63
98

PRINT PREVIEW USING FINITE STATE MACHINE EMULATION OF AN IBM 3812 PAGE PRINTER

by

Gregg Allen Thomas

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

APPROVED:

Charles E. Nunnally

Dr. Charles E. Nunnally, Chairman

[illegible]

Dr. Joseph G. Tront

John C. McKeeman

Dr. John C. McKeeman

May, 1990

Blacksburg, Virginia

LD
5655
V855
1990
T566
C.2

PRINT PREVIEW USING FINITE STATE MACHINE EMULATION OF AN IBM 3812 PAGE PRINTER

by

Gregg Allen Thomas

Dr. Charles E. Nunnally, Chairman

(ABSTRACT)

A methodology and prototype has been developed which enables users of GML and SCRIPT on an IBM 3090 to preview documents on a locally attached personal computer before printing. Currently, no utility exists to accomplish this activity. This new preview process is graphical in nature and provides an absolute picture of the document, exactly as it will be printed on an IBM 3812 laser printer.

Acknowledgements

I would like to thank Dr. Charles E. Nunnally for introducing me to this challenging endeavor. I also thank him for all of the guidance, support, and encouragement he has given throughout the course of the development of this work. I truly feel that this is an effort which would not have reached completion without his backing. I also thank Dr. Tront and Dr. McKeeman for serving on my committee.

As always, my parents have been there for me during my studies at Virginia Tech. There are not enough ways to thank them for their infinite love and support. Thanks also to my fellow C.E.L.-mates: Sandeep, Tom, Chang, Rajan, Sanjay, Scott, Raghu, and Chak for making my stay at Virginia Tech more enjoyable. A special thanks goes to Manish Modi, who has become a much valued friend whom I shall miss upon his return to India.

I would also like to thank Wanda Baber and Mark Potter for providing much-apprieciated technical assistance. I am also in debt to Bob Lineberry, whose advice I often sought made my life much easier, and to Karen Snider for always being so helpful and nice.

Table of Contents

1.0 Introduction 1

2.0 The Printing Process Overview 6

2.1 Existing Printing System 6

2.1.1 YTERM 8

2.1.2 The Text Editor 9

2.1.3 SCRIPT Engine 10

2.1.4 Printer Service Machine 11

2.1.5 IBM 3812 Page Printer 11

2.2 Ideal Printing System 14

2.2.1 Improved Text Editor 14

2.2.2 Print Preview 15

2.3 Print Preview Development 16

2.3.1 CPDS Data Stream 16

2.3.2 IPDS Data Stream 19

3.0 Printer Finite State Machine 21

3.1 PMP Command Structure 21

3.1.1 Page Commands 23

3.1.2 Cursor Commands 25

3.1.3 Font Commands 27

3.1.4 Generation Commands 29

3.1.5 Macro Commands 29

3.2 3812 Printer Finite State Machine 32

3.3 PMP Data Stream Capture 36

4.0 Capabilities of the Personal Computer 37

4.1 Graphics Modes of the Personal Computer 37

4.2 Memory of the Personal Computer 39

4.3 Visual Specifications of Preview 40

4.3.1 General Visual Flow 40

4.3.2 Condensed Page 41

4.3.3 Explode Area 46

4.4 Preview Techniques 48

4.4.1 Memory Mapping 48

4.4.2 PMP Command String Interpretation 49

5.0 State Machine Mapping on the Personal Computer **50**

5.1 PMP Data Stream Capture 50

5.2 State Machine Mapping 51

5.2.1 PMP Command Subset 51

5.2.2 Print Preview State Machine 54

5.3 Preview Implementation 61

5.3.1 PMP Font Index 61

5.3.2 PMP Print Block Buffer 66

6.0 Further Development of the Preview Methodology **68**

6.1 General Steps for Transfer of the Methodology 68

6.2 Enhanced Communications Platform 69

6.3 Modifications to the Personal Computer Preview State Machine 71

6.4 Transfer of the Methodology to the Host 72

6.5 Means of PMP Data Stream Capture 74

7.0 Conclusion **76**

BIBLIOGRAPHY **78**

APPENDIX A - PMP Command Listing **81**

APPENDIX B - Capturing the PMP Data Stream **88**

APPENDIX C - Implementation of the Print Preview Methodology **90**

APPENDIX D - Extended Example of PMP Command Flow **91**

VITA **117**

List of Illustrations

Figure 1. Printing Process Overview 7

Figure 2. Improved Printing System 17

Figure 3. IPDS vs. CPDS Data Streams 18

Figure 4. 3812 Printer Finite State Machine 22

Figure 5. States of PMP Page Commands 24

Figure 6. States of PMP Cursor Commands 26

Figure 7. States of PMP Font Commands 28

Figure 8. States of PMP Generation Commands 30

Figure 9. States of PMP Macro Commands 31

Figure 10. PMP Command Flow Example 35

Figure 11. User Interface Visual Flow 42

Figure 12. User Interface Layout 43

Figure 13. Condensed Page Filter 44

Figure 14. Explode Area Filter 47

Figure 15. PMP Data Stream Capture 52

Figure 16. Print Preview Finite State Machine 56

Figure 17. States of PMP Page Commands 57

Figure 18. States of PMP Cursor Commands 58

Figure 19. States of PMP Font Commands 59

Figure 20. States of PMP Generation Commands 60

Figure 21. Typical Character Raster Pattern 62

Figure 22. Font Index Structure 64

Figure 23. Font Reference Overview 65

Figure 24. Page Print Block Structure Overview 67

Figure 25. Further Development of Preview Methodology 70

Figure 26. PC to PC Dialogue Development 73

Chapter 1. Introduction

Type has evolved from blocks of wood and metal bearing the raised character shape to the many and varied digitized representations of characters that are available by using current computer technology. Typography is the art or technique of composing printed material from type. The evolution of typography into digital type as found in computers has introduced a higher level of complexity into document composition. Further, presentation service software development has proceeded at exponentially increasing rates to accommodate these services. As a result, a diverse group of products are being developed to address this new technology. This in turn has created many techniques of accomplishing document presentation, but without a standard for typographic logic and taste.[16]

Documents composed with typographic fonts attain visible distinction and clarity. Typeset copy, like any attractive object, invites one to look at it and encourages the eye to keep moving through the text. One can use any of a great variety of faces and styles for each distinct editorial element of a message. Headings, text, captions, footnotes, passages to be stressed, anything written that plays a role in getting the writer's thoughts across to the reader.[16] As new methods of document composition and presentation achieve greater levels of functionality, the levels of complexity in overall document preparation will increase as well.

With most document preparation systems, the novice is often left to face the trial and error method of achieving the end result during document preparation. With the power of the computer and its innate ability to display graphical images, it has become possible to aid document preparation by providing a preview function which allows the user to view the document on a computer screen. This is done without having to complete the printing cycle. This function is commonly known as print previewing and aids the user by providing a method of catching and preventing printing errors. Pagination, paragraph formatting, font changes, etc., can be done while the user is still at the computer used to prepare the document. By providing the platform which eliminates many printing mistakes, the print preview function has achieved its objective, to increase user productivity and reduce overall operating costs of the printing system.

The intent of this work was to develop a methodology and system which uses standard personal computers as graphical output devices, attached to mainframe computers such as an IBM 3090. Specifically, the personal computers are used to develop advanced document preparation including exact printer previewing. This methodology will offer exact replication of the document prior to printing. The methodology developed in this research supports documents which include general fonts, mathematical symbols, vectors, and bit-image graphical images. This particular implementation focuses on advance printer previewing as it relates to an IBM 3812 page printer. During the course of this development it will become more obvious to the reader that complete graphical emulation of the IBM 3812 page printer has been accomplished. This was the main objective of the work. While being classified as a preview function, the methodology is unique in the approach. Commercial software available for personal computers provide preview functions by operating upon source documents while employing a particular printer driver. In general, the software provides the preview function by utilizing outline fonts which are scaleable in size and also reflect attribute changes. The new methodology differs in that the personal computer is emulating a device, the 3812 page printer. By attacking the problem in this manner exact printer output previews are available as opposed to representations of printer output. This preview function also allows the opportunity of mapping the 3812 data stream into other printing devices data streams such as HP LaserJet printers using HPGL, standard dot matrix printers, and also to printers supporting PostScript.

An additional theme of this work is the reality that this methodology can be implemented between a host machine and a remote intelligent graphical device. This study also gives creditability to the idea that the methodology will be useful when applied across low speed serial data connections. This fact exists primarily because the basic software tool utilized is a simple text editor implementing a text formatting style of development. This software platform is fundamentally character based, while the new methodology presented permits attachment of full graphical previewing of the documents.

The need for such a preview function is simple. The rewards for successful implementation of a preview function are substantial savings in the form of faster turnaround time and lowered production costs.[17] Today laser printers provide a good, inexpensive printing solution for those who do not make enormous demands on typographic quality. A good program will provide special Greek (for scientific notation) characters and the ability to handle formulas.[1] The new methodology fully supports extended characters and Greek symbol sets, as well as all output created from the powerful General Markup Language (GML) formula directives.

In general, printing data streams are limited in function by the very nature of their implementation. Typically the data streams incorporate fonts and images which make use of font raster patterns. Therefore, to change font point sizes in fact means that a separate font file has to be accessed and used. This is the method of implementation within the IBM 3090 printing system. The font access is almost virtually unlimited, since the user has access to a wide array of fonts which are stored on the host. However, to be truly "unlimited", fonts need to be absolutely scaleable. One such printing description language is PostScript, developed by Adobe Systems. The concept and goal for PostScript is to be able to "print anything" and be output device independent.[19] Since a great number of printers now support the PostScript printing description language, a logical extension to this work is a translation of the data stream used in the IBM 3090 printing system to that used by PostScript.

The logical solution to providing the proper means of print previewing is to define a computer whose display architecture is identical or substantially close to the printing architecture. Adobe Systems, the company who first developed PostScript, has provided a sound base towards that solution. In conjunction with Steve Jobs, the founder of Apple Computers, the NeXT computer has such an architecture with the appropriate software drivers. The screen display and the printer output are both driven by Display PostScript, which is the first method of providing device independent graphics for computer screens.[19]

Due to the proprietary nature of current commercial printing techniques, no technical references are available for comparison to the printing methods described in this work. All technical references are documents published and maintained by IBM Corporation.

The body of this work is divided into two general categories. Chapters 2 and 3 address the printing system of the IBM 3090 and the IBM 3812 page printer. Chapters 4 through 6 address the function and implementation of the preview methodology.

Chapter 2. Overview of the Printing Process

2.1 Existing Printing System

The process to create and print a document on an IBM 3090 has several steps. Figure 1 shows the printing process overview for the IBM 3090 printing system. A user may connect to the host by using a number of communications packages. Shown in the figure is the connection provided to the host by using the 3270 terminal emulation program YTERM. The text editor typically used to compose documents is XEDIT. Once the document has been prepared, the SCRIPT print invocation sends the source document to the VM3812 SCRIPT Engine, whose output is the Composed Page Data Set (CPDS) data stream. The 3812 Service Machine creates the Intelligent Print Data Stream (IPDS) by using the CPDS data stream and CPDS resource objects. The IPDS Data Stream is sent directly to the IBM 3812 Page Printer, where the finite state machine of the printer composes the pages of the document in memory and generates the output document.

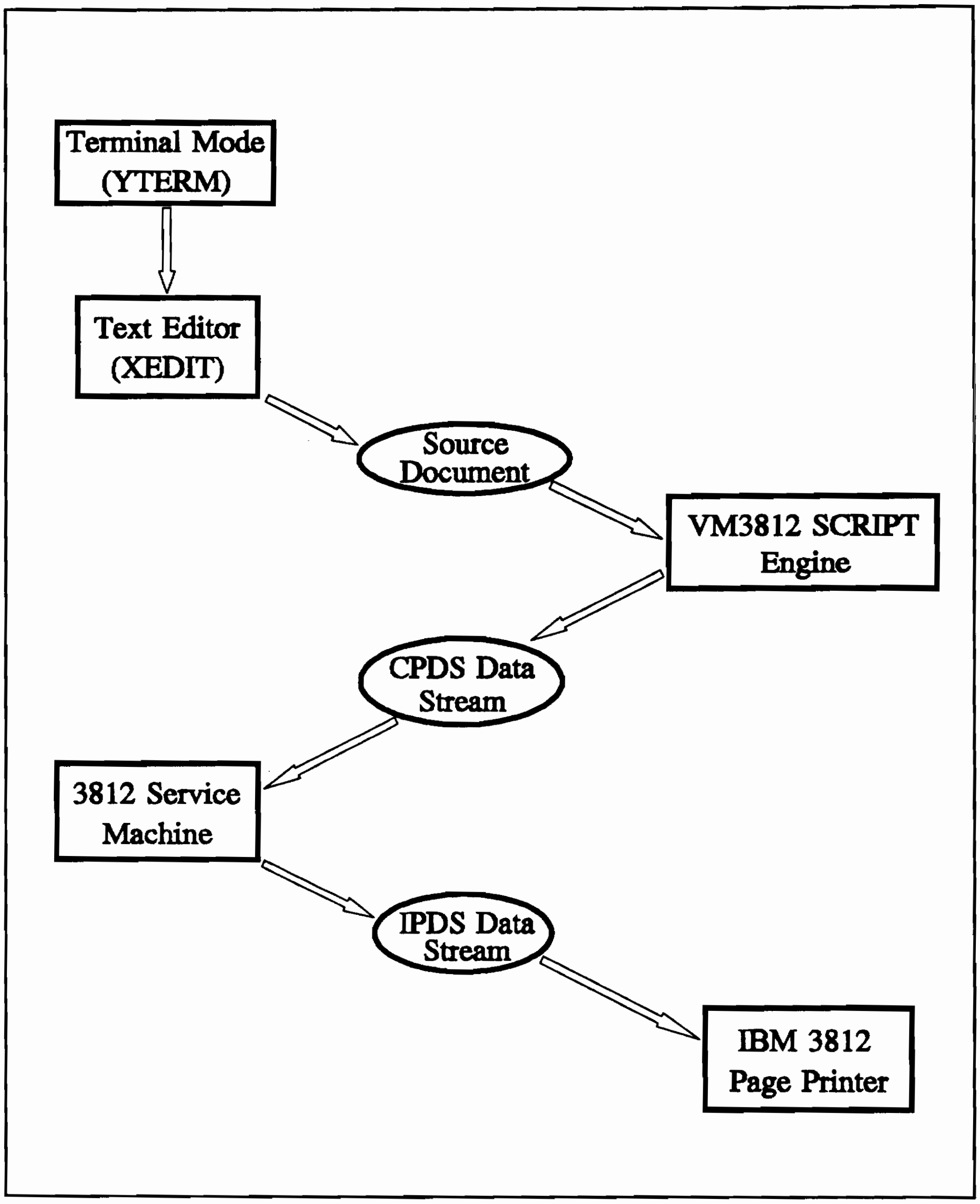


Figure 1. Printing Process Overview

The individual components of the printing system will be discussed in the following sections.

2.1.1 YTERM

Users must first connect to the IBM 3090 by using a standard communications package such as YTERM. The communications software typically uses a data switch line to call one of the modems in the IBM 3090 modem pool, which enables a personal computer to communicate with the IBM 3090. The BAUD rate of the data lines is 19,200 BAUD. The software used by the IBM 3090 as well as the software used by the personal computer system are text based. None of these packages use the power or the capabilities of the attached personal computer. Therefore, using the document preparation facilities on the IBM 3090 provides a computing paradox. The host offers vasts amounts of speed and memory to the user and a very powerful document preparation tool in GML, as well as the printing capabilities of an IBM 3812 laser printer. However, the graphics capabilities of the attached personal computer are wasted due to the lack of connectivity and appropriate software protocol. Further, the additional memory that a personal computer may have is not used by the IBM 3090, or host, and the host cannot make use of input devices attached to the personal computer, such as a mouse, when the personal computer is attached to a host. Therefore, all personal computers, regardless of model, are reduced to the role of acting as a dumb terminal when attached to the mainframe.

2.1.2 The Text Editor

The first step in the printing process is to create and compose the document. Documents are created on the mainframe by using a simple text editor, XEDIT. Print commands are embedded within the document as tags, or print directives. The tags are collectively known as the General Markup Language (GML). GML allows the user to identify logical document structures, such as chapters and paragraphs and specify formatting for each logical structure. For example, the control **:H1** defines a heading level 1, which typically corresponds to the chapter heading in a book. The **:H1** control can result in the following specific formatting steps:

- Skip down one inch.
- Turn on centering.
- Put the text of the heading in uppercase 14 point bold italic Times Roman.
- Skip one inch after the heading.
- Put the heading text into the table of contents.
- Put the heading text in a running footline at the bottom of pages in this chapter.[2]

These printing directives control such aspects as paragraph formatting, font changes, page numbering, etc. Every change in the document, such as bold characters, italic characters, size of characters, or a change in the font style itself is invoked by a GML tag. Thus, once a document has been composed and is ready for printing, it is fully character based and contains no control characters or special printing characters.

Currently, the only method of previewing a document is the DOIT directive within the text editor. The DOIT exec is a method of previewing the document prior to printing but has limited functionality. It allows the user to examine pagination, paragraph formatting, and page numbering. This method is severely handicapped however, in that it is a text-mode based tool and is also limited by size of the terminal or display device, which is typically twenty-five lines. Therefore, the usefulness of this utility is ineffectual and does not contribute towards increased productivity.

2.1.3 SCRIPT Engine

Once the document, including GML print directives, has been fully developed using the text editor, the user sends the file to the IBM 3812 Page Printer by invoking the SCRIPT print command. Higher level printing directives, such as the number of copies to be printed, printer destination, etc., are given as part of the SCRIPT print command. The SCRIPT command sends the document to the SCRIPT Engine which processes the document and converts it to a CPDS data stream. This data stream is subsequently routed to the VM3812 printer service machine.

In the CPDS data stream, a formatted document is represented as a document object stored in a CPDS print file. A CPDS document object is a hierarchical structure of other CPDS objects and CPDS structured fields that mirror the structure of the physical document it describes. Each of these objects is composed of other CPDS objects and of structured fields that specify the details of how characters and images are to be placed on the page.[4]

2.1.4 Printer Service Machine

Once the document has been processed by the SCRIPT Engine, all CPDS resource objects have their reference included within the output CPDS data stream. Higher level printing directives, such as the number of copies to be printed, printer destination, etc., have also been incorporated as part of the CPDS data stream. All CPDS resource objects are referenced and incorporated into the IPDS data stream format. CPDS resource objects include:

- Print files created by SCRIPT/VS Release 3
- Code page files
- Code font files
- Page segment files
- Overlay files [4]

Users of the printing process using SCRIPT have a virtually unlimited font library size. This is because each printer has access to every font in the host based library of fonts.[13] The VM3812 printer service machine interprets the CPDS data stream and formats all CPDS object references together to form the IPDS. The IDPS data stream is sent to and interpreted by the IBM 3812 printer.

2.1.5 IBM 3812 Page Printer

Generally, the IBM 3812 Page Printer is centrally located to its users or at a remote printing site. During the final phase of the printing process, the IPDS data stream containing the necessary print information is sent from the VM3812 printer service machine to the IBM 3812 printer. This IPDS data stream that is sent to the printer is composed of Page Map Primitive (PMP) commands. The VM3812 Service Machine also drives the printer. The service machine must understand the communications interface to each printer and provide the appropriate protocol, buffering, and device-management commands so that the printing jobs are delivered and correctly executed by the printer. The IPDS data stream provides a two-way communication path to the printer. This path is used to query the printer, initialize the printer, load and manage resources, and validate the receipt and successful processing of each message.[2] The IPDS data stream is interpreted by the finite state machine resident in the printer which composes the pages of the document.

When the 3812 is attached to a VM3812 printer service machine, only the raster bit patterns for each character in the font that are actually used in the job are included in the IPDS data stream. The font bit patterns for the characters that are not referenced by the data stream are not sent to the printer. Of the 256K RAM available to the controller over 130K remains for storage of font patterns. The VM3812 Service Machine automatically manages the available font memory in the printer. The character bit patterns loaded into the printer are maintained on a least recently used basis, character by character, across jobs.[15]

The IBM 3812 printer is a page printer, which means that each page is composed entirely in memory prior to actual printing of the document. This memory area is known as the page memory map.

A pel is defined as the smallest addressable point on a piece of paper. Since the resolution of the IBM 3812 page printer is 240 pels per inch, the page memory map within the printer itself contains almost seven million addressable bits, which represent as many unique addressable points on a piece of paper. By using PMPs, pels may be turned on or off either explicitly, or implicitly, by printing a character pattern or a vector.

The printer finite state machine interprets the PMP commands and builds the page map in stages as subsequent PMP commands are read. Once a page has been fully composed in the page memory map, a print page PMP command is issued. The printer uses the pel patterns contained in the page memory map to drive a light emitting diode (LED) array which is one page in width. The array projects points of light onto the surface of a rotating photoconductor belt. Static charges are discharged to the belt at points where the LED light falls. As the belt rotates past a toner source, the discharged areas attract a fine black powder, known as the toner. The photoconductor belt then carries this toner image to the paper. A static charge attracts the toner from the belt to the paper. Heat and pressure fuse the toner to the paper to form the printed page.[6]

2.2 Ideal Printing System

The individual components of the printing process and the printing system show that the system for composing and printing documents can be improved. The first step in improving the system is to incorporate a communications package which uses the capabilities of the personal computer. The new communications platform should take advantage of the graphical nature of the personal computer and also any of its attached pointing devices. Additional memory installed in the personal computer should be utilized by the communications platform to enhance the performance of the system.

The lack any of graphic preview capability for users of the IBM 3090 presents a need for a preview option. Because of the graphical capabilities of attached personal computers, the technical means for providing such a preview function also exist.

The following two sections describe improvements to the editor and implementation of the new preview methodology respectively.

2.2.1 Improved Text Editor - GXEDIT

By using a graphically based communications platform, the host editor, XEDIT, can use the graphical capabilities of the attached personal computer. The new host editor, GXEDIT, can also be enhanced by incorporating options that are found in most personal computer based word processing packages. By having the mouse as a pointing device, the host editor can be made more powerful and functional as editor option selections can be made available to the user through pull down menus. Commands which were previously awkward or required memorization by the user are made easier to use and more intuitive. This allows the user to create documents faster than before. Also, printer directives such as printer destination, number of copies, etc. can also be selected through the use of pull down menus.

2.2.2 Print Preview - GDOIT

In order to facilitate the preparation of documents, a graphical preview option is provided to replace the DOIT directive. The new preview function, GDOIT, makes use of the graphics available with an attached personal computer as well as the mouse as a pointing device. The preview function is a graphics based program which enables the user to view the document at the full page layout level. The character patterns which are displayed match those which are utilized at printing time. This makes the preview methodology unique in that the preview is in fact an exact replica of the printed page prior to printing as opposed to a mapping or representation. An index window provides the user a means of selectively "exploding" areas of the page for closer inspection. Both the full page view and the "explode" area are "what you see is what you get" (WYSIWYG) views of the document.

A preview option for the IBM 3090 printing process will provide an increase in productivity. Trial and error printing techniques are reduced. Page layout, multiple character sets, character sizes and attributes, math symbols, as well as vectors may be previewed, allowing the user to fully compose a document which is of final draft quality before it is printed. This saves a great deal of time in retrieving printouts from centrally located or remote printing sites. Additionally, the amount of paper consumption itself will be vastly reduced as more and more users become aware of and use the new preview utility as part of the printing process. Figure 2 shows the printing system with the added features and improvements.

2.3 Print Preview Development

The data streams used in the printing process may be captured as files at two intermediate steps. All GML printing tags within the document are processed and used to create the first of the two data streams used in the process, the CPDS data stream.

2.3.1 CPDS Data Stream

Figure 3 illustrates the function of the VM3812 Service Engine. The CPDS data stream contains references to all fonts used when printing the document, but not the actual raster patterns themselves. The font files are contained on the VM3812 print utilities disk. Overlays are also referenced CPDS objects which are not included within the CPDS data

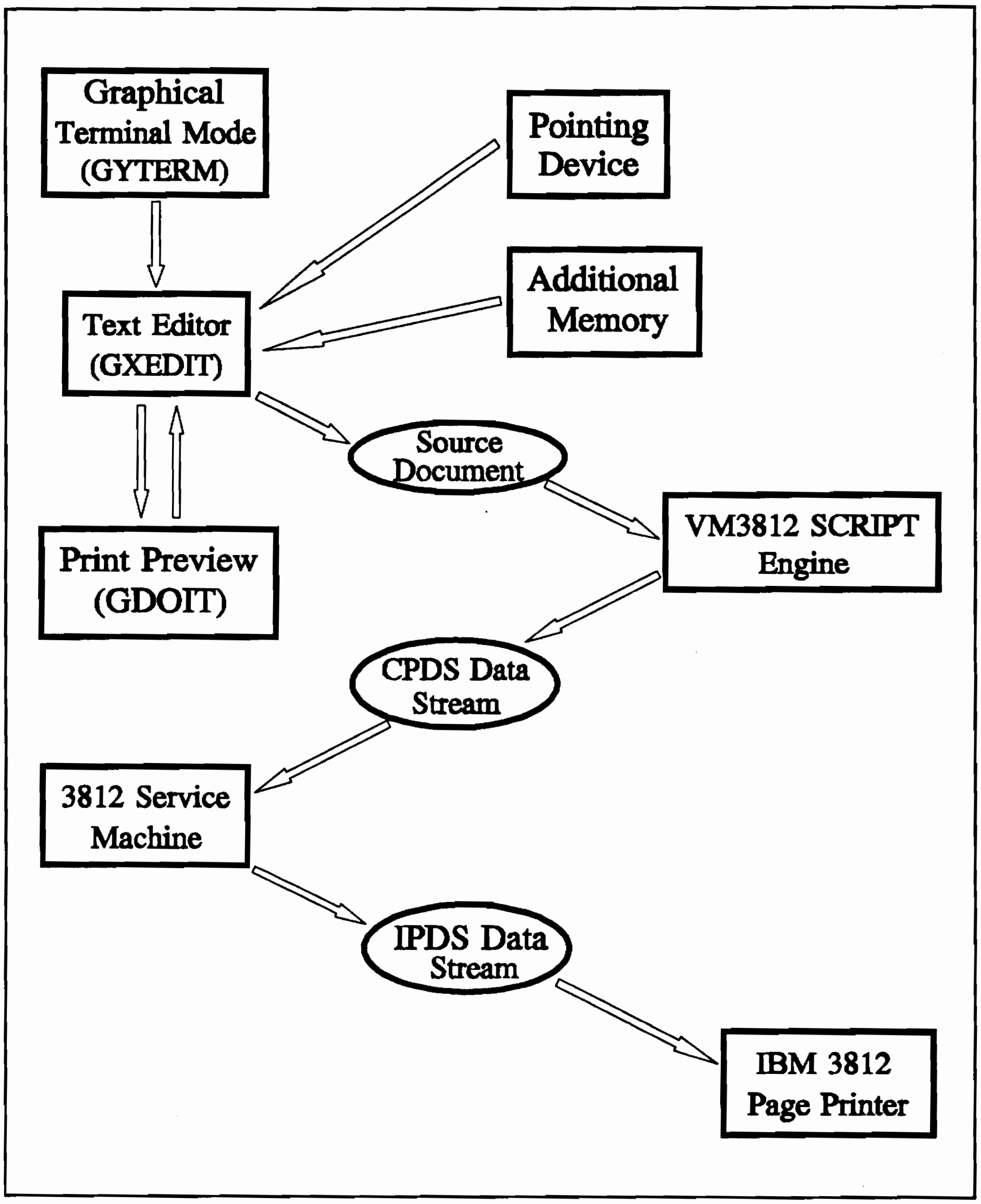


Figure 2. Improved Printing System

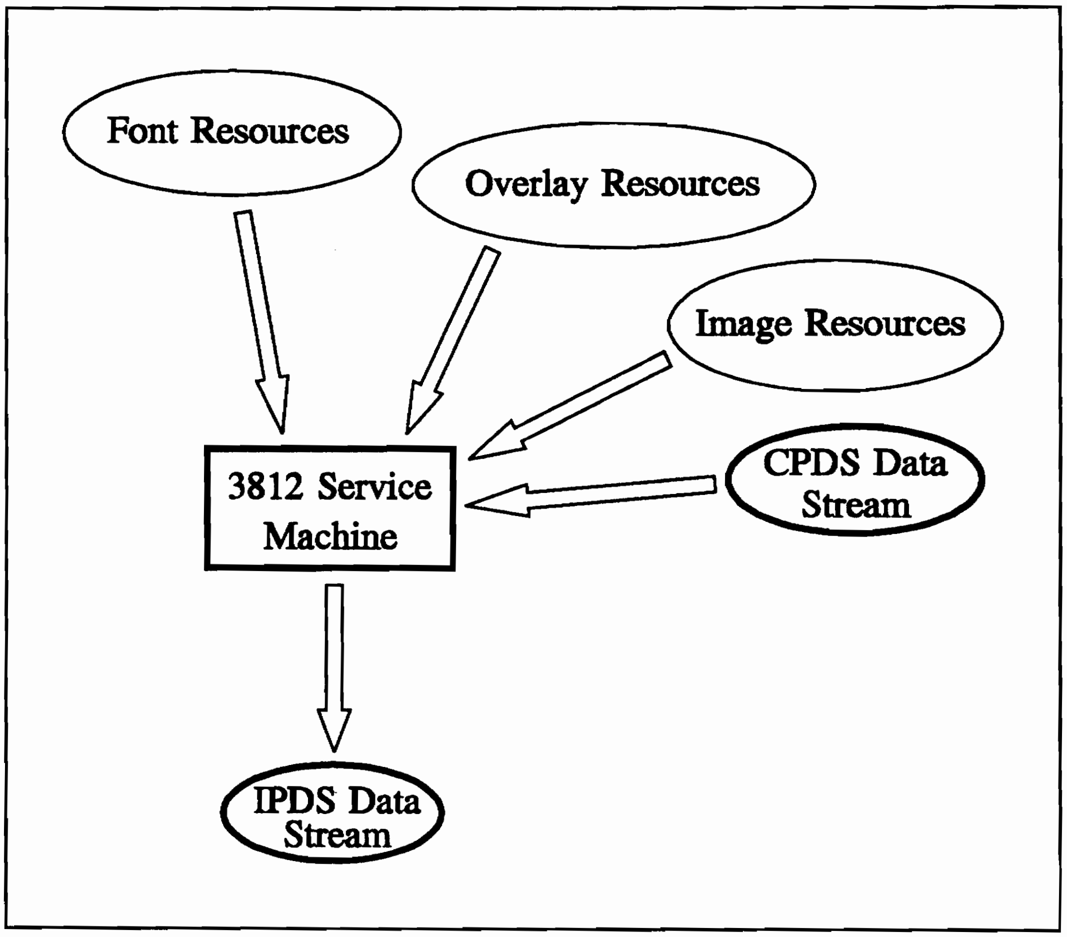


Figure 3. IPDS vs. CPDS Data Streams

stream itself. From this data stream the VM3812 printer service machine gathers everything which is needed for printing the document. For every character used in the document, the correct raster pattern is obtained from the appropriate font by referencing its CPDS font resource object. These are included as part of the PMP data stream, which is the output from the VM3812 printer service machine. Any bit image graphics also have their CPDS object resources referenced and included in the IPDS data stream as well as their cursor positioning information.

