FlexFAX − A Network-based Facsimile Service

Samuel J. Leffler

Silicon Graphics, Inc.
2011 N. Shoreline Boulevard
P.O. Box 7311
Mountain View, CA 94039-7311

ABSTRACT

The FAX machine is now a standard piece of office equipment for businesses and is becoming more commonplace in the home. FAX machines with a computer interface are relatively expensive, but FAX modems are not. This paper describes FlexFAX, an effort to provide low-cost, easy to use, FAX communication services to computer-based users through a network-based FAX server. FlexFAX uses commercially available FAX modems and can run on any UNIX-based platform to which a FAX modem is interfaced.

Introduction

FlexFAX is a UNIX-based software system that provides a network-accessible facsimile service. FAX transmission is as simple as printing the contents of a file or sending electronic mail. When combined with a suitable imaging processor, such as a PostScript™ interpreter, a wide range of documents can be sent as facsimile. Facsimile reception is automatically handled by a server and administrators may request automatic notification of arriving facsimile. When suitable routing information is included in a received FAX, FlexFAX automatically dispatches the message to the user, or users, through electronic mail. Users may interactively examine facsimile on a graphics display and print facsimile on common PostScript printers.

The facilities offered by FlexFAX are similar to commercially available systems for personal computers; e.g. [Aba89]. These systems however do not serve a community of users, but are, instead, designed for personal use; i.e. one modem, one user. Server-based systems have been built for communities of users [Uni89, PC-89, Sol90]; but none is as general-purpose as FlexFAX − [Sol90] does not support automatic routing of incoming messages, while [PC-89] requires special-purpose hardware and communication services to do automatic routing of incoming facsimile. FlexFAX supports automatic routing of incoming facsimile, using a routing scheme that works with most any FAX modem and requires only the cooperation of the sender.

Section 1 of this paper describes the origin of the system and identifies the important features of a good facsimile system. Section 2 briefly describes existing systems. Section 3 gives an overview of the architecture of the system, identifying the major software components and their interactions. Section 4 discusses issues involved in imaging facsimile, while section 5 covers the problem of printing facsimile on a PostScript printer. Section 6 describes the scheme used by FlexFAX to do automatic routing of incoming facsimile. Section 7 summarizes current and future work.

1. Background

Silicon Graphics, like most companies, has a number of facsimile machines scattered around the facilities. These machines tend to be located in each group or department, with access treated like a copy machine: first come first served, through a sign-up sheet, or through a facsimile machine owner. There is also a central ‘‘FAX Mail service’’ similar to a company copy service. This organization typically results either in under-utilization of resources (too many FAX machines) or extended delays and wasted time (not enough machines). Furthermore, incoming facsimile are always printed; all too often resulting in piles of unwanted or undelivered FAX messages. In response to this situation (and an abandoned FAX modem that was left in the corner of a colleague's office) FlexFAX was conceived as an attempt to eliminate all FAX machines at Silicon Graphics.

For FlexFAX to be a successful replacement for the installed facsimile machines however, the services it offers need to be equivalent or better to those available with the existing scheme.[1] In particular, the typical services offered by a facsimile machine can be itemized as:

1. Transmission of printed matter (text and graphics),
2. Reception of printed matter,
3. Printing of received materials, and
4. Scanning of printed matter.

Furthermore, any computerized alternative to a FAX machine must preserve the properties that make FAX communication popular:

- Immediacy of delivery (no postal system, electronic relaying, or other queueing),
- Universality and simplicity of addressing (phone numbers and surnames),
- Low cost of communication (the cost of a phone call),
- Unattended operation (no ''telephone tag'' or answering machines).

2. Existing Systems

FAX modems and software for personal computers have been widely available for some time [Aba89, PC-89, Uni89]. These systems use one of two types of modem hardware: an add-in board or an external device that is connected through a serial port. Examples of the former are the Hayes JT FAX 9600B, the Brooktrout TR111/112 family of boards, and the Intel Connection Coprocessor; an example of the latter is the Abaton Interfax 24/96 modem (made by Everex Systems). PC- and Macintosh-based add-in boards are more prevalent than external modems for the obvious economic reasons, but also because there is no standard for communicating with the external modem. This situation will change with the availability of ''Class 1'' and ''Class 2'' modems (probably sometime this fall).

