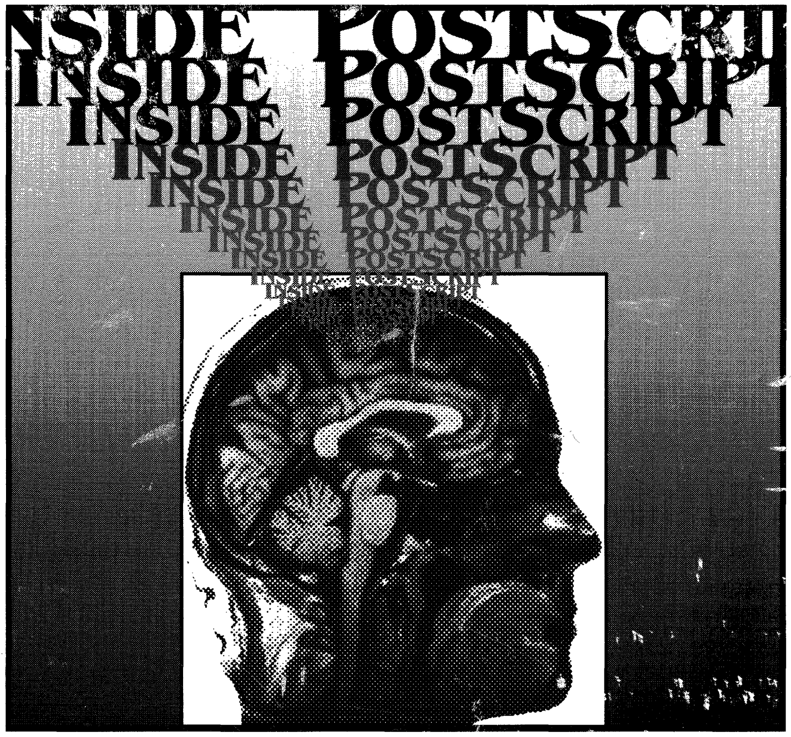
INSIDE POSTSCRIPT®



An analysis of the PostScript Interpreter
on
Canon® Print Engines

by Frank Merritt Braswell

$37.50

INSIDE POSTSCRIPT®

FRANK
MERRITT
BRASWELL

Copyright © 1989 by Systems of Merritt, Incorporated.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher.

Trademark Acknowledgments

PostScript® is a registered trademark of Adobe Systems Incorporated.

QMS® is a registered trademark of QMS, Incorporated.

QMS-PS 800 is a trademark of QMS, Incorporated.

Apple®, AppleTalk®, and LaserWriter® are registered trademarks of Apple Computer, Incorporated.

Diablo® is a registered trademark of Xerox Corporation.

IBM® and IBM PC® are registered trademarks of International Business Machines Corporation.

Lasertalk is a trademark of Emerald City Software.

Procomm is a trademark of Datastorm Technologies, Incorporated.

Times* and Helvetica* are trademarks of Linotype AG and/or its subsidiaries.

Canon® is a registered trademark of Canon Incorporated.

The brain image was obtained from a Picker VISTA® Magnetic Resonance Scanner.

The Name PostScript

The name *PostScript®* is a registered trademark of Adobe Systems Incorporated. All instances of the name *PostScript* in the text are references to the PostScript Language as defined by Adobe Systems Incorporated, unless otherwise stated. The name *PostScript* also is used as a product trademark for Adobe Systems' implementation of the PostScript language interpreter.

Special Thanks

Thanks and praise to the Lord for giving me the gift of an analytical and creative mind.

This book is dedicated to my wife Cindy, who put up with my many hours of research and writing. I love you. To my children, Rebekah, Joshua, Miriam, and Hannah, thanks for your patience.

Thanks to Melissa Higdon for the great cover art and help with the formatting.

Thanks to Kathy Hancock for help with the editing.

Thanks to my reviewers, Randy Adams, Pat Wood, Val Stentz and David Holzgang for many helpful suggestions.

Published by:

Systems of Merritt, Inc.
2551 Old Dobbin Drive East
Mobile, Alabama 36695
(205) 660-1240

First Printing, March 1989

Table of Contents

Appendices

Part I

The Introductory Group

Chapter 1

Introduction to Inside PostScript

Introduction

Anyone who takes the time to read through all this material and actually ***enjoy*** it (like me) is a hopelessly brain-warped PostScript junkie. With the growth in popularity of the PostScript language however, you may take comfort in the fact that you are not alone.

Once you became familiar with the PostScript language and began to delve more deeply into its details, you began to discover operators and dictionaries which were undocumented. As you studied Adobe's documentation, you found mysterious references to additional undocumented features of the language. Then perhaps you tried the following.

```
PS>serverdict {} forall pstack
```

You were further intrigued by the many undocumented serverdict procedures and variables. The purpose of this book is to document that which is not documented by Adobe.

While the documented language standard may be sufficient for describing a printed page, the undocumented language is necessary for controlling user job execution and the print engine. In other words, the documented language is device independent, while much of the undocumented language is device dependent.

Benefits of Inside PostScript

What would be the reason for studying the undocumented portion of the PostScript language? There are several reasons.

- Studying Adobe's use of the language will give you ideas for improving your own PostScript programming techniques.
- Understanding how Adobe controls job execution will help you better control your own PostScript program execution.
- Current PostScript literature gives very few examples of advanced error handling techniques, file stream manipulation, or the use of immediately evaluated names, to name a few areas covered in this book.

A word of caution concerning the use of undocumented features is in order here which goes to the heart of the reason Adobe did not document them in the first place. To make use of the undocumented features of the language is to make your PostScript program device dependent. So don't expect Adobe to provide support for printer drivers which make use of undocumented commands.

Also be aware of the fact that the information in this book may not be 100% accurate since I developed *Inside PostScript* apart from any Adobe internal documentation. Only Adobe can speak with complete authority on the subjects presented herein. Neither can I assume any responsibility for the use of this information in PostScript programs.

Common PostScript Structure

Having said that, what good is a book which documents the specific internal procedures of PostScript version 38 on the QMS-PS 800?

By studying a variety of PostScript printers, I found a basic common structure runs through them all, and to understand one will open the door to understanding the PostScript printer you are working with. At QMS I have worked with PostScript printers such as the ColorScript 100, JetScript, the PS 810, PS 800II, PS 800+, PS 1500, PS 2200 and others. On each printer it was my job to evaluate the Adobe alpha and beta software releases and work with Adobe to correct bugs found. Working with such a variety of PostScript printers led to my interest in the inner workings of the interpreter as it controlled different print engines.

