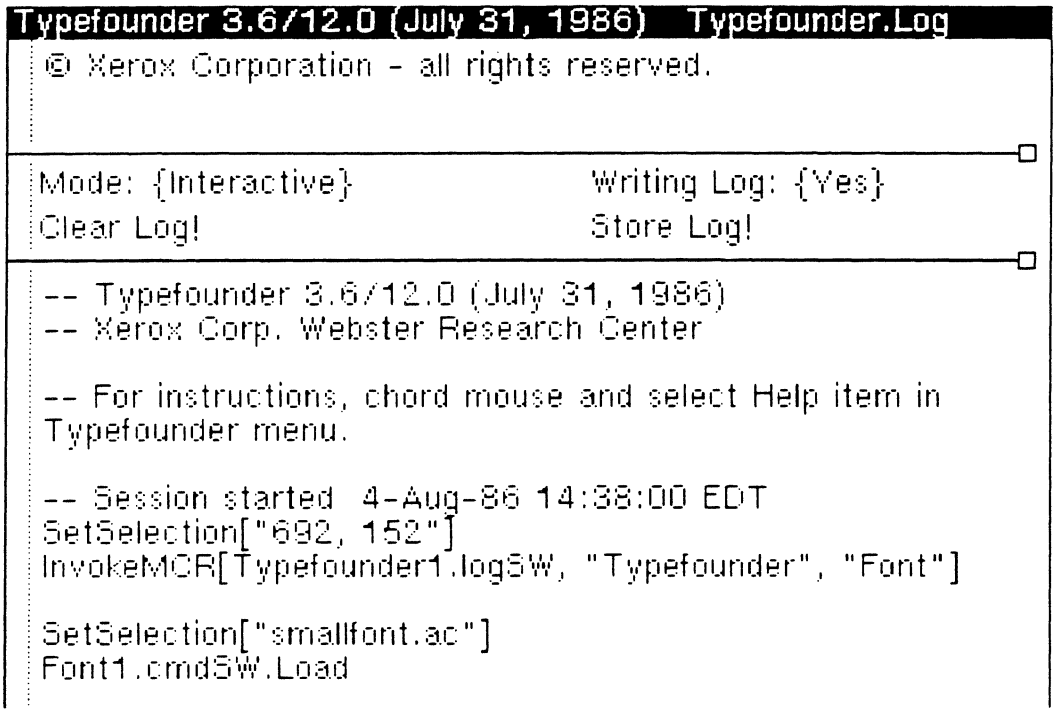
Like other edits, the effects of transforms are not saved unless you do a Save! or Store! command in the character window.

Rotations performed here are independent of font orientation.

Slanting and thickening can be done on entire fonts with the Variations tool.

Typefounder

Typefounder controls operating mode, maintains a log file, and reports certain warning messages.



Commands

Clear Log! erases the current log.

Store log! writes the current log on the file named by the current selection.

Parameters

Mode: overall operating mode of Typefounder. In *Interactive* mode, all tools can operate simultaneously. This is the normal case. In *Teaching ToolDriver* mode, one operation must be completed before another can be started. This mode should be used for creating scripts for execution by ToolDriver. *Running ToolDriver* mode is used for running a script.

Switches

Writing Log: controls whether a log file is written. When on, all commands and parameter changes are recorded in the file Typefounder.log in a form suitable for use with ToolDriver.

Subtools

Barcode: draws bar code images.

Dictionary: builds and lists PrePress-style font dictionaries.

Emergency: use this if a window freezes up. (See Appendix E for more information.)

Font: displays and manipulates a raster, contour, or metrics font.

Font Finder: lists fonts on the local volume.

Raster Image: displays AIS, RES, and other raster format images.

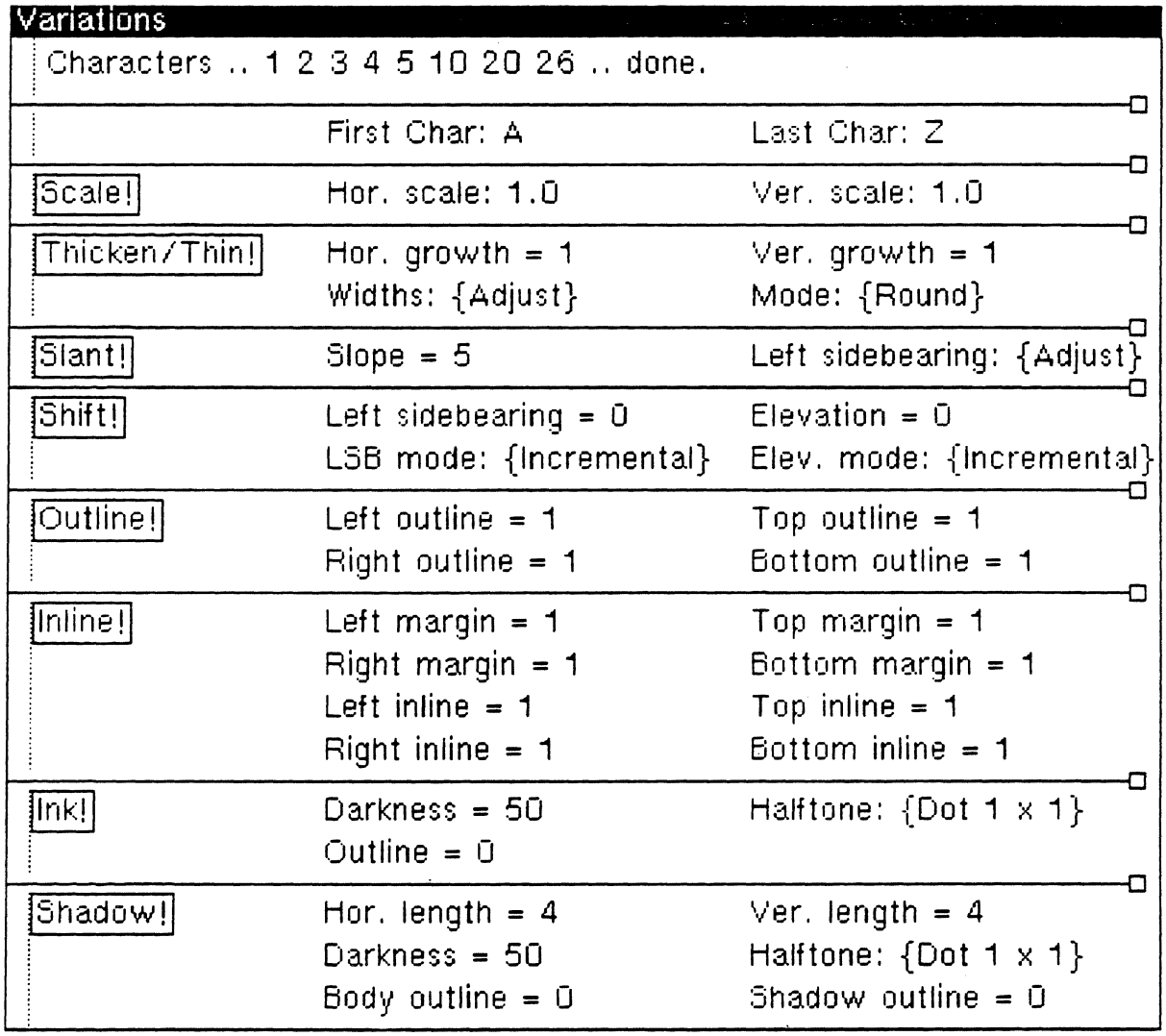
Scratchpad: displays sample character set, and provides a place to type.