The software for personal computer-based facsimile systems varies, but typically provides the following features:

- Unattended transmission and reception of facsimile,
- Viewing of facsimile,
- Spooling with scheduling capabilities (for sending when phone rates are low),
- Printing to local printers,
- Dialing directories,

Other features often found are: support for locally-connected scanners, notification of transmission/reception, enhanced imaging capabilities (e.g. support for PostScript). A few products provide gateway facilities and automatic routing of incoming messages with a Direct Inward Dialing (DID) telephone trunk line – a facility supplied by the phone company whereby blocks of phone numbers are assigned to one incoming phone line.

The most common scenario is a personal computer with a *dedicated* FAX modem connected to an office line on a shared or dedicated basis – basically the standard FAX machine model, but with a computer. This usage model simplifies/eliminates certain problems (access control, routing, scheduling, accounting), but limits potential resource sharing.

[1] Actually, the promise of eliminating wasted unrecyclable paper alone might have been enough to make the plan worthwhile.

3. System Architecture

A simplified view of the FlexFAX architecture is shown in Figure 1. One FAX server process exists for each FAX modem on a system. This process implements the CCITT Group 3 protocols [CCI80] for transmitting and receiving facsimile, and is also responsible for handling the FAX modem. An ancillary queueing agent is responsible for handling system-related issues such as spooling, scheduling, notification, and administrative tasks. When multiple FAX modems are present, the queueing agent is responsible for balancing load and, potentially, for selecting the best FAX server/modem to use in sending a document.

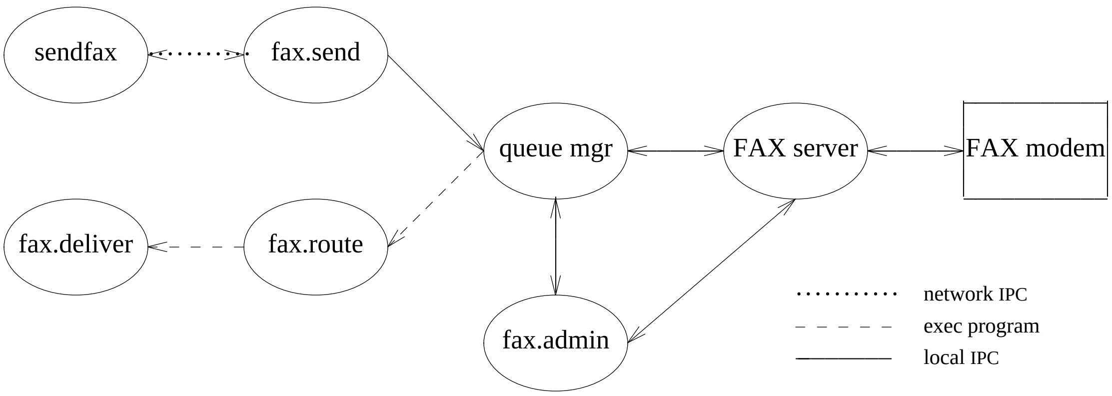

Figure 1. FlexFAX Architecture.

3.1. Transmission

To send a facsimile a user invokes the *sendfax* application, specifying a set of files and a set of destinations (possibly through a mailing list). *sendfax* establishes a network connection to a transmission agent (*fax.send* in Figure 1) and submits the job for transmission. The information passed to the transmission agent includes the destination facsimile machines (i.e. phone numbers), the time to send the documents, and the information required to generate cover sheet(s). The documents to be transmitted are passed to the server either pre-formatted for transmission or in a format that requires imaging support to generate a facsimile. Formatting the documents on the client end usually reduces load on the server. In some cases however, it may be preferable and/or necessary to format and image a document on the server side; for example, when an imaging service is available only on the server. There are also issues related to generating a match between the imaged document and the capabilities of the receiving FAX machines; e.g. matching paper width and/or length.

The transmission agent places the documents to be sent in a spooling area and crafts a *job description* file that is then sent to the queueing agent (*queue mgr* in Figure 1). The queueing agent then schedules the job for transmission, potentially at some time in the future. When a job is ready to be sent, it is passed to a FAX server which manages the transmission process. As documents are transmitted by the FAX server the queueing agent is notified and the associated job's state is updated. Jobs that need to be retransmitted or rescheduled are handled by the queueing agent.