It all began with the statement:

```
PS>serverdict {} forall pstack
```

Join me as we begin our journey *Inside PostScript.*

Organization of Inside PostScript

Our journey into the depths of PostScript is broken into five distinct sections or groups. As you browse through the table of contents you will find each group consists of one or more chapters. The groups are listed as follows.

- Introductory Group - Introduction to basic interpreter structure and concepts.
- Error Handling Group - Error handling concepts and procedures.
- Interactive Mode Group - Discussion of interactive mode and the executive procedure.
- Printer Control Group - Print engine control and PostScript job execution.
- Test Page Group - The power-up test page is documented.

- Three appendices feature tables to help the reader with a further analysis of PostScript.

How to Use This Book

Inside PostScript can be used in several ways.

Inside PostScript can be read as an overview of the PostScript interpreter without a detailed study of each code example since each PostScript procedure is summarized prior to the presentation of the code.

Those desiring detailed study of the code examples will find the PostScript source code comments helpful to quickly understanding how Adobe has implemented each function.

Inside PostScript can also be used as a complete reference manual to the PostScript interpreter. The index contains references to each documented procedure, plus the appendices show the structure of the interpreter through various tables and lists.

Chapter 2

PostScript Interpreter Structure

Introduction

This chapter introduces the overall structure of the PostScript interpreter, plus the documentation conventions used in this book for its study.

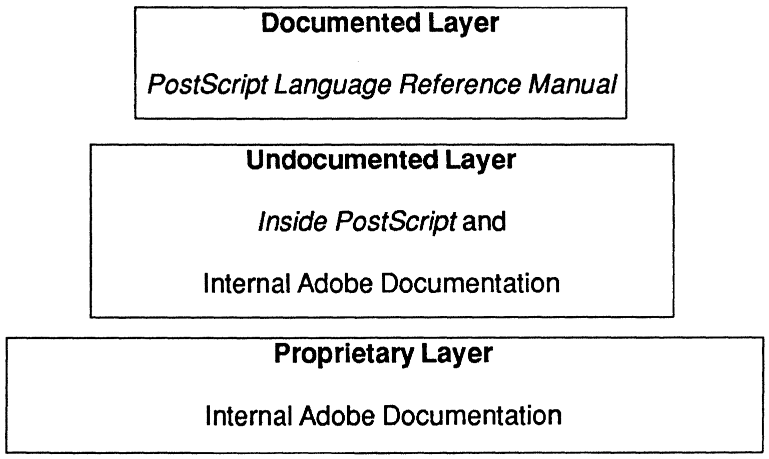
As a PostScript programmer, you know that PostScript is a very powerful page description language with over 200 operators. With this great power and variety of operators however, comes much complexity and confusion due to the nature of the language itself. PostScript's ability to redefine operators (like Forth), make it a joy to program, difficult to debug, and a nightmare to maintain.

While the vast majority of PostScript users are only concerned with the language through an application program, and never hand-code a single program, there are those of us who are blessed with the task of writing drivers and other programs which directly exercise the language.

To directly exercise the language for the description of the printed page is one thing, but to use the PostScript language for control of the print engine is another matter. This is a completely new concept to any PostScript programmer outside of Adobe and covers a totally different application of the language. Very few of the procedures used for control of the printer are actually locked such that the programmer cannot actually view them or deduce their function by trial or error. The difficulty in analyzing these internal programs is the volume of code involved and the complex interconnectivity among procedures.

Layers of the PostScript Interpreter

There are three layers of the PostScript interpreter: the documented layer, undocumented layer and the proprietary layer.


Documented Layer
PostScript Language Reference Manual
Undocumented Layer
Inside PostScript and
Internal Adobe Documentation
Proprietary Layer
Internal Adobe Documentation

Documented Layer The documented layer involves PostScript operators and procedures whose functions are documented in Adobe publications such as the three language manuals and various printer supplements. These documented operators are sufficient for device independent page description and some specific printer control functions such as tray switching.

Undocumented Layer The undocumented layer is the topic of this book and involves the use of PostScript to control the print engine and user job execution. This layer is visible to the user and is very device dependent.

Proprietary Layer The proprietary layer involves additional control code which is purposely hidden from the users view such as locked procedures and actual operator code.

Structure

The following chapters attempt to break the interpreter into logical groupings of procedures. In some cases, the procedures group nicely within dictionaries such as $idleTimeDict, and in other cases the procedures are grouped by function like the communication procedures.

The first of the technical chapters covers PostScript error handling. The next three chapters cover the PostScript interactive mode and related procedures. These chapters can logically stand separate from the printer control chapters which follow.

The printer control chapters are sequenced to build upon each other and climax with the discussion of the job control and server loop which tie all the preceding chapters together.

Finally, the test page code is presented which is different in nature from all the previous code.

Analysis Techniques

It would have been impossible to write such a book as this without first laying an analytical foundation that involved writing numerous PostScript programs to systematically display the dictionaries, procedures and the cross reference relationships of all internal names.

I list the results of this analysis in the appendices. Before digging into the technical discussions, I recommend a basic familiarization with these tables to help visualize the overall dictionary structures and relationships. This information is invaluable as you study the structure of the PostScript interpreter in this book.

Documentation Conventions

All of the program listings presented in this book were extracted from the PostScript interpreter itself and are given the appearance of documented code fragments. All nested procedures are indented for readability and both open and close braces for procedures are at the same level of indention so the eye can quickly pick out the beginning and end of the procedure.

All the internal procedures have been bound, even though binding is not explicitly indicated in the procedure listings. Immediately evaluated names are flagged when they appear within other procedures. The concepts of binding and immediately evaluated names are explained in detail in the next chapter.

Also, the listings are not necessarily equivalent to the original Adobe source code. They may be close, but only Adobe has the source code and knows how the interpreter is built into its runtime form.

One term used in this book which may not be familiar to some is reverse channel. Reverse channel refers to the communication channel on which information travels from the printer to a host computer. Reverse channel information can be PostScript error or user program messages.

I chose not to highlight PostScript operators and procedures in this book with a font change as is done is other books on PostScript. On some pages, so many procedures are discussed that the reader would be distracted from the discussion by font changes.

Extensive use of the PostScript executive procedure or interactive mode was used in the research for this book and many interactive session fragments are presented in the following chapters. These examples are identified by the interactive prompt "PS>" at the beginning of each command line.

In order to fully understand many of the concepts presented in this book, the reader may wish to explore the features of the PostScript interpreter using the interactive mode. To help you with your study, several useful tools are available.

PostScript Tools

Several items are worth mentioning which are extremely helpful in the study and programming of PostScript.