Notes

The log produced by this tool can be used with ToolDriver (described in the *XDE User's Guide*) to effect a 'batch' operation, in which commands are taken from a 'script' (the log file) rather than from you. See the section in this manual on "Using Typefounder with Tool Driver". The log may be turned off in Interactive or Running ToolDriver mode to conserve disk space.

Variations

Variations performs design variations, such as thickening or outlining, on some or all of the characters in a font.



Commands

Scale: scales a font by the indicated factors (fractions, such as 1.5, are allowed).

Thicken/Thin: thickens a font horizontally and/or vertically by the indicated number of pixels. Negative numbers cause thinning.

Slant: slants the image one pixel horizontally for every Slope pixels vertically. Slants to right if the slope is positive and to left if the slope is negative.

Shift: changes the left sidebearing and elevation of the characters. In absolute mode, these offsets are equal to the specified numbers. In incremental mode, the specified numbers are added to the existing offsets. Also see the Fitter tool.

Outline: makes outline characters. The outline thicknesses on the four sides are independently controlled.

Inline: makes inline characters. The inline thicknesses on the four sides are independently controlled. The margins between the edge of the character and the inline are similarly controlled.

Ink: paints the characters with a uniform halftone pattern. Darkness ranges from 0 (white) to 100 (black). The style may be a 45-degree dot pattern or horizontal or vertical stripes. Four grain sizes are available. The character is outlined if the outline thickness specified is non-zero.

Shadow: makes the characters with drop shadows. Use a negative lengths to drop down or left. Shadow gray parameters are the same as those of Ink (see above). The shadow is outlined if the outline thickness specified is non-zero. The main character may be outlined independently.

Parameters

Parameters are grouped with the commands that use them, and should be self-explanatory.

Notes

While this tool is generally used on raster fonts, the Shift! and Scale! commands work on contour fonts as well.

The parent font is altered in place instead of being copied to a new Font window, so several operations may be cascaded. If you wish to undo what you have done with this tool, Reset! the parent font to recover the original version.

The Converter tool can also be used to scale fonts.

The Transforms subtool of the Character tool can be used to preview some of these operations on individual characters.

3. Using Typefounder with Tool Driver

The Typefounder log, which is automatically written to the bottom subwindow of the Typefounder window as you use Typefounder tools, can be used as a Tool Driver script to execute Typefounder functions in batch mode. This section describes how to prepare and execute such a script. It also describes how to use a companion tool, Next Item Tool (helpful in constructing scripts with DO loops), and how to modify tool.sws, a file needed by Tool Driver. Finally, two example scripts are discussed. Files referenced in this section are found on the release directory. Tool Driver is discussed in the *XDE User's Guide*.

Preparing a script

1. Start Typefounder.
2. Set Mode to Teaching ToolDriver.
3. Activate windows and perform desired functions. Note the Typefounder log is written as each function is executed. Clean up the screen at the end of the session by destroying all tools except the top-level Typefounder tool.
4. Store the log in the desired filename, using the Store Log! command in Typefounder.
5. Load the log file into a file window and modify it as necessary, Change Teaching to Running where indicated; change filenames or parameters where needed; add loop control as described in the *XDE User's Guide*, under Tool Driver. Be sure the last statement in the script is followed by a carriage return.
6. If the Next Item Tool is to be used in loop control, make an item list file for it.

Running a script

1. Deactivate Typefounder if active.
2. Start Tool Driver.
3. Start Typefounder. This starting sequence is important because ToolDriver must be started before any tool

commands it must recognize, otherwise, the tool is ignored.

4. Set the Script: parameter in Tool Driver to the script name.

5. If the script contains NextItem calls, start Next Item Tool and Load File! the item list file .

6. Click Go! in Tool Driver.

Next item tool

Next Item Tool, an independent tool not part of Typefounder, can be called from a Tool Driver script to augment Tool Driver's simple loop control. NextItemTool has one command, "Next Item!" and a display field called "Current Item". It works with a list of values previously typed into a file known as an item list file. Next Item! steps to the next item in the item list file, displays it in "Current Item" and sets the current selection to it.

Since many Typefounder commands take the current selection as the argument, this provides a way to supply script parameters with a file. The value of "Current Item" can also be referenced in scripts, and may be used to change user-writable parameter values in tools. The item list file contains a list of items which can be numbers, characters, strings, filenames, and so on, separated by a space. No comments are allowed in the file. The end-of-list condition causes a message which may be checked in the script with the ToolDriver LastMessage statement. Use of Next Item Tool is demonstrated in the second example, which also includes a sample item list.

Tool.sws

A list of tools and subwindow names which Tool Driver recognizes is contained in Tool.sws. This file must be on your volume if you are using Tool Driver. Its format is described in the Tool Driver section of the *XDE User's Guide*.

This version of tool.sws allows you to have up to three instances of each Typefounder tool active at a time. If your task requires more than three instances of a tool, you can modify tool.sws to allow the number of instances you require. Add [toolName#] sw1, sw2, sw3 as necessary using the first entry for that tool as a guide. The carriage return between each tool section is required. For example, if you need four Fonts, add [Font4] msgSW, cmdSW, paramSW, formatSW to the list. Tool Driver and any tools it will be "driving" must be deactivated and reactivated for the revised Tool.sws to take effect.

Example 1 - making a printing resolution font from a master

The following example may also be found on Sample.script. This script was made by performing the following actions in Teaching Tool Driver mode in Typefounder and saving the log in "Sample.script." Some comments were added later. A Font is started and a master spline font is loaded. A Converter is started and its parameters are set to make a 10 point font at 300 spots per inch without a template font. After the printing resolution font is made, it is stored in a file. A Proofer is started, a text file is loaded and a proof of the new font is made. It is easy to modify the script to make a 12 point at 300 spi font from the same master font: just change the two fields indicated by the comments in the script from 10 to 12. Before you run the script, retrieve Xerox.text, the proof text.

```
-- Sample.script
-- Typefounder 3.6
-- Xerox Corp. Webster Research Center

-- Session started 18-Mar-86 16:08:22 EDT
--Before running this script in ToolDriver change 'Teaching ' to 'Running ' in following statement.
Typefounder1.cmdSW.Mode ← "Running ToolDriver" -- 'Teaching' changed to 'Running '
SetSelection["367, 31"]
InvokeMCR[Typefounder1.logSW, "Typefounder", "Font"]

SetSelection["XeroxLogo.sd"]
Font1.cmdSW.Load

SetSelection["13, 317"]
InvokeMCR[Font1.formatSW, "Typefounder", "Converter"]

-- Set up the parameters to make a 10 point font and scan convert.
Converter1.paramSW.Pointsize ← "10" -- Change the 10 to 12.
Converter1.paramSW.Hor'.' Thickening ← "0.5"
Converter1.paramSW.Ver'.' Thickening ← "0.5"
Converter1.cmdSW.Convert

-- Store the new font.
SetSelection["XeroxLogo10x300.ac"] -- Change the 10 to 12.
Font2.paramSW.Type ← "raster"
Font2.formatSW.File' Format ← "AC"
Font2.cmdSW.Store

-- Make a proof of the font
SetSelection["364, 529"]
InvokeMCR[Font2.formatSW, "Typefounder", "Proofer"]

Proofer1.paramSW.Leading' in' Bits' ← 10
SetSelection["Xerox.text"]
Proofer1.cmdSW.Load' Text' File

Proofer1.cmdSW.Proof

-- Clean up the screen
InvokeMCR[Font1.formatSW, "Typefounder", "Destroy"]
```