3.2. Reception

An incoming facsimile is handled initially by the FAX server process that services the modem on which the call is received. This process answers the phone, manages the Group 3 receiving protocol, and passes the document pages to the queueing agent. The queueing agent then spawns a program that is responsible for routing incoming facsimile to a user. One of the important features of FlexFAX is that it can automatically route an incoming facsimile directly to a user by scanning the facsimile for addressing information. This information takes the form of a bar code that usually appears on the cover page of a facsimile. A database of registered bar codes is used to map a bar code to a user who then receives the facsimile as an electronic mail message (section 6 discusses the routing scheme in more detail.) If a received facsimile can not be automatically routed to its destination, it is dispatched to an administrator named the *FaxMaster*. (The

FaxMaster is modeled after the *PostMaster* user associated with electronic mail systems.)

Received facsimile can be viewed on a bitmap display and printed with various utility programs. The efficient printing of facsimile on PostScript printers turns out to be an interesting problem; it is discussed in section 5.

3.3. Document Storage

The Group 3 facsimile protocols specify a compressed format to be used in sending and receiving data. This format is encapsulated in the Class F recommendation [Cyg89] of the Tagged Image File Format (TIFF) [Ald88]; a file format that augments the compressed data with information about the structure, photometric interpretation, and origin of the data. Multiple document pages may be stored in a single TIFF file, simplifying management of documents.

4. Imaging Facsimile

Aside from handling the CCITT facsimile protocols, the main effort required to build a facsimile service is in providing tools to image outgoing facsimile. FlexFAX handles normal ASCII text, output from the *nroff* and *troff* family of text processing tools, as well as colour and greyscale images in a variety of formats.

4.1. Imaging Text

The conversion of text-oriented input is easily accomplished by canibalizing a screen-oriented previewer. Public domain previewing programs that handle *Device Independent Troff* (DIT) [Ker84] or that handle TeX DVI [Knu84] are readily available. The main stumbling block in realizing high quality output is the availability of high quality fonts. Publicly available fonts, such as those included with the X Window System [Sch86], often do not include high quality, high resolution fonts, or these fonts lack certain glyphs. It is laborious to cleanup and hand-tune appropriate fonts; consequently most vendors charge a lot of money for these fonts. Automated font machinery such as the Adobe Type Manager (ATM) [Ado90] Folio font manager, [Fol89], or Apple TrueType font system [App90] is a desirable alternative to bitmap fonts, but is also rather expensive. The cost of add-on fonts and font technology is a particularly good reason to image all facsimile on the FAX server machine rather than the client machine. This way only a single copy of the fonts and software need to be purchased, although licensing may restrict this type of use.

Another issue in imaging text is the font *face* to use in imaging the facsimile. Typical laser printers have a 300 dots/inch (dpi) output resolution while Group 3 facsimile machines support 200 dpi in the horizontal direction and either 100, 200, or 400 lines/inch (lpi) in the vertical direction. Each increase in resolution effectively doubles the time it takes to transmit a page. Consequently most facsimile are not transmitted at the highest resolution and so printing on a laser printer results in a loss of information.[2] A recent study by Adobe [Jey90] considered this problem in terms of the most readable Adobe font faces to use when imaging a facsimile that is to be sent at various resolutions and then printed on a 300 dpi laser printer at the receiving end. Their results, while very subjective, indicated that Lucida and Lucida Sans fonts were the most readable, although several were found to show good results.

The *troff* conversion program used in FlexFAX was derived from a local previewer and uses a combination of NeWS, X, and Silicon Graphics-proprietary fonts. Both text and graphics commands are correctly processed so that the *pic*, *tbl*, *eqn*, and *grap* preprocessing tools can be used in preparing facsimile. For example, the following could be used to format this paper for transmission:

```
refer ?.t | pic | eqn | tbl | troff -ms | dit2tif -o fax.tif
```

(*dit2tif* is a conversion program that generates a Class F TIFF file from *troff* output.)