First, if you don't have a PostScript Programmer's Instant Reference Card you need to get one. This card lists all the PostScript operators and useful code fragments which are helpful to beginning and experienced PostScript programmers. Contact Micro Logic at (201) 342-6518 for details on ordering.

Next, if you are a Macintosh (or IBM PC) user you need to get a software package called Lasertalk. This package is excellent for learning PostScript, debugging programs, and exploring the language. For information contact Emerald City Software at (415) 324-8080.

For PC users communication packages such as Procomm can be used to communicate with PostScript printers over a serial RS 232 channel. Direct interaction with the PostScript interpreter is possible along with program uploading and recording of interactive sessions. Procomm is available free on many bulletin boards.

Finally of course, you need a PostScript printer. The research for this book was done on a QMS-PS 800 printer. PostScript printers (such as the Apple LaserWriter) based on a Canon print engine are very similar in nature and can be studied along with the material in this book.

Chapter 3

Efficient PostScript

Introduction

Because of their extensive use by Adobe throughout the code, the concepts of binding and immediately evaluated names need to be discussed in detail before moving into discussions of the internal PostScript procedures. Efficiency of execution and compactness of code are the reasons Adobe makes use of these concepts, and they can be of great value to any PostScript programmer once they are understood. Binding is discussed first followed by immediately evaluated names.

Binding

According to the *PostScript Language Reference Manual*, the bind operator replaces executable operator names with their values within a procedure. **Only operator type objects are affected by the bind operator.** The way to tell if an operator has been bound is to use the == or pstack procedures in interactive mode to display the items on the stack. If a procedure is displayed and operators within the procedure are shown with dashes (`--add--`) it means they have been bound. There are several advantages to binding which are explained below. First, lookup speed is explained, then name changes, and finally, dictionary lookup.

Lookup Speed One of the primary benefits of binding is that execution speed can be increased. Every programmer wants his code to execute faster, and here is a way to do it.

Normally, the PostScript interpreter must move through procedures placing items on the various stacks. If a name is encountered which is associated with an operator type, the interpreter first scans the name and then looks the name up in the dictionaries on the dictionary stack. Once found, the operator is executed.

If the procedure is bound prior to invocation, the bind operator does all the work of looking up the operator so that when the procedure is invoked later and the interpreter encounters the operator, it is executed immediately. It is especially critical that procedures used in loops be bound. If a procedure is only executed once, binding is of little value. For example:

```
PS>/Helvetica findfont 25 scalefont setfont
PS>72 72 translate 1 setgray
PS>.95 -.1 0
PS>{setgray 0 0 moveto (Bind Test) show -1 -1 translate}
PS> bind for          % bind loop procedure on stack
PS>showpage
PS>
```

More technically, the bind operator substitutes the character string representing the name of the operator with a pointer to the operator function itself.

The following example shows how to tell the difference between a normal procedure and a procedure which has been bound.

At first the procedure containing the add operator is placed on the stack and displayed. The pstack operator indicates that add is simply a name (string of characters). The bind operator is applied to the same procedure and then the add operator is displayed as "`--add--`" indicating it has been bound.

Also, notice there is no effect on the result of executing the procedure before and after the binding. The same result, "5", is displayed. The only change is the speed of execution.

```
PS>{2 3 add ==} pstack  % place the procedure on the stack
{2 3 add == }           % display the procedure
PS>dup exec             % execute the procedure
5                       % display procedure result
PS>bind pstack          % bind the procedure
{2 3 --add--== }        % display the procedure
PS>exec                 % execute the bound procedure
5                       % display the procedure result
```

Name Changes Changing the name of an operator used in a bound procedure after binding has no effect on the execution of the procedure. This is because the procedure no longer uses the name to access the procedure.

Dictionary Lookup When the bind operator is applied to a procedure the same rules for dictionary lookup of operator names is used as if during the execution of the procedure. However, during execution of the bound procedure, it is not necessary to have the dictionaries opened which contain the bound operators since the lookup step has already been performed.

In the example below, the procedure containing the statusdict operator checkpassword does not require statusdict to be open when it is executed since the bind operation performed the lookup.

```
PS>statusdict begin               % open statusdict
PS>{0 checkpassword ==} pstack    % create procedure
{0 checkpassword == }             % display procedure
PS>bind pstack                    % bind procedure
{0 --checkpassword--== }          % display bound procedure
```

```
PS>end                            % close statusdict
PS>dup exec                       % execute procedure
true                              % display procedure result
PS>pstack                         % display procedure again
{0 --checkpassword--== }          % with statusdict closed
PS>                               % end of example
```

Immediately Evaluated Names

Information on this concept can only be found in Adobe PostScript Language supplements for recent printers, yet it has been available since version 25.0 and is an extremely important part of the PostScript interpreter.

Syntax and Usage The syntax for immediately evaluated names is //name, and when the interpreter encounters this token, it immediately looks up the name and substitutes the value in its place. As with the bind operator, the normal rules for dictionary lookup apply at the time that //name is evaluated.

Unlike the bind operator, immediately evaluated names can be keys associated with arrays, dictionaries or other objects. Greater care must be exercised when using immediately evaluated names (as opposed to bind) because of its substitution nature. The important thing to remember is that **substitution** does not mean **execution.** The effect of this is the difference between placing a procedure on the stack verses executing a procedure on the stack. The following examples illustrate how an immediately evaluated name is different than the name itself.

This first example shows an incorrect usage of an immediately evaluated name.

```
PS>/= load pstack   % first show the procedure
{{--dup----type--/stringtype --ne--{(
stringtypeflowflow
)--cvs--}--if----print--}--exec--(
)--print--}
```

Define a procedure with an incorrect usage of //=.

```
PS>/x               % define the x procedure
PS>{(test 1) = (test 2) //=} def
PS>x                % execute the x procedure
test 1              % result of x procedure
PS>                 % = proc and "test 2" are left on stack
PS>pstack           % because second = is never executed
{{--dup----type--/stringtype --ne--{(
 stringtypeflowflow
)--cvs--}--if----print--}--exec--(
)--print--}
(test 2)
PS>/x load ==       % show contents of x procedure
{(test 1)= (test 2){{--dup----type--/stringtype --ne--{(
stringtypeflowflow
)--cvs--}--if----print--}--exec--(
)--print--}}
```

The next example shows how to use the exec operator to properly execute the immediately evaluated name.

```
PS>/x                    % proper use
PS>{(test 1 = (test 2) //= exec} def
PS>x                     % execute x procedure
test 1                   % the proper results are displayed
test 2
PS>/x load pstack        % show contents of x procedure
{(test 1)= (test 2){{--dup----type--/stringtype --ne--{(
stringtypeflowflow
)--cvs--}--if----print--}--exec--(
)--print--}exec }
PS>                      % end of example
```

These examples show how the immediately evaluated name "//=" is actually substituted into the /x procedure, and why the exec operator must be applied to immediately evaluated names which refer to procedures.