2.3.2 IPDS Data Stream

The reason for intercepting the data stream at the point after it has been processed by the VM3812 printer service machine in the printing process versus prior to the VM3812 printer service machine is straightforward. By intercepting the IPDS data stream as opposed to the CPDS data stream, all required CPDS objects have already been referenced and gathered together in one data stream. Thus, the IPDS data stream is a stand alone entity used by the finite state machine of the 3812 printer to create the document. That is, all fonts, images, and overlays used during printing and formatting information are already embedded within the IPDS data stream, and no further referencing to fonts or other formatting tools is necessary. By operating on the IPDS data stream, the new methodology provides the preview function by emulating the IBM 3812 Page Printer finite state machine. In this way, instead of having to replicate the printing process following the output CPDS data stream from the SCRIPT engine, the methodology uses the VM3812 service machine to gather and incorporate all CPDS resource objects.

A personal computer in a stand alone environment was used to develop the new preview methodology. Documents were SCRIPTed to a "dummy" VM3812 printer service machine using the PUNCH option. The PUNCH option creates an output data stream which the IBM 3812 printer uses when it is attached to personal computers. The PMP data stream contains all of the PMP commands used to print the documents, but does not include the query language and dialogue information. The PMP data stream is returned to the user's reader list. These PMP data stream files were received from the reader list and subsequently downloaded to a disk on the attached personal computer. The preview program uses these PMP data stream files as input to emulate the IBM 3812 Page Printer finite state machine. The user sees the document graphically in a full page mode with the option to selectively display portions of the page for exact previewing. The IBM 3812 Page Printer finite state machine as well its mapping into the preview state machine contained on the personal computer will be discussed further in Chapter Three.

Chapter 3. Printer Finite State Machine

3.1 PMP Command Structure

To develop the IBM 3812 finite state machine, the input data stream (PMP) to the printer must first be examined. This data stream is captured and saved as a file when the PUNCH option is used. The captured PMP data stream file is made up entirely of PMP commands and is void of extraneous control printing commands or printing codes. The higher level PMP commands define the finite state machine of the 3812 page printer. All commands of the PMP data stream command set fall into five general categories:

- **Page Commands**
- **Cursor Commands**
- **Font Commands**
- **Generation Commands**
- **Macro Commands**

The PMP command categories will be discussed further in the following sections. Figure 4 shows the printer finite state machine with the PMP states grouped into the five categories. A complete listing of the PMP command set with definitions and functions of the commands is shown in Appendix A.

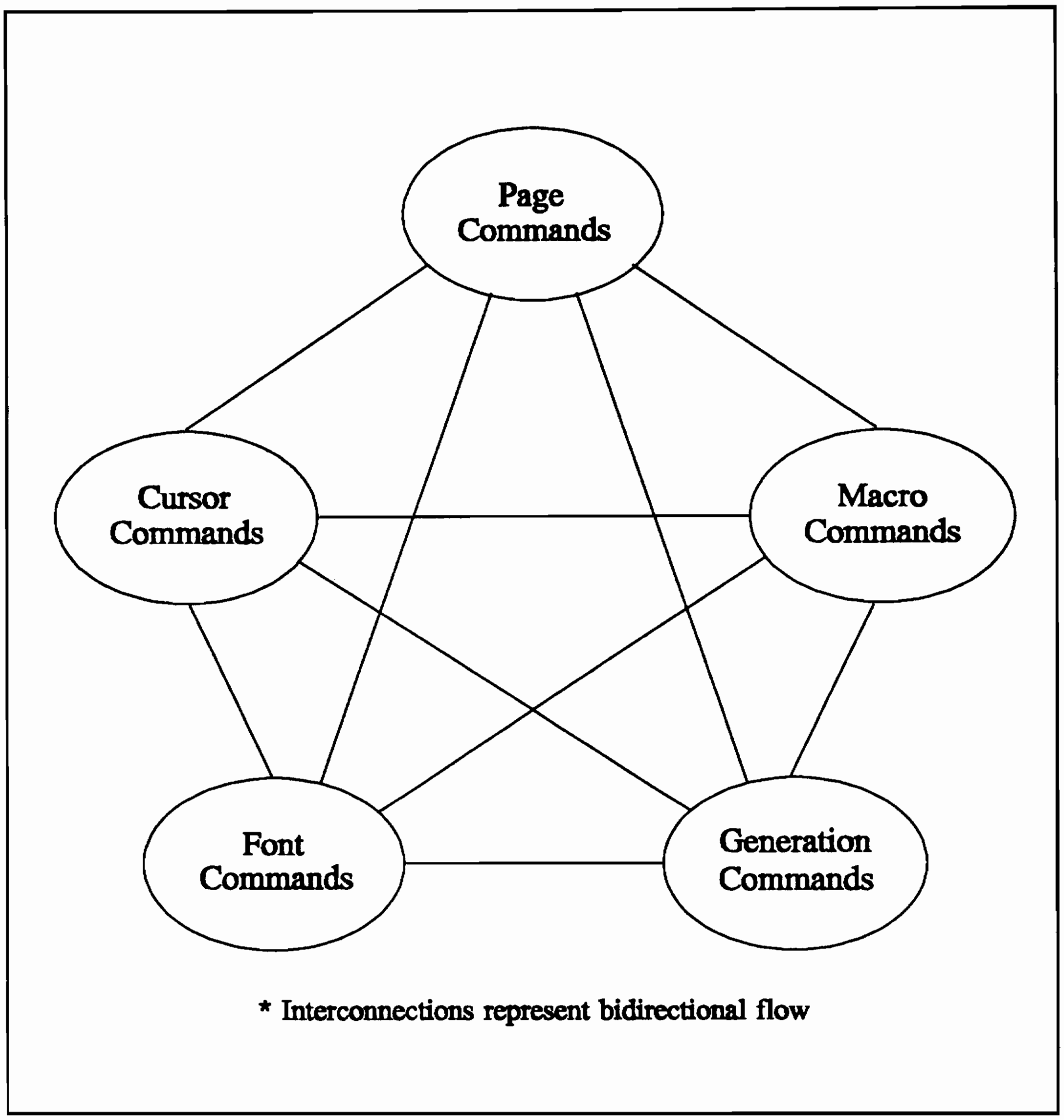


Figure 4. 3812 Printer Finite State Machine

3.1.1 Category One - Page Commands

Page commands are placed at the beginning of every set of PMP commands within the data stream required to form a page memory map. In other words, these commands set overall parameters prior to generating each page of the document as a page memory map within the printer. They include such directives such as page size and printing orientation, such as landscape versus portrait. The most important page command is the print page directive, which causes the composed page within the printer page memory map to be printed to paper. Figure 5 shows the states of the PMP Page Command category.

The first two states of the figure, Set Page Size and Set Page Orientation, are the first two states of the printer finite state machine. After these two states are reached, the third state is found in the Generation Command category. The Print Page command is the final state of the finite state machine and may be reached from all of the other four categories of commands.

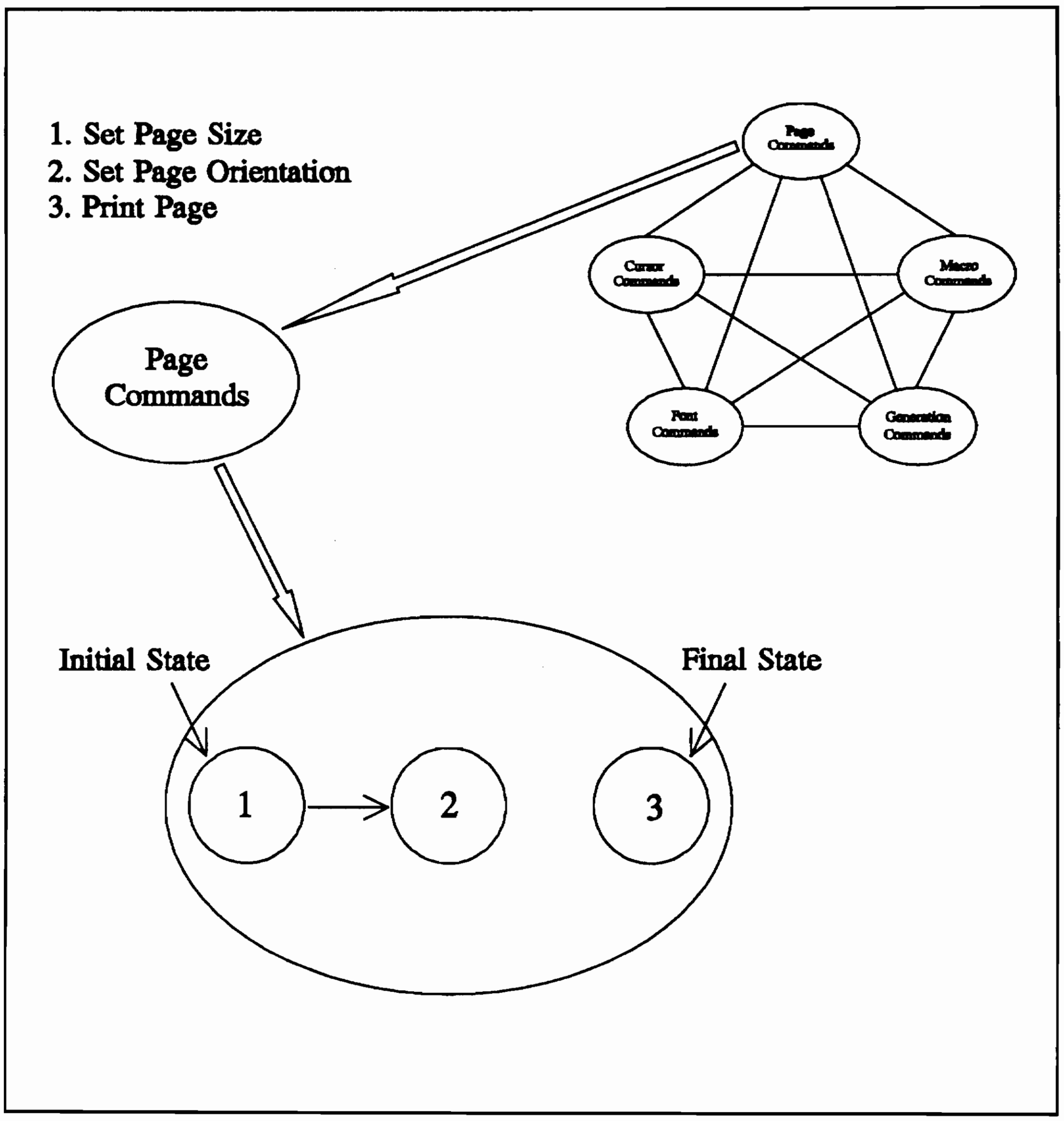


Figure 5. States of PMP Page Commands

3.1.2 Category Two - Cursor Commands

The page memory map within the printer is treated as a matrix by the printer finite state machine. As each page of the document is composed within the page memory map in the 3812 printer, the printer resident program monitors the active location of the matrix, called the cursor. Since the page map is a two-dimensional matrix, the cursor has two components, the horizontal and vertical displacement. The active cursor location is stored in a memory location known as the cursor register. The states in the Cursor command category are used to set the cursor register, update the cursor register, and also to save and restore the cursor register. The active cursor location is used as a reference point in order to copy font raster patterns, vectors, and bit images to the correct location within the page memory map.

Figure 6 shows the states of PMP category number two, Cursor Commands. The initial states of this category are one and two, Set Cursor Horizontally and Set Cursor Vertically. Additionally, state five, Save Cursor, much be reached before states six or seven, Restore Cursor and Restore Cursor Component respectively, can be reached.

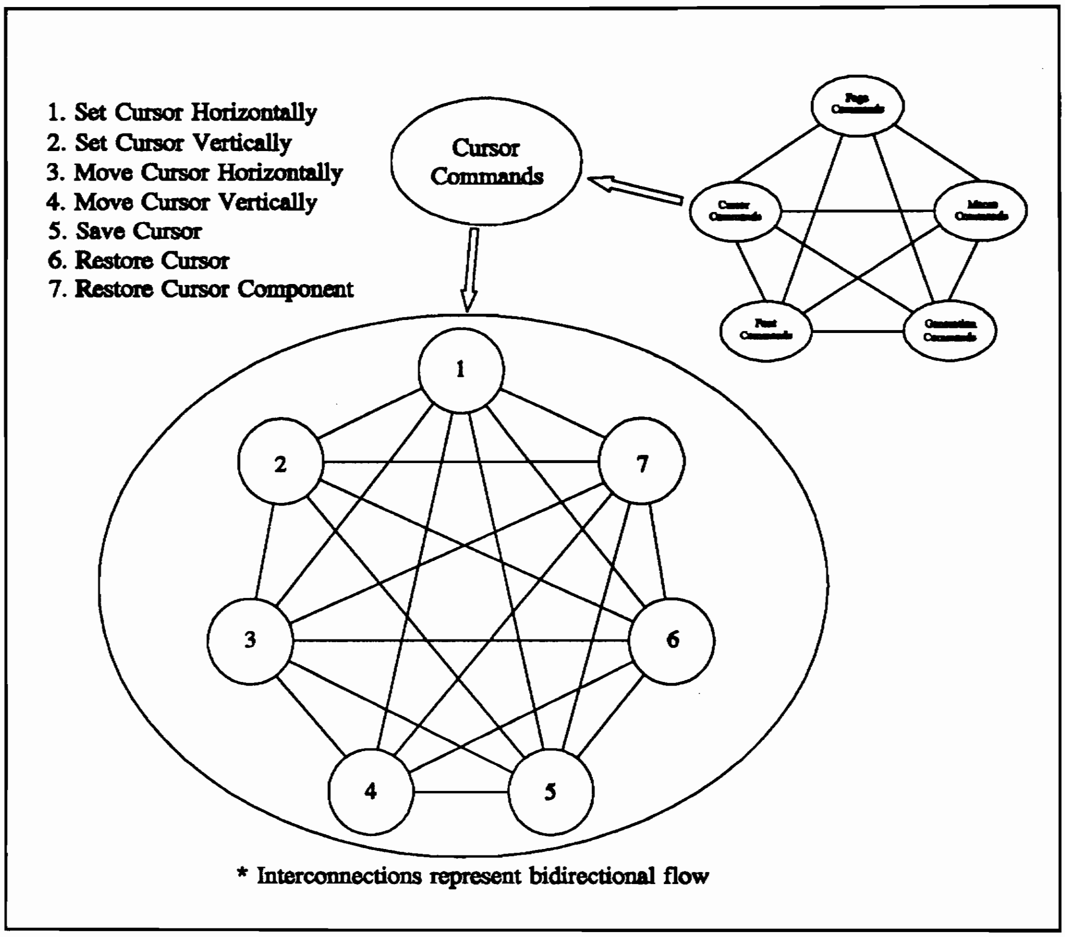

1. Set Cursor Horizontally
2. Set Cursor Vertically
3. Move Cursor Horizontally
4. Move Cursor Vertically
5. Save Cursor
6. Restore Cursor
7. Restore Cursor Component
Cursor Commands
1
2
3
4
5
6
7
* Interconnections represent bidirectional flow

Figure 6. States of PMP Cursor Commands

3.1.3 Category Three - Font Commands

At the time that the PMP data stream is downloaded to the 3812 printer, all of the font raster patterns necessary for forming the page map are contained within the data stream. None of the fonts used for forming the page memory map are contained within the printer.

Font commands are used to initially store the fonts in the PMP data stream to the memory on the 3812 printer. They are also used to manipulate the fonts while forming the page map. Fonts are activated and deactivated as needed when the characters within the document change characteristics, such as a change of font or a change in attribute, such as bold or italic. Fonts downloaded into the 3812 printer memory may also be stored as bold fonts, even if the original font patterns do not have the bold attribute.

Figure 7 shows the states of PMP category number three, PMP Font Commands. States four and five are the initial states within this category. Once the fonts have been downloaded, all other states may be reached.

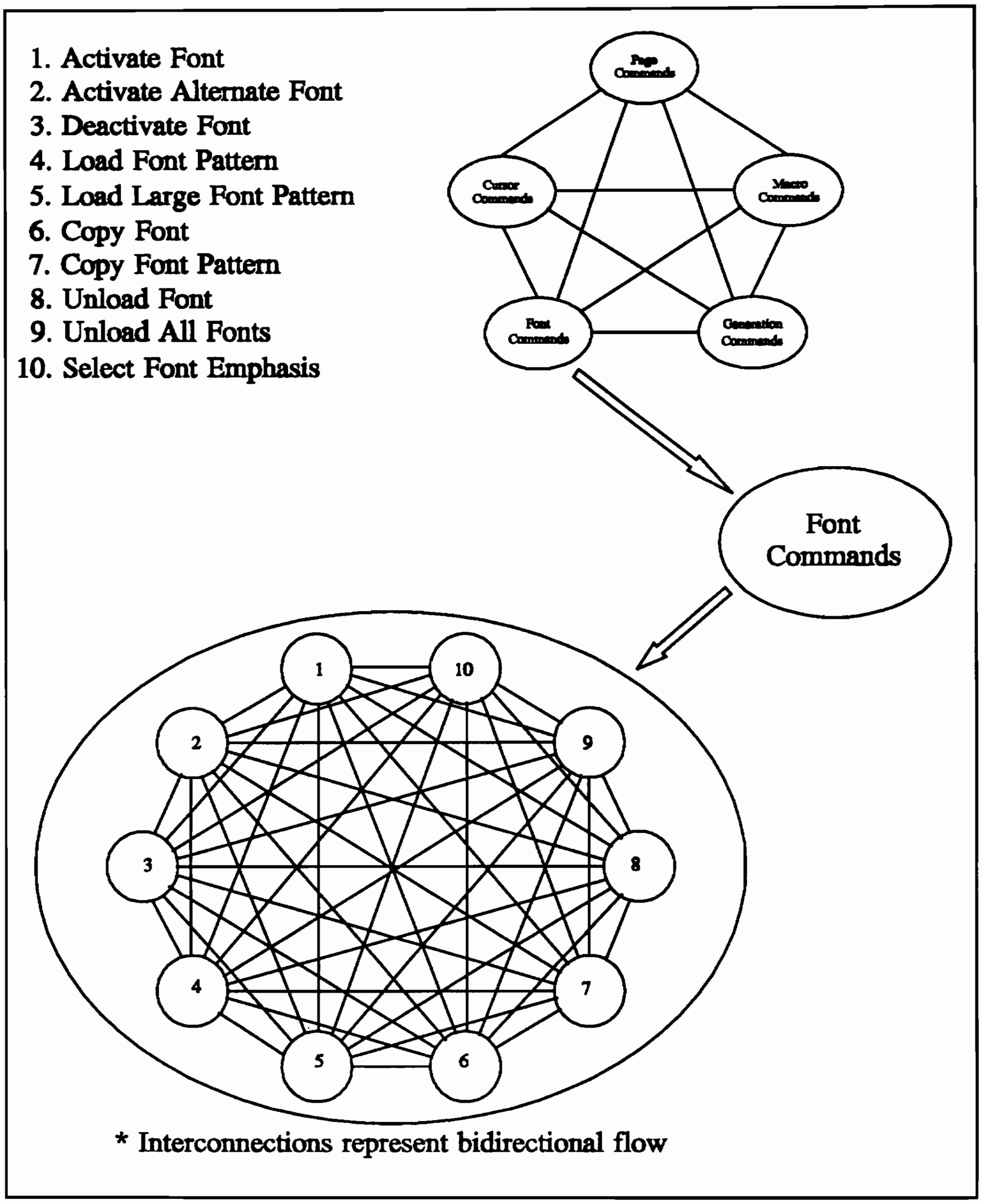


Figure 7. States of PMP Font Commands

3.1.4 Category Four - Generation Commands

Generation commands are used to turn pels on or off within the page memory map. Specifically, the commands are used to reference the fonts stored in the 3812 printer memory. The necessary character raster patterns are copied to the page map when the page is being generated. The commands are also used to generate vectors on the page memory map of varying lengths and thicknesses. Bit mapped images can also be generated directly from the PMP data stream and copied to the page memory map.

Figure 8 shows the states of PMP category number four, PMP Generation Commands. States one and two, Set Generation Mode and Set Font Pattern Controls, are states three and four of the finite state machine and initial states of the category. State three, Generate Font Patterns, may only be reached after the Load Font Pattern or Load Large Font Pattern states of category three, PMP Font Commands.

3.1.5 Category Five - Macro Commands

Macros allow for complex PMP command sequences to be recorded and referenced. This allows complex PMP command sequences to be invoked as a macro instead of having to repeat the sequence. By using macros, complex sequences of PMP commands may be executed and new PMP commands may be defined. Also, macros may be nested within other macros, up to a maximum nesting level of eight. Figure 9 shows the states of PMP category number five, PMP Macro Commands.