[2] Many Group 3 facsimile machines do not support 400 lpi, in which case there will always be a resolution loss due to transmission.

4.2. Converting Images

The handling of colour and greyscale images is more difficult than the conversion of text and text-oriented graphics. Since Group 3 facsimile are bilevel images this conversion requires the use of *halftoning* or *dithering* techniques [Uli87]. The following scheme is used by FlexFAX:

1. Convert colour images to greyscale by taking a simple percentage combination of the red-green-blue components; by default 28-59-11 percent, respectively.
2. Resize the image to fit the page (or the desired area); a filtered scaling algorithm is used.
3. Apply a high pass filter to sharpen features.
4. Correct the greyscale colour based on an idealized scanner.
5. Convert from greyscale to bilevel with an ordered dither.

Step 2 uses a filtered scaling algorithm with blur (when scaling up); by default a triangular filter is used. Step 3 uses a 3 by 3 kernel; its' use is very important in producing good bilevel images. Step 4 passes the data through a transfer function to alter colour so that it approximates what would be obtained by scanning the image on an idealized facsimile machine. Step 5 uses an 8 by 8 ordered dither over a rectangular grid. The choice of an ordered dither was based on subjective experiments with a sampling of scanned and computer-generated images.

The default conversion scheme is done automatically by FlexFAX when it is given an image to transmit. Changes to the default parameters or alternative schemes, such as using dot diffusion [Knu87] in step 5, might be more appropriate for converting certain images, in which case the sender can select from a set of image processing tools that are part of the programming environment on a Silicon Graphics machine [Sil89]. There are also public domain image manipulation packages available such as [Pos89].

5. Printing Facsimile

Facsimile are large bilevel images; typically 1728 by 1066 lines per page.[3] While only 1 bit deep, this works out to ˜225 Kbytes/page or ˜457 Kbytes/page at high resolution. Most printers found in the workstation environment today are connected to a host through a 9600 baud or 19200 baud serial line. Assuming a 9600 baud serial line, it takes 192 seconds or 3.2 minutes to transfer the raw bits that make up a single facsimile page − assuming full transfer rates. At 19200 baud this rate drops to 1.6 minutes; a more tolerable interval.

Unfortunately for this application, many of the printers are PostScript printers and PostScript does not directly support the transfer and printing of binary images. Instead the image must be expressed in the PostScript language and, typically, presented to the printer as an ASCII representation.[4] The most straightforward scheme for printing an image is to use the PostScript *image* operator. When presented in this form, the data is most conveniently represented as an ASCII hexadecimal string that represents the binary image data. In this format twice as much data must be transferred to the printer as the raw format since each byte of data must be specified as two ASCII hexadecimal digits (two bytes). Furthermore, the PostScript interpreter must parse the input data, execute the PostScript language operation (the *image* operator), and, potentially, resize the image to fit the physical characteristics of the printer. Experiments with the printing of facsimile data in this manner indicated an average printing time of 15 minutes per low resolution page for a printer connected by 9600 baud line to a fast host. Time can vary significantly depending on the memory configuration, printer cpu, and whether or not any scaling is required.

The first solution to this problem that was tried was to implement the Group 3 facsimile decompression algorithm in PostScript. The decompression algorithm was then prepended to each print job and used to interpret raw binary data that was supplied to a PostScript *image* operator (as before). This solution represents a tradeoff between the time spent interpreting PostScript code and the time spent transferring data along the serial line. The PostScript code was carefully tuned to make the inner loop of the

[3] A standard ISO A4 page is guaranteed to have a reproducible area that is 281.46 mm long [CCI80]. This translates to about 1083 lines at low resolution (3.85 line/mm) or 2166 at high resolution (7.7 line/mm).

[4] A forthcoming revision of the PostScript language includes an scheme for the efficient transmission of compressed image data [Ado90].

decompression algorithm as low cost as possible – all symbols were bound (to avoid dictionary name lookups), conditional and looping statements were minimized, etc. An average low resolution facsimile page comprised mostly of text (about 70 Kbytes), requires less than a minute to transfer at 9600 baud. By way of comparison, a low resolution copy of test chart No. 2 of CCITT Recommendation T.21 (mixed text and graphics) is about 55 Kbytes; or about 43 seconds of transfer time. A suite of pages was printed on several different PostScript printers of varying memory configurations and PostScript engines. The tests indicated that no page took less than 1.5 times the amount of time required for sending the decompressed data. Many pages took significantly longer than the raw transfer time; especially on the slower printers.