Dictionary Substitutions Another feature extensively used by Adobe is the substitution of dictionaries into procedures.

```
PS>                      % create procedure
PS>{//statusdict begin 0 checkpassword end ==}
PS>pstack                % view procedure
{-dictionary- begin 0 checkpassword end == }
PS>exec                  % execute procedure
true                     % view results
PS>                      % end example
```

The effect of immediately evaluating dictionary names is similar to what bind does for operator names.

The one negative side effect of this is that the procedures are almost impossible to follow without the source code because the word -dictionary- is substituted for the actual name when displayed with pstack or ==.

Part II

The Error Handling Group

Chapter 4

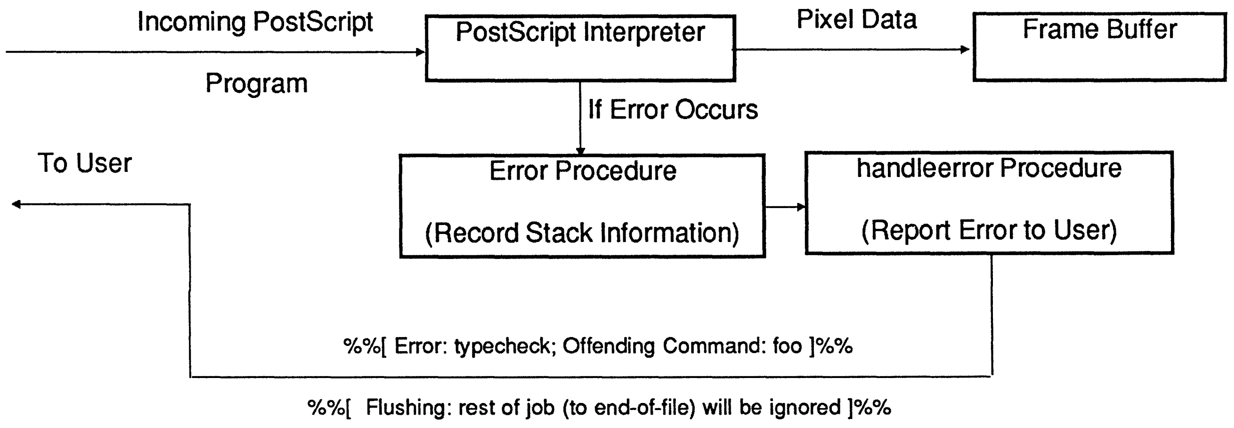
Error Handling and errordict

Introduction

Error handling (assuming the user has not altered it) is a two step process in PostScript. First, a procedure is invoked which takes a snapshot of the three PostScript stacks; second the handleerror procedure reports the error to the user over the reverse channel.

Block Diagram

The default method of handling errors is diagramed below.



In this chapter the error procedures and handleerror are discussed along with how they affect the PostScript programmers debugging efforts.

Error Procedures

The errordict dictionary contains all the procedures which are invoked when PostScript encounters an error. Following is a complete list of all PostScript errors and procedures as found in errordict.

Errorname / Procedure

1. `/VMerror`
   ```
   {/VMerror //.error exec} def
   ```
2. `/dictfull`
   ```
   {/dictfull //.error exec} def
   ```
3. `/dictstackoverflow`
   ```
   {/dictstackoverflow //.error exec} def
   ```
4. `/dictstackunderflow`
   ```
   {/dictstackunderflow //.error exec} def
   ```
5. `/execstackoverflow`
   ```
   {/execstackoverflow //.error exec} def
   ```
6. `/invalidaccess`
   ```
   {/invalidaccess //.error exec} def
   ```
7. `/invalidexit`
   ```
   {/invalidexit //.error exec} def
   ```

```
8. /invalidfileaccess
     {/invalidfileaccess //.error exec} def
9. /invalidfont
     {/invalidfont //.error exec} def
10. /invalidrestore
     {/invalidrestore //.error exec} def
11. /ioerror
     {/ioerror //.error exec} def
12. /limitcheck
     {/limitcheck //.error exec} def
13. /nocurrentpoint
     {/nocurrentpoint //.error exec} def
14. /rangecheck
     {/rangecheck //.error exec} def
15. /stackoverflow
     {/stackoverflow //.error exec} def
16. /stackunderflow
     {/stackunderflow //.error exec} def
17. /syntaxerror
     {/syntaxerror //.error exec} def
18. /timeout
     {/timeout /timeout //.error exec} def
19. /typecheck
     {/typecheck //.error exec} def
20. /undefined
     {/undefined //.error exec} def
21. /undefinedfilename
     {/undefinedfilename //.error exec} def
22. /undefinedresult
     {/undefinedresult //.error exec} def
23. /unmatchedmark
     {/unmatchedmark //.error exec} def
24. /unregistered
     {/unregistered //.error exec} def
```

The interesting thing about the different error procedures is that they are all identical except for the name of the error. This name is placed on the stack at the beginning of the procedure. The one exception is the timeout error procedure where an additional object (the name /timeout) is pushed on the stack. In this one case, timeout is both the offending command and the name of the error procedure.

Each procedure is called internally from the interpreter, thus no calls are made to the error procedures from any of the PostScript control procedures except for two special cases. Before the call is made, the offending command is left on the stack for the error procedure.

In two special instances, error procedures are invoked from resident control procedures. The names of the timeout and undefinedfilename error procedures are referenced in execjob and executive respectively where they are explicitly trapped by the stopped operator.

Another interesting observation is that the ".error" procedure in systemdict is a subset of every error procedure as an immediately evaluated name. This is flagged in the documentation of the error procedure.

The default PostScript error procedures make entries in the $error dictionary. These entries include the name of the offending command, error name, and snapshots of the execution, dictionary, and operand stacks. The error procedures do not print and send no data to the reverse channel. Printing is done by the procedure called handleerror.

In the $error dictionary, six arrays (ostackarray and ostack for the operand stack; estackarray and estack for the execution stack; dstackarray and dstack for the dictionary stack) are used to handle the snapshots of the stacks. The arrays ostackarray, estackarray, and dstackarray are allocated once when the first error occurs. Subsequent errors reuse these same arrays.

The arrays ostack, estack, and dstack are subsets of ostackarray, estackarray and dstackarray respectively. These smaller subsets contain the actual snapshots. A study of the error procedure below illustrates the difference between the two types of arrays.