Example 2 - Loops

The following example may also be found on SampleLoop.script. The script from the above example was modified to make a series of printing resolution fonts from the same master font using loop control. This example also illustrates use of Next Item Tool. The added statements are marked by comments. Before you run this script be sure to start NextItemTool and Load File! item. list from Item.list and retrieve Xerox.text.

```
-- SampleLoop.script
-- Typefounder 3.6
-- Xerox Corp. Webster Research Center

-- Session started 18-Mar-1986 16:08:22 EDT
-- Before running this script in ToolDriver change Teaching to Running in following statement.
Typefounder1.cmdSW.Mode ← "Running ToolDriver" -- "Teaching" changed to "Running"

SetSelection["367, 31"-]
InvokeMCR[Typefounder1.logSW, "Typefounder", "Font"]

SetSelection["XeroxLogo.sd"]
Font1.cmdSW.Load

SetSelection["13, 317"]
InvokeMCR[Font1.formatSW, "Typefounder", "Converter"]

DO -- New loop control.

-- Set up the parameters to make a font of the desired point size and scan convert.
NextItemTool.cmdSW.Next' Item -- new. Use Next Item Tool to get several different point sizes.
IF LastMessage[NextItemTool.MsgSW] = "End of list." THEN EXITLOOP;
--new. Put the Check for the last item after the first Next Item Tool command inside the loop.

Converter1.paramSW.Pointsize ← NextItemTool.cmdSW.Current' Item -- Changed to use Next
Item Tool.
Converter1.paramSW.Hor'.' Thickening ← "0.5"
Converter1.paramSW.Ver'.' Thickening ← "0.5"
Converter1.cmdSW.Convert

-- Store the new font.
NextItemTool.cmdSW.Next' Item -- Replaces SetSelection[XeroxLogo10x300.ac]. We want a
different file name for each font.
Font2.paramSW.Type ← "raster"
Font2.formatSW.File' Format ← "AC"
Font2.cmdSW.Store

-- Make a proof of the font.
SetSelection["364, 529"]
InvokeMCR[Font2.formatSW, "Typefounder", "Proofer"]

Proofer1.paramSW.Leading' in' Bits' ← 10
SetSelection["Xerox.text"]
Proofer1.cmdSW.Load' Text' File

NextItemTool.cmdSW.Next' Item -- We want a different file name for each proof.
Proofer1.cmdSW.File ← NextItemTool.cmdSW.Current' Item
Proofer1.cmdSW.Proof
```

```
ENDLOOP; -- new

-- Clean up the screen after the last loop.
InvokeMCR[Font1.formatSW, "Typefounder", "Destroy"]

InvokeMCR[Font1.formatSW, "Typefounder", "Destroy"]
```

Sample next item Tool list, Items.list

```
6 XeroxLogo6x300.ac XeroxLogo6x300text.ip
7 XeroxLogo7x300.ac XeroxLogo7x300text.ip
8 XeroxLogo8x300.ac XeroxLogo8x300text.ip
9 XeroxLogo9x300.ac XeroxLogo9x300text.ip
```

Sample text display text file, Xerox.text

```
XEROX
XEROX
XEROX
XEROX
```

Typefounder limitations

Large fonts and images are inherently stressful on workstation resources, such as disk space and virtual memory. Typefounder is designed to continue operation with warning messages if it needs more disk or virtual memory than is available. Operations that result in 'Sorry - out of memory' conditions can sometimes be made to work by deactivating other tools that are not in use. Low disk space conditions can be remedied by deleting files on the disk (or moving them to file servers). Keep a cushion of a few hundred free disk pages when running Typefounder, as both Typefounder and other tools have trouble if virtual memory gets completely exhausted, and disk space is usually needed as backing files for virtual memory.

Images are limited to 32767 by 32767 pixels in size. The Character tool's image subwindow can display at most a portion 1000 pixels wide by 700 high, though larger images can be loaded and edited by using scrolling.

The product of scale factor and image size must be under 32767 to display a raster character.

(This page intentionally blank)

A. A few hints on how to do things

Finding out what is in a font file

Font Finder lists a few useful attributes about all font files on your local volume. To find out more about an individual font, load the font into a font window. To see exactly which characters are in the font, start a Lister and list those attributes of interest. To see the character shapes, bring up a Text Display window, and run Show All Characters. To examine font metrics in detail, start a Metrics window.

Changing the format of a font file (like AC to Strike)

Load the font into a font window, chord the mouse over the Format item and select the new format (like "strike"), type (in the Scratchpad or Executive window or message subwindow) a new name for the strike file, select it, and click Store! in the Font window.

Constructing a master font or log from artwork

First scan the artwork on a scanner to produce an AIS or compressed AIS image file. Load that file onto your workstation, start a Raster Image window, and load it with the image file. Start a new font window and click Edit! in it. Start a Segmenter in the Raster Image window, and pick the new font as destination for the characters. By holding down Point (the left mouse button) in the displayed image, draw a box around the first character in the image (include tic marks indicating baseline or side bearings, if present). Type the character name for this part of the scanned image into the Char: field of the Segmenter, and click Stuff!. Repeat box drawing and stuffing until all the characters are isolated.

In the new font, start a Character window, and click Next! to get to the first character. Turn on baseline markers and editing. Set elevation, left side bearing, and width by dragging the small square markers with the mouse. Use one of the larger editing brushes as an eraser to clean up extraneous marks, Click Trim! to remove any white margins, and Save! to store the cleaned up master character.

When finished, set the appropriate parameters in the Font window and store the new master font in a file.

The Typefounder tutorial and Appendix B both cover this topic - see them for more details and useful hints.

Making a family of printing resolution fonts

This is usually done by building and running a ToolDriver script, as it's generally a time consuming process. The idea is to load either a contour master or a high resolution raster master into a font window, start a Converter, fill in the point size and resolution, sometimes (especially with smaller point sizes) make adjustments to the default conversion constraints, and run Convert! This is repeated for each point size and resolution needed.

Appendix B deals with this task in great detail.

Moving a character from one character code to another in a font

Load the font into a Font window, and bring up a Merger. Under Chars To Be Altered, type the name or character code of the new position of the character in First1, and the name or character code of its current position under Source in Start1. Click Merge! to move the character. When all moves are accomplished, click Save! in the Font window to write the revised font back into the file.

Standard character code assignments are given in the Xerox *Character Code Standard*, Xerox System Integration Standard XSIS 058404.

Using existing characters to build new ones

Since letterforms in a font often have common sub-parts (such as a serif, the curved section of lower case p and q), raster characters can often be built up from pieces of existing characters faster than drawing them with the mouse.

Some characters, like certain parentheses and quotes, are mirror images of other characters. Once the first of each pair is made, the second of the pair is then constructed with the Mirror X! or Mirror Y! commands in the Transforms tool. Adjust the left side bearing and elevation if necessary, and store the result in the correct character code.

Common strokes can be combined into new characters as follows:

1. With a character displayed in the Character window, set the Mouse to "draws box", and draw a box around the stroke of interest.

2. In a second Character window (the one in which you will construct new characters), start a Background window, run Grab!, and point to the first character window. The stroke will show in the background.

3. Set the scale mode to absolute, and Change Scale so that background scale is the same as foreground scale.

4. Move the stroke into position with the Shift! command.

5. Merge it into the character under construction with Add To Foreground!.

Multiple background windows can be used for complex constuctions.