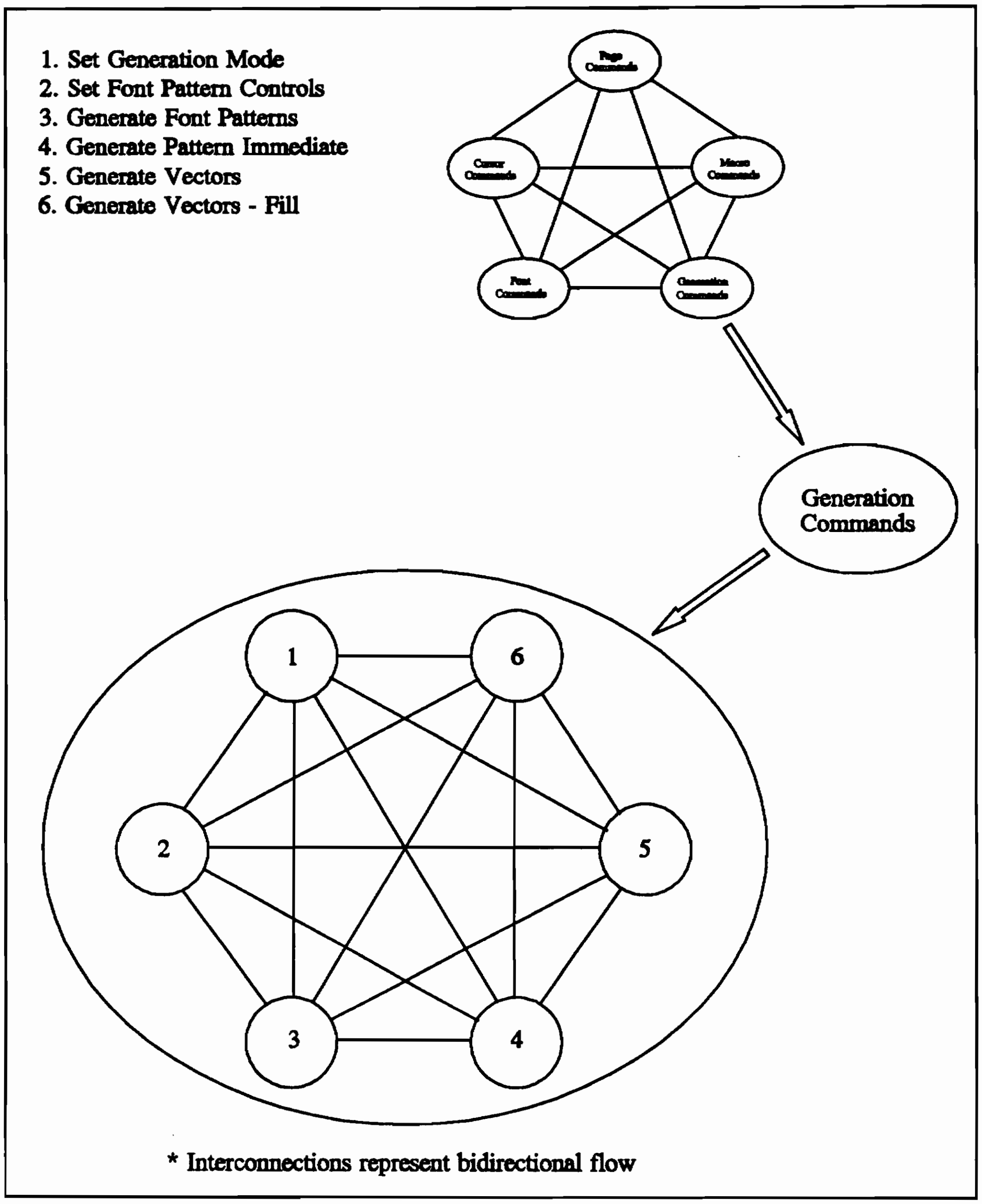

1. Set Generation Mode
2. Set Font Pattern Controls
3. Generate Font Patterns
4. Generate Pattern Immediate
5. Generate Vectors
6. Generate Vectors - Fill
Page Commands
Cursor Commands
Macro Commands
Font Commands
Generation Commands
Generation Commands
1
6
2
5
3
4
* Interconnections represent bidirectional flow

Figure 8. States of PMP Generation Commands

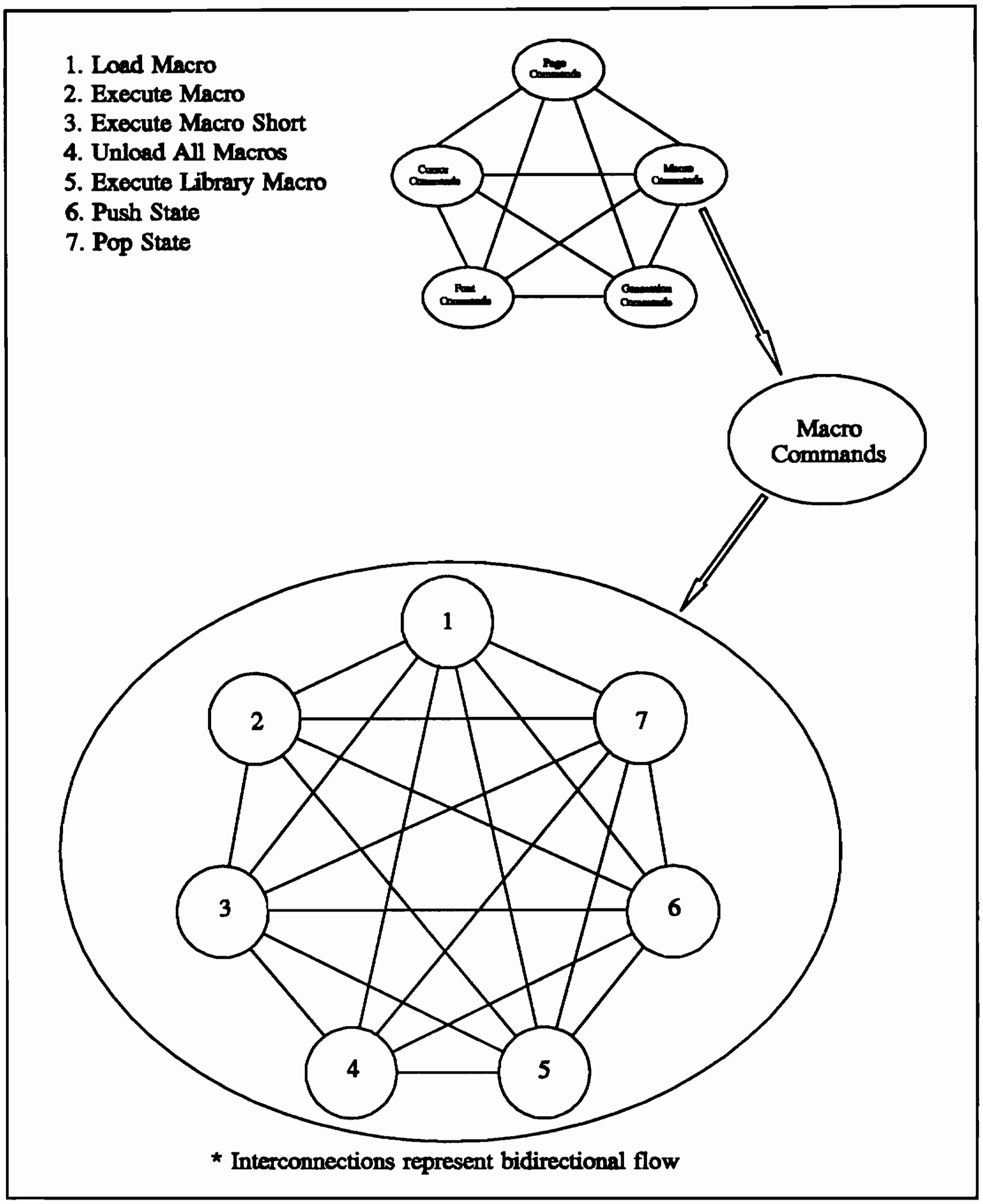


Figure 9. States of PMP Macro Commands

State one, Load Macro, must be reached before states two, three, and four can be reached. State five is somewhat independent in that a library macro may be invoked as long as it is resident on the printer disk. State six, Push State, must be reached before state seven, Pop State, can be reached.

3.2 3812 Printer Finite State Machine

The 3812 printer finite state machine is defined by the higher level PMP commands. In examining the flow of the state machine, the term "PMP Page Print Block", or PMP PPB, is first defined. The PMP PPB is the subset of PMP commands within the PMP data stream which are required to print a single page of a document. Thus, for any document there is a corresponding PMP PPB for any page which leads to its composition to the page memory map within the 3812 printer. Also, once a particular font raster pattern has been downloaded from the PMP data stream for use in forming a page, it may be referenced by all subsequent pages and does not require downloading.

Therefore, the finite state machine of the IBM 3812 Page Printer is derived from the PMP PPB subset of the PMP data stream. To process documents which consist of more than one page is simply the finite state machine reapplied to each page print block.

The first two initial states of the finite machine are derived from two of the commands within the Page Command category of PMP commands. The first state sets the page size and the second state sets the page orientation. Once these initial parameters have been set, the finite state machine may move to any of the other four categories of PMP commands, Cursor Commands, Font Commands, Generation Commands, and Macro Commands until the page has been fully composed within the page memory map. When the page memory map has been fully composed, the finite state machine reaches its final state and issues the print page directive.

Font raster patterns are loaded to printer memory prior to being used to form the page memory map. Once the font raster pattern has been downloaded however, it may be referenced in order to compose subsequent pages of the document. In general, font raster patterns for a particular font are loaded prior to generating the patterns for the words within a page to the page memory map.

Figure 10 shows the section of PMP commands which cause the generation of "This is a test." to the 3812 page memory map. To print "This is a test." the raster patterns for "T", "h", "i", "s", "a", "t", "e", and "." are downloaded to printer memory. The cursor is then positioned to the correct location within the page memory map, and then the font raster patterns for the first word, "This", are referenced and copied to the page memory map. The cursor is then repositioned and the font raster pattern for "a" is copied to the page memory map, followed by the font raster patterns for "is". The cursor is then positioned and the patterns for "test." are copied to the page memory map. In order to create the page number, the font pattern for "1" is then downloaded to printer memory, the cursor repositioned, and the pattern copied to the page memory map.

An extended example is demonstrated in Appendix D.

"This is a test."

```
D8 00                   Set Font Pattern Controls
D9 00                   Set Generation Mode         Black on white
E0 00 00                Set Cursor Horizontally
E1 00 00                Set Cursor Vertically
C2                      Deactivate Font
D2 00                   Set Page Orientation
E1 00 97                Set Cursor Vertically
E0 01 2C                Set Cursor Horizontally
E2 00 20                Move Cursor Horizontally
E2 00 C0                Move Cursor Horizontally
C2                      Deactivate Font
D2 00                   Set Page Orientation        Normal portrait
E1 00 FB                Set Cursor Vertically
E0 01 2C                Set Cursor Horizontrally
D3 03                   Activate Font               Activate Font #3
F0 E3 18 16 01 01 18    Load Font Pattern           Load "T"
F0 88 19 13 00 00 19    Load Font Pattern           Load "h"
F0 89 19 09 00 00 19    Load Font Pattern           Load "i"
F0 A2 10 0A 02 01 10    Load Font Pattern           Load "s"
F0 40 00 00 00 0B 00    Load Font Pattern           Load " "
F0 81 10 0F 01 00 10    Load Font Pattern           Load "a"
F0 A3 15 0A 00 01 15    Load Font Pattern           Load "t"
F0 85 10 0D 01 01 10    Load Font Pattern           Load "e"
F0 4B 04 04 03 02 04    Load Font Pattern           Load "."
04 E3 88 89 A2          Generate Font Patterns      Print "This"
E2 00 0B                Move Cursor Horizontally
02 89 A2                Generate Font Patterns      Print "is"
E2 00 0B                Move Cursor Horizontally
01 81                   Generate Font Patterns      Print "a"
E2 00 0B                Move Cursor Horizontally
05 A3 85 A2 A3 4B       Generate Font Patterns      Print "test."
C2                      Deactivate Font             Deactivate Font #3
D2 00                   Set Page Orientation
E1 09 D7                Set Cursor Vertically
E0 01 2C                Set Cursor Horizontally
E2 00 0B                Move Cursor Horizontally
D3 04                   Activate Font               Activate Font #4
F0 F1 15 08 04 04 15    Load Font Pattern           Load "1"
01 F1                   Generate Font Patterns      Print "1"
C2                      Deactivate Font
D2 00                   Set Page Orientation
E0 00 00                Set Cursor Horizontally
E1 00 00                Set Cursor Vertically
D1 00                   Print Page
```

Figure 10. PMP Command Flow Example

The 3812 printer finite state machine moves interactively within the previously mentioned four categories of PMP commands until the page of the document is fully composed within the page memory map. During the final state, a print page directive is issued. This is a PMP command contained within the Page Command category which causes the page memory map to be printed.

3.3 PMP Data Stream Capture

Having defined the 3812 printer finite state machine from the PMP data stream, a method must be developed to capture the data stream. By capturing the PMP data stream, the 3812 printer state machine input sequence can be processed on a personal computer and therefore can be used to map the 3812 printer state machine into a similar state machine contained in the personal computer. Once this new state machine is developed, it will be possible to fully emulate the functions of the IBM 3812 Page Printer by using the graphical capabilities of the personal computer. This is the underlying method for providing the new print preview methodology for the printing system contained within the IBM 3090 system.

Chapter 4. Capabilities of the Personal Computer

4.1 Graphics Modes of the Personal Computer

By using standard personal computers, it is now possible to map the finite state machine of the IBM 3812 Page Printer into a similar state machine contained on the personal computer. Similar in nature to the page memory map of the IBM 3812 Page Printer, the graphics mode of a personal computer can be used to generate a given page of a document onto the personal computer screen. Instead of pels turned on or off within a page memory map, pixels can be toggled on the personal computer screen.

However, due to the nature of graphics modes, there are unavoidable tradeoffs with regard to resolution versus the number of colors that are able to be displayed in a certain pixel. In general, if a given graphics video mode has a high pixel to screen resolution, there will be less colors that can be displayed per pixel. The reverse is also true in that if the set of colors that can be displayed per pixel is very high, there will less pixels that can be displayed on the screen. Of course this means that there are certain resolution constraints imposed by the set of graphics modes within the video configuration of a personal computer. Once the graphics limitations and constraints are resolved, all that remains to provide the finite state machine is to map the finite state machine of the IBM 3812 Page Printer into the finite state machine within the personal computer.

The finite state machine on the personal computer is created to replicate the finite state machine of the 3812 printer state machine. The values of the bits within the page memory map may be turned either on or off, representing pels on the actual print media being on or off. Thus only two colors are needed to represent the page memory map. This means that a graphics mode may be chosen without regard to color resolution, since two colors is the minimum set of colors that is used in any given graphics mode. This offers the highest pixel to screen resolution possible, regardless of the choice of graphics mode. However, if the graphics mode of an attached personal computer is several years old, then the pixel to screen resolution is limited by the technology itself. By the same token, if the process expects fairly recent technology in the video display card, then the process will effectively be eliminating the ability of older model personal computers to utilize the print preview function.

It has been decided that the print preview process will use one of the more recent video display options, the Multi Color Graphics Adapter, or MCGA. This particular graphics mode was chosen for its high pixel to screen resolution in its MCGA HI video mode. Also, the latest available configuration, Video Graphics Array, or VGA, offers within its set of video modes a complete mapping of the highest of the MCGA video modes as far as pixel to screen resolution, MCGA HI RES mode. However, since even the highest resolution of the personal computer video modes does not approach the resolution of the printed page of the 3812 printer, the print preview process can never be an absolute mapping of an entire page. Rather, the area of the page memory map must somehow be mapped into the smaller resolution display area of the personal computer. The Condensed Page section in this chapter will discuss this mapping.

4.1 Memory of the Personal Computer

The DOS operating system of personal computers is currently configured to operate within a base memory size of 640K RAM. Personal computers may have additional RAM added to their systems, but in general DOS and DOS applications do not make use of this additional RAM without special system drivers. Also, lower end machines typically do not run at speeds which would warrant the extra RAM.

Within the IBM 3812 printer, approximately 130K of RAM is made available for font storage. Also, the complete working RAM of the printer is only 256K. In comparison, the 3812 printer RAM is considerably less in size than the 640K base memory size of the personal computer.

The two points mentioned above lead to the conclusion that the methodology for the print preview process should be developed with regard to a 640K RAM limit, minus the memory required to run DOS and the communications software. By keeping the memory size within 640K, this also permits most personal computers to make use of the preview function.

4.3 Visual Specifications of Preview

4.3.1 General Visual Flow

The new print preview system must allow a user to interactively use the function in a manner that is smooth and user friendly. This means that the User Interface (UI) of the methodology must be implemented in such a way as to stay within the memory and graphical constraints of the attached personal computer.

The design of the visual flow of the UI was outlined primarily with the user in mind. At the most basic level, the UI has four major components:

- **Condensed Page Display**
- **Explode Area Display**
- **Index Window**
- **Command Instruction Line**

The print preview process visual flow is fairly simple. The user is first presented with a Condensed Page on the left hand side of the personal computer screen. A command instruction line is shown on the bottom of the screen which provides instructions to function key usage within the preview process UI. A non-destructive Index Window is then overlaid on the Condensed Page. The user may move the Index Window about the Condensed page until an area is selected for an exploded view. The area within the Index Window, which represents approximately a one square inch area on the printed page, can be selected to be shown in the Explode Area on the right hand side of the personal computer screen. Figure 11 shows the process as it relates to the visual flow within the UI. Figure 12 shows a representation of the layout of the UI on a computer terminal.

4.3.2 Condensed Page

Since the graphics mode has been chosen the display parameters for the condensed page can be calculated. Figure 13 shows the algorithm used to map the pels within the page memory map into pixels within the condensed page window. The limiting factor for displaying a full page in condensed mode is the vertical length of the page, due to the fact that the vertical aspect of the personal computer screen offers less pixels for display than the horizontal aspect of the screen. For an eight and a half by eleven inch piece of paper, the vertical eleven inches corresponds to 2640 pels of vertical pel resolution, due to the 240 pel per inch resolution of the 3812 printer. The video graphics mode chosen offers a pixel screen resolution of 640 horizontal pixels and 480 vertical pixels. Thus the 2640 vertical pels must somehow be mapped into the 480 vertical pixels available for display on the personal computer screen.

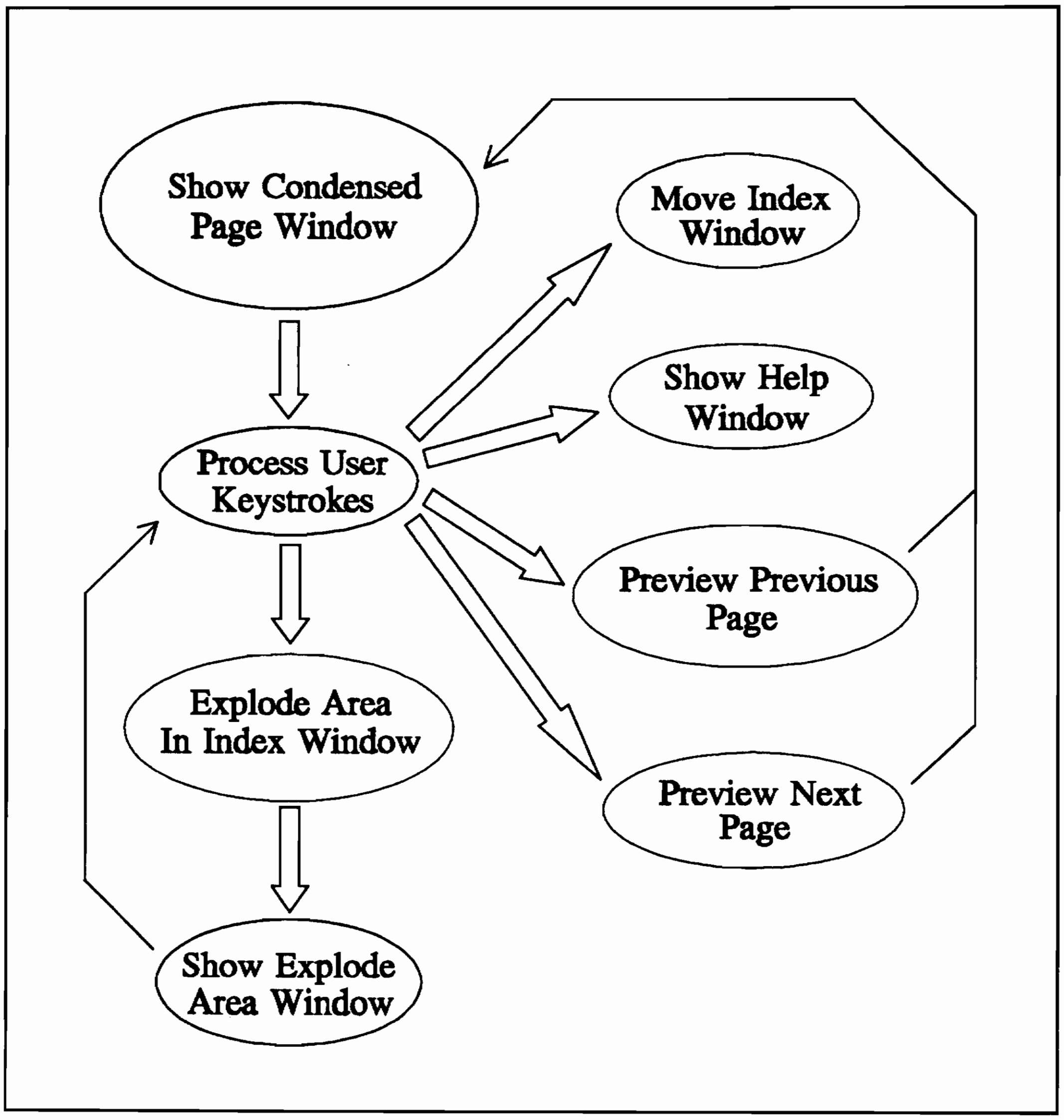


Figure 11. User Interface Visual Flow

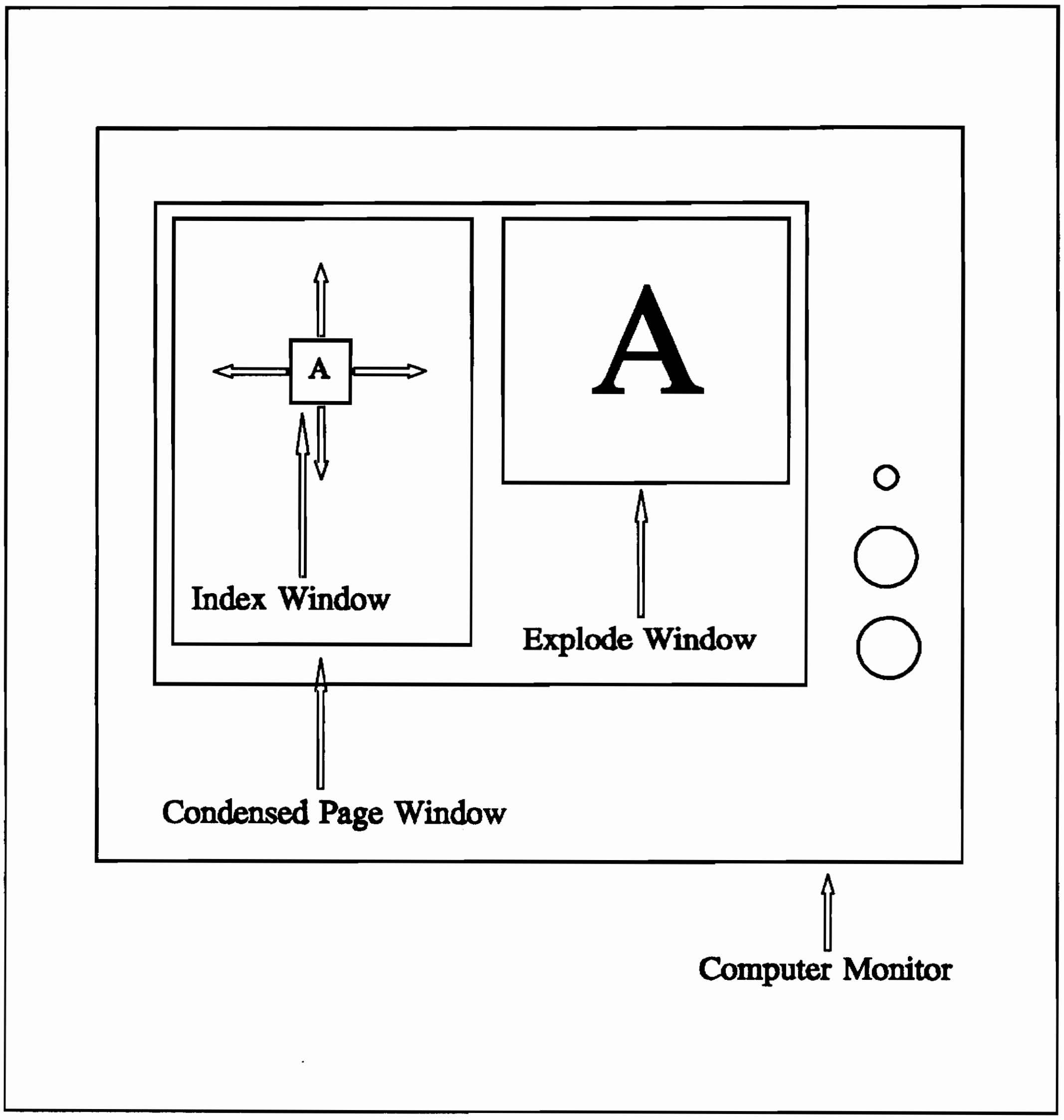


Figure 12. User Interface Layout

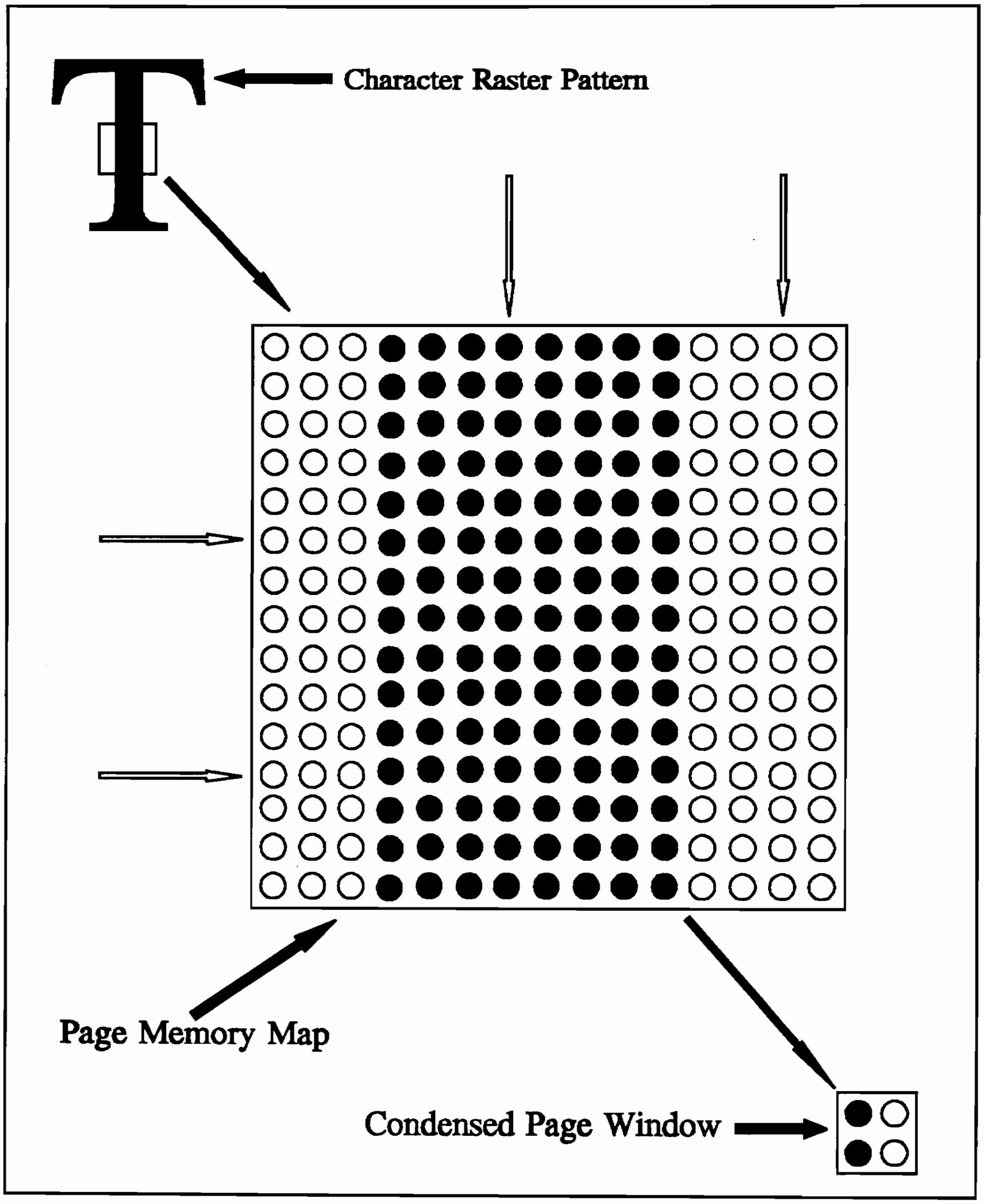


Figure 13. Condensed Page Filter

By using the two vertical resolutions, a vertical filter value can be found for use when displaying the condensed page. The personal computer vertical pixel resolution divides the vertical pel resolution of a page to give a result of six after rounding the result to the next highest integer value.

This means that at most the vertical resolution is one sixth of the actual resolution of the page in pels. The value for the vertical filter will be six, meaning that when displaying the condensed page on the personal computer screen, only every sixth pel will be displayed. The pels not used for display are discarded. This corresponds to 440 pixels being used for displaying the vertical portion of the page on the personal computer screen.

If the same filter value is used for the horizontal portion of the condensed page, this corresponds to 340 horizontal pixels being used. However, due to the aspect ratio of the personal computer screen, approximately 0.32, using the same filter value for both the vertical and horizontal portions of the condensed page proves to be impractical and results in a condensed page which is unrealistic in appearance.

In order to obtain a condensed page which is realistic in appearance, a value of seven must be used for the horizontal filter value. This results in 292 pixels being used to display 2040 pels for the horizontal portion of the condensed page.

The horizontal and vertical filter values generate the condensed page window, which represents the eight and one half by eleven page of the document within the UI of the print preview process. In other words the mapping of the page memory map into the condensed page involves translating the 240 pel per inch resolution of a page into the condensed page window. It is interesting to note that in spite of the dense resolution of the page memory map, only a very small amount of this data is used to generate the visual representation in the UI. The condensed page uses data which is approximately 2.4% of the original data that composes the page memory map.