The typecheck procedure is documented as an example of each of the error procedures named above. The only difference between the error procedures is the name of the procedure, and the error name placed on the stack at the beginning of the procedure. To change to the syntaxerror procedure for example, simply substitute the name syntaxerror for typecheck.

Overview of Error Procedures

The error procedures start by initializing the errorname, command and newerror variables in the $error dictionary. Next, a check is made to see if the error was a VMerror, in which case none of the allocating or processing of the arrays is done (since we are out of VM). If there is adequate VM available, a check is made to see if ostackarray is null. If it is null, then this is the first error since power-up and space is then allocated for estackarray, ostackarray and dstackarray.

The next step is to store the stacks in ostackarray, dstackarray, and estackarray. Then the stack which was emptied into ostackarray is restored to its original state so that the stack is undisturbed upon exiting the procedure.

Finally, handleerror is executed if the initializing flag is true. This covers the case of an error during the printing of the test page.

The typecheck Procedure

The typecheck procedure is documented in detail below.

```
%________________________________________________________________
% calling format
% "offending command" typecheck
% found in:  errordict
                         %
/typecheck               %
{                        %
  /typecheck             % place the error name on the stack
                         % this error name is the only
                         % difference between the error
                         % procedures
                         %
                         % .error procedure begins here
                         % .error is placed on the stack
                         % here in preparation for exec
  {                      %
    //$error             % $error
    exch                 %
    /errorname           % get the error name (from above)
    exch                 % and put it in $error
    put                  %
                         %
    //$error             % $error
    exch                 %
    /command             % get the offending command off
    exch                 % the stack and put it in $error
    put                  %
                         %
    //$error             % $error
    /newerror            % newerror is set to true
    true                 % and put into $error
    put                  %
                         %
    //$error             % $error
    /errorname           % get the error name back
    get                  % from $error
                         %
    /VMerror             %
    ne                   % is the error a VMerror?
                         %
                         % if we are out of VM then we
                         % shouldn't be allocating arrays
```

```
                                % as is done in the
                                % following if code
                                %
{                               % if not a VMerror
  //$error                      % $error
  /ostackarray                  % get the ostackarray
  get                           % from $error
                                %
  null                          %
  eq                            % is ostackarray a null array
                                %
                                % if ostackarray is null then arrays
                                % for the operand stack, execution
                                % stack and dictionary stack need to
                                % be allocated
                                %
  {                             % if arrays not allocated do this
    //$error                    % $error
    /estackarray                %
    250                         % allocate a 250 element array for
    array                       % estackarray and put it
    put                         % in $error
                                %
    //$error                    % $error
    /ostackarray                %
    500                         % allocate a 500 element array for
    array                       % ostackarray and put it
    put                         % in $error
                                %
    //$error                    % $error
    /dstackarray                %
    20                          % allocate a 20 element array for
    array                       % dstackarray and put it
    put                         % in $error
  } if                          % if ostackarray eq null
                                %
                                % this next major section of
                                % code takes the snapshot of the
                                % three stacks
                                %
                                % first, take snapshot of the
                                % operand stack
                                %
  count                         % count the objects on stack
  //$error                      % $error
  /ostackarray                  % retrieve ostackarray
  get                           % from $error
                                %
  exch                          %
  0                             % start index for getinterval
  exch                          %
```

```
getinterval      % put a subarray from ostackarray
                 % on top of the stack
                 % the subarray starts at index 0 of
                 % ostackarray and is count objects
                 % long
                 %
astore           % now store the entire stack into
                 % the array we just put on the stack
                 %
//$error         % $error
exch             %
/ostack          % take the array on the top of
exch             % the stack and call it ostack
put              % put ostack in $error
                 %
                 % at this point there is nothing
                 % on the stack
                 %
                 % second, take a snapshot of the
                 % dictionary stack
                 %
//$error         % $error
/dstack          % prepare to store dstack
                 %
//$error         % $error
/dstackarray     % fetch dstackarray
get              % from $error
dictstack        % store the dictstack in dstackarray
                 % (a subarray containing only the
                 % dicts on the dictionary stack is
                 % returned)
put              % store this array in dstack
                 %
                 % third, take a snapshot of the
                 % execution stack
                 %
//$error         % $error
/estack          % prepare to store estack
                 %
//$error         % $error
/estackarray     % fetch estackarray
get              % from $error
execstack        % dump the execution stack
                 % into estackarray
dup              % get the length of
length           % the array
2                %
sub              % sub 2 from the length
0                %
exch             %
getinterval      % get everything  from the array
```

```
                        % but the last two items
                        %
    put                 % store this array in estack
                        %
                        % at this point, the stack is empty
                        %
                        % next, the stack is restored
                        % to its original state by unloading
                        % ostack
    //$error            % $error
    /ostack             %
    get                 % fetch ostack from $error
    aload               % unload the array onto the stack
    pop                 % dump the array left on the top
                        % of the stack
  } if                  % if errorname ne VMerror
                        %
                        % take care of possible errors
                        % during the printing of the
                        % test page
                        %
  //$error              % $error
  /initializing         %
  get                   % fetch initializing flag
  {                     % if initializing = true
    handleerror         % run handleerror
  } if                  % if initializing = true
  interrupt             % interrupt = stop
 }                      %
                        %
                        % .error procedure ends here
                        %
 exec                   % execute .error
} def                   % typecheck ends here
```

The timeout Procedure

The only difference between timeout and other error procedures is that the name timeout is placed on the stack twice at the beginning of the procedure. This is because timeout is both the offending command and error name.

```
%________________________________________________
% calling format
% timeout
% found in:  errordict
                        %
/timeout                %
{                       %
 /timeout               % place the offending command name
                        % on the stack
                        %
```

```
  /timeout              % place the error name on the stack
                        %
                        % timeout is both the offending
                        % command and the error name
                        %
                        % .error procedure begins here
                        % .error is placed on the stack
                        % here in preparation for exec
  {                     %
    //$error            % $error
    exch                %
    /errorname          % put the error name (from above)
    exch                % in $error
    put                 %
                        %
    //$error            % $error
    exch                %
    /command            % get the offending command off
    exch                % the stack and put it in $error
    put                 %
                        %
    .
    .
    . remainder of the code is the same as
    . in the error procedure above
    .

  }                     %
  exec                  %
} def                   % timeout ends here
```

The handleerror Procedure

As described in chapter 3 of the *PostScript Language Reference Manual*, handleerror is invoked automatically by an internal PostScript control program through the use of the stopped operator. The internal procedure is the execjob procedure which is discussed in the chapter on job execution.

The stopped operator tends to be a bit mysterious at first, so I will refer you to the discussions of the executive and execjob procedures for some examples of how Adobe uses stopped to trap error conditions. Understanding stopped is the key to implementing your own error control procedures.