Sometimes it is not even necessary to extract a stroke. Suppose, for example, you wish to build some fractions ($\frac{1}{2}$,$\frac{3}{4}$, etc) for an 18 point font, and an 8 point bold version of the same numerals is already available.

1. Start a character window in the 18 point font, and load it with a hyphen, or draw the horizontal line by hand.

2. Start a font window, and load it with the 8 point bold font.

3. Start a background window, turn off tracking, and use Pick Font! to set the background font to the 8 point bold.

4. Select a "1", and use Load! to bring up the one as background. Use Shift! to move it into position as a numerator and Add To Foreground! to paint it into the character.

5. Repeat, using "2" as a background and moving it below the line. Touch up by bit editing if neccessary, and store as character code 275. You now have the $\frac{1}{2}$; other fractions can be built in similar fashion.

Drawing character shapes by hand

Raster characters can be drawn from scratch with the pixel editor in the Character tool (though it's usually easier to start with an existing character and modify it). You'll find that editing goes quicker if you block out the area you wish to work in by making black pixels in opposite corners (this is because there is a noticable delay if your edits expand an existing raster image. Expanding it once at the start is faster). The larger edit brushes are helpful in roughing out the design.

Contour characters cannot currently be started from scratch. Instead, load an existing contour character and delete all its knots. You can then begin adding new ones.

A sophisticated painting and drawing program is available in Viewpoint in the Free-Hand Drawing feature. See the *ViewPoint Series Reference Library*, 610E01030. Free-hand drawing canvas produced with this illustrator program can be read by Typefounder through the Raster Image tool.

B. Making fonts by scanning artwork

This appendix discusses the process of making digital printing fonts with Typefounder by scanning artwork. Digitized character images from a scanned image are arranged into a font file and edited where necessary to clean up stray bits. These master characters are aligned on a baseline and assigned widths. After the master fonts are proofed and checked for accuracy, the printing resolution size fonts are made using ToolDriver to run Typefounder in batch mode. The printing fonts are also proofed and finally, installed on the printers.

The artwork

Artwork, or photographs of the characters, may be obtained from commercial typeface vendors such as International Typeface Corporation or Merganthaler, or from in-house graphic arts groups, or from books of public domain typefaces.

The input scanner used to digitize the images should have sufficient resolution to capture enough detail in the characters. The master font's em size (pixels per em) provides a good measure of character detail. To calculate the em size of a font multiply the resolution of the scanner in spots per inch by the point size of the artwork divided by 72. To make printing fonts up to 36 points at resolutions up to 1200 spots per inch (spi), a master font em size of about 600 is adequate. Master characters much larger than that become unwieldy and do not improve the resulting printing fonts appreciably. If the resolution of the scanner is too low, the artwork should be photographically enlarged. Conversely, if the resolution is too high, the artwork should be photographically reduced.

It is very important to carefully align the artwork on the scanner. Small angular misalignments in the master font can be magnified when converted to the printing resolution sizes. Also it is difficult to determine the baseline location of tilted characters.

Some scanners produce 8 bit per pixel (gray scale) digital images. Since Typefounder uses 1 bit per pixel images the resulting images must be thresholded before they are usable. The digitized artwork should be converted to one of the formats read by the Raster Image Tool. CIS format is recommended because it saves a considerable amount of file space.

Depending upon the source of the artwork, various amounts of character metric information, such as baseline and side bearing locations, may be supplied. For example, some artwork may

have no information, obliging you to set side bearings and baselines by eye, or the characters may have tick marks for these metrics as well as width tables and kerning tables.

The master font

After the character photographs are digitized, preparation of the digital font master begins. Making the master is the most time consuming task, yet the most important part of the process. This section describes the three steps of making the master: assembling the digitized characters into a font, assigning character metrics, and proofing. The last two are repeated until the appearance of the master is satisfactory.

Assembling the font

First, the characters in the digitized artwork must be assembled into a font. A digital font is a collection of digitized character pictures, each with a numeric identifier known as a character code. In assembling the font, each character picture must be associated with an octal character code and stored into a font file. This is done by loading the scanned image into a Raster Image window, drawing boxes around individual characters, and using Segmenter to Stuff them into a new Font window.

A suggested mapping between character pictures and character codes is presented in the XC1-1-1 *Character Code Standard* Xerox System Integration Standard XSIS 058404. Each font is made up of one or more character sets consisting of up to 189 characters each. There may be as many as 256 character sets in a font; but, in practice, there are usually fewer than 10. Character set 0 is usually the main character set of the font. It contains the alphabet, punctuation, and numerals, most of the characters of a standard typewriter keyboard. Refer to the charts in the *Character Code Standard* while assembling the font with the Segmenter to determine which octal number to use for each character.

Shown below is a sample basic character collection, containing 115 characters. This collection is essentially the union of ASCII (for computer terminals) and the characters on the XDE and Viewpoint keyboards, plus some useful dashes and spaces.

Character Set 0

Symbol	ASCII	XC1-1-1	Description
		40	word space, w = 1/3 em or 1/3 M
!	41	41	exclamation
"	42	42	neutral double quote (XDE keyboard)
#	43	43	number sign
%	45	45	percent sign
&	46	46	ampersand
'	47	47	apostrophe (XDE keyboard)
(	50	50	open parenthesis

)	51	51	close parenthesis
*	52	52	asterisk
+	53	53	plus sign
,	54	54	comma
-		55	neutral dash, w between hyphen and minus
.	56	56	period
/	57	57	slash
0-9	60-71	60-71	numerals (all the same width)
:	72	72	colon
;	73	73	semicolon
<	74	74	less than
=	75	75	equals
>	76	76	greater than
?	77	77	question mark
@	100	100	commercial at
A-Z	101-132	101-132	upper case roman characters
[	133	133	open bracket
\	134	134	back slash (XDE keyboard)
]	135	135	close bracket
^		136	circumflex
		137	low bar
`	140	140	grave, spacing (XDE keyboard)
a-z	141-172	141-172	lower case roman characters
{	173	173	open curly brace
|	174	174	vertical bar (XDE keyboard)
}	175	175	close curly brace
~	176	176	tilde, low, spacing (XDE keyboard)
¢		242	cent sign (Viewpoint keyboard)
$	44	244	dollar sign (width same as numerals)
'		251	single open quote (Viewpoint keyboard)
"		252	double open quotes (Viewpoint keyboard)
←	137	254	left arrow (XDE keyboard)
↑	136	255	up arrow (XDE keyboard)
'		271	single close quote (Viewpoint keyboard)
"		272	double close quotes (Viewpoint keyboard)
_		314	underline (Viewpoint keyboard), w = 0, l = 3/4 em

Character Set 41

Symbol	ASCII	XC1-1-1	Description
-		76	hyphen, w = 1/3 em, l = 1/4 em

Character Set 356

Symbol	ASCII	XC1-1-1	Description
		43	hair space, w = 1/9 em
–		55	minus, w = FIGURE, l = 1/2 em
~		176	similar to (same as 0|176)

Character Set 357

Symbol	ASCII	XC1-1-1	Description
-		43	discretionary hyphen, same as hyphen
–		44	en dash, w = 1/2 em, l = 1/3
—		45	em dash, w = em, l = 3/4 em
‒		46	figure dash, w = figure, l > 3/4w, or l = w
'		47	neutral single quote

54	en quad (space), w = 1/2 em
55	em quad (space), w = em
56	figure space, w = figure
57	thin space, w = 1/5 em

Character Set 360

Symbol	ASCII	XC1-1-1	Description
		312	substitution char, black rectangle, w = space, h = cap

After the characters are put in a font with the Segmenter, the parameters shown in the Font window should be filled in. Use the full family name of the font for Family Name (no spaces). Point size should be the point size of the artwork. The resolution should be set to the resolution at which the fonts were scanned. **Note:** for character sets other than 0, enter the character set in octal, followed by a "b" in the vertical resolution field. Set the appropriate face information and make the encoding XC1. Compressed OC is suggested for File Format to save space and Orientation A3 (90) to save file loading time. Now select a name for the font file and Store the font.