4.3.3 Explode Area

The area on the right hand side of the screen is reserved for showing selected areas of the condensed page. Within this area, pels from the 3812 printer page memory map are mapped on a one to one basis with pixels on the personal computer screen. When previewing, the explode area represents a 1.06 inch by 1 inch area with regard to an eight and one half inch by eleven inch piece of paper. Figure 14 depicts the area filter used to show portions of the page memory map to the explode area window. This means that the index window covers approximately 1.14% of the condensed page. Within the explode area characters, math symbols, images, and vectors are shown exactly as they would be printed. As soon as the index window is moved by the user and a new area is selected for display, the explode window is updated and the new area is shown.

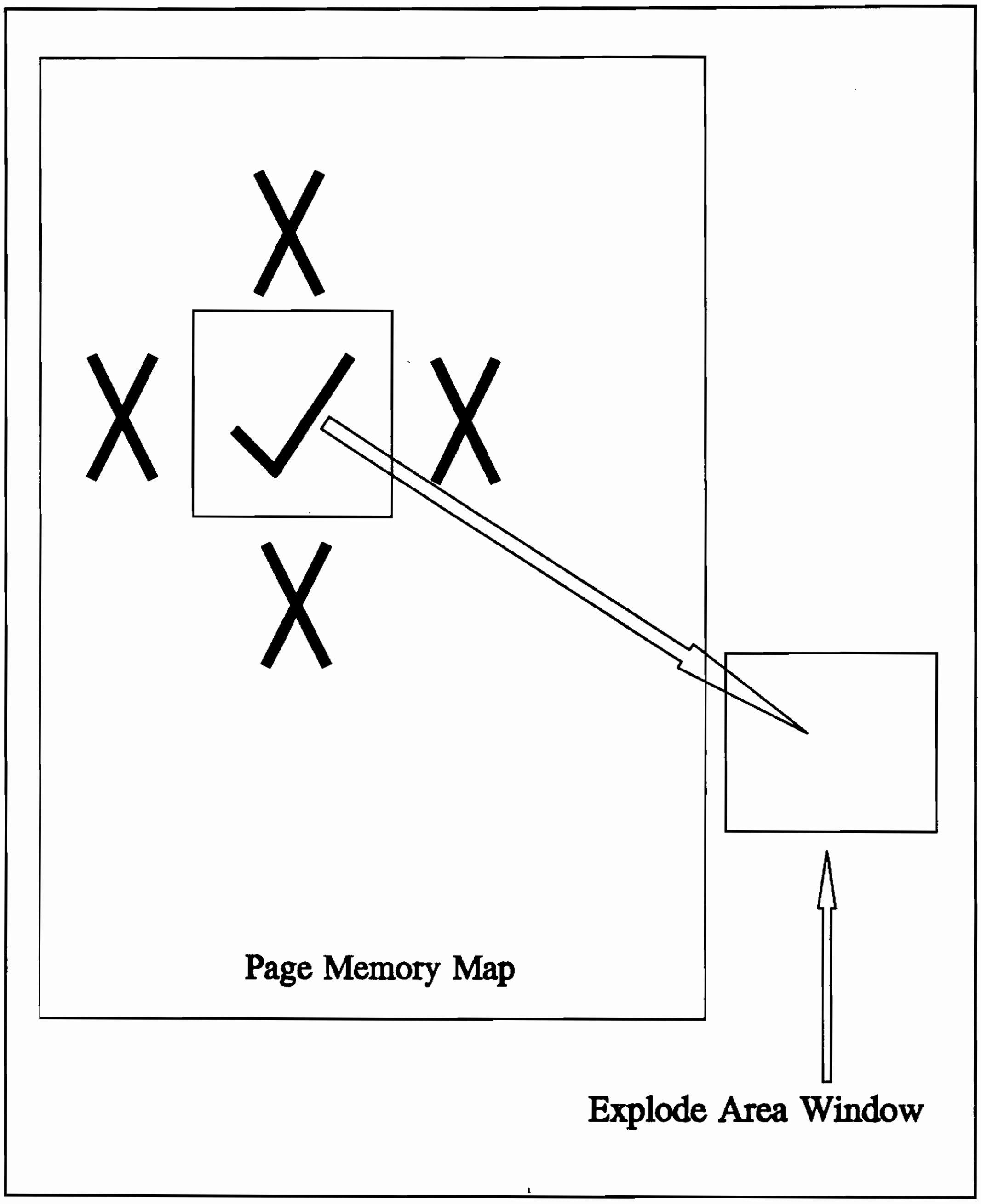


Figure 14. Explode Area Filter

4.4 Preview Techniques

4.4.1 Memory Mapping

One possible algorithm for implementing the preview process calls for forming the pages of the document within the memory of the personal computer in the same manner as they are formed within the 3812 printer page memory map. However, the 240 pel resolution of the printer necessitates 5,385,600 bits (657 K-bytes) per page to store the entire page layout. A comparison with the base memory configuration on the personal computer of 640K shows that the page map obviously cannot be formed in memory.

Another alternative again involves forming the page of a document as in the 3812 printer page memory map, but not in the personal computer's memory, but instead as a file to be saved to disk. This would require the system to have a hard disk, which would place limitations on the number and types of personal computers which could use the preview function.

Also, in order to follow either of the two methods above would involve forming the complete page map and then use it as a reference to show specific areas for previewing. This would involve extra overhead in computing time prior to being able to show any type of preview. Once the index window is positioned over an area on the condensed page, an index is calculated in order to select the exact area for previewing and show it in the explode area window. The next method discussed eliminates the need for forming the complete page and instead uses the captured data stream as an operator in order to form the condensed page and the explode area.

4.4.2 PMP Command String Interpretation

This approach calls for saving the PMP bytes within the captured data stream that are necessary to compose the document to memory. As shown in the discussion on the PMP data stream, a file can be captured which contained the necessary PMP command bytes for forming the document without any external references. Thus, if this file is intercepted and stored to the memory on the personal computer, each page can be composed from the PMP data stream as needed. By using the PMP data stream to interactively create the condensed page window and the explode window as needed, added computing overhead is eliminated. This also eliminates the need for a hard disk, and also stays within the memory constraints needed for the program.

Chapter 5. Print Preview State Machine

5.1 PMP Data Stream Capture

In order to emulate the 3812 printer finite state machine, its input sequence, the PMP data stream must be further analyzed. In order to operate on the PMP data stream, a "dummy" VM3812 printer service machine has been set up and configured as resident on the IBM 3090. Instead of the PMP data stream being sent to a 3812 printer, it is instead returned to the user's reader list on the IBM 3090.

Since the PMP data stream is the data stream used to print documents when the 3812 printer is attached to the personal computer under DOS, the new preview method also lends itself to extending its use beyond previewing from the host alone. The new preview method may also be implemented for the 3812 printer when attached to a single personal computer under DOS, or with further modification, to a distributed network with an attached 3812 printer, such as a token ring network.

See Appendix B on exact instructions on capturing the PMP data stream. Figure 15 illustrates this process.

5.2 State Machine Mapping

5.2.1 PMP Command Subset

In order to map the 3812 printer finite state machine into a new print preview finite state machine to be contained on the personal computer, the PMP command structure must be examined again. The finite state machine of the printer is easily implemented on the personal computer. However, there are some structural changes in the state machine brought about simply due to the differences in architecture as well as the printing process system in general.

Since the preview function is for use with documents that are the end result of the SCRIPT process, each document has a job header page which includes such information as the job number, date, user ID, etc. Since the methodology that has been developed is directed solely towards previewing the document itself, it becomes well within the bounds of the exercise to exclude this job header page and effectively render it transparent to the user.

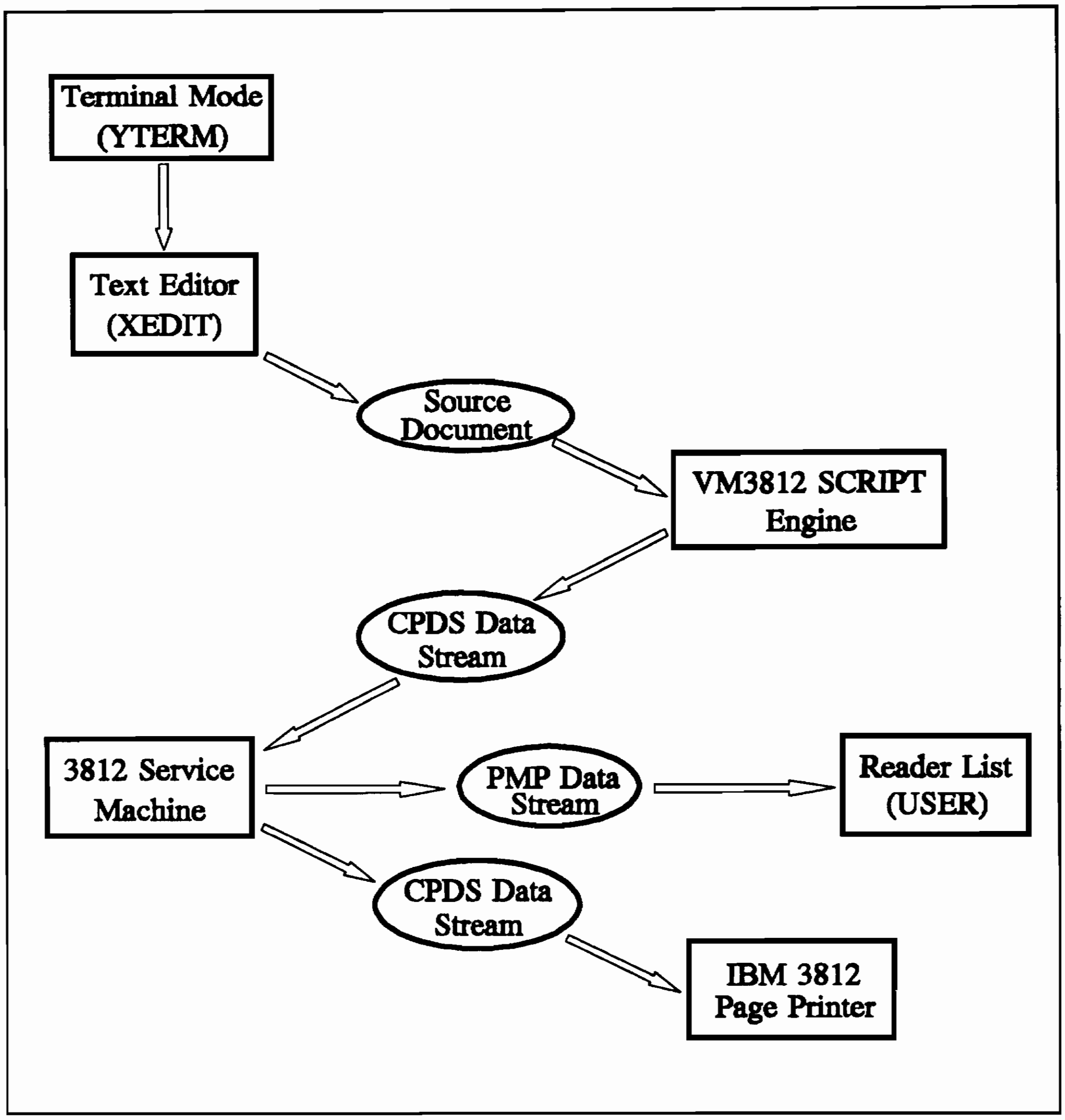


Figure 15. PMP Data Stream Capture

Additionally, some job printouts routed through the 3812 Page Printer such as program listings and program output require landscape printouts. The landscape printing style would require a different visual layout on the personal computer screen. This does not lend itself towards meeting the visual criteria which has been set for the print preview process, so the landscape printing mode has been excluded as a permissible input to the print preview process developed in this document.

These generalizations inherently reduce the subset of PMP commands within category one, PMP Page Commands, to that of a single command, Print Page.

There are a total of thirty-three PMP commands which compose the PMP data stream. All of these commands fall within the five categories shown in Chapter Two. The printer service machine interprets the file which has been SCRIPTed and composes the PMP data stream to be sent to the 3812 Page Printer. Presently the VM3812 service machine for the 3812 Page Printer does not include any of the macro PMP commands in its output, which reduces the set of PMP commands to only four general categories and a total PMP set of twenty-six commands. Macro commands are provided mainly for use with the IBM 3812 Page Printer when used as a printing device from within the personal computer DOS environment. Therefore, the 3812 printer state machine being discussed and developed does not include macro commands as inputs. However, the macro commands are included and defined in Appendix A, the complete listing of the PMP command set.

Furthermore, in examining the Font Command subset of PMP commands, there are only three out of ten PMP commands which are used in the state machine on the personal computer. On the IBM 3812 Page Printer, there is always an active font. The fonts are also loaded and unloaded into the active font workspace as they are needed. However, within the personal computer state machine, all fonts are stored to memory and referenced as necessary, so no font manipulation is necessary with regard to the active font workspace. This further reduces the command set to a total of seventeen PMP commands which are actually used within the personal computer state machine. This subset of PMP commands is still contained within the four general categories of PMP commands mentioned earlier.

5.2.2 Print Preview State Machine

Although the original set of PMP commands used by the 3812 printer state machine has been reduced, full emulation of the 3812 printer is possible. Once again, by using the concept of the PMP Page Print Block, the new state machine can be derived from the high level PMP commands in the new reduced subset of PMP commands.

By reducing the subset of PMP commands within category one within the printer state machine, the initial state of the state machine has also been changed. The printer state machine structure is basically the same, however three of four of the categories of PMP commands have been reduced, and one category eliminated. This basic structure creates a new print preview state machine that is simply a subset of the 3812 printer state machine. The flow of the subset of PMP commands remains the same as in the full 3812 printer state machine.

Figure 16 shows the new finite state machine PMP categories as they are mapped into the personal computer. Figure 17 shows PMP category one, Page Commands, Figure 18 shows PMP category two, Cursor Commands, Figure 19 shows category three, Font Commands, and Figure 20 shows PMP category four, Generation Commands.

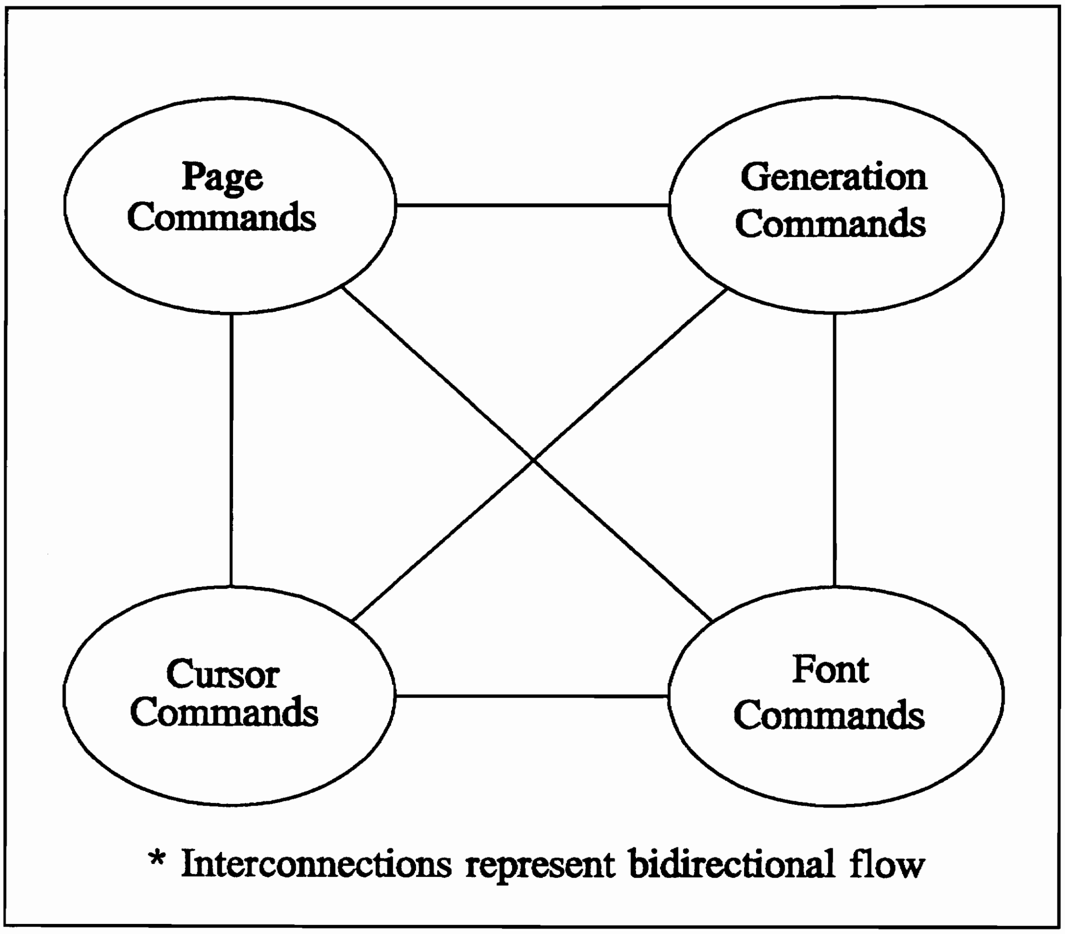


Figure 16. Print Preview Finite State Machine

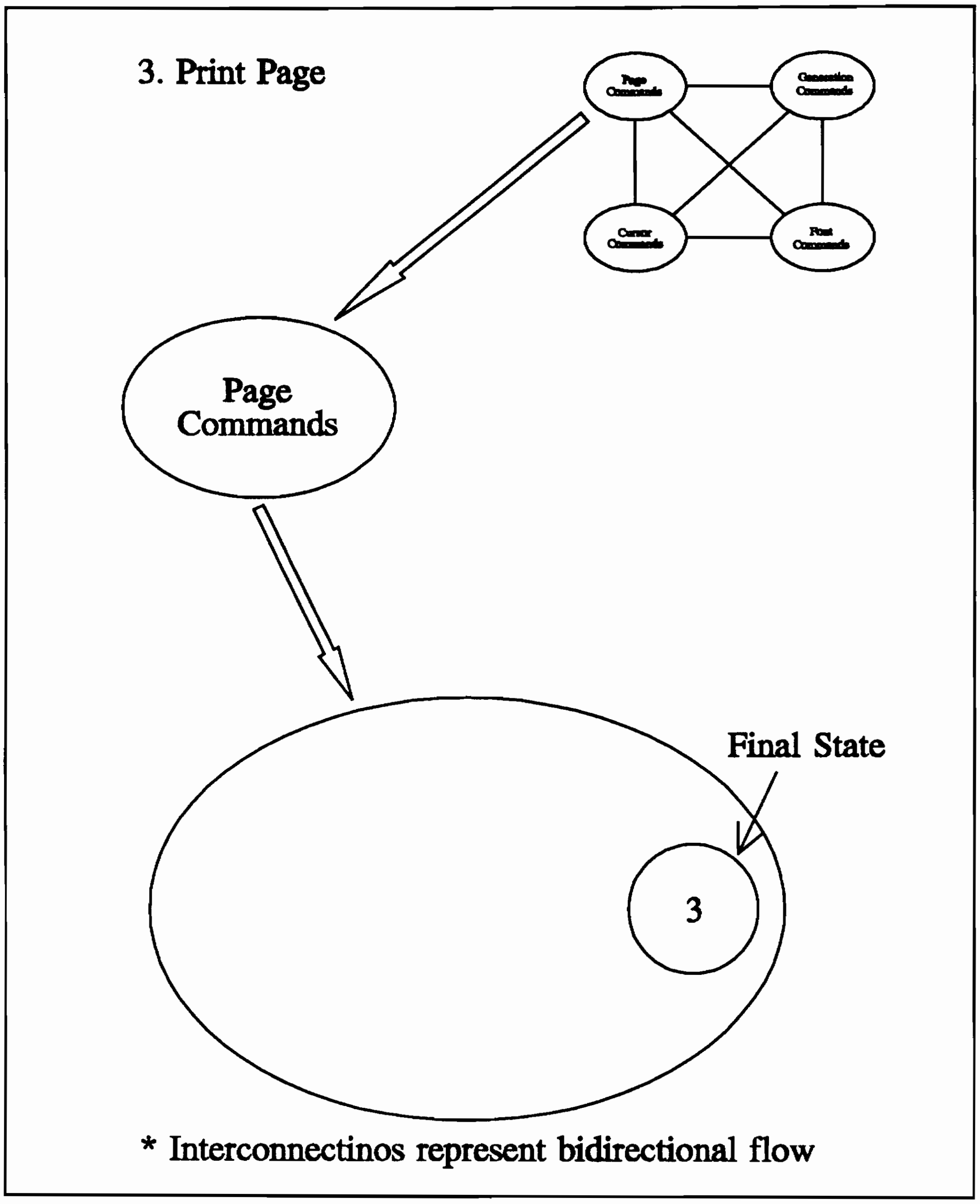


Figure 17. States of PMP Page Commands

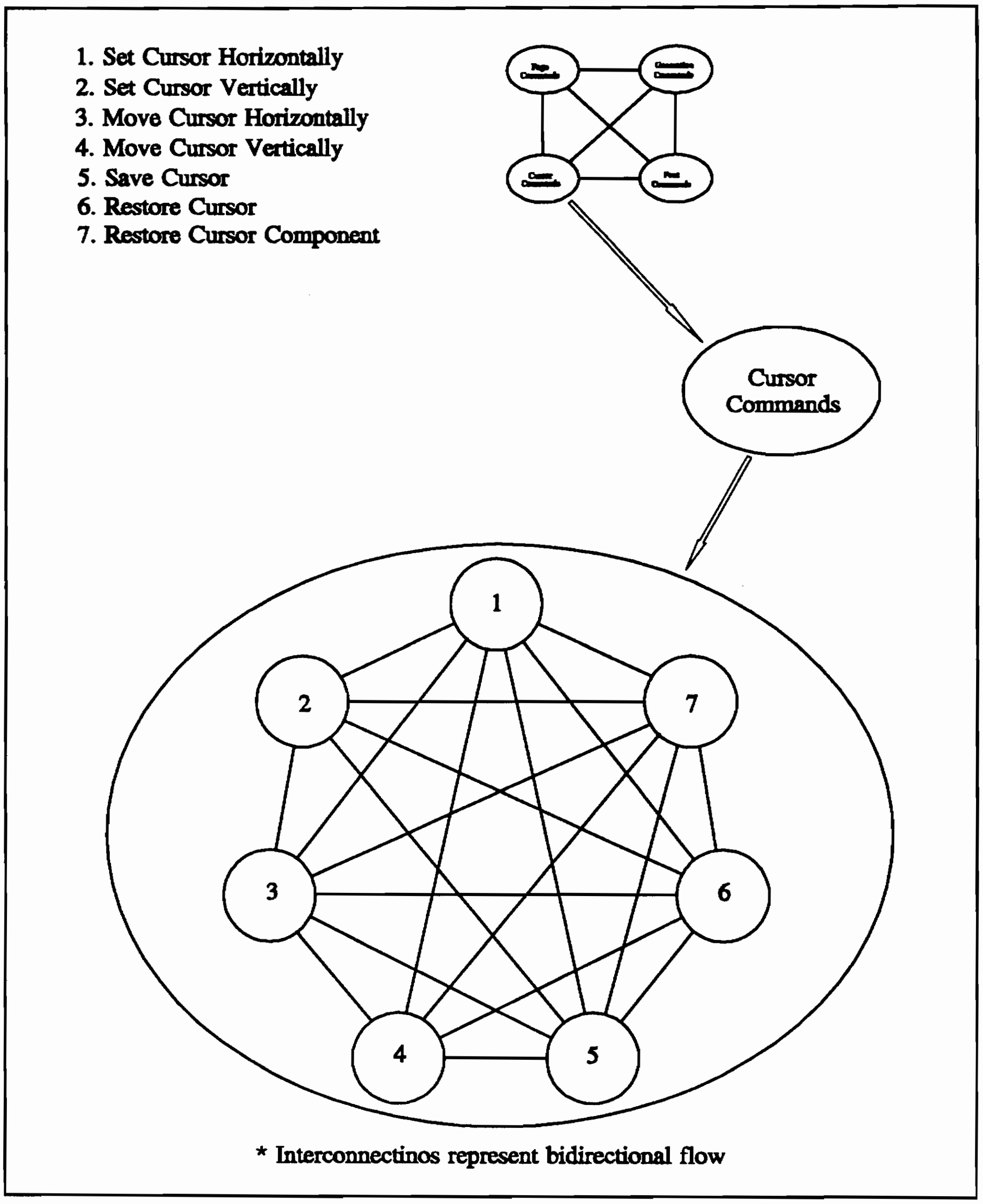

1. Set Cursor Horizontally
2. Set Cursor Vertically
3. Move Cursor Horizontally
4. Move Cursor Vertically
5. Save Cursor
6. Restore Cursor
7. Restore Cursor Component
Cursor Commands
1
2
3
4
5
6
7
* Interconnectinos represent bidirectional flow

Figure 18. States of PMP Cursor Commands

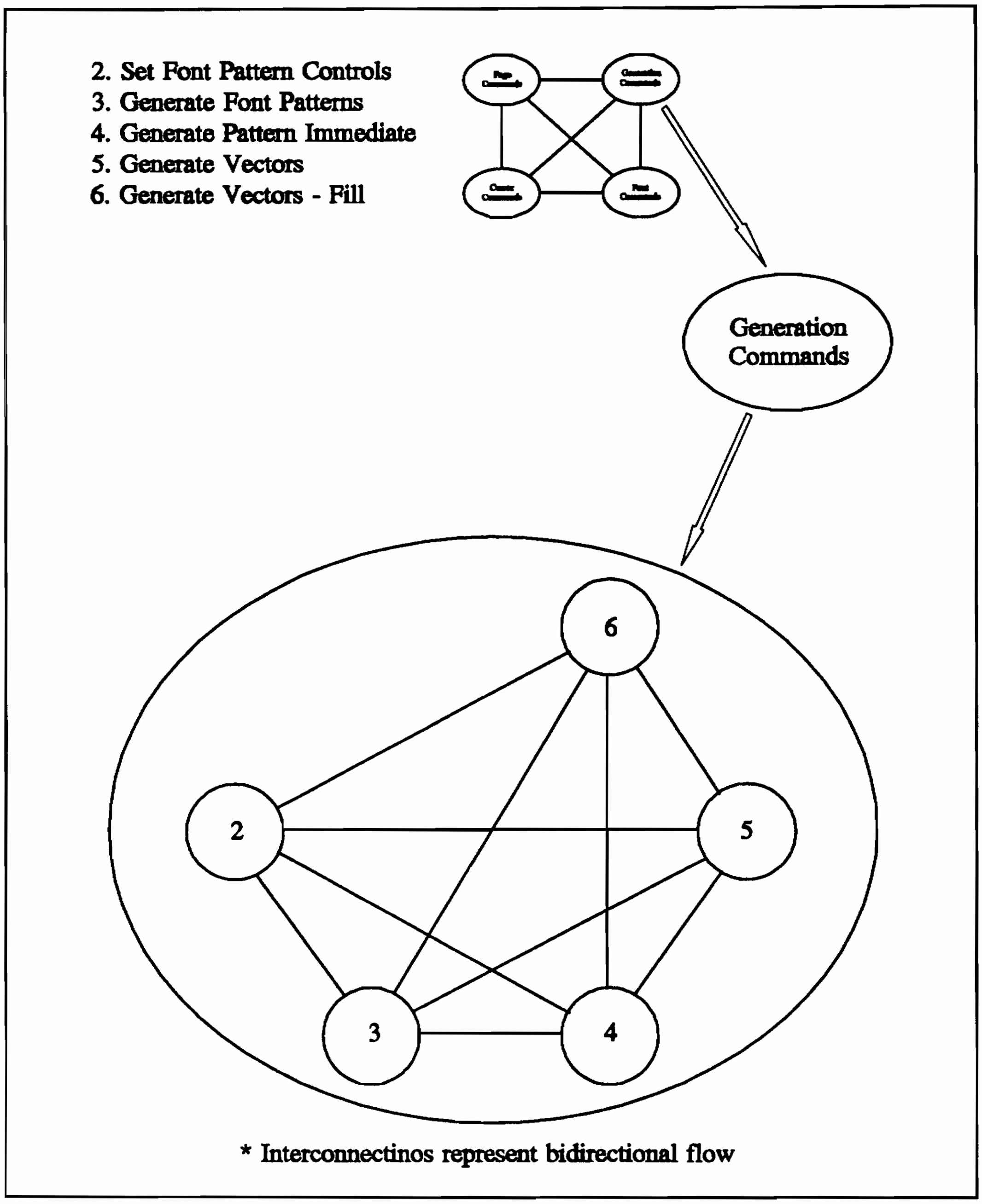

2. Set Font Pattern Controls
3. Generate Font Patterns
4. Generate Pattern Immediate
5. Generate Vectors
6. Generate Vectors - Fill
Generation Commands
6
2
5
3
4
* Interconnectinos represent bidirectional flow