It was clear from this work that two factors were important in reducing the printing time: use the simplest straight-line PostScript code (no looping or conditional constructs) and reduce the data transferred on the serial line. The scheme that was selected is based on the observation that the facsimile data can be viewed as a series of horizontal *spans* of black pixels painted on a white background. (In fact, this is also the model on which the Group 3 compression algorithm works.) The spans are imaged using the PostScript line drawing operators. The coordinate system is setup so that a unit square corresponds to a single pixel of facsimile data. A page of facsimile data then becomes simply a series of PostScript *moveto*, *lineto*, and *stroke* operations with coordinates expressed in rows and pixels. Since all the lines are horizontal, most PostScript interpreters can image the operations very quickly. This satisfies the first of the two criteria for a fast printing method (straight-line code), but even with abbreviations for the PostScript operators a lot of data is required to express an image in this way.

To reduce the amount of data transferred, each page of facsimile data is preprocessed to build a compact encoding of the series of line drawing operations. Specifically, each span is treated as an <*rmoveto*, *rlineto*> pair. A histogram of all pairs that appear in a page is constructed and PostScript token names are assigned based on the frequency of their use. The token names are generated from the PostScript character set; avoiding those characters that either have special meaning, or that might result in a valid PostScript operator name (e.g. *add*). The compressed image for each page then takes the form of a prologue to set up the page transformation and define encoding, followed by the encoded image data.

This technique works very well with text or mixed text and graphics but is poor for halftone images. Experiments indicate a printable page of text requires about 1.4-3.0 times as much data as the raw binary compressed Group 3 facsimile data. The time to print a page is entirely dominated by the time required to transfer the encoded PostScript. This is very encouraging as interfaces with higher transfer rates are becoming more commonly available.

A number of additional techniques can be applied to the problem to reduce and/or optimize this encoding technique. The algorithm assumes coherence in one direction. If the image is not oriented in the direction it is scanned, the coherence can be improved by rotating the image. Inverting the photometric interpretation of data (using white on black instead of black on white) can also improve the encoding. Preprocessing an image with a filter that removes some noise can significantly improve coherence, although it can also result in a loss of image detail. Finally, the encoding algorithm can be made more complex by considering strings of move-draw operations. The latter technique however is very difficult because the cost of specifying an encoded token requires that any improvements obtained from multi-pair encoding be significant. The time of additional processing intended to improve the encoding must also be weighed against the time to just transfer the data. Since the encoding process is typically done between the time a request is made to print a facsimile and the time it appears in a printer's output tray, reducing the amount of data through additional preprocessing may not reduce the overall time to print a facsimile.

6. Automatic Facsimile Routing

Existing and proposed facsimile standards define the protocols used for communicating data between FAX machines, but they do not specify any type of *envelope* or *addressing information* other than a phone number. Consequently, FAX systems that serve multiple users either use groups of phone numbers tied to a single phone line, Direct Inward Dialing (DID), or human intervention to automatically route incoming facsimile to a destination. DID requires special FAX modem hardware and requires multiple phone lines for an additional cost. Human intervention has the obvious downsides. FlexFAX uses a routing scheme based on a per-user *bar code* that is scanned from a cover page of each received facsimile. This scheme is independent of any hardware technology, but requires prior knowledge of the destination user's assigned bar code.

This can easily be provided to a sender through an initial facsimile. It is expected that people will accumulate bar code symbols for specific people in much the same way that they collect business cards or addresses in an address book.

6.1. Bar Codes

Bar codes were introduced about 20 years ago as a means of labelling material goods. They are most commonly encountered in supermarkets and departments stores. The bar code industry uses the term *symbology* to denote each particular bar code scheme, while the term *symbol* refers to the bar code label itself [Pav90]. There are many different symbologies, including UPC (Universal Product Code) [Sav75] and Code 39 (three of nine), a scheme standardized by the United States Department of Defense since 1980 [Pal89].