If the artwork did not include all the characters in the character collection, it is now time to construct (or adapt from other fonts) the missing characters. Some are most easily made from existing characters by using the bit editor in the Character Tool. For example, the em and en dashes can be made by extending the minus. Other characters which are difficult to make, such as the commercial at (@) or braces, can be taken from an existing font and modified slightly if necessary to complement the design. If this approach is taken, follow the typographic industry standards below.

1. Certain characters should not be italicized, but should be emboldened to match the weight of the font. These are:

@	commercial at
®	registered
©	copyright
/	slash
\	backslash
~	tilde
=	equals
+	plus
-	minus
<	less than
>	greater than
→←↑↓	arrows
|	vertical bar

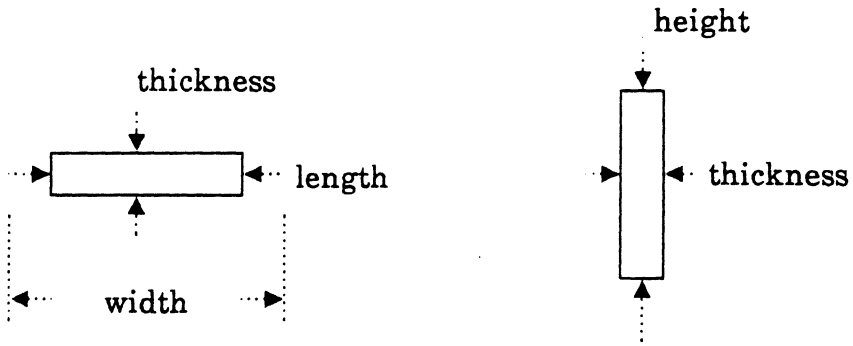
2. The mathematical characters, plus, equal, greater than, less than, minus, times, and divide, normally have the same width as the figures.

3. One design for the commercial at sign is normally used for several similar fonts, but the weight is matched to the weight of the font. Therefore, only two commercial at

signs are needed, one for serif and one for sans serif for a variety of fonts.

4. The widths of the dashes, em, en, and figure dashes, are such that several consecutive dashes do not make a solid line. Special form characters are used for making solid lines.

Use the following figure and the dimensions given in the above character list to determine the dimensions of dashes for dashes not provided with the artwork.



All the dashes, the minus, and the underline characters have the same thickness as the crossbar of the plus sign. The dashes and minus all are the same height above the baseline. The top of the hyphen may be slightly higher than the top of the dashes. The math characters, =, –, +, (equal, minus, plus) have the same horizontal stroke thickness, length and width. The length of the minus is the same as the length of the crossbar of the plus sign, and its width is the same as the width of the plus sign. The hyphen is usually thicker than the dashes, and is centered in the x height (whereas dashes are centered in the cap height). The vertical bar has the same thickness and height as the slashes. The em is defined as the point size in pixels at the specified resolution.

Assigning character metrics

This section describes the methods used to assign the baselines, left side bearings, and widths to the characters. The technique used depends on the metric information supplied with the artwork. The Character and Background are the main Typefounder tools used.

Scanned character images often contain stray bits caused by dirt or noise in the scanning process. It is important to clean these up before proceeding. The large bit editing brush can be used for erasures. The stray bits are easier to find if the character Color is Black. The Trim command and Boundary Markers can be used to check for leftover bits. There should be no white border around the character within the markers.

Some artwork contains baseline, side bearing, and width information. The baseline and one or both side bearings are marked on the artwork as tick marks next to the character. To enter this information into the font, Load the character into a

Typefounder Character window and turn on Mouse Edits and Baseline Markers. Grab the small boxes at the end of the baseline or side bearing with the mouse and move the line until it coincides with the tick mark in the image. Usually the tick marks are more than one pixel wide and may not even be uniformly wide. For the baseline, use the top edge of the tick mark as the location of the baseline unless that row has only a few pixels in it, in which case use the next row. Use the rightmost edge for the side bearings and the same caveat. It is helpful to have a scale greater than one for this process.

Sometimes widths are supplied in tables. The values in the tables are in fixed units, eighteenths of an em, for example. A one em wide grid divided into eighteenths is provided on each sheet of the artwork (sometimes each unit is further subdivided). Determine how many pixels are in an em using the width of the grid in pixels. If all the grids are not the same width, use the average. Divide the width by 18 and round to the nearest integer. The result is the number of pixels for each width unit.

To specify the width of a character if the right side bearing only is marked, put the grid into an empty character location using the Segmenter. Load the grid character into a Background and turn off Tracking so the grid character will remain in the background as foreground characters are changed. Load the character whose width is to be specified into a Character window. Be sure the background and foreground scales are the same. Use the Background Shift command to align the 0 position on the grid with the tick mark on the character. Look up in the table the width for that character and drag the side bearing to coincide with the proper grid division. Be sure the width in pixels, shown in the third subwindow of the Character window, is the number of units in the table multiplied by the number of pixels per unit.

If only the left side bearing of the character is marked, the procedure is less complicated. Simply change the Width parameter in the third subwindow to be the number of units in the table multiplied by the number of pixels per unit.

The grid can also be used to check that the em size determined by the point size and resolution of the font is correct. The value for Horizontal Em Size shown in the Metrics for the Font should be the same as the width in pixels of a grid unit multiplied by 18. If it is not, the point size or the resolution of the font should be changed so the em size and grid unit size correspond. As the width for each character is specified, the tick marks may be removed by editing as described above.

Other artwork may not include tick marks to specify the location of the baseline and side bearings or the width of the character. Phototypeset artwork may have a rule drawn under the lines of the characters, the same distance from the baseline for every line of characters. If this is the case, as each character is put into the font, include part of that rule in the box drawn around the character. The Segmenter automatically sets the baseline to the bottom most black bit in the character, which should be the bottom of the rule. Check the characters for stray bits below the rule and remove them. An easy way to do this is to turn Boundary Markers on and check that the bottom boundary coincides with the bottom of the rule. Determine the distance from the bottom of the rule to be the true

baseline using the upper case H. Use the Variations tool to Shift the baseline up this amount for all the characters.

Any tick marks or grids used in assigning metrics should be removed from the character images after use.

If no baseline information is included with the artwork, the baselines for character set 0 may be set with the Align Baseline command of the Fitter tool. It is sometimes necessary to refine the results by manually setting baseline alignment with the Character Tool. Use editing mode with Metrics Markers, and drag the baseline until alignment is correct. The Text Display may be used to show several characters at once to facilitate this process.

Spacing the font (setting character widths), is usually done with the Fitter's Center & Space command for character set 0. Again, it is important that any baseline marks and stray bits be removed. It is also important that the correct point size and resolution be specified for the font before this is done, since the algorithm uses this information. Spacing can take up to an hour for large fonts. After Center & Spacing the font, the Set Fixed Pitch command may be used on the numbers and math symbols to make them all the same width. Widths for character sets other than zero are set by hand.