Figure 19. States of PMP Font Commands

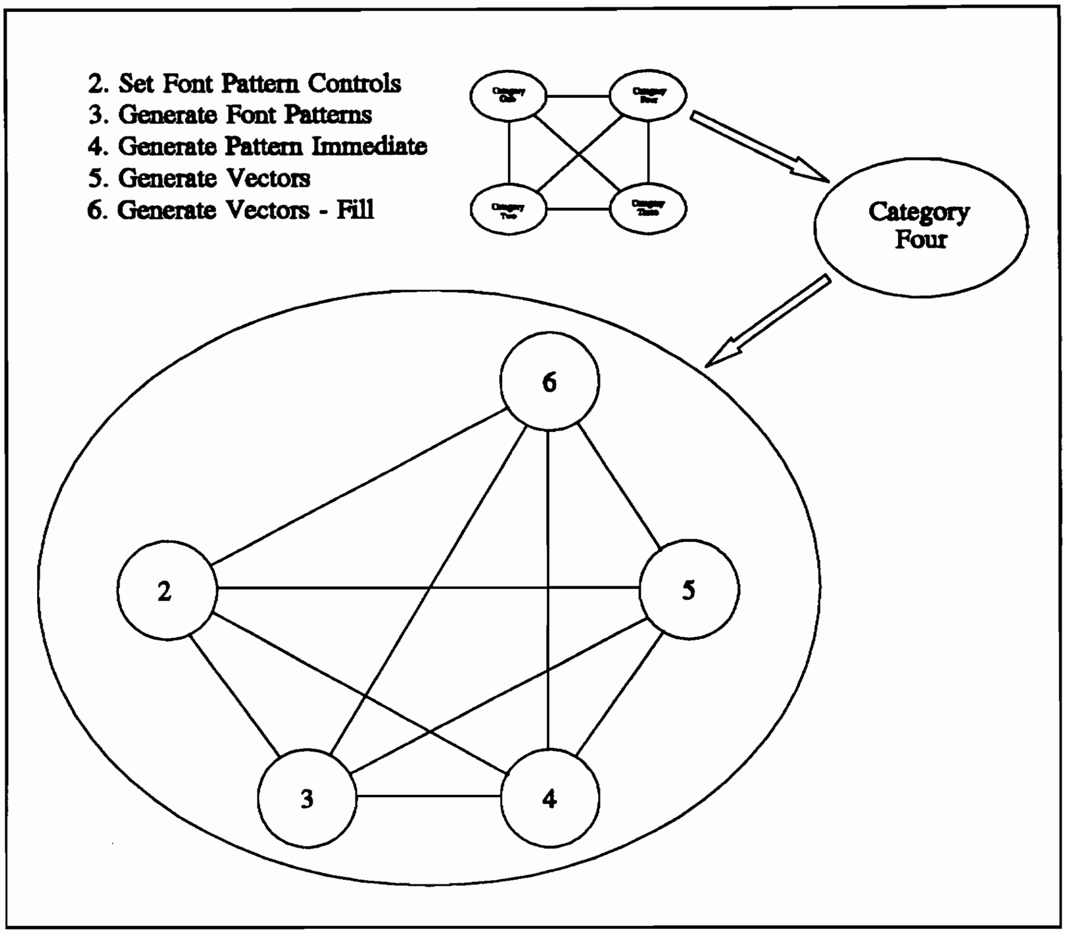

2. Set Font Pattern Controls
3. Generate Font Patterns
4. Generate Pattern Immediate
5. Generate Vectors
6. Generate Vectors - Fill
Category Four
6
2
5
3
4

Figure 20. States of PMP Generation Commands

5.3 Preview Implementation

5.3.1 PMP Font Index

Fonts contained within the PMP data stream are stored and referenced from the memory on the personal computer. A font index table is created in order to reference the fonts while creating the condensed page window and the explode window. The VM3812 printer service machine for the IBM 3812 Page Printer stores only those characters from a font used during printing. Characters within a font that are not used for printing are not incorporated into the PMP data stream file. This is a result of trying to minimize the size of the PMP data stream, which indirectly reduces the amount of memory needed to store and index the fonts when using them in the new preview state machine.

The characters which are printed are stored as simple raster patterns. Each character has several control bytes used by the printer state machine. The control bytes are the character reference number, the width of the character pattern, height of the character pattern, and the spacing of the character pattern with regard to the cursor line. Shown in Figure 21 is a typical character raster pattern with its control bytes.

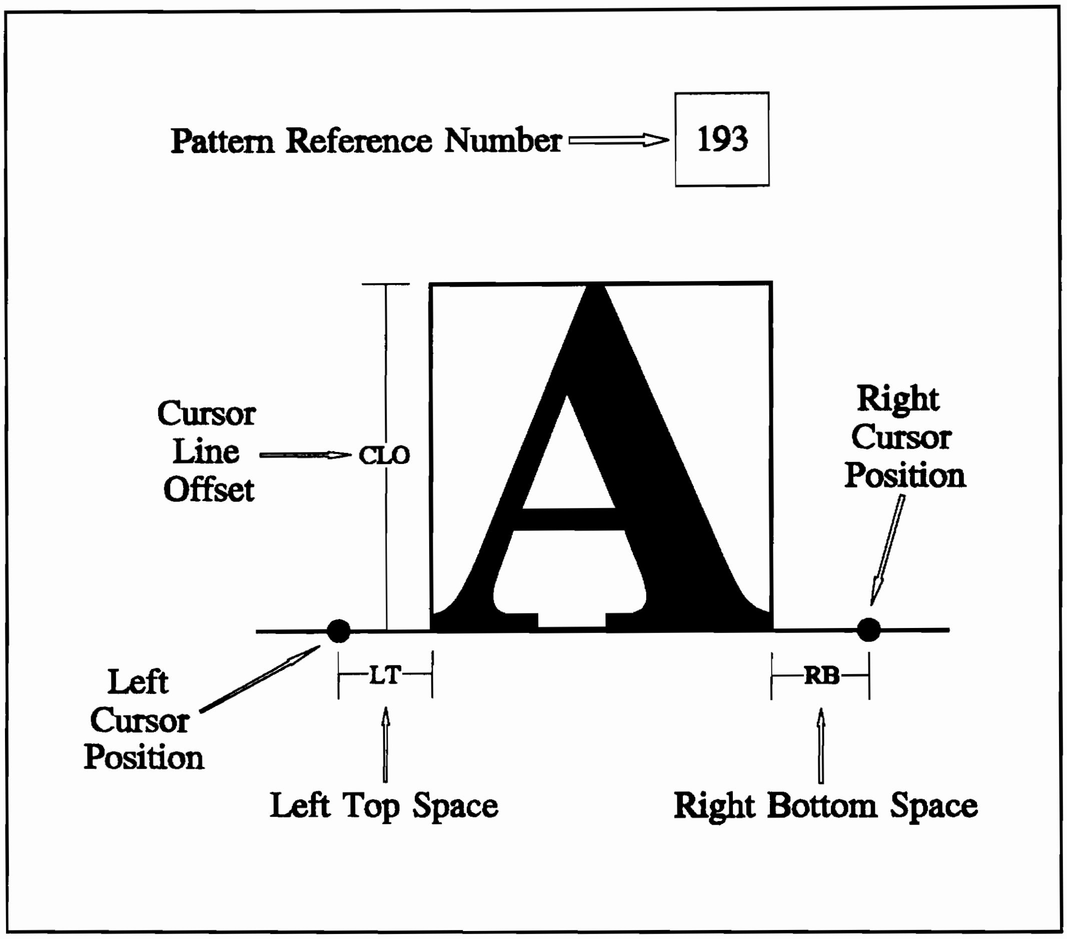


Figure 21. Typical Character Raster Pattern

On the IBM 3812 Page Printer, fonts are stored separately from each other and are loaded into the working memory as they are needed when forming the page memory map. On the personal computer, the font character patterns themselves are stored in memory as a linked list. Each member of the list has three distinct components. The first component is the font pattern reference number. This number is derived from a font code page.

The character pattern reference number is made up of a hexadecimal byte. The first digit of the byte is from the top row of the font table, and the second digit is from the left hand column of the table. Since the font tables are similar and in most cases the same from font to font, the same characters in different fonts usually have the same reference number. The structure of the font index with its components is shown in Figure 22.

The algorithm to copy the character raster patterns to the screen must first verify the character by referencing the font index. Each element in the font index has the font index structure shown in Figure 22. The font index structure itself is shown in Figure 23. The requested character number is first compared with the font index reference number, then the active font number is compared again the font index character font number, and if both match the character is verified and printed.

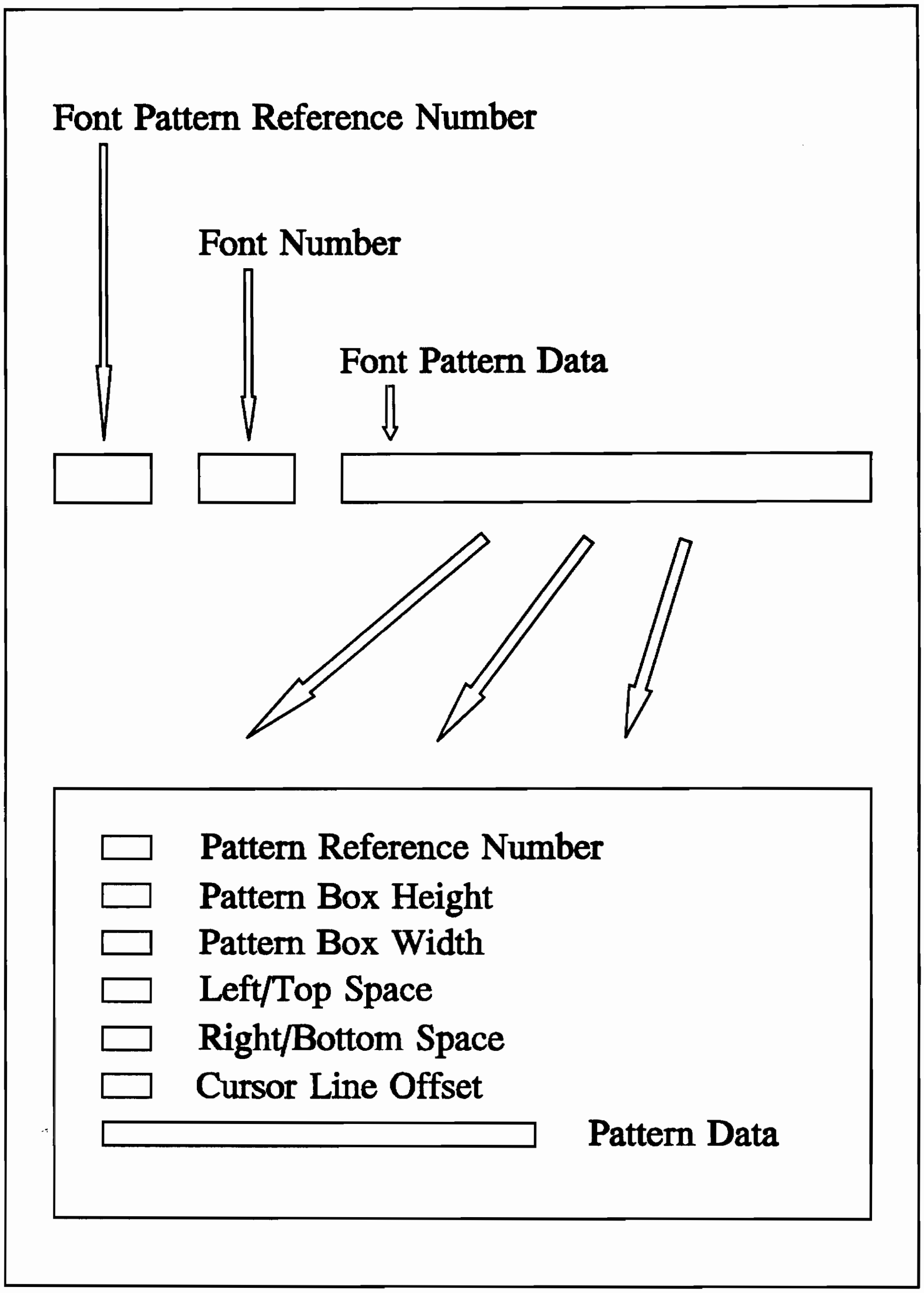


Figure 22. Font Index Structure

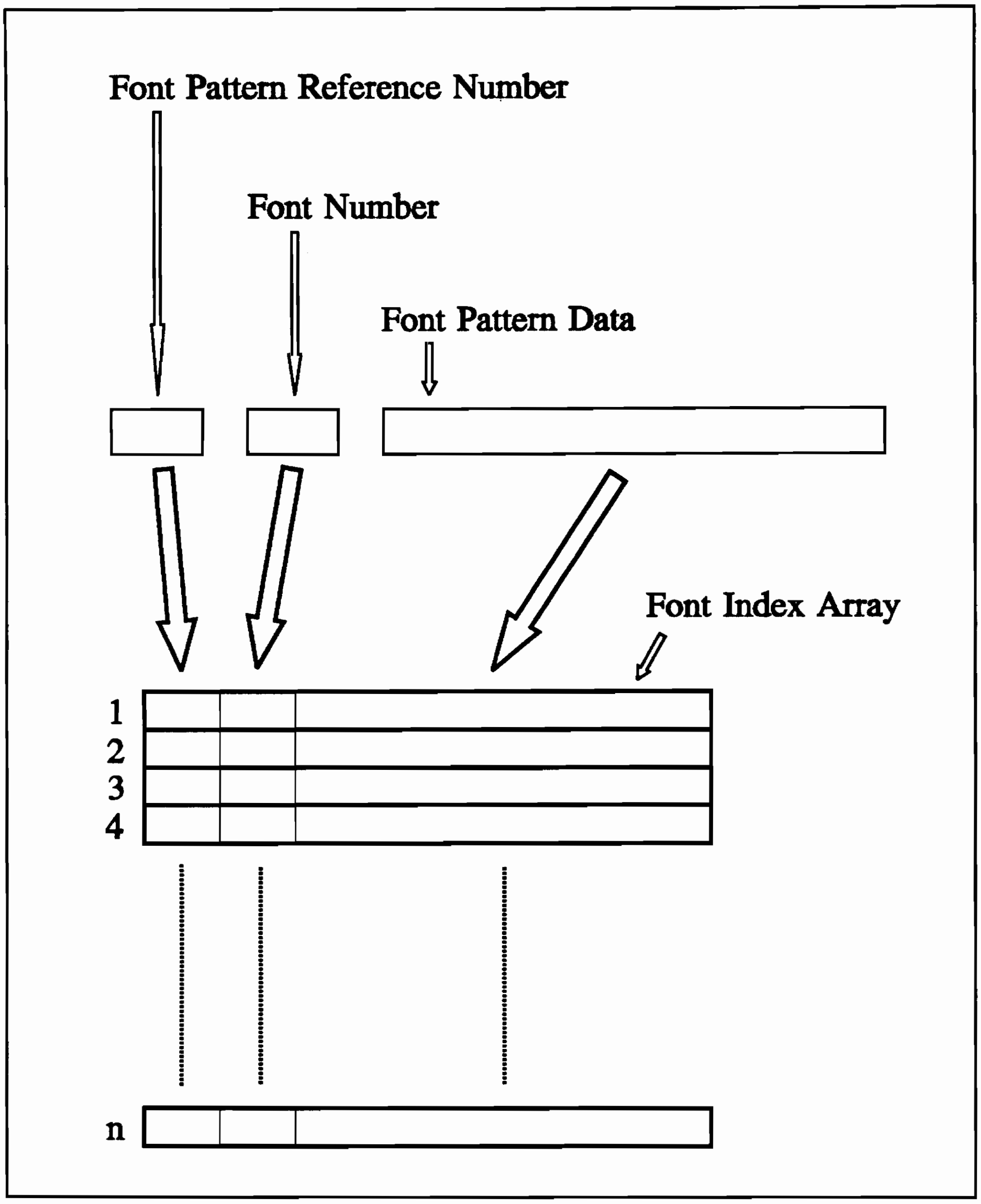


Figure 23. Font Reference Overview

5.3.2 PMP Print Block Buffer

A page print block is again defined as those PMP bytes which are needed in order to print an individual page of the document. These bytes within the page print blocks are also summed in order to reserve the correct amount of memory for the page print block index. An index by page number is created in order to reference the page print blocks. Figure 24 depicts the page block print buffer data structure. This enables the preview function to process a page without having to repeatedly process the entire PMP file. When a particular page is to be previewed, the program references the print block buffer index, and uses the appropriate page print block of PMP commands to generate the condensed page window and the explode area window. All PMP bytes which are not used to print, such as the load font command, are then discarded. Following the creation of the font index and the page print block index all processing of the PMP data stream is complete, and the PMP data stream file is closed. Characters are generated by referencing the font index, and images and vectors are generated by processing the respective PMP commands for images and vectors. All further action taken by the preview function uses the print page block index and the font index, which are both stored in memory.

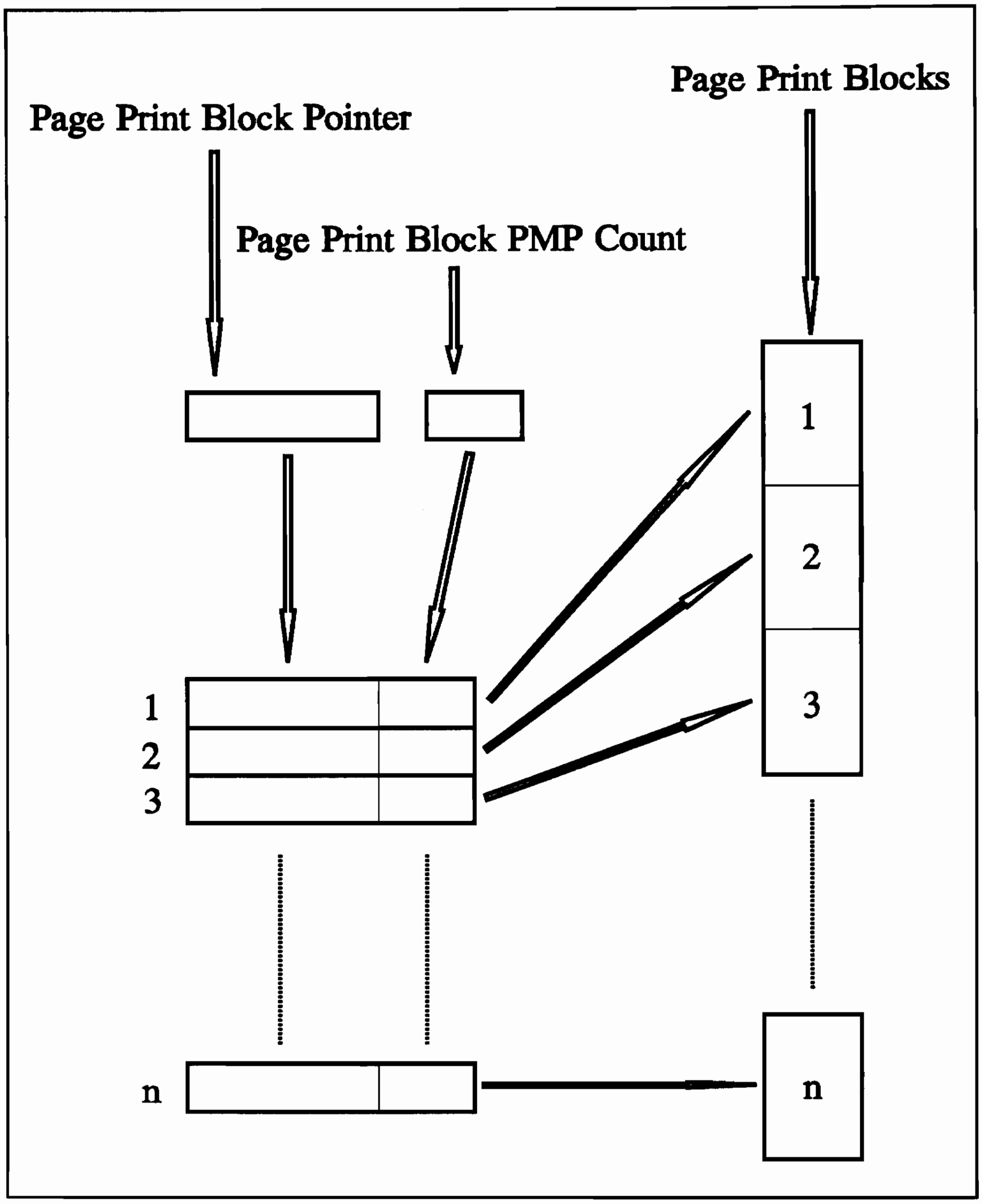


Figure 24. Page Print Block Structure Overview

Chapter 6. Further Development

6.1 General Steps for the Transfer of the Technology

The 3812 Printer state machine has been fully developed in a stand alone personal computer DOS environment. Chapter Five describes the steps of this development. However, in order to fully realize the objective, providing the new print preview methodology to the users of the SCRIPT process, the methodology implementation must be modified and reorganized further. The process is fully functional at this time in that SCRIPT documents may be fully previewed. However, in order to utilize the preview function the input to the 3812 printer, the PMP data stream, must be downloaded from the host and transported to a personal computer meeting the hardware and software prerequisites as outlined in Chapter Four.

Further development of the new preview methodology involves several steps. These steps include development of a new communications platform, personal computer to personal computer dialogue, transferal of the preview implementation to the host, and finally host to personal computer dialogue. The culmination of the development process will result in an interactive flow between the host and the personal computer during the print preview process. Figure 25 shows the development steps overview.

Parts of the implementation such as the graphical display, user interface, and the mouse driver will remain resident within the new communications software platform. The host will simply transfer data streams to the attached computer which consist of pure graphical data. Once the print preview has been invoked, the contents of the condensed page window, pure graphical data, will be transferred by the host to the personal computer. Once the user positions the index window on the condensed page window and selects an area for preview, the communications platform passes the index parameters to the host. Using these index parameters, the explode window is generated and sent to the personal computer. As far as the user is concerned, changes to the structure and implementation are transparent and the visual flow remains unchanged from that which is described in Chapter Four. The two data stream packets sent from the host containing the condensed page and the explode area represent data transfers sizes of 15.68K and 7.62K respectively. If the personal computers are attached to the host via data lines with reasonable bandwidths, the data transfer rates become negligible.

6.2 Enhanced Communications Platform

The first of the development steps as shown in Figure 25 is the enhancement and improvement of the communications software platform as described in Chapter Two. Full development of the communications software will not be discussed at this time.

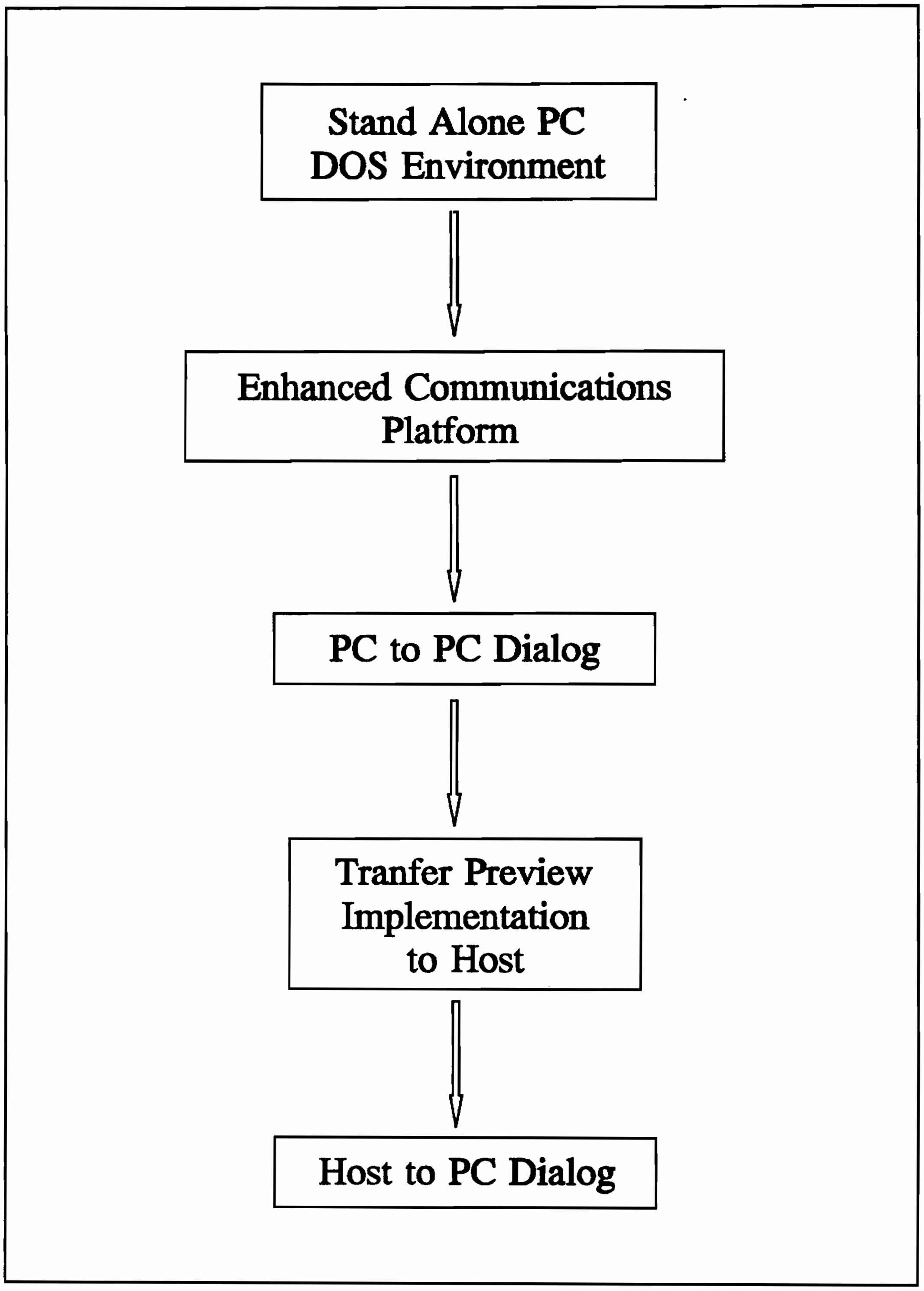


Figure 25. Further Development of Preview Methodology

However, an essential component of the print preview system is the correct implementation of dialogue between the host and the attached personal computer. This requires the communications platform to recognize sends and requests of certain parameters between the attached device and the host and also for the personal computer to receive data transferral of print preview data. As the condensed page and the explode area are sent to the attached personal computer, the resident communications software platform will dynamically place the graphical data in the appropriate location in video memory.