FlexFAX uses a Code 39-like bar code symbology for routing facsimile directly to users. The scheme used by FlexFAX differs from Code 39 mainly in the dimensions of the symbols. Code 39 was selected because it is widely known and has several properties that are important in recognizing symbols in facsimile:

- They are easy to recognize,
- They can be scanned in either direction,
- They are *self-checking* (i.e. single-bit errors can be immediately detected), and
- There is no specification for the arrangement of code words.

It should be noted that Code 39 is just one of many symbologies that would satisfy our needs.

6.2. FlexFAX Integration

FlexFAX users are automatically assigned a bar code symbol the first time they send a facsimile. These symbols are maintained in databases located at each server and typically are just the given name of the user (e.g. ‘‘Sam Leffler’’). Name clashes are resolved by appending a unique digit to a clashing name. Alternative schemes, such as using the users’ mail address (‘‘sam@flake.asd.sgi.com’’) are possible with Code 39; and would eliminate the need for an intermediate database to map user name to mail address. Mail addresses however tend to be longer than (shortened) user names and so require more effort to scan.

The bar code symbol for a user is normally included on cover sheets of outgoing facsimile sent by each user. These cover sheets are designed so that they may be turned upside down and used on any response.[5] It is also possible to place bar code symbols in the margin of pages. This scheme was rejected because the space needed to get a reliably scanned symbol tends to infringe on the working area of a page.

When a facsimile is received by FlexFAX it is passed to a scanning program that processes the document looking for a bar code symbol. If a symbol that maps to a known user is found, the document is converted to an ASCII format compatible with *uudecode* and dispatched to the user through the mail system. If no bar code is found, notification that an unrouted document exists is mailed to a distinguished user named the *FaxMaster*.

The bar code scanning algorithm sequentially searches documents looking for a bar code. Only the first page is typically scanned so that bar code symbols enclosed in a document do not give false results. This can be changed by a configuration parameter so that some number of pages are scanned. Unrouted facsimile may also be scanned in their entirety, although more commonly they are just viewed on-screen and manually dispatched.

7. Current and Future Work

The existing system would benefit greatly by direct support for PostScript imaging in sending facsimile. If a PostScript interpreter were available, all the input materials except images could be handled with a single imaging processor. Experiments using NeWS 1.1 as a PostScript interpreter have not been satisfactory. Public domain interpreters such as GhostScript do not appear to be a solution either. We are currently

[5] The bar code scanning algorithm is designed to also recognize upside down symbols since this was found to be a common operational error.

examining alternatives that would provide a good solution to this problem.

The use of bar code symbols for addressing appears to be a success. Some problems can arise in scanning bar codes on cover sheets that have not been copied onto plain paper. These problems are due to the ease with which the thermal paper normally used in FAX machines stretches during scanning, often causing distortion of the bar code symbols. The current bar code recognition algorithm handles scaling and rotation, but can be fooled by nonlinear warping. More sophisticated schemes that can handle nonlinear warping are possible, but incur more computation and may not be worthwhile. Alternatives such as re-orienting the bar code or using marking symbols other than bars to minimize the distortions are also a possibility.

Another issue associated with the bar codes is their content and related maintenance. It would be better to use a ‘‘White Pages’’ service to map names derived from bar codes to electronic mail addresses. This would avoid having to maintain databases and also avoid having to resolve name clashes (something that is not currently handled between FAX servers on separate machines.)

The integration between the mail system and the FAX service also needs to be improved. The current system uses a scheme based on Internet RFC1049 [Sir88], but a more general scheme for handling multimedia documents would improve usability. As it is now, a user must explicitly view a facsimile enclosed in a mail message, rather than the mail system automatically recognizing the content of the mail message and presenting it in the most appropriate format. The mail reader for the NeXT machine is an example of a system that recognizes non-text enclosures and automatically presents them in their ‘‘native format.’’

Finally, there is a wide range of potential work associated with the CCITT Group 4 facsimile standards [CCI84]. These standards are likely to be accompanied by the availability of greyscale and colour facsimile devices. The wide availability of Group 4 devices, however, appears to be a few years off due to their dependence on high speed communication facilities such as ISDN.