For characters included in more than one character code, (for instance, in our suggested character collection the hyphen, discretionary hypen, circumflex, and tilde) enter these characters into one character set and use all the alignment tools first. When the character is spaced and aligned correctly, use the Merger to copy it to its second location.

Proofing the master fonts

After the baselines and widths are set for all the characters, the master fonts must be proofed on paper to evaluate overall baseline alignment and character fit. Baseline alignment is especially critical because small misalignments at the master size translate to noticeable misalignments at the printing resolution sizes. Use the Proofer to make charts or text samples of the masters. The Load All Chars command can be used to automatically get a list of all the non-empty characters in the font. The optional Baseline and Side Bearings, Cap Height Line, and x Height Line features are useful for checking baseline alignment and character widths. Because Proofer output can take a long time to print, try to keep the prints under two pages long.

It is also useful to make a sample printing resolution font to check for baseline misalignments. Use the Converter specifying 14 for the Point Size. Save the new font and use the Proofer without Baseline, x-Height Line or Cap Height Line to print some text in the font.

Master fonts are the most important part of font production, especially if the printing fonts are not bit edited. Check the master fonts to be sure the character codes are correct, that they include at least the characters in your basic character collection, and that they follow typographic industry standards

where necessary. Also check the baseline alignment and character fit carefully.

Making the contour master

In most cases, it's desirable to convert the completed raster master to a contour master to provide higher effective resolution and more accurate automatic half-bitting when converting to the printing sizes. This is especially true if the scanner resolution is not high enough to produce characters of em size 600. Use the Converter with Type set to contour (the other parameters may be left at default settings). It takes about 1.5 hours to convert a typical font of em size 500 with 150 characters. In some cases, it may be necessary to edit the contour characters with the contour editor in the Character window.

The printing resolution fonts

Once the master fonts are completed, the Converter is used to make a complement of printing fonts. If many point sizes and faces are involved, converting the fonts is a time consuming process and the printing fonts are often made using the XDE ToolDriver to run the Typefounder tools in batch mode overnight. The ToolDriver uses a file called a script which lists the commands to be executed (see the section Using Typefounder with ToolDriver). Most Typefounder operations, with the exception of bit editing, can be performed with ToolDriver.

In general, the default values of the Converter parameters are appropriate. However, when converting certain styles of fonts or fonts for a specific printer, it may be desirable to modify those values. For example, when converting a very high contrast font, such as one with very thin horizontal strokes, the Minimum Horizontal Stroke Width parameter might be adjusted. If your printer is known to erode strokes, thickening might be applied in the horizontal or vertical directions or both. The values for these parameters are best determined by converting the master to one point size several times using different values for each time and evaluating the results.

When converting from contour masters, convert character set 0 before other character sets of the same font. The converter needs metrics obtained during processing of character set 0 to correctly process other character sets. For character sets other than 0, edit the master font before the Converter is started and set the Character Set field to the character set number (because the Converter uses this information to determine certain parameters during the conversion and it is not stored in the contour font file). After the conversion is complete, the master font can be reset.

After the printing fonts are made they are proofed. The following sections describe the sizes of printing fonts to make,

how to make the widths files needed for various text editors, and how to proof and install 300 spi fonts.

The printing resolution sizes

A common set of point sizes is shown below with the corresponding mica sizes. Because the printers use mica sizes for determining the font size to use in printing a document, specify mica size for more accuracy in the Converter.

Points	Micas
5	176
6	212
7	247
8	282
9	318
10	353
11	388
12	423
13	459
14	494
16	564
18	635
24	847
30	1058
36	1270

About 2,000 free disk pages are required to make the 300 spi fonts for a typical font (one face) of 150 characters. This includes the space required for the contour master font files. It takes about 8.5 hours to convert to the 300 spi fonts from contour masters, or about 2 hours to convert from raster masters.

Widths files

Certain text editing programs require printer widths files in order to calculate justification and line breaks. If the fonts you are making are for such editors, you must make a widths file. Generally, only one widths file is necessary for each font family. It contains the widths for all the character sets of all the faces of a family. The widths file for a single character set of a single face of a font family can be made from the master font using the Converter by specifying Type metrics and Units milliems. Be sure the orientation of the widths font is A8. The widths files for one family can be packaged together in a dictionary using the Merge command in the Font window. Load the first widths file into the Font window, select the dictionary name and click Store. Load the subsequent widths files into the Font window, select the dictionary name and click Merge. The ToolDriver can be used to make the widths dictionary in batch mode.

XDE volumes usually contain a special file called fonts.widths which contains the widths files for all the fonts on local printers. If the printing fonts are usable from XDE, merge with character widths into fonts.widths as well.

Proofing and installing 300 SPI fonts

It can be difficult to make good quality 300 spi fonts automatically because of their relatively low resolution (compared to metal type). Therefore, several cycles may be necessary, varying the parameters of the Converter to obtain optimum results. The parameters most often varied are the horizontal and vertical thickening.

Proofing the new fonts is easiest with the Proofer tool. Make up suitable proof text in XDE files, start a proofer under the font to be proofed, and use Load Text File to load the text. The Proof command then produces an interpress file, which can be printed with the Print utility. Proof text might look like the following:

> Modern
> 10 point
> !"#%&'()* + ,-./0123456789:;< = >?
> @ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^
> 'abcdefghijklmnopqrstuvwxyz{|}~
> öåüçƒ‡¶
> Excellence in typography is the result of nothing more than an attitude. Its appeal comes from the understanding used in its planning: the designer must care. In contemporary advertising the perfect integration of design elements often demands unorthodox typography. It may require the use of compact spacing, minus leading, unusual sizes and weights; whatever is needed to improve appearance and impact.

When multiple character sets are involved, it sometimes saves time to merge all the characters into another temporary font. Then the Proofer can be used to make an interpress file of all the characters in one run.

To install fonts on Xerox 8044 print servers, the fonts must be compiled into a dictionary. Use the Dictionary window Build Dictionary command to merge AC files of all the point sizes and character sets of one face of a font family into a single dictionary. For fonts used within Xerox, the dictionary name should follow this pattern : Xerox.XC1-1-1.*Family.Face*, such as Xerox.XC1-1-1.Souvenir.Bold.Italic. Set New Dict. Format to prepress, select a list containing the names of all the AC files to be merged into the dictionary and click Build Dictionary. Expanding a file name pattern in the Executive window is often a handy way to produce the list of AC files. The Build operation typically takes about 15 minutes for a dictionary containing 18 point sizes and 5 character sets in each size. Write the dictionaries to a floppy disk by typing in the Executive window:

```
Floppy Write 4290/t dictionaryName
```

Take the floppy to the printer, enter administrative mode by typing Enable (you may have to enter a password; see your system administrator) and install the font by typing Install From Floppy.

C. Creating a logo

If you are using Typefounder to create logos to use in ViewPoint, follow the procedures below. These procedures assume you are familiar with the basic operations of the Font, Raster Image, Segmenter, Variations, and Character windows. If you aren't familiar with them, go through the Typefounder online tutorial first.

Logos for ViewPoint can be created from existing XDE brush files (XDE brush software is available on your Unsupported/Data release directory), ViewPoint Freehand drawing canvases, or from any scanned image in AIS format. The procedures assume you have made the logo by one of these methods.