The handleerror procedure handles the reporting of the error messages back to the user over the reverse channel. If you've ever wondered where those informative and user friendly error messages like:

```
%%[ Error: typecheck; Offending Command: show ]%%
```

come from, read on as we tackle the details of handleerror.

Overview of handleerror

When handleerror is first entered, the newerror boolean is used to decide if any error processing is done. This is to protect the interpreter from entering an unending or long error chain. Only the first error is reported to the user.

If the error is processed, the first thing done is to set newerror to false so that additional calls to handleerror do not generate more error messages.

The remainder of the handleerror procedure is responsible for building the error message piece by piece in the form:

```
%%[ Error: typecheck; Offending Command: show ]%%
```

where the error name and offending command are placed in their appropriate places.

Let's take a detailed look at handleerror.

```
%_____________________________________________________________
% calling format
% handleerror
% found in:  errordict
% referenced in all error
% procedures in errordict,
% executive, execjob, .error and
% handleerror (in systemdict)
/handleerror                %
{                           %
  //$error                  % $error
  begin                     %
                            %
  newerror                  % fetch newerror boolean
  {                         % if newerror true
    /newerror               %
    false                   % set newerror false
    def                     %
                            % string length =  11
    (%%[ Error: )
                            %
    print                   % print the first part of
                            % the error message
                            %
    errorname               % put errorname on stack
    {                       % place proc on stack for exec
                            % this proc prints the name of the
                            % error
      dup                   %
      type                  % get errorname type
      /stringtype           %
      ne                    %
      {                     % errorname type ne stringtype?
```

```
                    % string length =  128
    (converted error name will end up in this string)
                    %
    cvs             % convert errorname to string
  } if              % if errorname type ne stringtype
  print             % print the name of the error
}                   %
exec                % execute above proc
                    %
                    % string length =  20
(; OffendingCommand: )
                    %
print               % print the next part of the error
                    % string
                    %
/command            %
load                %
{                   % place proc on stack for exec
                    % this proc prints the offending
                    % command
  dup               %
  type              % get command type
  /stringtype       %
  ne                %
  {                 % command type ne stringtype?
                    % string length =  128
    (converted command name will end up in this string)
                    %
    cvs             % convert command to string
  } if              % if command type ne stringtype
  print             % print the name of the offending
                    % command
}                   %
exec                % execute the above proc
                    %
                    % place the last few characters
                    % of the message on the stack
                    % string length =  4
( ]%%)
                    %
{                   % place proc on stack for exec
                    % this proc prints the message
                    % tail with a carriage return
                    %
  {                 % place another proc on
                    % stack for exec
                    % this proc prints the
                    % message tail
                    % (don't ask me why they do it
                    % this way)
    dup             %
```

```
            type            % get the type of the tail
            /stringtype     %
            ne              %
            {               % tail type ne stringtype
                            % string length =  128
            (message tail will go in this string)
                            %
              cvs           % convert tail to string
            } if            % if tail type ne stringtype
            print           % print the message tail
          }                 %
          exec              % execute the above proc
                            % place a carriage return on stack
                            % string length =  1
          (\n)
                            %
          print             % print the carriage return
        }                   %
        exec                % execute the above proc
        flush               % flush the output buffer so that
                            % entire message is be printed NOW
      } if                  % if newerror true
                            %
      end                   % close $error
                            %
    } def                   % handleerror ends here
```

The handleerror Entry in systemdict

The above handleerror procedure in errordict is actually called indirectly from a procedure called handleerror in systemdict which is listed below. This is done because systemdict is always present on the dictionary stack, and handleerror must always be accessible for error handling.

```
%____________________________________________________________
% calling format
% handleerror
% found in:  systemdict
                        % referenced in executive
                        % and execjob
/handleerror            %
{                       %
  //$error              % errordict
  /handleerror          %
  get                   % fetch handleerror from
                        % errordict
                        %
  exec                  % execute handleerror
                        %
} def                   % handleerror ends here
```

PostScript Debugging

Even though the three stacks have been recorded in $error by the error procedure, handleerror in its default form does not report any of this information to the user.

For more sophisticated debugging of PostScript programs, the user must alter the default error handling process to give a more detailed report of the error condition. Typically, handleerror is redefined to accomplish this.

Adobe has been distributing a PostScript program called ehandler.ps which redefines handleerror to print a page with the offending command and operand stack information displayed. The ehandler.ps program does not list the contents of the dictionary or execution stack however.

More elaborate error handling procedures display the contents of all three stacks, jump into interactive mode, or use an alternate I/O channel for error reporting. There are many possibilities.

The stopped Operator

Additional error trapping techniques may be implemented by the use of the stopped operator as demonstrated in the following pseudocode example.

```
user code .... % user program code

{ procedure being debugged
}              % place procedure on top of stack
stopped        % execute procedure and return boolean
               % boolean = false if no error
               % boolean = true if error encountered
{ error handling procedure
               % execute error handling procedure if
} if           % stopped operator returned true

additional user code ...      % remainder of user code
```

The stopped operator executes the procedure on the top of the stack and returns a boolean which indicates if an error was found. If the boolean is true, then an error handling procedure can be executed.

The "error handling procedure" can be handleerror, or the programmer may wish to handle errors in this section of code completely differently. PostScript offers the programmer total flexibility in error handling.

As *Inside PostScript* covers the execjob and executive procedures, we find that Adobe uses the above technique to trap user program errors. In these cases, the "procedure being debugged" is the user program itself.

Part III

The Interactive Mode Group

Chapter 5

The PostScript Interactive Mode

Introduction

This chapter explores the workings of the PostScript executive or "interactive" mode. Interactive mode is entered by executing the executive procedure in the userdict. This mode allows the user to execute PostScript statements on a line by line basis and is an excellent way to become familiar with the language as new operators and concepts can be tried in interactive sessions. This mode was used extensively in the research for this book as resident procedures, operators and variables were tested.

Benefits of Studying executive

By studying the executive and related procedures, a better understanding of how to use the stopped operator to control program execution, and %statementedit in creating file objects for interactive input is gained.

The stopped operator is used to control error handling during the interactive session. In the interactive mode, if an error occurs, an error message is displayed, and the user is prompted for the next line of input as follows:

```
PostScript(tm) Version 38.0
Copyright (c) 1985 Adobe Systems Incorporated.
PS>1 show
%%[ Error: typecheck; OffendingCommand: show ]%%
PS>
```

Also notice, that everything the user types is echoed back ("1 show" in this case) to the host computer. Character echoing and the ability to edit an input statement by using backspaces is a characteristic of using a file object created with %statementedit.