8. Current Status

FlexFAX is working, but still under development. It has yet to be made generally available to the computer users at Silicon Graphics, so it is too early to see if the original goal of replacing the FAX machines at Silicon Graphics will be attained. It has, however, demonstrated that it is possible to deliver a shared service that provides most all the services associated with a facsimile machine.

Acknowledgements

Paul Haeberli designed the image conversion scheme. Mark Linton’s feedback helped improve this paper. Eric Allman bought a FAX machine so that I could have a machine close by to work with (or so he said).

References

Aba89.Abaton, *InterFax Fax,* Abaton, Fremont, CA, 1989. Facsimile system for Macintosh systems.

Ado90.Adobe Corp, *Adobe Type Manager,* Mountain View, CA, 1990.

Ado90.Adobe Corp., *PostScript Language Reference Manual, 2nd Edition,* Addison Wesley, Reading, MA, 1990.

Ald88.Microsoft Corp., *Tagged Image File Format, Revision 5.0,* Aldus Corp., Seattle, WA, August 8, 1988.

App90.Apple Computer, *TrueType Font Manager*, Sunnyvale, CA, 1990.

CCI80.CCITT, *Standardized Test Charts for Document Facsimile Transmission*, 1980. Recommendation T.21.

CCI80.CCITT, *Standardization of Group 3 Facsimile Apparatus for Document Transmission*, 1980. Recommendation T.4. Amended 1984.

CCI84.CCITT, *Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus*, 1984. Recommendation T.6. Amended 1988.

Cyg89.Cygnet Technologies, *The Spirit of TIFF Class F,* Berkeley, CA, December 1989.

Fol89.Folio Corp, *Folio Font Manager*, Mountain View, CA, 1989.

Jey90.Ross A. Jeynes, ''Answers about fax fonts,'' *comp.lang.postcript*, March 1990. A summary of an article by Adobe that appeared in the Fall 1989 issue of the Font & Function catalog.

Ker84.B. Kernighan, ''Device Independent Troff,'' in *Documenters Workbench*, Bell Laboratories, Murray Hill, NJ, 1984.

Knu84.Donald E. Knuth, ''The TeXbook,'' ISBN 0-201-13447-0, Addison Wesley, Reading, MA, 1984.

Knu87.Donald E. Knuth, ''Digital Halftones by Dot Diffusion,'' *ACM Transactions on Graphics*, vol. 6, no. 4, pp. 245-273, Association for Computing Machinery, New York, NY, October 1987.

Pal89.R. C. Palmer, *The Bar Code Book*, Helmers Publishing, Peterborough, NH, 1989.

Pav90.Theo Pavlidis, Jerome Swartz, and Ynjiun P. Wang, ''Fundamentals of Bar Code Information Theory,'' *IEEE Computer*, pp. 74-86, 1990.

PC-89.PC Research, *FaxiX – Facsimile System for UNIX*, PC Research, Tinton Falls, NJ, 1989.

Pos89.Jef Poskanzer, *Extended Portable Bitmap Toolkit Distribution*, November 22, 1989. Included on MIT X11R4 software distribution.

Sav75.D. Savir and G. Laurer, ''The Characteristics and Decodability of the Universal Product Code,'' *IBM Systems Journal*, vol. 14, pp. 16-33, 1975.

Sch86.Robert W. Scheifler and Jim Gettys, ''The X Window System,'' *ACM Transactions on Graphics*, vol. 5, no. 2, pp. 79-109, Association for Computing Machinery, New York, NY, April 1986.

Sil89. Silicon Graphics Corp., *4Dgifts Software*, Mountain View, CA, 1989. Part of IRIX software distribution.

Sir88. M. Sirbu, ''A Content-Type Header Field for Internet Messages,'' RFC-1049, Network Information Center, March 1988.

Sol90.Solutions, *FAXGATE Plus*, FaxGATE Widget Company, Williston, VT, 1990. Facsimile gateway software for Macintosh systems.

Uli87.Robert Ulichney, *Digital Halftoning,* The MIT Press, Cambridge, MA, 1987. ISBN 0-262-21009-6

Uni89.UniFax Communications, *UniFax*, UniFax Communications Inc., Vancouver, BC, 1989. Facsimile gateway software for UNIX systems.