Four different files must be made to transform a logo image into a ViewPoint logo:

1. A **font file** for the 8044 printer so a ViewPoint document containing the logo can be printed.

2. A **screen display font file** in AC format so the logo can be displayed in a document window.

3. A **widths file** containing width information for the screen display font.

4. A **novaFont file** containing the actual logo.

These files are created in the order listed because they build from each other. The screen display font file and widths file is generated from the printer font file, and the novaFontfile is generated from the screen display font file and widths file.

Making the font file for the 8044 printer

To work with the logo image file, it must be loaded into XDE and set up in Typefounder. To do this:

1. Start a Font window from the Typefounder window.

2. Start a Raster Image window from the Typefounder window.

3. Load the brush file or Freehand drawing canvas into the Raster Image window by doing the following:

   a. type in the filname of the brush file/canvas and select it.

   b. select Load! in the Raster Image window.

Now that the logo image is loaded in Typefounder, it must be associated with a keyboard character and a Font window. To do this:

1. Start a Segmenter window from the Raster Image window.

2. Select Pick Font! in the Segmenter window and point to the Font window.

3. Enter a keyboard character in the Char: field of the Segmenter window.

4. If you don't want the entire logo image in the Raster Image window, then draw a box around the portion of the image you would like for the logo. You don't have to draw a box around the image if you want the whole image.

5. Select Edit! in the Font window.

6. Select Stuff! in the Segmenter window.

The next steps involve setting the parameters of the Font window for the logo and storing the printer font:

1. Calculate the Pointsize for the font: pt = rasterheight/300 * 72. You can find the raster height by starting a Character window from the font window and loading the keyboard character you assigned earlier. **Note:** The point size for a ViewPoint font or logo must be less than 99.

2. Enter the following information in the Font window:

Family: full family name of the font (such as Logo1)	
Character Set = usually 0B	Weight: medium
Pointsize: value from Step 1	Posture: roman
Size in micas =	Setwidth: regular
Hor. Resolution = 300	Coding: XC1
Ver Resolution = 300	
Type: raster	Units: pixels
File Format: AC	Scan Mode: A8(0 deg)

3. Type in the name of the new AC file (such as Logo1Printer.ac), select the text, and select the Store! command in the Font window.

The logo height must be scaled within the range of 250 to 350 for the 8044 printer. The Variations window's Scale! command can be used to change the size of the logo image. If you need to correct the scale, do the following:

1. Start a Variations window from the Font window.

2. Select Edit! in the Font window.

3. Change the Hor. and Ver. scale fields:

   a. Calculate the scale value with **one** of these formulas:

   - scale = (desired raster height)/(current raster height)
   - scale = (desired point size)/(current point size)

b. Enter the number calculated in step a. into Hor. and Ver. scale fields.

4. Select the Scale! command in the Variations window.

Reload the changed image into a Character window to see your changes. Repeat the above steps until the logo image is scaled correctly.

Note: Any Scale! operation from the Variations window operates on the image currently loaded in the Character window. If you need to repeat the Scale! command, the last scaled image is used unless you re-load the Font.

Once the image scale is correct it may need touching up a bit. Typefounder's Character window provides a simple bitmap editor to do this. Do the next steps if your logo image needs touching up:

1. Select Edit! in the font window.

2. Start a Character window from the Font window.

3. Load! the keyboard character which the brush or canvas was assigned to in an earlier step.

4. Touch up the image in the Character window as needed.

5. Select Save! in the Character window.

A printer font's point size must be fairly accurate. A more accurate point size can be obtained by calculating the size in micas. When the Size in micas field in the Font window is filled in, Typefounder automatically fills in the Pointsize accurate to two decimal places. Be sure to resave the printer font after changing the point size. Do the following:

1. Select Edit! in the font window.

2. Use the following formula to calculate the Size in micas for the printer font:

   Size in micas = (point size rounded to an integer)*(2540/72)

3. Enter the Size in micas into the Font window (the Pointsize will automatically be entered), and Save! the printer font.

Now the printer font must be "packaged" so it can be loaded on the 8044 printer. Follow these steps:

1. If there is only one printer font, use the Rename.~ command in the Executive window to rename the print font file to match the form, Xerox.XC1-1-1.*Family*. For example, Xerox.XC1-1-1.Logo1.

If there is more than one printer font, do the following:

1. Start a Dictionary tool from the Typefounder window, or use one which is already on the XDE desktop.

2. Change the New Dict. Format to PrePress.

3. Enter a name for the printer font in the New Dictionary File: field. **Note**: The dictionary name should follow the pattern:

   Xerox.XC1-1-1.*Family*, where *Family* matches the family name defined in the Font window.

4. Type the names of the printer font files (such as Logo1Printer.ac, Logo2Printer.ac, and so on) into a window and select the names.

5. Select the Build! command to create the dictionary of printer fonts.

You're now ready to write the printer font or dictionary of fonts to a floppy disk with the Executive command:

Floppy write 4290/t Xerox.XC1-1-1 .*Family*

Finally, the printer font or dictionary of fonts must be loaded and cataloged on the 8044 printer. Take the floppy disk to a printer and load the new logo font or dictionary as follows:

1. Insert the floppy disk into the printer's floppy drive.

2. Logon to the Print Service. A System Administrator must be logged on to load fonts. See your administrator for a name and password.

3. Enter the following commands:

   a. Enable

   b. Print Service

   c. Stop Printing (enter a reason when prompted)

   d. Install From Floppy (the files on the floppy will be listed one at a time. Enter a Y after each file to be loaded as a printer font).

   e. Start Printing (a font re-cataloging operation occurs, after which the new fonts are available for use).

   f. Logoff the Print Service.

Making the screen display font file

The screen display font file is generated from the printer font file. To do this:

1. Load! the printer font (such as Logo1Printer.ac) into a Font window, and select Edit!.

2. Use the Variations Tool to modify the Hor. scale: and Ver. scale: fields to make the Raster width less than 256 and the

Raster height less than 99. Modify the image by doing the following:

a. Start a Variations window from the Font window.

b. Change the Hor. and Ver. scale: fields by entering the number from this calculation:

scale = 6/25 (same as 72/300)

c. Select the Scale! command in the Variations window.

Reload the image in a character window to view the changes. Repeat these steps until the image is scaled correctly.

The resolution fields are the only parameters which differ from parameters in the printer font file. Change the Hor. and Ver. Resolution fields in the Font window to equal 72 and leave the rest of the fields the same.

If you need to touch up the logo image, do the following:

1. Select Edit! it in the Font window.
2. Start a Character window from the Font window.
3. Load! the keyboard character which the brush or canvas was assigned to earlier.
4. Touch up the image in the Character window as needed.
5. Select Save! in the Character window.

Finally, the new screen display font must be stored in a file for later use:

1. Make sure the Size in micas listed in the Font window is the same as the Size in micas for the printer font (Logo1Printer.ac).
2. Store! the screen font into a new file, such as Logo1.ac.

If you have more than one printer font file, follow the above steps for each file.

Making the widths file

The widths file is also generated from the printer font file. The File Format field is the only field which needs to be changed. Remember to store the widths file under a new name.

1. Load! the printer font (such as Logo1Printer.ac) into a Font window.
2. Select Edit! in the Font window.
3. Change the File Format: to WD (absolute).