With regard to this particular point within the development chart, the component of the communications software platform which involves the print preview may be developed independently of the entire communications platform itself. Only a resident program need be developed which simply handles the user interface and placement of the condensed page and explode area on the personal computer screen.

6.3 Modifications to the Personal Computer Preview State Machine

Once the preview components of the new communications platform have been developed, the state machine implementation must also be modified. Instead of operating in a stand alone DOS environment on a personal computer, the preview process will instead operate between two personal computers. One machine will contain the print preview state machine, and one will contain the communications platform preview module.

The preview state machine resident on the personal computer will assume as its input sequence the PMP data stream. It will also assume this data stream is to be sent to a specified location when the print preview process is invoked by the user. Therefore, the interactive dialogue at this point in the development process involves a request from one personal computer (simulating the attached device) invoking the print process on the second personal computer (simulating the host). Figure 26 shows the dialogue flow as related to the print preview process during this stage of development.

6.4 Transfer of the Methodology to the Host

Once the communications platform preview module and the preview state machine are communicating satisfactorily, the print preview state machine must be transported to the host. This main complexity with regard to implementation relocation will involve language differences between the personal computer and the host. The communications dialogue will remain the same as that used during the personal computer to personal computer simulation. Thus the dialogue flow is the same as that shown above in Figure 26. Relocation of the preview state machine to the host should result in much faster computation times for the generation of the condensed page and the explode area. A design consideration at this point involves the generation and manipulation of the graphical data, the condensed page and the explode area. These data blocks may either

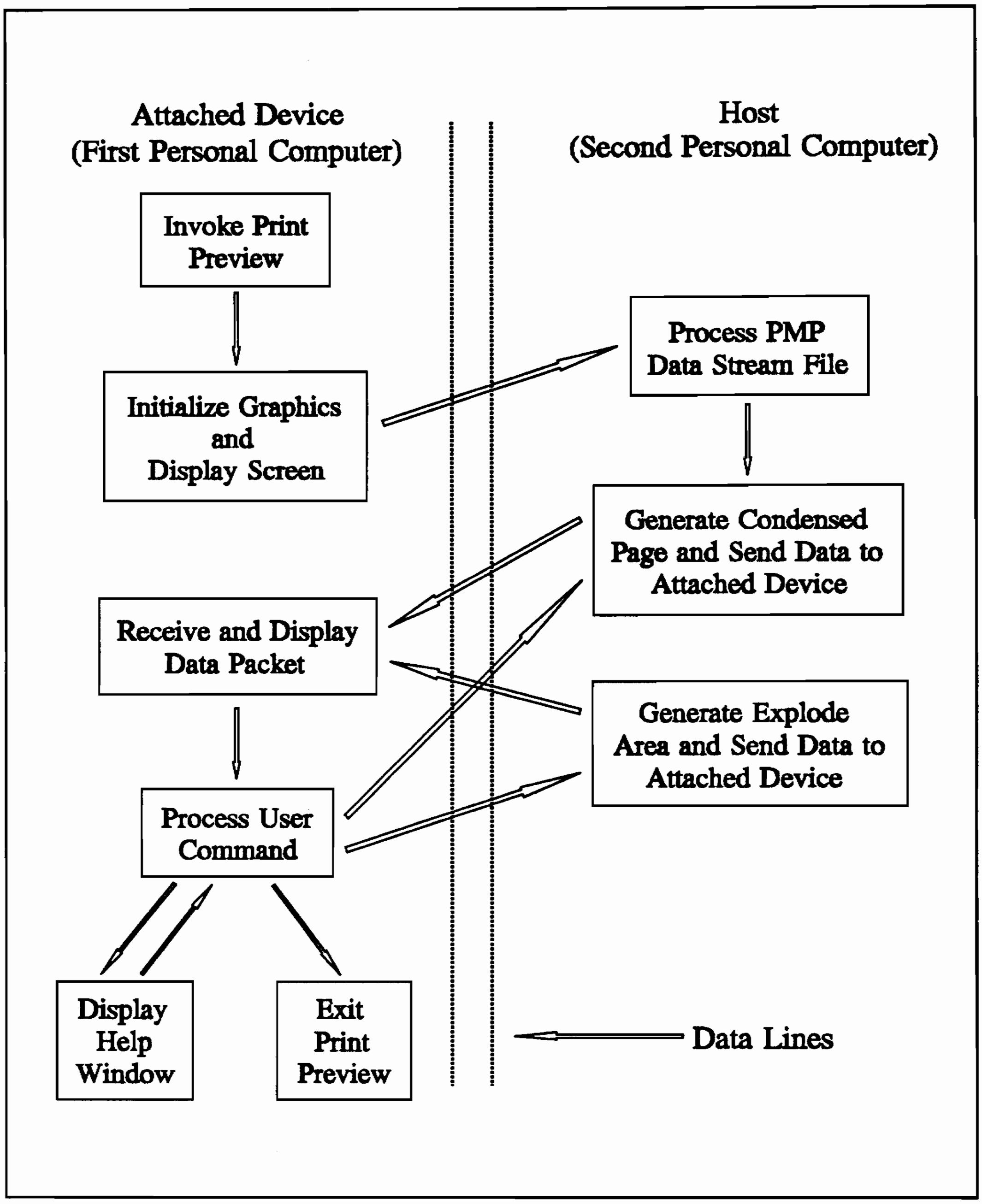

Attached Device
(First Personal Computer)
Host
(Second Personal Computer)
Invoke Print Preview
Process PMP Data Stream File
Initialize Graphics and Display Screen
Generate Condensed Page and Send Data to Attached Device
Receive and Display Data Packet
Generate Explode Area and Send Data to Attached Device
Process User Command
Display Help Window
Exit Print Preview
Data Lines

Figure 26. PC to PC Dialogue Development

be generated and dynamically sent to the attached personal computer as shown in Figure 15, or may be saved as a file and then transferred to the personal computer. The personal computer resident portion of the preview function will interpret both of these methods in the same manner. Either type of data block is first acknowledged by the personal computer. Then as the data blocks themselves are sent across the data lines, the preview component on the personal computer simply positions the graphical data into the appropriate window.

6.5 Means of PMP Data Stream Capture

Once the print preview process is invoked by the user on the attached personal computer, the preview state machine on the host must operate on the PMP data stream as discussed in Chapters Three, Four and Five. The difference however is the method of operation on the PMP data stream.

In order to properly make use of the preview methodology, the PMP data stream is still required as an input sequence to the preview state machine. However, a new method of capturing this data stream must be implemented. One possible solution is of course to employ a null VM3812 service machine in the same manner as discussed in Chapter Five. This printer service machine would capture the PMP data stream and save it as a file. The PMP data stream file would then be returned to the user's reader list. The preview state machine would then access by default the PMP data stream file through the user's reader list.

Chapter 7. Conclusion

The SCRIPT printing system which currently resides on the IBM 3090 provides excellent tools for users to edit, format, and print documents. However, the overall system configured with attached personal computers to the host has been described and its limitations have been outlined. Because of the limitations on this larger connective system between the attached personal computers and the host, there are inherent limitations placed on the printing system.

Improvements to the connective system have been discussed. Particularly, enhancements to the printing system, a subset or component of the connective system have been discussed and a new print preview methodology has been developed. This new print preview methodology maps the IBM 3812 printer finite state machine into a preview finite state machine. This process currently operates on a stand alone personal computer DOS environment. The methodology allows its users the benefit of creating a final draft prior to ever using a printer, thus saving both time and effort. Currently there are no preview methods which match the newly developed preview process methodology. The process creates an exact replica of documents during the preview process, which allows for tremendous accuracy when displaying all facets of a document, but limits the preview process in speed in comparison to other preview processes which operate on the personal computer.

Future research involves transporting the preview state machine to the host. Once the preview state machine is implemented on the host, the speed and power of the mainframe will increase the speed of the limiting factors of the preview process tremendously. When fully implemented on the host, the new print preview methodology will grant its users the best of both worlds - the complex graphical nature of the personal computer coupled with the speed and power of the IBM 3090 mainframe computer.

BIBLIOGRAPHY

1) J. Cavuoto, "**Successful Publishing's Secret?**", Computer Graphics World, Vol. 9, No. 11, November 1986, pp. 63-70.

2) R.K. deBry and B.G. Platte, "**Advanced Function Printing: A Tutorial**", IBM Systems Journal, Vol. 27, No. 2, pp. 219-233.

3) R.K. debry, B.G. Platte, C.L. Berinato, and J.W. Martin, "**Architectures of Advanced Function Printing**", IBM Systems Journal, Vol. 27, No. 2, pp. 234-245.

4) IBM Corporation, "**VM3812-IBM 3812 Pageprinter VM Support: Application Programmer's Guide**", 1985.

5) IBM Corporation, "**VM3812 User's Guide**", 1987.

6) IBM Corporation, "**Pageprinter 3812 Hardware Reference Library: Programming Reference**", 1985.

7) IBM Corporation, "**Pageprinter 3812 Hardware Reference Library: Introduction and Planning Guide**", 1985.

8) IBM Corporation, "**Pageprinter 3812 Hardware Reference Library: Guide to Operations**", 1985.

9) IBM Corporation, "**About Type, IBM's Technical Reference for 240-pel Digitized Type**", 1988.

10) IBM Corporation, "**IBM 3812 Pageprinter Technical Update**", 1987.

11) IBM Corporation, "**Intelligent Printer Data Stream Reference**", 1987.

12) IBM Corporation, "**Print Services Facility, Data Stream Reference**", 1988.

13) IBM Corporation, PROFS Note, "**The VM3812 Environment**", 1989.

14) IBM Corporation, PROFS Note, "**VM3812 Functional Characteristics**", 1989.

15) IBM Corporation, PROFS Note, "**VM3812 Performance**", 1989.

16) A.K. Griffee and C.A. Casey, "**An Introduction to Typographic Fonts and Digital Font Resources**", IBM Systems Journal, Vol. 27, No. 2, 1988, pp. 206-218.

17) D. McCammish, "**Desktop Publishing Drives a High-Tech Company**", Modern Office Technology, Vol. 33, No. 5, May 1988, pp. 80-88.

18) R.A. McGrath, "**Affordable PC Publishing**", Computer Graphics World, Vol. 10, No. 3, March 1987, pp. 113-118.

19) T.S. Perry, "**PostScript Prints Anything: A Case History**", IEEE Spectrum, Vol. 25, No. 5, May 1988, pp. 42-46.

Appendix A - PMP Command Listing

As discussed in Chapter Three, there are a total of thirty- three PMP commands which fall into five general categories. The PMP commands are given below in conjunction with a brief function description.

Page Commands

F6H - Set Page Size

The Set Page Size command defines the paper size to be used and lines up the page map with the correct paper tray.

D2H - Set Page Orientation

The Set Page Orientation command defines which side of the paper is to be treated as the top of the paper. Landscape and Portrait modes are standard modes, but the orientation of a page may be rotated by 90, 180, and 270 degrees, allowing for printing in any direction.

Page Commands (continued)

D1H - Print Page

The Print Page command causes the page memory map to be printed to paper. There are also options which allow the page map to be saved following the command, as well as an option which allows selection of an alternate paper tray source.

Cursor Commands

E0H - Set Cursor Horizontally

The Set Cursor Horizontally command sets the horizontal component of the cursor.

E1H - Set Cursor Vertically

The Set Cursor Vertically command sets the vertical component of the cursor.

E2H - Move Cursor Horizontally

Moves the position of the cursor right or left.

E3H - Move Cursor Vertically

Moves the position of the cursor either up or down.

Cursor Commands (continued)

8xH - Save Cursor

Saves the current cursor position in one of 15 cursor registers.

9xH - Restore Cursor

Changes the cursor position to the position saved in one of the 16 cursor registers.

E4H - Restore Cursor Component

Restores one of the cursor components - either horizontal or vertical - while leaving the other component unchanged.

Font Commands

D3H - Activate Font

This command is used for two purposes. If no patterns have been previously loaded in the font number being activated, it prepares that number for loading. If font patterns have been previously loaded into that font number, it makes those patterns available for immediate generation into the page memory map.

Font Commands (continued)

D7H - Activate Alternate Font

Activates the an alternate font. An alternate font is not mandatory. The alternate font may be used to back up the active font if the active font is missing some font patterns.

C2H - Deactivate Font

Deactivates the currently active font and the alternate font.

F0H - Load Font Pattern

Loads into the active font a specified pattern.

F4H - Load Large Font Pattern

Loads a large pattern into the active font.

E5H - Copy Font

Copies one loaded font to another.

FAH - Copy Font Pattern

Copies a single pattern from a specified font into the active font.

Font Commands (continued)

D4H - Unload Font

Clears a font from working storage.

C7H - Unload All Fonts

Removes all fonts from working storage and deactivates the active and alternate fonts.

DCH - Select/Deselect Font Emphasis

Allows any pattern loaded through the Load Font Pattern command to be loaded as a bold character.

Generation Commands

D9H - Set Generation Mode

Selects how pattern data bits are generated into page memory map pels. For example, it controls whether a 1 data bit means "black" or "white."

D8H - Set Font Pattern Controls

Sets up options for the command "Generate Font Patterns."

Generation Commands (continued)

0xxxxxxx - Generate Font Patterns

Generates a pattern or patterns form the active font into the page memory map. This is the PMP command you use for generating text.

F5H - Generate Pattern Immediate

Places a bit pattern directly into the page memory map at the current cursor position. This is the command used to generate bit image graphics.

F8H - Generate Vectors

Generates vectors in the page memory map by drawing lines between successive points, starting with the current cursor position.

F9H - Generate Vectors - Close and Fill

Creates a closed polygon that the Page Printer fills in after drawing the vectors.

Macro Commands

F7H - Load Macro

Lets a macro be defined.

Macro Commands (continued)

DAH - Execute Macro

Runs a macro from working storage.

A0H-BFH - Execute Macro - Short

Allows a one byte invocation of any one of the first thirty two macros in Page Printer working storage.

C8H - Unload All Macros

Unloads all normal macros from working storage.

FBH - Execute Library Macro

Searches for and runs a macro from the printer diskette.

C9H - Push State

Saves the state of the PMP processor.

CAH - Pop State

Restores the PMP state variables saved by a previous Push State command.

Appendix B - Capturing the PMP data stream

The input sequence to the IBM 3812 Page Printer is the PMP data stream. In order to develop the print preview state machine, the PMP data stream is captured as a file. To do this, a document is created using XEDIT. Once the document has been formatted using GML, it may be printed by using the SCRIPT process.

However, in order to capture the PMP data stream, the file is SCRIPTed to a dummy printer service machine. The command used to capture all of the test PMP data files used during the development of the new preview methodology is as follows:

```
SCRIPT filename filetype userdisk (d 3812 dest VM3812C
```

The invocation of this command simply returns the PMP data stream as a file to the users reader list with the same filename and filetype PMP3812. The file may then be received into the user's filelist. To transport the PMP data files to the attached environment, the following command is used:

```
PCTRANS DOWNB [drive:filename.ext] [filename PMP3812 userdisk]
```

a. "drive" is the destination drive on the personal computer.
b. "filename.ext" is the filename of the file on the personal computer.
c. "filename" is the filename of the document on the host.
d. "userdisk" is the drive where the file is located on the host.

Appendix C - Print Preview Implementation

The new print preview methodology was implemented in the C programming language. This appendix describes the steps taken to create the executable form of the program which describes the preview state machine.

The program, GDOIT.EXE, was written in C and was compiled using Turbo C version 2.0. There are twenty different source files used to create the program. The source files are given in Appendix D. In order to re-create the executable file, the project file must be loaded into Turbo C and autodependencies should be turned on. Code optimization should be set for speed, and code generation should be set for 8088/8086 code or 80186/80286 code generation as appropriate.

After GDOIT.EXE has been created, the files CGA.BGI and SANS.CHR must be present in the same directory at execution time in order to run the application. CGA.BGI is the graphics driver for MCGA and VGA HI graphics modes, and SANS.CHR is the character font used during the user interface (UI) display.

Appendix D - Extended PMP Flow Example

As an extended example of the PMP command structure the "SLIDE" SCRIPT document is presented. This document makes use of the majority of PMP commands, including multiple fonts, special math symbols, and vector graphics.

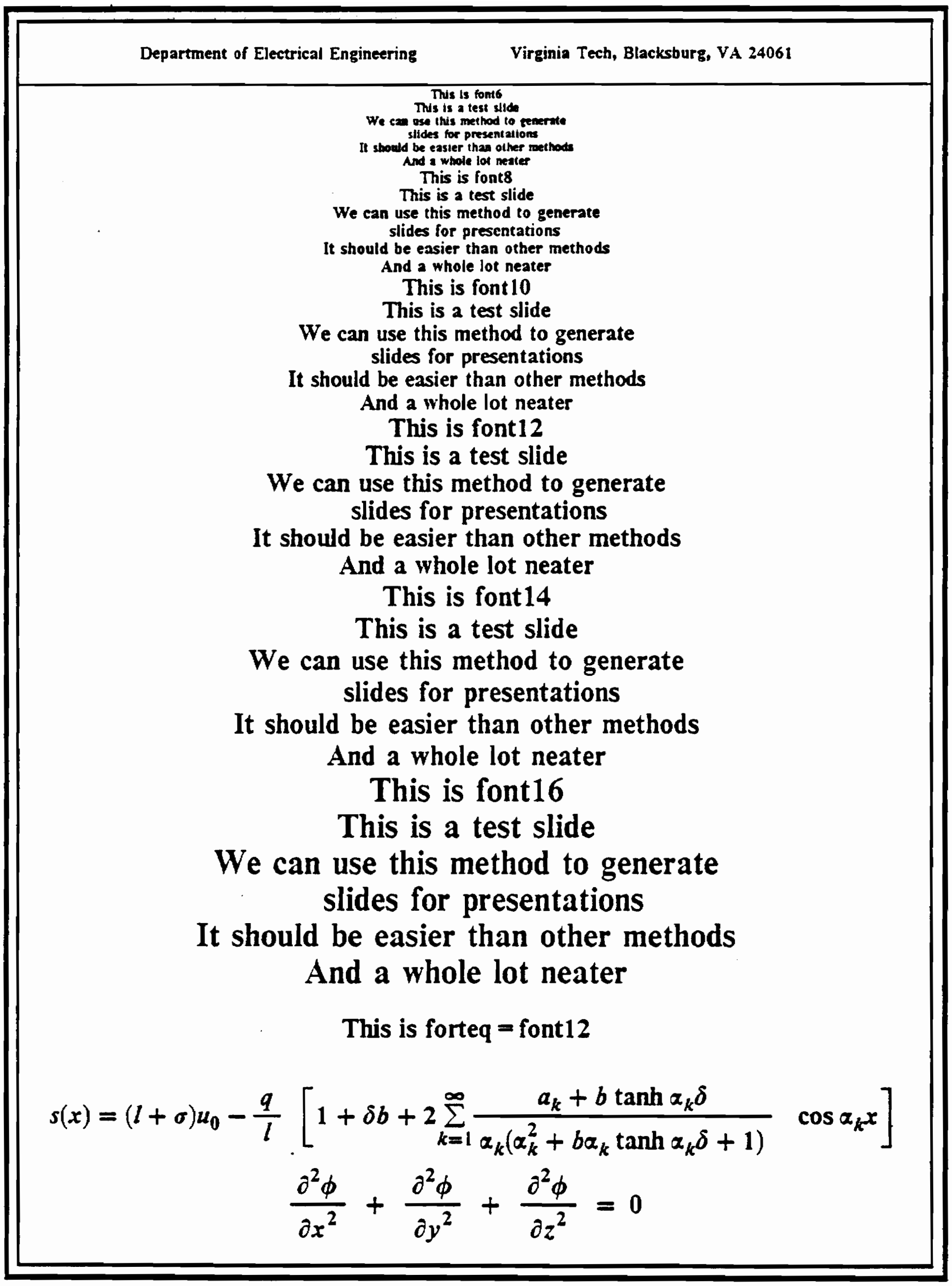
The example is given in four parts:

- SCRIPT Source file, p. 92
- Actual printed page of "SLIDE", p. 94
- Screen dump of GDOIT processing "SLIDE", p. 95
- PMP Command Flow Listing, pp. 96-116

SCRIPT source file for "SLIDE"

```
.pn off
.dr thick weight 1mm
.dr med   weight .5mm
.dr thin  weight .1mm
.df font16 type(16 bold)
.df font14 type(14 bold)
.df font12 type(12 bold)
.df font10 type(10 bold)
.df font8  type(8  bold)
.df font6  type(6  bold)
.df fonteq type(12 bold)
.bx thick 0cm 15.5cm
.bx new med 0.25cm 15.28cm
.in +0.75cm
.ir +0.75cm
.pn off
.ce on
.bf font8

Department of Electrical Engineering
Virginia Tech, Blacksburg, VA 24061

.pf
.hr thin 0.25cm 15.28cm
.bf font6
This is font6
This is a test slide
We can use this method to generate
   slides for presentations
It should be easier than other methods
And a whole lot neater
.pf
.bf font8
This is font8
This is a test slide
We can use this method to generate
   slides for presentations
It should be easier than other methods
And a whole lot neater
.pf
.bf font10
This is font10
This is a test slide
We can use this method to generate
   slides for presentations
It should be easier than other methods
And a whole lot neater
.pf
.bf font12
This is font12
This is a test slide
We can use this method to generate
   slides for presentations
It should be easier than other methods
And a whole lot neater
```

```
.pf
.bf font14
This is font14
This is a test slide
We can use this method to generate
   slides for presentations
It should be easier than other methods
And a whole lot neater
.pf
.bf font16
This is font16
This is a test slide
We can use this method to generate
   slides for presentations
It should be easier than other methods
And a whole lot neater
.pf
.bf fonteq

This is forteq=font12

.setsym smff
.se ask 'alpha sub k'
:df frame=none align=center.
s(x) = (1 + sigma) u sub 0 - q over 1 %%
  left lb 1 + delta b+2
    sum from <k=1> to infinity of
      <a sub k + b tanh &ask delta>
                  over
      <&ask (&ask sup 2 + b &ask
             tanh &ask delta + 1 ) > %%
    cos &ask x right rb
:edf.
:df frame=none align=center.
 <vardelta sup 2 phi> over <vardelta x sup 2> % + %
 <vardelta sup 2 phi> over <vardelta y sup 2> % + %
 <vardelta sup 2 phi> over <vardelta z sup 2> % = % 0
:edf.
.pf
.ce off
.bx off
.bx off
.pn off
```



Department of Electrical Engineering Virginia Tech, Blacksburg, VA 24061

This is font6
This is a test slide
We can use this method to generate
slides for presentations
It should be easier than other methods
And a whole lot neater

This is font8
This is a test slide
We can use this method to generate
slides for presentations
It should be easier than other methods
And a whole lot neater

This is font10
This is a test slide
We can use this method to generate
slides for presentations
It should be easier than other methods
And a whole lot neater

This is font12
This is a test slide
We can use this method to generate
slides for presentations
It should be easier than other methods
And a whole lot neater

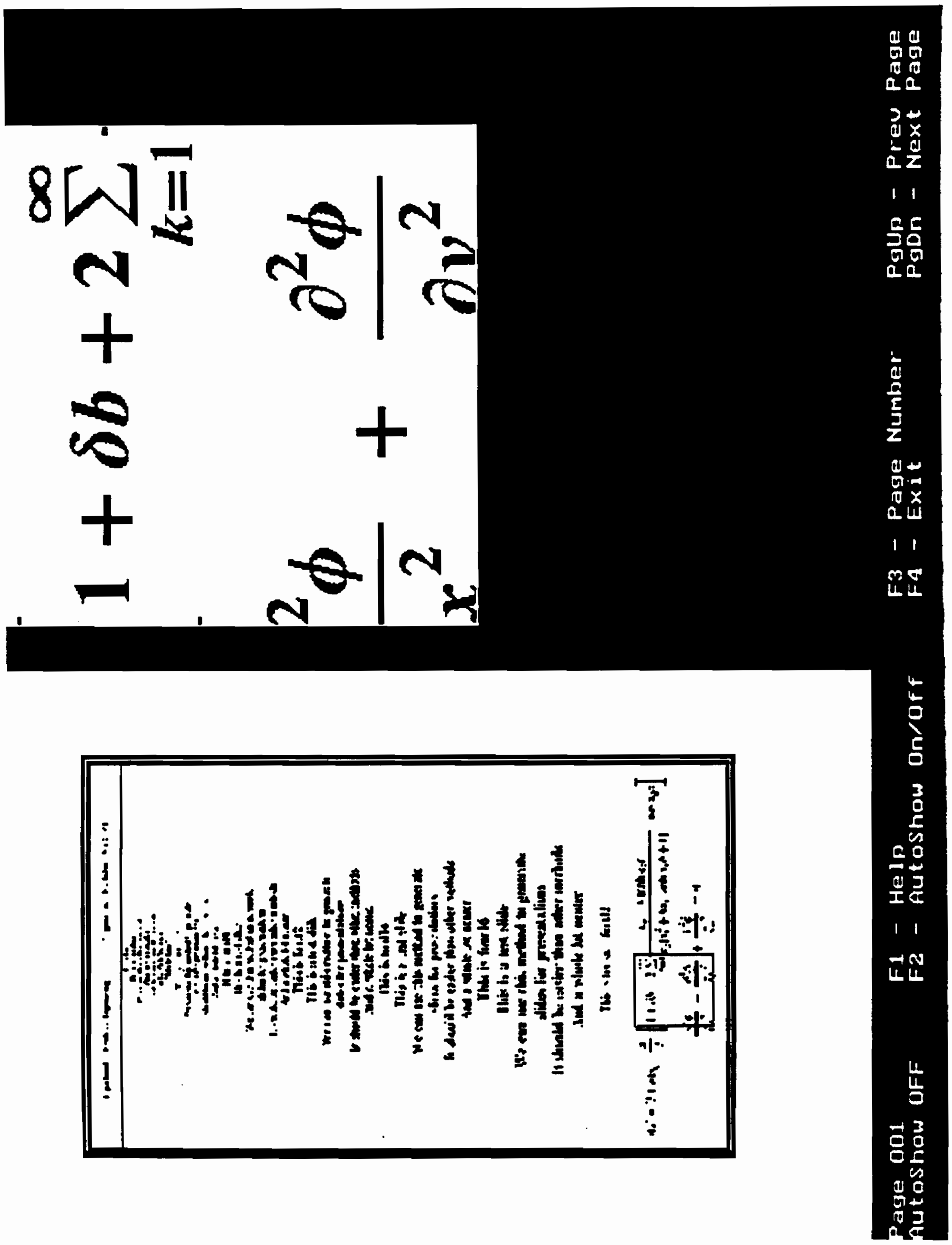
This is font14
This is a test slide
We can use this method to generate
slides for presentations
It should be easier than other methods
And a whole lot neater

This is font16
This is a test slide
We can use this method to generate
slides for presentations
It should be easier than other methods
And a whole lot neater