PostScript's normal mode of operation, batch mode, is more suited for computer to computer communication. No characters are echoed back, and typing errors cause the following response:

```
%%[ Error: undefined; OffendingCommand: foo ]%%
%%[ Flushing: rest of job (to end-of-file) will be ignored ]%%
```

The above error is enough to strike terror in the heart of even the most experienced PostScript programmer. Batch mode processing and error handling are discussed in other chapters.

In order to use the interactive mode from an IBM PC or compatible, a serial communication cable is needed along with a communications software package such as Procomm or Lasertalk (IBM PC version). On the Macintosh, packages like Lasertalk can be used with AppleTalk communication.

Supporting Procedures

Before executive is examined, the supporting procedures and variables are documented.

Welcome to the interactive world of PostScript.

The Prompt Procedure This procedure is used to print the "PS>" prompt string each time the executive is ready to receive input. The execdepth variable is used to keep track of how many levels deep the executive is. For example, if you entered:

```
executive executive executive
```

the executive prompt would look like "PS>>>". Try it and see. The benefits of this are discussed later in the chapter.

```
%______________________________________________________________
% calling format
% prompt
% found in:  userdict
                        % referenced in executive
                        %
/prompt                 %
{                       %
                        % string length =  2
  (PS)
                        %
  print                 % print "PS"
                        %
  execdepth             % set up for repeat
                        % repeat the loop
                        % execdepth times
                        %
  {                     % start repeat loop
                        % string length =  1
    (>)
                        %
    print               % print ">"
  } repeat              % end repeat loop
  flush                 % flush output buffer
} def                   % prompt ends here
```

The quit Procedure The quit procedure is used to exit the executive by setting the quitflag to true. The quitflag boolean is tested each time through the executive loop and if it is true, interactive mode is exited.

The use of the interrupt command at the end of the procedure is to prohibit commands after the quit command from being executed. According to the *PostScript Language Reference Manual*, the interrupt operator is by default equal to the stop operator.

For example, in the following command sequence, only the commands before the quit procedure are executed since the interrupt or stop operator causes the remainder of the line to be ignored.

```
PostScript(tm) Version 38.0
Copyright (c) 1985 Adobe Systems Incorporated.
PS>(before quit\n) print quit (after quit) print
before quit
```

As you can see, nothing after quit is executed, plus the executive has been exited.

The quit procedure is documented below.

```
%________________________________________________________
% calling format
% quit
% found in:  userdict
% unreferenced
/quit                   %
{                       %
  //execdict            % execdict
  /quitflag             % set quitflag to true in
  true                  % execdict to prepare for
  put                   % exiting the executive
                        %
  interrupt             % execute a stop command
} def                   % quit ends here
```

The quit Operator in systemdict There are two forms of quit, and it is appropriate to describe the differences here. The procedure form of quit in userdict is as described above, however there is an operator called quit in the systemdict which performs a different function.

When executed, the quit operator restarts the printer just as if it has just been turned on. This may come in handy if you don't wish to power off/on your printer to clear virtual memory, however it still takes about the same amount of time to prepare for executing the first job.

Since the quit **operator** is in the systemdict and the quit **procedure** is in the userdict, the quit operator cannot be executed directly, but must be loaded and executed as follows. PostScript also requires that this operation be performed outside the server loop.

```
serverdict begin 0 exitserver
systemdict /quit get exec
```

This prevents the quit operator from inadvertently being executed.

The checkquit Procedure This procedure is used to check the terminating condition of executive. Two conditions are checked, the state of the quitflag variable, and if the communication mode (thumbwheel switch) has been changed. If either condition is true, the exit command is issued to exit the executive loop.

NOTE: Not all PostScript printers use thumbwheel switches as their means to change communication modes. DIP switches and front panel keypads are used on other printers.

```
%___________________________________________________________
% calling format
% checkquit
% found in:  execdict
% referenced in executive
/checkquit              %
{                       %
  quitflag              % put quitflag on stack
                        %
  switchsetting         % what is the position of the
                        % thumbwheel switch?
                        %
  //serverdict          % serverdict
  /saveswitch           % get the last known value of the
  get                   % switch from the serverdict
                        %
                        % see if the current setting is
                        % the same as the last known setting
                        %
  ne                    % switchsetting ne saveswitch?
                        %
  or                    % (switchsetting ne saveswitch) or
                        % (quitflag eq true)
                        %
  {                     % if time to exit
    exit                %
  } if                  % if time to exit
                        %
} def                   % checkquit ends here
```

The intidleproc and batchidleproc Procedures

These two procedures are used upon first entering the executive in the call to idleproc. They are responsible for seting timeouts and performing idle time font caching. The procedures intidleproc (interactive mode idleproc) and batchidleproc (batch mode idleproc) are never referred to by name, but these procedures are imbedded in procedures from which executive can be invoked. If the executive is invoked from batch mode in the procedures 0, 1 or 3 (in serverdict) then the batchidleproc procedure is used as idleproc. If the executive is invoked through the thumbwheel switch setting 2 from the procedure 1 (in specialswitch) then the intidleproc procedure is used as idleproc. The executive is the only procedure which calls idleproc.

A glance at the beginning of these procedures reveals that when idleproc is called the first time it redefines itself as a null procedure so that subsequent calls to idleproc in the executive loop do nothing.

The first call to idleproc is designed to set the job and wait timeouts to 0 (infinite timeout). This is the way the interactive mode is designed to work. In addition, intidleproc performs idle time font caching prior to the user's first input. This is because if executive is invoked from batch mode, idle time font caching has already been done (see discussion of execjob), but if executive is invoked from the thumbwheel switch setting 2, there may be time for idle time font caching before user input is received.

Please refer to the chapter on the thumbwheel switch and the chapter on idle time font caching for details on these issues. The overall concept is briefly presented below.