4. Store! the new widths file under a new name, such as Logo1.wd. When the widths file is stored, the following fields will automatically be changed:

   Hor. and Ver. Resolution = 0
   Type: metrics Units: micas

Making the nova font file for ViewPoint

The screen display and widths files must be packaged to create a file which can be loaded into ViewPoint. This is the novaFont file. Do the following:

1. Start a Dictionary window from the Typefounder Tool window.

2. Change the field New Dict. Format to equal ViewPoint.

3. Turn on the Override Family boolean, and enter the family name, such as Logo1. The family name must correspond to the family name given in the printer, screen and widths font files.

4. Enter a name for the novaFont in the New Dictionary File: field of the form: *name*.novaFont.

Once the parameters for the Dictionary tool have been set, the novaFont file needs to be made:

1. Type the names of the screen font, such as Logo1.ac, and the widths file, such as Logo1.wd into a window and select both names. If using more than one printer font file, type and select the names of all the screen font and width files.

2. Select the command Build Dictionary!.

The novaFont file is now complete and ready to load intoViewPoint. Follow the directions in "Making Screen Font Dictionaries" under the Dictionary section of this manual to load the logo in ViewPoint.

If you have more than one printer font file, follow the above steps for each file.

Using your logo in ViewPoint

Once the printer font file is loaded in the 8044 printer and the novaFont file is loaded into ViewPoint, you can use the logo in a ViewPoint document. Follow these steps:

1. Open a ViewPoint document.

2. Select the position in the document window where you want the logo to be.

3. Press the PROP'S key to bring up the Property Sheet and select the parameters for "character." Display the font menu to find the name of your logo (the name will be the family name you chose when creating the novaFont file), and select it as the current font. Apply this change and then select Done.

4. Enter the logo by pressing the character key you assigned to your logo. For example, if you assigned the letter 'y,' you'd press the Y key to enter your logo into the document at the position selected in Step 1. Your logo displays on the screen, however, you won't see the entire image if the line height for that position isn't large enough to contain the image. If this happens, just select the logo, bring up the Property Sheet, and change the "Line Height" parameter under "paragraph" to accommodate the logo.

5. Switch back to your regular font by selecting that font from the Properties Sheet and applying the change.

(This page intentionally blank)

D. Customizing with TypefounderUser.cm

Typefounder requires a Xerox 8010 or 6085 workstation with at least 768 kilobytes of memory, and the XDE 4.0 Desktop software.

With a special file named TypefounderUser.cm, you can define window places and sizes for Typefounder windows and initialization sequences for starting tools automatically. The format for the file is identical to that of User.cm. TypefounderUser.cm can be left as a separate file or can be merged with User.cm. During loading, Typefounder looks first for TypefounderUser.cm; if it is not present, then it looks in User.cm for a Typefounder section. If neither is present, only the top level Typefounder window appears after loading.

When TypefounderUser.cm is used, loading the TypefounderTool will take much longer than without it, but startup time after loading will be the same.

Each section of the TypefounderUser.cm corresponds to a parent:child tool pair. If the parent tool is not specified, the section applies for every instance of that tool. There must not be a space after the colon. The entries (all of which are optional) are:

WindowBox for defining the location and/or size of the initial window for the subtool. WindowBox locations are relative to the origin of the tool's parent except for Typefounder, whose location is relative to the upper left corner of the screen. Some tools have fixed relative locations and ignore the WindowBox place (Help, for example). The built-in sizes of most windows can be overridden using this entry .

InitialState for suggesting the initial state for the tool (active, inactive, or tiny). The default value is active. See the *XDE User's Guide* for more information on tool states and windows)

TinyPlace for suggesting the location relative to the upper right corner of the screen of the tool window in its tiny state. The size of a tiny box is 30 high by 60 wide.

AutoBoxSelection for automatically choosing the location of the tool window instead of asking you to point. The location is defined in the WindowBox entry. The default value is FALSE. This applies for the first instance of a subtool. Additional instances ask you to select the place.

AutoStartWithParent for specifying that a tool should be started automatically upon activation of its parent tool. The default value is FALSE.

The following is a simple example TypefounderUser.cm.

```
[Typefounder]
    WindowBox: [x: 0, y: 50]
    AutoBoxSelection: TRUE

[Typefounder:Font]
    WindowBox: [x:0, y:160]
    AutoBoxSelection: TRUE

Font:Character]
    WindowBox: [x:0, y:190]
    AutoBoxSelection: TRUE

[Font:TextDisplay]
    WindowBox: [x:350, y:0]
    AutoBoxSelection: TRUE

[Font:Converter]
    WindowBox: [x:0, y:205]
    AutoStartWithParent: TRUE
    AutoBoxSelection: TRUE

[Typefounder:Scratchpad]
    WindowBox: [x:420, y:0, w:410, h:150]
    AutoStartWithParent: TRUE
```

E. What to do if Typefounder breaks

Please report problems with Typefounder to your local XDE support group. Include as much detail as possible about what you were doing at the time. A copy of Typefounder.log is usually very helpful. If you are familiar with the XDE Debugger, retrieve TypefounderTool.symbols to your debugger volume, display the stack and variables, and either append the debug log to the message or save it on a file server and reference it in the message.

The Emergency subtool of the Typefounder window provides a way to free a window that is stuck in a busy state (as indicated by the Busy cursor when the cursor is inside the command subwindow). Using Emergency may have detrimental side effects, so after using it, save your work and destroy and restart Typefounder.

If you land in the debugger with an uncaught signal, it is sometimes possible to get back to the client volume (where Typefounder was being run) by typing "Q" (after any XDE debugger work that you wish to do is complete). You will probably have to do this twice. This does not always work, and if it does, the window that caused the problem will probably be dead, but you may be able to use other windows. If you take this route, save all work in progress and reboot the volume.

If you plan to save work in progress after a crash, you may want to make a debugger volume for the volume on which you run Typefounder. For example, if you run Typefounder on a Tajo volume, it is recommended that you have a CoPilot volume. If space does not permit having a debugger volume, you can perform remote debugging from another machine that has a CoPilot volume (see the Debugger section of the XDE User's Guide).

If you fill up your disk volume (indicated by a very small number of pages showing in the herald window), you will probably have to delete some files or reboot the volume before you can restart Typefounder. In some cases, you'll have to scavenge the volume before rebooting will work.

(This page intentionally blank)

F. Error messages

Though most Typefounder error messages are self-explanatory, some can be confusing. Below are some of the less obvious messages, with an explanation of how they happen and what to do about them.

Access conflict or Check Access

This usually means that a file the tool needs is in use by some other tool , such as one tool is busy reading a font when another tool wants to write it. Try to stop the conflicting tool, or wait until it is done.

Background character not available

The Background tool is in tracking mode, and you have loaded the character window with a character not present in the background font. This message is just a warning that no background is being displayed at the moment.

Error reading file

This is from Character Alias, Proofer, or TextDisplay. The file you are trying to read may be too long (more than 65,535 pages), or in use by another tool, or loaded into another window. The file may also be trashed.

Rename access not available for font ...

This is from Distributor. The tool is unable to create a backup copy of an output font (probably due to conflicting access). The font window for it is left on the screen, but the font is not Saved as it usually would be. You may want to Store this font under a new name.

Slope is too large to change font

This is from Fitter. Change the slope to be less than the number of pixels in the em height of the font.

Space too low

This can occur during operation of any tool. You may be very low on disk space, in which case you should delete some files and try again. If disk space seems adequate (over 200 free pages), your workstation's virtual memory may be full or fragmented, in which case deactivating some tools may help. If neither one of these gets around the problem, Destroy Typefounder and reboot your volume.