This is forteq = font12

$$s(x) = (l + \sigma)u_0 - \frac{q}{l}\left[1 + \delta b + 2\sum_{k=1}^{\infty}\frac{a_k + b\tanh\alpha_k\delta}{\alpha_k(\alpha_k^2 + b\alpha_k\tanh\alpha_k\delta + 1)}\cos\alpha_k x\right]$$

$$\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2} = 0$$


Page 001
AutoShow OFF
F1 - Help
F2 - AutoShow On/Off
F3 - Page Number
F4 - Exit
PgUp - Prev Page
PgDn - Next Page

```
Input file:  d:\slide.txt              20880 bytes

C6                          Jog Exit Tray
D8 00                       Set Font Pattern Controls
D9 90                       Set Generation Mode
E0 00 00                    Set Cursor Horizontally
E1 00 00                    Set Cursor Vertically
C2                          Deactivate Font
E1 00 97                    Set Cursor Vertically
E0 01 2C                    Set Cursor Horizontally
E2 02 C0                    Move Cursor Horizontally
E2 00 20                    Move Cursor Horizontally
C2                          Deactivate Font
E1 00 EC                    Set Cursor Vertically
E0 01 2C                    Set Cursor Horizontally
E3 FF F8                    Move Cursor Vertically
83                          Save Cursor to Register #3
E2 00 04                    Move Cursor Horizontally
E3 00 04                    Move Cursor Vertically
F8 C9 00 00 05 B3           Generate Vectors
93                          Restore Cursor from Register #3
E3 00 08                    Move Cursor Vertically
E1 00 ED                    Set Cursor Vertically
E0 01 2C                    Set Cursor Horizontally
83                          Save Cursor to Register #3
E3 00 04                    Move Cursor Vertically
E2 00 04                    Move Cursor Horizontally
F8 C9 00 08 00 00           Generate Vectors
93                          Restore Cursor from Register #3
E0 06 DF                    Set Cursor Horizontally
83                          Save Cursor to Register #3
E3 00 04                    Move Cursor Vertically
E2 00 04                    Move Cursor Horizontally
F8 C9 00 08 00 00           Generate Vectors
93                          Restore Cursor from Register #3
E1 00 FD                    Set Cursor Vertically
E0 01 40                    Set Cursor Horizontally
E3 FF FC                    Move Cursor Vertically
83                          Save Cursor to Register #3
E2 00 02                    Move Cursor Horizontally
E3 00 02                    Move Cursor Vertically
F8 C5 00 00 05 8C           Generate Vectors
93                          Restore Cursor from Register #3
E3 00 04                    Move Cursor Vertically
E1 00 FE                    Set Cursor Vertically
E0 01 2C                    Set Cursor Horizontally
83                          Save Cursor to Register #3
E3 00 04                    Move Cursor Vertically
E2 00 04                    Move Cursor Horizontally
F8 C9 07 9B 00 00           Generate Vectors
93                          Restore Cursor from Register #3
E0 01 40                    Set Cursor Horizontally
83                          Save Cursor to Register #3
E3 00 02                    Move Cursor Vertically
E2 00 02                    Move Cursor Horizontally
F8 C5 07 9F 00 00           Generate Vectors
```

```
93                         Restore Cursor from Register #3
E0 06 CC                   Set Cursor Horizontally
83                         Save Cursor to Register #3
E3 00 02                   Move Cursor Vertically
E2 00 02                   Move Cursor Horizontally
F8 C5 07 9F 00 00          Generate Vectors
93                         Restore Cursor from Register #3
E0 06 DF                   Set Cursor Horizontally
83                         Save Cursor to Register #3
E3 00 04                   Move Cursor Vertically
E2 00 04                   Move Cursor Horizontally
F8 C9 07 9B 00 00          Generate Vectors
93                         Restore Cursor from Register #3
E1 01 35                   Set Cursor Vertically
E0 01 2C                   Set Cursor Horizontally
E2 00 CE                   Move Cursor Horizontally
D3 04                      Activate Font #4
F0 C4 12 13 01 01 12       Load Font Pattern
F0 85 0D 0B 00 01 0D       Load Font Pattern
F0 97 12 0C 00 02 0D       Load Font Pattern
F0 81 0D 0C 01 01 0D       Load Font Pattern
F0 99 0D 0A 00 01 0D       Load Font Pattern
F0 A3 11 08 00 01 11       Load Font Pattern
F0 94 0D 14 00 01 0D       Load Font Pattern
F0 95 0D 0D 00 01 0D       Load Font Pattern
F0 40 00 00 00 09 00       Load Font Pattern
F0 96 0D 0B 01 02 0D       Load Font Pattern
F0 86 13 09 00 FF 13       Load Font Pattern
F0 C1 12 13 00 01 12       Load Font Pattern
F0 A2 0D 0A 00 01 0D       Load Font Pattern
F0 83 0D 0B 00 01 0D       Load Font Pattern
F0 C5 12 10 01 01 12       Load Font Pattern
F0 87 12 0C 01 01 0D       Load Font Pattern
F0 89 14 06 00 01 14       Load Font Pattern
0A C4 85 97 81 99 A3 94    Generate Font Patterns
85 95 A3                   Generate Font Patterns
E2 00 09                   Move Cursor Horizontally
02 96 86                   Generate Font Patterns
E2 00 09                   Move Cursor Horizontally
09 C1 85 99 96 A2 97 81    Generate Font Patterns
83 85                      Generate Font Patterns
E2 00 09                   Move Cursor Horizontally
0B C5 95 87 89 95 85 85    Generate Font Patterns
99 89 95 87                Generate Font Patterns
E2 00 90                   Move Cursor Horizontally
F0 E5 12 13 00 01 12       Load Font Pattern
F0 E3 12 10 01 01 12       Load Font Pattern
F0 88 12 0D 00 01 12       Load Font Pattern
F0 6B 09 05 01 01 05       Load Font Pattern
F0 C2 12 11 01 01 12       Load Font Pattern
F0 93 12 06 00 01 12       Load Font Pattern
F0 92 12 0E 00 00 12       Load Font Pattern
F0 82 12 0C 00 02 12       Load Font Pattern
F0 A4 0D 0D 00 01 0D       Load Font Pattern
F0 F2 12 0C 01 01 12       Load Font Pattern
```

```
F0 F4 12 0D 00 01 12        Load Font Pattern
F0 F0 12 0C 01 01 12        Load Font Pattern
F0 F6 12 0C 01 01 12        Load Font Pattern
F0 F1 12 09 02 03 12        Load Font Pattern
08 E5 89 99 87 89 95 89     Generate Font Patterns
81                          Generate Font Patterns
E2 00 09                    Move Cursor Horizontally
05 E3 85 83 88 6B           Generate Font Patterns
E2 00 09                    Move Cursor Horizontally
0B C2 93 81 83 92 A2 82     Generate Font Patterns
A4 99 87 6B                 Generate Font Patterns
E2 00 09                    Move Cursor Horizontally
02 E5 C1                    Generate Font Patterns
E2 00 09                    Move Cursor Horizontally
05 F2 F4 F0 F6 F1           Generate Font Patterns
E1 01 5E                    Set Cursor Vertically
E0 01 43                    Set Cursor Horizontally
E3 00 00                    Move Cursor Vertically
83                          Save Cursor to Register #3
E2 00 00                    Move Cursor Horizontally
E3 00 00                    Move Cursor Vertically
F8 C1 00 00 05 8C           Generate Vectors
93                          Restore Cursor from Register #3
E3 00 00                    Move Cursor Vertically
E1 01 73                    Set Cursor Vertically
E0 01 2C                    Set Cursor Horizontally
E2 02 98                    Move Cursor Horizontally
D3 05                       Activate Font #5
F0 E3 0E 0D 01 00 0E        Load Font Pattern
F0 88 0E 0A 00 00 0E        Load Font Pattern
F0 89 0E 05 00 01 0E        Load Font Pattern
F0 A2 0A 07 01 01 0A        Load Font Pattern
F0 40 00 00 00 07 00        Load Font Pattern
F0 86 0E 08 00 FF 0E        Load Font Pattern
F0 96 0A 09 00 01 0A        Load Font Pattern
F0 95 0A 0A 00 00 0A        Load Font Pattern
F0 A3 0E 06 00 01 0E        Load Font Pattern
F0 F6 0E 09 00 01 0E        Load Font Pattern
04 E3 88 89 A2              Generate Font Patterns
E2 00 07                    Move Cursor Horizontally
02 89 A2                    Generate Font Patterns
E2 00 07                    Move Cursor Horizontally
05 86 96 95 A3 F6           Generate Font Patterns
E1 01 88                    Set Cursor Vertically
E0 01 2C                    Set Cursor Horizontally
E2 02 7E                    Move Cursor Horizontally
F0 81 0A 09 01 00 0A        Load Font Pattern
F0 85 0A 08 00 01 0A        Load Font Pattern
F0 93 0E 05 00 01 0E        Load Font Pattern
F0 84 0E 0A 00 00 0E        Load Font Pattern
04 E3 88 89 A2              Generate Font Patterns
E2 00 07                    Move Cursor Horizontally
02 89 A2                    Generate Font Patterns
E2 00 07                    Move Cursor Horizontally
01 81                       Generate Font Patterns
```

```
E2 00 07                    Move Cursor Horizontally
04 A3 85 A2 A3              Generate Font Patterns
E2 00 07                    Move Cursor Horizontally
05 A2 93 89 84 85           Generate Font Patterns
E1 01 9D                    Set Cursor Vertically
E0 01 2C                    Set Cursor Horizontally
E2 02 34                    Move Cursor Horizontally
F0 E6 0E 15 00 00 0E        Load Font Pattern
F0 83 0A 07 01 01 0A        Load Font Pattern
F0 A4 0A 0A 00 00 0A        Load Font Pattern
F0 94 0A 0F 00 00 0A        Load Font Pattern
F0 87 0E 09 01 00 0A        Load Font Pattern
F0 99 0A 08 00 00 0A        Load Font Pattern
02 E6 85                    Generate Font Patterns
E2 00 07                    Move Cursor Horizontally
03 83 81 95                 Generate Font Patterns
E2 00 07                    Move Cursor Horizontally
03 A4 A2 85                 Generate Font Patterns
E2 00 07                    Move Cursor Horizontally
04 A3 88 89 A2              Generate Font Patterns
E2 00 07                    Move Cursor Horizontally
06 94 85 A3 88 96 84        Generate Font Patterns
E2 00 07                    Move Cursor Horizontally
02 A3 96                    Generate Font Patterns
E2 00 07                    Move Cursor Horizontally
08 87 85 95 85 99 81 A3     Generate Font Patterns
85                          Generate Font Patterns
E1 01 B2                    Set Cursor Vertically
E0 01 2C                    Set Cursor Horizontally
E2 02 60                    Move Cursor Horizontally
E2 00 15                    Move Cursor Horizontally
F0 97 0E 0A 00 01 0A        Load Font Pattern
06 A2 93 89 84 85 A2        Generate Font Patterns
E2 00 07                    Move Cursor Horizontally
03 86 96 99                 Generate Font Patterns
E2 00 07                    Move Cursor Horizontally
0D 97 99 85 A2 85 95 A3     Generate Font Patterns
81 A3 89 96 95 A2           Generate Font Patterns
E1 01 C7                    Set Cursor Vertically
E0 01 2C                    Set Cursor Horizontally
E2 02 29                    Move Cursor Horizontally
F0 C9 0E 07 01 00 0E        Load Font Pattern
F0 82 0E 0A 00 01 0E        Load Font Pattern
02 C9 A3                    Generate Font Patterns
E2 00 07                    Move Cursor Horizontally
06 A2 88 96 A4 93 84        Generate Font Patterns
E2 00 07                    Move Cursor Horizontally
02 82 85                    Generate Font Patterns
E2 00 07                    Move Cursor Horizontally
06 85 81 A2 89 85 99        Generate Font Patterns
E2 00 07                    Move Cursor Horizontally
04 A3 88 81 95              Generate Font Patterns
E2 00 07                    Move Cursor Horizontally
05 96 A3 88 85 99           Generate Font Patterns
E2 00 07                    Move Cursor Horizontally
```

```
07 94 85 A3 88 96 84 A2    Generate Font Patterns
E1 01 DC                   Set Cursor Vertically
E0 01 2C                   Set Cursor Horizontally
E2 02 6D                   Move Cursor Horizontally
F0 C1 0E 0F 00 00 0E       Load Font Pattern
F0 A6 0A 0E 00 00 0A       Load Font Pattern
03 C1 95 84                Generate Font Patterns
E2 00 07                   Move Cursor Horizontally
01 81                      Generate Font Patterns
E2 00 07                   Move Cursor Horizontally
05 A6 88 96 93 85          Generate Font Patterns
E2 00 07                   Move Cursor Horizontally
03 93 96 A3                Generate Font Patterns
E2 00 07                   Move Cursor Horizontally
06 95 85 81 A3 85 99       Generate Font Patterns
E1 01 F8                   Set Cursor Vertically
E0 01 2C                   Set Cursor Horizontally
E2 02 87                   Move Cursor Horizontally
D3 04                      Activate Font #4
F0 F8 12 0C 01 01 12       Load Font Pattern
04 E3 88 89 A2             Generate Font Patterns
E2 00 09                   Move Cursor Horizontally
02 89 A2                   Generate Font Patterns
E2 00 09                   Move Cursor Horizontally
05 86 96 95 A3 F8          Generate Font Patterns
E1 02 14                   Set Cursor Vertically
E0 01 2C                   Set Cursor Horizontally
E2 02 67                   Move Cursor Horizontally
F0 84 12 0D 01 00 12       Load Font Pattern
04 E3 88 89 A2             Generate Font Patterns
E2 00 09                   Move Cursor Horizontally
02 89 A2                   Generate Font Patterns
E2 00 09                   Move Cursor Horizontally
01 81                      Generate Font Patterns
E2 00 09                   Move Cursor Horizontally
04 A3 85 A2 A3             Generate Font Patterns
E2 00 09                   Move Cursor Horizontally
05 A2 93 89 84 85          Generate Font Patterns
E1 02 30                   Set Cursor Vertically
E0 01 2C                   Set Cursor Horizontally
E2 02 00                   Move Cursor Horizontally
F0 E6 12 1B 00 00 12       Load Font Pattern
02 E6 85                   Generate Font Patterns
E2 00 09                   Move Cursor Horizontally
03 83 81 95                Generate Font Patterns
E2 00 09                   Move Cursor Horizontally
03 A4 A2 85                Generate Font Patterns
E2 00 09                   Move Cursor Horizontally
04 A3 88 89 A2             Generate Font Patterns
E2 00 09                   Move Cursor Horizontally
06 94 85 A3 88 96 84       Generate Font Patterns
E2 00 09                   Move Cursor Horizontally
02 A3 96                   Generate Font Patterns
E2 00 09                   Move Cursor Horizontally
08 87 85 95 85 99 81 A3    Generate Font Patterns
```

```
85                            Generate Font Patterns
E1 02 4C                      Set Cursor Vertically
E0 01 2C                      Set Cursor Horizontally
E2 02 3E                      Move Cursor Horizontally
E2 00 1B                      Move Cursor Horizontally
06 A2 93 89 84 85 A2          Generate Font Patterns
E2 00 09                      Move Cursor Horizontally
03 86 96 99                   Generate Font Patterns
E2 00 09                      Move Cursor Horizontally
0D 97 99 85 A2 85 95 A3       Generate Font Patterns
81 A3 89 96 95 A2             Generate Font Patterns
E1 02 68                      Set Cursor Vertically
E0 01 2C                      Set Cursor Horizontally
E2 01 F1                      Move Cursor Horizontally
F0 C9 12 08 01 01 12          Load Font Pattern
02 C9 A3                      Generate Font Patterns
E2 00 09                      Move Cursor Horizontally
06 A2 88 96 A4 93 84          Generate Font Patterns
E2 00 09                      Move Cursor Horizontally
02 82 85                      Generate Font Patterns
E2 00 09                      Move Cursor Horizontally
06 85 81 A2 89 85 99          Generate Font Patterns
E2 00 09                      Move Cursor Horizontally
04 A3 88 81 95                Generate Font Patterns
E2 00 09                      Move Cursor Horizontally
05 96 A3 88 85 99             Generate Font Patterns
E2 00 09                      Move Cursor Horizontally
07 94 85 A3 88 96 84 A2       Generate Font Patterns
E1 02 84                      Set Cursor Vertically
E0 01 2C                      Set Cursor Horizontally
E2 02 4B                      Move Cursor Horizontally
F0 A6 0D 11 00 01 0D          Load Font Pattern
03 C1 95 84                   Generate Font Patterns
E2 00 09                      Move Cursor Horizontally
01 81                         Generate Font Patterns
E2 00 09                      Move Cursor Horizontally
05 A6 88 96 93 85             Generate Font Patterns
E2 00 09                      Move Cursor Horizontally
03 93 96 A3                   Generate Font Patterns
E2 00 09                      Move Cursor Horizontally
06 95 85 81 A3 85 99          Generate Font Patterns
E1 02 A8                      Set Cursor Vertically
E0 01 2C                      Set Cursor Horizontally
E2 02 6C                      Move Cursor Horizontally
D3 06                         Activate Font #6
F0 E3 18 16 01 00 18          Load Font Pattern
F0 88 18 10 01 01 18          Load Font Pattern
F0 89 18 07 01 01 18          Load Font Pattern
F0 A2 10 0C 01 00 10          Load Font Pattern
F0 40 00 00 00 0B 00          Load Font Pattern
F0 86 18 0D 01 FD 18          Load Font Pattern
F0 96 10 10 01 00 10          Load Font Pattern
F0 95 10 10 01 01 10          Load Font Pattern
F0 A3 16 0A 01 01 16          Load Font Pattern
F0 F1 18 0A 04 03 18          Load Font Pattern
```

```
F0 F0 18 10 01 00 18       Load Font Pattern
04 E3 88 89 A2             Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
02 89 A2                   Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
06 86 96 95 A3 F1 F0       Generate Font Patterns
E1 02 CC                   Set Cursor Vertically
E0 01 2C                   Set Cursor Horizontally
E2 02 4D                   Move Cursor Horizontally
F0 81 10 10 02 FF 10       Load Font Pattern
F0 85 10 0E 01 00 10       Load Font Pattern
F0 93 18 07 01 01 18       Load Font Pattern
F0 84 18 10 01 00 18       Load Font Pattern
04 E3 88 89 A2             Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
02 89 A2                   Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
01 81                      Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
04 A3 85 A2 A3             Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
05 A2 93 89 84 85          Generate Font Patterns
E1 02 F0                   Set Cursor Vertically
E0 01 2C                   Set Cursor Horizontally
E2 01 CB                   Move Cursor Horizontally
F0 E6 18 23 00 FF 18       Load Font Pattern
F0 83 10 0E 01 00 10       Load Font Pattern
F0 A4 10 10 01 01 10       Load Font Pattern
F0 94 10 19 01 01 10       Load Font Pattern
F0 87 17 11 01 00 10       Load Font Pattern
F0 99 10 0C 01 00 10       Load Font Pattern
02 E6 85                   Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
03 83 81 95                Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
03 A4 A2 85                Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
04 A3 88 89 A2             Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
06 94 85 A3 88 96 84       Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
02 A3 96                   Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
08 87 85 95 85 99 81 A3    Generate Font Patterns
85                         Generate Font Patterns
E1 03 14                   Set Cursor Vertically
E0 01 2C                   Set Cursor Horizontally
E2 02 1B                   Move Cursor Horizontally
E2 00 21                   Move Cursor Horizontally
F0 97 17 10 01 00 10       Load Font Pattern
06 A2 93 89 84 85 A2       Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
03 86 96 99                Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
0D 97 99 85 A2 85 95 A3    Generate Font Patterns
```

```
81 A3 89 96 95 A2          Generate Font Patterns
E1 03 38                   Set Cursor Vertically
E0 01 2C                   Set Cursor Horizontally
E2 01 BA                   Move Cursor Horizontally
F0 C9 18 0C 01 00 18       Load Font Pattern
F0 82 18 10 01 00 18       Load Font Pattern
02 C9 A3                   Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
06 A2 88 96 A4 93 84       Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
02 82 85                   Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
06 85 81 A2 89 85 99       Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
04 A3 88 81 95             Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
05 96 A3 88 85 99          Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
07 94 85 A3 88 96 84 A2    Generate Font Patterns
E1 03 5C                   Set Cursor Vertically
E0 01 2C                   Set Cursor Horizontally
E2 02 2A                   Move Cursor Horizontally
F0 C1 18 18 01 00 18       Load Font Pattern
F0 A6 10 17 00 00 10       Load Font Pattern
03 C1 95 84                Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
01 81                      Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
05 A6 88 96 93 85          Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
03 93 96 A3                Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
06 95 85 81 A3 85 99       Generate Font Patterns
E1 03 87                   Set Cursor Vertically
E0 01 2C                   Set Cursor Horizontally
E2 02 57                   Move Cursor Horizontally
D3 07                      Activate Font #7
F0 E3 1C 19 01 02 1C       Load Font Pattern
F0 88 1C 15 00 00 1C       Load Font Pattern
F0 89 1D 0A 00 01 1D       Load Font Pattern
F0 A2 14 0F 01 00 14       Load Font Pattern
F0 40 00 00 00 0D 00       Load Font Pattern
F0 86 1D 10 00 FE 1D       Load Font Pattern
F0 96 14 12 01 02 14       Load Font Pattern
F0 95 14 15 00 00 14       Load Font Pattern
F0 A3 1B 0D 00 01 1B       Load Font Pattern
F0 F1 1C 0D 03 05 1C       Load Font Pattern
F0 F2 1C 12 01 02 1C       Load Font Pattern
04 E3 88 89 A2             Generate Font Patterns
E2 00 0D                   Move Cursor Horizontally
02 89 A2                   Generate Font Patterns
E2 00 0D                   Move Cursor Horizontally
06 86 96 95 A3 F1 F2       Generate Font Patterns
E1 03 B2                   Set Cursor Vertically
E0 01 2C                   Set Cursor Horizontally
```

```
E2 02 33                    Move Cursor Horizontally
F0 81 14 13 01 00 14        Load Font Pattern
F0 85 14 10 01 01 14        Load Font Pattern
F0 93 1C 0A 00 01 1C        Load Font Pattern
F0 84 1C 13 01 01 1C        Load Font Pattern
04 E3 88 89 A2              Generate Font Patterns
E2 00 0D                    Move Cursor Horizontally
02 89 A2                    Generate Font Patterns
E2 00 0D                    Move Cursor Horizontally
01 81                       Generate Font Patterns
E2 00 0D                    Move Cursor Horizontally
04 A3 85 A2 A3              Generate Font Patterns
E2 00 0D                    Move Cursor Horizontally
05 A2 93 89 84 85           Generate Font Patterns
E1 03 DD                    Set Cursor Vertically
E0 01 2C                    Set Cursor Horizontally
E2 01 99                    Move Cursor Horizontally
F0 E6 1C 29 00 00 1C        Load Font Pattern
F0 83 14 10 01 01 14        Load Font Pattern
F0 A4 14 15 00 00 14        Load Font Pattern
F0 94 14 20 00 00 14        Load Font Pattern
F0 87 1C 12 01 02 14        Load Font Pattern
F0 99 14 0F 01 00 14        Load Font Pattern
02 E6 85                    Generate Font Patterns
E2 00 0D                    Move Cursor Horizontally
03 83 81 95                 Generate Font Patterns
E2 00 0D                    Move Cursor Horizontally
03 A4 A2 85                 Generate Font Patterns
E2 00 0D                    Move Cursor Horizontally
04 A3 88 89 A2              Generate Font Patterns
E2 00 0D                    Move Cursor Horizontally
06 94 85 A3 88 96 84        Generate Font Patterns
E2 00 0D                    Move Cursor Horizontally
02 A3 96                    Generate Font Patterns
E2 00 0D                    Move Cursor Horizontally
08 87 85 95 85 99 81 A3     Generate Font Patterns
85                          Generate Font Patterns
E1 04 08                    Set Cursor Vertically
E0 01 2C                    Set Cursor Horizontally
E2 01 F6                    Move Cursor Horizontally
E2 00 27                    Move Cursor Horizontally
F0 97 1C 13 01 01 14        Load Font Pattern
06 A2 93 89 84 85 A2        Generate Font Patterns
E2 00 0D                    Move Cursor Horizontally
03 86 96 99                 Generate Font Patterns
E2 00 0D                    Move Cursor Horizontally
0D 97 99 85 A2 85 95 A3     Generate Font Patterns
81 A3 89 96 95 A2           Generate Font Patterns
E1 04 33                    Set Cursor Vertically
E0 01 2C                    Set Cursor Horizontally
E2 01 83                    Move Cursor Horizontally
F0 C9 1C 0D 01 02 1C        Load Font Pattern
F0 82 1C 13 00 02 1C        Load Font Pattern
02 C9 A3                    Generate Font Patterns
E2 00 0D                    Move Cursor Horizontally
```

```
06 A2 88 96 A4 93 84       Generate Font Patterns
E2 00 0D                   Move Cursor Horizontally
02 82 85                   Generate Font Patterns
E2 00 0D                   Move Cursor Horizontally
06 85 81 A2 89 85 99       Generate Font Patterns
E2 00 0D                   Move Cursor Horizontally
04 A3 88 81 95             Generate Font Patterns
E2 00 0D                   Move Cursor Horizontally
05 96 A3 88 85 99          Generate Font Patterns
E2 00 0D                   Move Cursor Horizontally
07 94 85 A3 88 96 84 A2    Generate Font Patterns
E1 04 5E                   Set Cursor Vertically
E0 01 2C                   Set Cursor Horizontally
E2 02 0A                   Move Cursor Horizontally
F0 C1 1C 1D 00 01 1C       Load Font Pattern
F0 A6 14 1B 00 01 14       Load Font Pattern
03 C1 95 84                Generate Font Patterns
E2 00 0D                   Move Cursor Horizontally
01 81                      Generate Font Patterns
E2 00 0D                   Move Cursor Horizontally
05 A6 88 96 93 85          Generate Font Patterns
E2 00 0D                   Move Cursor Horizontally
03 93 96 A3                Generate Font Patterns
E2 00 0D                   Move Cursor Horizontally
06 95 85 81 A3 85 99       Generate Font Patterns
E1 04 90                   Set Cursor Vertically
E0 01 2C                   Set Cursor Horizontally
E2 02 4E                   Move Cursor Horizontally
D3 08                      Activate Font #8
F0 E3 1E 1B 01 02 1E       Load Font Pattern
F0 88 1E 15 01 00 1E       Load Font Pattern
F0 89 1F 0A 01 01 1F       Load Font Pattern
F0 A2 16 10 01 00 15       Load Font Pattern
F0 40 00 00 00 10 00       Load Font Pattern
F0 86 1F 10 01 FE 1F       Load Font Pattern
F0 96 16 14 01 01 15       Load Font Pattern
F0 95 15 15 01 00 15       Load Font Pattern
F0 A3 1D 0E 00 01 1C       Load Font Pattern
F0 F1 1F 0F 04 03 1F       Load Font Pattern
F0 F4 1E 15 01 00 1E       Load Font Pattern
04 E3 88 89 A2             Generate Font Patterns
E2 00 10                   Move Cursor Horizontally
02 89 A2                   Generate Font Patterns
E2 00 10                   Move Cursor Horizontally
06 86 96 95 A3 F1 F4       Generate Font Patterns
E1 04 C2                   Set Cursor Vertically
E0 01 2C                   Set Cursor Horizontally
E2 02 23                   Move Cursor Horizontally
F0 81 16 14 02 00 15       Load Font Pattern
F0 85 16 11 02 01 15       Load Font Pattern
F0 93 1E 0A 01 01 1E       Load Font Pattern
F0 84 1F 14 01 01 1E       Load Font Pattern
04 E3 88 89 A2             Generate Font Patterns
E2 00 10                   Move Cursor Horizontally
02 89 A2                   Generate Font Patterns
```

```
E2 00 10                    Move Cursor Horizontally
01 81                       Generate Font Patterns
E2 00 10                    Move Cursor Horizontally
04 A3 85 A2 A3              Generate Font Patterns
E2 00 10                    Move Cursor Horizontally
05 A2 93 89 84 85           Generate Font Patterns
E1 04 F4                    Set Cursor Vertically
E0 01 2C                    Set Cursor Horizontally
E2 01 7D                    Move Cursor Horizontally
F0 E6 1F 2C 00 00 1E        Load Font Pattern
F0 83 16 11 01 01 15        Load Font Pattern
F0 A4 15 15 01 00 14        Load Font Pattern
F0 94 15 20 01 00 15        Load Font Pattern
F0 87 1F 14 02 00 15        Load Font Pattern
F0 99 15 10 01 00 15        Load Font Pattern
02 E6 85                    Generate Font Patterns
E2 00 10                    Move Cursor Horizontally
03 83 81 95                 Generate Font Patterns
E2 00 10                    Move Cursor Horizontally
03 A4 A2 85                 Generate Font Patterns
E2 00 10                    Move Cursor Horizontally
04 A3 88 89 A2              Generate Font Patterns
E2 00 10                    Move Cursor Horizontally
06 94 85 A3 88 96 84        Generate Font Patterns
E2 00 10                    Move Cursor Horizontally
02 A3 96                    Generate Font Patterns
E2 00 10                    Move Cursor Horizontally
08 87 85 95 85 99 81 A3     Generate Font Patterns
85                          Generate Font Patterns
E1 05 26                    Set Cursor Vertically
E0 01 2C                    Set Cursor Horizontally
E2 01 E1                    Move Cursor Horizontally
E2 00 30                    Move Cursor Horizontally
F0 97 1F 14 01 01 15        Load Font Pattern
06 A2 93 89 84 85 A2        Generate Font Patterns
E2 00 10                    Move Cursor Horizontally
03 86 96 99                 Generate Font Patterns
E2 00 10                    Move Cursor Horizontally
0D 97 99 85 A2 85 95 A3     Generate Font Patterns
81 A3 89 96 95 A2           Generate Font Patterns
E1 05 58                    Set Cursor Vertically
E0 01 2C                    Set Cursor Horizontally
E2 01 66                    Move Cursor Horizontally
F0 C9 1E 0F 01 01 1E        Load Font Pattern
F0 82 1F 14 01 01 1E        Load Font Pattern
02 C9 A3                    Generate Font Patterns
E2 00 10                    Move Cursor Horizontally
06 A2 88 96 A4 93 84        Generate Font Patterns
E2 00 10                    Move Cursor Horizontally
02 82 85                    Generate Font Patterns
E2 00 10                    Move Cursor Horizontally
06 85 81 A2 89 85 99        Generate Font Patterns
E2 00 10                    Move Cursor Horizontally
04 A3 88 81 95              Generate Font Patterns
E2 00 10                    Move Cursor Horizontally
```

```
05 96 A3 88 85 99          Generate Font Patterns
E2 00 10                   Move Cursor Horizontally
07 94 85 A3 88 96 84 A2    Generate Font Patterns
E1 05 8A                   Set Cursor Vertically
E0 01 2C                   Set Cursor Horizontally
E2 01 F7                   Move Cursor Horizontally
F0 C1 1F 20 00 00 1F       Load Font Pattern
F0 A6 15 1E 00 00 14       Load Font Pattern
03 C1 95 84                Generate Font Patterns
E2 00 10                   Move Cursor Horizontally
01 81                      Generate Font Patterns
E2 00 10                   Move Cursor Horizontally
05 A6 88 96 93 85          Generate Font Patterns
E2 00 10                   Move Cursor Horizontally
03 93 96 A3                Generate Font Patterns
E2 00 10                   Move Cursor Horizontally
06 95 85 81 A3 85 99       Generate Font Patterns
E1 05 C3                   Set Cursor Vertically
E0 01 2C                   Set Cursor Horizontally
E2 02 39                   Move Cursor Horizontally
D3 09                      Activate Font #9
F0 E3 24 1F 02 02 24       Load Font Pattern
F0 88 24 19 00 01 24       Load Font Pattern
F0 89 24 0D 01 00 24       Load Font Pattern
F0 A2 1A 12 01 01 19       Load Font Pattern
F0 40 00 00 00 12 00       Load Font Pattern
F0 86 25 13 01 FD 25       Load Font Pattern
F0 96 1A 17 01 01 19       Load Font Pattern
F0 95 19 18 01 01 19       Load Font Pattern
F0 A3 22 0F 01 01 21       Load Font Pattern
F0 F1 24 12 04 04 24       Load Font Pattern
F0 F6 25 17 02 01 24       Load Font Pattern
04 E3 88 89 A2             Generate Font Patterns
E2 00 12                   Move Cursor Horizontally
02 89 A2                   Generate Font Patterns
E2 00 12                   Move Cursor Horizontally
06 86 96 95 A3 F1 F6       Generate Font Patterns
E1 05 FC                   Set Cursor Vertically
E0 01 2C                   Set Cursor Horizontally
E2 02 07                   Move Cursor Horizontally
F0 81 1A 17 02 01 19       Load Font Pattern
F0 85 1A 15 01 01 19       Load Font Pattern
F0 93 24 0D 01 00 24       Load Font Pattern
F0 84 25 18 01 01 24       Load Font Pattern
04 E3 88 89 A2             Generate Font Patterns
E2 00 12                   Move Cursor Horizontally
02 89 A2                   Generate Font Patterns
E2 00 12                   Move Cursor Horizontally
01 81                      Generate Font Patterns
E2 00 12                   Move Cursor Horizontally
04 A3 85 A2 A3             Generate Font Patterns
E2 00 12                   Move Cursor Horizontally
05 A2 93 89 84 85          Generate Font Patterns
E1 06 35                   Set Cursor Vertically
E0 01 2C                   Set Cursor Horizontally
```

```
E2 01 47                          Move Cursor Horizontally
F0 E6 25 33 00 00 24              Load Font Pattern
F0 83 1A 15 01 01 19              Load Font Pattern
F0 A4 19 19 00 01 18              Load Font Pattern
F0 94 19 25 01 01 19              Load Font Pattern
F0 87 24 17 02 01 19              Load Font Pattern
F0 99 19 13 00 01 19              Load Font Pattern
02 E6 85                          Generate Font Patterns
E2 00 12                          Move Cursor Horizontally
03 83 81 95                       Generate Font Patterns
E2 00 12                          Move Cursor Horizontally
03 A4 A2 85                       Generate Font Patterns
E2 00 12                          Move Cursor Horizontally
04 A3 88 89 A2                    Generate Font Patterns
E2 00 12                          Move Cursor Horizontally
06 94 85 A3 88 96 84              Generate Font Patterns
E2 00 12                          Move Cursor Horizontally
02 A3 96                          Generate Font Patterns
E2 00 12                          Move Cursor Horizontally
08 87 85 95 85 99 81 A3           85 Generate Font Patterns
E1 06 6E                          Set Cursor Vertically
E0 01 2C                          Set Cursor Horizontally
E2 01 BC                          Move Cursor Horizontally
E2 00 36                          Move Cursor Horizontally
F0 97 24 18 01 01 19              Load Font Pattern
06 A2 93 89 84 85 A2              Generate Font Patterns
E2 00 12                          Move Cursor Horizontally
03 86 96 99                       Generate Font Patterns
E2 00 12                          Move Cursor Horizontally
0D 97 99 85 A2 85 95 A3           Generate Font Patterns
81 A3 89 96 95 A2                 Generate Font Patterns
E1 06 A7                          Set Cursor Vertically
E0 01 2C                          Set Cursor Horizontally
E2 01 2C                          Move Cursor Horizontally
F0 C9 24 12 01 01 24              Load Font Pattern
F0 82 25 19 00 01 24              Load Font Pattern
02 C9 A3                          Generate Font Patterns
E2 00 12                          Move Cursor Horizontally
06 A2 88 96 A4 93 84              Generate Font Patterns
E2 00 12                          Move Cursor Horizontally
02 82 85                          Generate Font Patterns
E2 00 12                          Move Cursor Horizontally
06 85 81 A2 89 85 99              Generate Font Patterns
E2 00 12                          Move Cursor Horizontally
04 A3 88 81 95                    Generate Font Patterns
E2 00 12                          Move Cursor Horizontally
05 96 A3 88 85 99                 Generate Font Patterns
E2 00 12                          Move Cursor Horizontally
07 94 85 A3 88 96 84 A2           Generate Font Patterns
E1 06 E0                          Set Cursor Vertically
E0 01 2C                          Set Cursor Horizontally
E2 01 D5                          Move Cursor Horizontally
F0 C1 25 24 00 01 25              Load Font Pattern
F0 A6 19 22 01 00 18              Load Font Pattern
03 C1 95 84                       Generate Font Patterns
```

```
E2 00 12                    Move Cursor Horizontally
01 81                       Generate Font Patterns
E2 00 12                    Move Cursor Horizontally
05 A6 88 96 93 85           Generate Font Patterns
E2 00 12                    Move Cursor Horizontally
03 93 96 A3                 Generate Font Patterns
E2 00 12                    Move Cursor Horizontally
06 95 85 81 A3 85 99        Generate Font Patterns
E1 07 36                    Set Cursor Vertically
E0 01 2C                    Set Cursor Horizontally
E2 02 0F                    Move Cursor Horizontally
D3 07                       Activate Font #7
F0 98 1C 13 01 01 14        Load Font Pattern
F0 7E 09 17 09 09 13        Load Font Pattern
04 E3 88 89 A2              Generate Font Patterns
E2 00 0D                    Move Cursor Horizontally
02 89 A2                    Generate Font Patterns
E2 00 0D                    Move Cursor Horizontally
0D 86 96 99 A3 85 98 7E     Generate Font Patterns
86 96 95 A3 F1 F2           Generate Font Patterns
E1 07 E4                    Set Cursor Vertically
E0 01 2C                    Set Cursor Horizontally
E2 00 47                    Move Cursor Horizontally
E3 FF DB                    Move Cursor Vertically
D3 0B                       Activate Font #11
F0 A2 13 0F 01 00 13        Load Font Pattern
01 A2                       Generate Font Patterns
D3 0C                       Activate Font #12
F0 4C 27 0A 03 01 1E        Load Font Pattern
01 4C                       Generate Font Patterns
D3 0B                       Activate Font #11
F0 A7 13 14 00 02 13        Load Font Pattern
01 A7                       Generate Font Patterns
D3 0C                       Activate Font #12
F0 5C 27 0A 01 02 1E        Load Font Pattern
F0 40 00 00 00 0E 00        Load Font Pattern
01 5C                       Generate Font Patterns
E2 00 0A                    Move Cursor Horizontally
D3 0A                       Activate Font #10
F0 7E 0C 1A 03 03 11        Load Font Pattern
F0 40 00 00 00 0E 00        Load Font Pattern
01 7E                       Generate Font Patterns
E2 00 0A                    Move Cursor Horizontally
D3 0C                       Activate Font #12
01 4C                       Generate Font Patterns
D3 0B                       Activate Font #11
F0 93 1D 0D 01 FD 1D        Load Font Pattern
F0 40 00 00 00 0D 00        Load Font Pattern
01 93                       Generate Font Patterns
E2 00 0A                    Move Cursor Horizontally
D3 0A                       Activate Font #10
F0 4E 1A 1A 03 03 18        Load Font Pattern
F0 A2 12 14 01 02 12        Load Font Pattern
01 4E                       Generate Font Patterns
E2 00 0A                    Move Cursor Horizontally
```

```
01 A2                           Generate Font Patterns
D3 0C                           Activate Font #12
01 5C                           Generate Font Patterns
D3 0B                           Activate Font #11
F0 A4 13 13 01 01 13            Load Font Pattern
01 A4                           Generate Font Patterns
E2 FF FF                        Move Cursor Horizontally
E3 00 0A                        Move Cursor Vertically
D3 03                           Activate Font #3
F0 F0 15 0E 01 01 15            Load Font Pattern
F0 40 00 00 00 09 00            Load Font Pattern
01 F0                           Generate Font Patterns
E2 00 0A                        Move Cursor Horizontally
E3 FF F6                        Move Cursor Vertically
D3 0A                           Activate Font #10
F0 60 04 1A 03 03 0D            Load Font Pattern
01 60                           Generate Font Patterns
E2 00 14                        Move Cursor Horizontally
E3 FF E3                        Move Cursor Vertically
D3 0B                           Activate Font #11
F0 98 1B 13 01 01 13            Load Font Pattern
01 98                           Generate Font Patterns
E2 FF E1                        Move Cursor Horizontally
E3 00 10                        Move Cursor Vertically
83                              Save Cursor to Register #3
E2 00 01                        Move Cursor Horizontally
E3 00 01                        Move Cursor Vertically
F8 C3 00 00 00 27               Generate Vectors
93                              Restore Cursor from Register #3
E2 00 2A                        Move Cursor Horizontally
E2 FF E5                        Move Cursor Horizontally
E3 00 28                        Move Cursor Vertically
01 93                           Generate Font Patterns
E2 00 33                        Move Cursor Horizontally
E3 00 08                        Move Cursor Vertically
83                              Save Cursor to Register #3
E2 00 01                        Move Cursor Horizontally
E3 00 01                        Move Cursor Vertically
F8 C2 00 00 00 0E               Generate Vectors
93                              Restore Cursor from Register #3
E2 00 10                        Move Cursor Horizontally
E2 FF F0                        Move Cursor Horizontally
E3 FF A4                        Move Cursor Vertically
83                              Save Cursor to Register #3
E2 00 01                        Move Cursor Horizontally
E3 00 01                        Move Cursor Vertically
F8 C2 00 00 00 0E               Generate Vectors
93                              Restore Cursor from Register #3
E2 00 10                        Move Cursor Horizontally
E2 FF F0                        Move Cursor Horizontally
83                              Save Cursor to Register #3
E3 00 02                        Move Cursor Vertically
E2 00 02                        Move Cursor Horizontally
F8 C4 00 5A 00 00               Generate Vectors
93                              Restore Cursor from Register #3
```

```
E2 00 04                   Move Cursor Horizontally
E2 00 10                   Move Cursor Horizontally
E3 00 39                   Move Cursor Vertically
D3 07                      Activate Font #7
01 F1                      Generate Font Patterns
E2 00 0A                   Move Cursor Horizontally
D3 0A                      Activate Font #10
F0 84 1E 12 01 02 1E       Load Font Pattern
01 4E                      Generate Font Patterns
E2 00 0A                   Move Cursor Horizontally
01 84                      Generate Font Patterns
D3 0B                      Activate Font #11
F0 82 1D 12 01 02 1D       Load Font Pattern
01 82                      Generate Font Patterns
E2 00 0A                   Move Cursor Horizontally
D3 0A                      Activate Font #10
01 4E                      Generate Font Patterns
E2 00 0A                   Move Cursor Horizontally
D3 07                      Activate Font #7
01 F2                      Generate Font Patterns
E2 00 0B                   Move Cursor Horizontally
E3 FF FD                   Move Cursor Vertically
D3 0C                      Activate Font #12
F0 80 25 20 00 03 1D       Load Font Pattern
01 80                      Generate Font Patterns
E2 FF DE                   Move Cursor Horizontally
E3 FF DC                   Move Cursor Vertically
D3 0E                      Activate Font #14
F0 B1 0B 1D 01 02 0B       Load Font Pattern
01 B1                      Generate Font Patterns
E2 FF D4                   Move Cursor Horizontally
E3 00 48                   Move Cursor Vertically
D3 0D                      Activate Font #13
F0 92 15 10 01 00 15       Load Font Pattern
01 92                      Generate Font Patterns
D3 0E                      Activate Font #14
F0 7E 09 14 02 02 0C       Load Font Pattern
01 7E                      Generate Font Patterns
D3 03                      Activate Font #3
F0 F1 15 08 04 04 15       Load Font Pattern
01 F1                      Generate Font Patterns
E2 00 63                   Move Cursor Horizontally
E3 FF C0                   Move Cursor Vertically
D3 0B                      Activate Font #11
F0 81 13 13 01 01 13       Load Font Pattern
01 81                      Generate Font Patterns
E2 FF FF                   Move Cursor Horizontally
E3 00 0A                   Move Cursor Vertically
D3 0D                      Activate Font #13
01 92                      Generate Font Patterns
F0 40 00 00 00 09 00       Load Font Pattern
E2 00 0A                   Move Cursor Horizontally
E3 FF F6                   Move Cursor Vertically
D3 0A                      Activate Font #10
01 4E                      Generate Font Patterns
```

```
E2 00 0A                    Move Cursor Horizontally
D3 0B                       Activate Font #11
01 82                       Generate Font Patterns
E2 00 0A                    Move Cursor Horizontally
D3 07                       Activate Font #7
04 A3 81 95 88              Generate Font Patterns
E2 00 0A                    Move Cursor Horizontally
D3 0A                       Activate Font #10
F0 81 13 13 01 01 13        Load Font Pattern
01 81                       Generate Font Patterns
E3 00 0A                    Move Cursor Vertically
D3 0D                       Activate Font #13
01 92                       Generate Font Patterns
E3 FF F6                    Move Cursor Vertically
D3 0A                       Activate Font #10
01 84                       Generate Font Patterns
E2 FE 94                    Move Cursor Horizontally
E3 00 12                    Move Cursor Vertically
83                          Save Cursor to Register #3
E2 00 01                    Move Cursor Horizontally
E3 00 01                    Move Cursor Vertically
F8 C3 00 00 01 CC           Generate Vectors
93                          Restore Cursor from Register #3
E2 01 CF                    Move Cursor Horizontally
E2 FE 3B                    Move Cursor Horizontally
E3 00 34                    Move Cursor Vertically
01 81                       Generate Font Patterns
E3 00 0A                    Move Cursor Vertically
D3 0D                       Activate Font #13
01 92                       Generate Font Patterns
E3 FF F6                    Move Cursor Vertically
D3 0C                       Activate Font #12
01 4C                       Generate Font Patterns
D3 0A                       Activate Font #10
01 81                       Generate Font Patterns
E3 FF EC                    Move Cursor Vertically
D3 03                       Activate Font #3
F0 F2 15 0E 01 01 15        Load Font Pattern
01 F2                       Generate Font Patterns
E2 FF F0                    Move Cursor Horizontally
E3 00 1E                    Move Cursor Vertically
D3 0D                       Activate Font #13
01 92                       Generate Font Patterns
E2 00 0A                    Move Cursor Horizontally
E3 FF F6                    Move Cursor Vertically
D3 0A                       Activate Font #10
01 4E                       Generate Font Patterns
E2 00 0A                    Move Cursor Horizontally
D3 0B                       Activate Font #11
01 82                       Generate Font Patterns
D3 0A                       Activate Font #10
01 81                       Generate Font Patterns
E3 00 0A                    Move Cursor Vertically
D3 0D                       Activate Font #13
01 92                       Generate Font Patterns
```

```
E2 00 0A                   Move Cursor Horizontally
E3 FF F6                   Move Cursor Vertically
D3 07                      Activate Font #7
04 A3 81 95 88             Generate Font Patterns
E2 00 0A                   Move Cursor Horizontally
D3 0A                      Activate Font #10
01 81                      Generate Font Patterns
E3 00 0A                   Move Cursor Vertically
D3 0D                      Activate Font #13
01 92                      Generate Font Patterns
E3 FF F6                   Move Cursor Vertically
D3 0A                      Activate Font #10
01 84                      Generate Font Patterns
E2 00 0A                   Move Cursor Horizontally
01 4E                      Generate Font Patterns
E2 00 0A                   Move Cursor Horizontally
D3 07                      Activate Font #7
01 F1                      Generate Font Patterns
D3 0C                      Activate Font #12
01 5C                      Generate Font Patterns
E2 00 34                   Move Cursor Horizontally
E3 FF D9                   Move Cursor Vertically
D3 07                      Activate Font #7
03 83 96 A2                Generate Font Patterns
E2 00 0A                   Move Cursor Horizontally
D3 0A                      Activate Font #10
01 81                      Generate Font Patterns
E3 00 0A                   Move Cursor Vertically
D3 0D                      Activate Font #13
01 92                      Generate Font Patterns
E3 FF F6                   Move Cursor Vertically
D3 0B                      Activate Font #11
01 A7                      Generate Font Patterns
E2 00 04                   Move Cursor Horizontally
E3 00 23                   Move Cursor Vertically
83                         Save Cursor to Register #3
E2 00 01                   Move Cursor Horizontally
E3 00 01                   Move Cursor Vertically
F8 C2 00 00 00 0E          Generate Vectors
93                         Restore Cursor from Register #3
E2 00 10                   Move Cursor Horizontally
E2 FF F0                   Move Cursor Horizontally
E3 FF A4                   Move Cursor Vertically
83                         Save Cursor to Register #3
E2 00 01                   Move Cursor Horizontally
E3 00 01                   Move Cursor Vertically
F8 C2 00 00 00 0E          Generate Vectors
93                         Restore Cursor from Register #3
E2 00 10                   Move Cursor Horizontally
E2 FF FC                   Move Cursor Horizontally
83                         Save Cursor to Register #3
E3 00 02                   Move Cursor Vertically
E2 00 02                   Move Cursor Horizontally
F8 C4 00 5A 00 00          Generate Vectors
93                         Restore Cursor from Register #3
```

```
E2 00 04                    Move Cursor Horizontally
E2 00 04                    Move Cursor Horizontally
E3 00 5E                    Move Cursor Vertically
E1 08 7A                    Set Cursor Vertically
E0 01 2C                    Set Cursor Horizontally
E2 01 BE                    Move Cursor Horizontally
D3 07                       Activate Font #7
E2 00 0A                    Move Cursor Horizontally
E3 FF B4                    Move Cursor Vertically
D3 0A                       Activate Font #10
F0 73 1D 12 01 02 1D        Load Font Pattern
01 73                       Generate Font Patterns
E3 FF EC                    Move Cursor Vertically
D3 03                       Activate Font #3
01 F2                       Generate Font Patterns
E3 00 14                    Move Cursor Vertically
D3 0A                       Activate Font #10
F0 86 26 18 01 03 1E        Load Font Pattern
01 86                       Generate Font Patterns
E2 FF B5                    Move Cursor Horizontally
E3 00 10                    Move Cursor Vertically
83                          Save Cursor to Register #3
E2 00 01                    Move Cursor Horizontally
E3 00 01                    Move Cursor Vertically
F8 C3 00 00 00 53           Generate Vectors
93                          Restore Cursor from Register #3
E2 00 56                    Move Cursor Horizontally
E2 FF B6                    Move Cursor Horizontally
E3 00 34                    Move Cursor Vertically
01 73                       Generate Font Patterns
D3 0B                       Activate Font #11
01 A7                       Generate Font Patterns
E2 00 02                    Move Cursor Horizontally
E3 FF EC                    Move Cursor Vertically
D3 03                       Activate Font #3
01 F2                       Generate Font Patterns
E2 00 2B                    Move Cursor Horizontally
E3 FF ED                    Move Cursor Vertically
D3 0A                       Activate Font #10
01 4E                       Generate Font Patterns
E2 00 29                    Move Cursor Horizontally
E3 FF E3                    Move Cursor Vertically
01 73                       Generate Font Patterns
E3 FF EC                    Move Cursor Vertically
D3 03                       Activate Font #3
01 F2                       Generate Font Patterns
E3 00 14                    Move Cursor Vertically
D3 0A                       Activate Font #10
01 86                       Generate Font Patterns
E2 FF B5                    Move Cursor Horizontally
E3 00 10                    Move Cursor Vertically
83                          Save Cursor to Register #3
E2 00 01                    Move Cursor Horizontally
E3 00 01                    Move Cursor Vertically
F8 C3 00 00 00 53           Generate Vectors
```

```
93                           Restore Cursor from Register #3
E2 00 56                     Move Cursor Horizontally
E2 FF B7                     Move Cursor Horizontally
E3 00 34                     Move Cursor Vertically
01 73                        Generate Font Patterns
D3 0B                        Activate Font #11
F0 A8 1B 16 FE 01 13         Load Font Pattern
01 A8                        Generate Font Patterns
E2 00 02                     Move Cursor Horizontally
E3 FF EC                     Move Cursor Vertically
D3 03                        Activate Font #3
01 F2                        Generate Font Patterns
E2 00 2B                     Move Cursor Horizontally
E3 FF ED                     Move Cursor Vertically
D3 0A                        Activate Font #10
01 4E                        Generate Font Patterns
E2 00 29                     Move Cursor Horizontally
E3 FF E3                     Move Cursor Vertically
01 73                        Generate Font Patterns
E3 FF EC                     Move Cursor Vertically
D3 03                        Activate Font #3
01 F2                        Generate Font Patterns
E3 00 14                     Move Cursor Vertically
D3 0A                        Activate Font #10
01 86                        Generate Font Patterns
E2 FF B5                     Move Cursor Horizontally
E3 00 10                     Move Cursor Vertically
83                           Save Cursor to Register #3
E2 00 01                     Move Cursor Horizontally
E3 00 01                     Move Cursor Vertically
F8 C3 00 00 00 53            Generate Vectors
93                           Restore Cursor from Register #3
E2 00 56                     Move Cursor Horizontally
E2 FF B8                     Move Cursor Horizontally
E3 00 34                     Move Cursor Vertically
01 73                        Generate Font Patterns
D3 0B                        Activate Font #11
F0 A9 16 11 01 00 13         Load Font Pattern
01 A9                        Generate Font Patterns
E2 00 02                     Move Cursor Horizontally
E3 FF EC                     Move Cursor Vertically
D3 03                        Activate Font #3
01 F2                        Generate Font Patterns
E2 00 2D                     Move Cursor Horizontally
E3 FF ED                     Move Cursor Vertically
D3 0A                        Activate Font #10
01 7E                        Generate Font Patterns
E2 00 15                     Move Cursor Horizontally
D3 07                        Activate Font #7
F0 F0 1C 12 01 02 1C         Load Font Pattern
01 F0                        Generate Font Patterns
E3 00 2F                     Move Cursor Vertically
E1 08 A1                     Set Cursor Vertically
E0 01 40                     Set Cursor Horizontally
E3 FF FC                     Move Cursor Vertically
```

```
83                        Save Cursor to Register #3
E2 00 02                  Move Cursor Horizontally
E3 00 02                  Move Cursor Vertically
F8 C5 00 00 05 8C         Generate Vectors
93                        Restore Cursor from Register #3
E3 00 04                  Move Cursor Vertically
E1 08 A2                  Set Cursor Vertically
E0 01 2C                  Set Cursor Horizontally
83                        Save Cursor to Register #3
E3 00 04                  Move Cursor Vertically
E2 00 04                  Move Cursor Horizontally
F8 C9 00 0C 00 00         Generate Vectors
93                        Restore Cursor from Register #3
E0 06 DF                  Set Cursor Horizontally
83                        Save Cursor to Register #3
E3 00 04                  Move Cursor Vertically
E2 00 04                  Move Cursor Horizontally
F8 C9 00 0C 00 00         Generate Vectors
93                        Restore Cursor from Register #3
E1 08 B6                  Set Cursor Vertically
E0 01 2C                  Set Cursor Horizontally
E3 FF F8                  Move Cursor Vertically
83                        Save Cursor to Register #3
E2 00 04                  Move Cursor Horizontally
E3 00 04                  Move Cursor Vertically
F8 C9 00 00 05 B3         Generate Vectors
93                        Restore Cursor from Register #3
E3 00 08                  Move Cursor Vertically
C2                        Deactivate Font
E1 09 D7                  Set Cursor Vertically
E0 01 2C                  Set Cursor Horizontally
E2 05 90                  Move Cursor Horizontally
D3 03                     Activate Font #3
01 F1                     Generate Font Patterns
C2                        Deactivate Font
E0 00 00                  Set Cursor Horizontally
E1 00 00                  Set Cursor Vertically
D1 00                     Print Page
D4 02                     Unload Font #2
D8 00                     Set Font Pattern Controls
D9 90                     Set Generation Mode

End of data file...
```

VITA

Gregg Allen Thomas was born on February 17, 1966. He attended the Virginia Military Institute and graduated in 1988 with a B.S. degree in Electrical Engineering with distinction. Following graduation from V.M.I., he enrolled in graduate studies at Virginia Tech to pursue a M.S. degree in Electrical Engineering.

He will enter active duty in the U.S. Air Force after graduation from Virginia Tech. His immediate plans also include pursuing an M.B.A. degree from Florida State University. Personal interests include photography, music, computer programming, and sports.