The intidleproc Procedure When executive is invoked through the special switch, idleproc is defined as intidleproc (interactive idleproc). A quick glance at the procedure reveals that it is only executed once before it is redefined as a null procedure. Thus, when executive is entered, idle time font caching is performed before any user activity, however after the interactive session is begun, no more idle time font caching is done. The first call to intidleproc also sets the job and waittimeout to 0 (infinite timeout).

```
%_____________________________________________________________
% calling format
% intidleproc
% found in:  serverdict
                        %
/intidleproc            % defined as idleproc in
                        % 1 (specialswitch)
{                       %
  /idleproc             % redefine idleproc as
  {                     % a null procedure after the
  }                     % first time it is
  def                   % executed
                        %
                        % define print in the current
                        % dictionary as the print operator
                        % in systemdict
                        %
  /print                % put /print on stack
                        %
  //systemdict          % systemdict
  /print                %
  get                   % fetch print from systemdict
                        %
  def                   % define as print in current dict
                        %
  //serverdict          % serverdict
  begin                 % open serverdict
                        %
  /watchstreams         % this procedure used by UseIdleTime
                        % to watch the serial channels for
                        % the first character of a new job
                        %
  load                  % place watchstreams on the stack in
                        % preparation for the
                        % call to UseIdleTime
                        %
  UseIdleTime           % begin idle time font caching
                        %
                        % idle time is over
                        %
  setrealdevice         % is the print engine ready?
                        %
```

```
                          % this next piece of code has the
                          % effect of setting the job and
                          % wait timeouts to 0 (or infinity)
                          % without changing the
                          % manualfeed timeout
  0                       %
  defaulttimeouts         % get the default time outs
  pop                     %
  exch                    %
  pop                     %
  0                       %
  settimeouts             % set the timeouts for executive
                          % with job and wait set to 0
                          %
  protect                 % this command is used whenever idle
                          % time font caching is finished and
                          % protects characters cached during
                          % idle time from being overwritten
                          %
  end                     % end serverdict
} def                     % intidleproc ends here
```

The batchidleproc Procedure The batchidleproc procedure does not perform idle time font caching since that has already been done by execjob. It only sets the job and waittimeout to 0 (infinite timeout).

```
%____________________________________________________________
% calling format
% batchidleproc
% found in:  serverdict
                          %
/batchidleproc            %
{                         %
  /idleproc               % redefine idleproc as
  {                       % a null procedure after the
  }                       % first time it is
  def                     % executed
                          %
  //serverdict            % serverdict
  begin                   %
                          % this next piece of code has the
                          % effect of setting the job and
                          % wait timeouts to 0 (or infinity)
                          % without changing the
                          % manualfeed timeout
  0                       %
  defaulttimeouts         % get the default time outs
  pop                     %
  exch                    %
  pop                  %
  0                    %
  settimeouts          % set the timeouts for executive
                       % with job and wait set to 0
                       %
  end                  % end serverdict
} def                  % batchidleproc ends here
```

The idleproc Procedure The idleproc is defined as a null procedure after the first statement is entered in interactive mode.

```
%_____________________________________________________________
% calling format
% idleproc
% found in:  execdict
                       %
/idleproc              %
{                      %
} def                  % idleproc ends here
```

Variables used by executive

The quitflag boolean and execdepth integer are both used by the executive procedure.

The quitflag Boolean The quitflag boolean causes the executive procedure to end when set to true.

```
%_____________________________________________________________
% quitflag is found in execdict; type = booleantype
% referenced in checkquit, quit
% executive, and execjob
/quitflag false def
```

The execdepth Integer The execdepth integer determines how many levels deep executive is. The number of levels is indicated by the "PS>" prompt as described in the prompt procedure above.

```
%_____________________________________________________________
% execdepth is found in execdict; type = integertype
% referenced in executive, prompt
% and start
/execdepth 1 def
```

The executive Procedure

At last it is time to dig into the executive procedure. The executive procedure breaks down into three logical sections: initialization, the executive loop, and cleanup. Within the loop, there are several distinct sections which deal with the input filestream, handling errors, and loop termination conditions.

Some excellent examples of the use of the stopped operator are present in the executive procedure. The stopped operator is used to capture errors during prompt execution, file opening, and user statement execution. Also, notice the way the dictionary execdict is opened and closed around the execution of the user's input.

Procedure Initialization The executive first increments the execdepth variable to keep track of how many levels deep the executive is, and then prints the banner and copyright information. It is now ready to enter the main processing loop.

Executive Loop The executive loop entered at this point is responsible for executing the user's input on a line by line basis each time through the loop. At the top of the loop, the quitflag and newerror booleans are set to false to clear the loop-exit and PostScript error conditions.

The prompt procedure then executes to print the "PS>" prompt. The prompt procedure is executed using a stopped operator and the error condition is trapped with an if statement such that if an error occurs, handleerror is executed, the message "Error during prompt execution" is printed and the executive loop is exited.

Next a procedure is placed on the stack and executed by a stopped operator. The procedure makes a call to idleproc to handle initial setup of timeouts and idle time font scan conversion (if executive is invoked through the special switch). This procedure is also responsible for opening the input filestream, %statementedit, which handles the user's input statement.

An ifelse statement is then executed to take action based on the success or failure of opening %statementedit.

The %statementedit open error condition is handled first. The newerror boolean is checked, and if a PostScript error is detected, the undefinedfilename error condition is checked. If true, handleerror is executed. The newerror boolean is then cleared, and the executive loop is exited.

If %statementedit was opened successfully, the section of code is executed which handles user input. First, the input filestream, stmtfile, is placed on the stack. Next, a stopped procedure is responsible for converting the filestream to an executable object and executing it. If an error occurred in the user's input statement it is trapped by an if statement after the stopped procedure. Errors are handled by executing handleerror and then closing the input filestream to flush the rest of the input statement to the end of the line.

At the bottom of the loop, checkquit is executed to test for any terminating conditions for the executive loop.

Procedure Cleanup After the executive loop is exited, quitflag is set to false and execdepth is decreased by 1 level. The doclose boolean is set to false. This tells execjob not to close stdin and stdout, and finally the interrupt (or stop) operator is executed which clears the stacks.

```
%___________________________________________________________
% calling format
% executive
% found in:  userdict
% referenced in 1 (specialswitch)
/executive              %
{                       %
  //execdict            % execdict
  begin                 % open execdict
                        %
  clearinterrupt        %
  disableinterrupt      %
                        %
  /execdepth            % add 1 level to
  execdepth             % execdepth
  1                     %
  add                   %
  def                   %
                        % print the opening banner and
                        % copyright information
                        %
                        % string length =  24
  (\nPostScript(tm) Version )
                        %
  print                 % print opening string
  version               %
  print                 % print version information
                        % string length =  48
  (\nCopyright (c) 1985 Adobe Systems Incorporated.\n)
                        %
  print                 % print copyright information
```

```
                                  %
{                                 % begin executive loop
                                  %
  /quitflag                       % set quitflag false at the
  false                           % top of the loop
  def                             %
                                  %
  //$error                        % $error
  /newerror                       % set newerror to false at top
  false                           % of loop
  put                             %
                                  %
  /prompt                         % place prompt procedure
  load                            % on stack
                                  %
  stopped                         % execute prompt procedure
                                  %
  {                               % if prompt error
                                  %
    handleerror                   % print PostScript error message
                                  %
                                  % string length =  30
    (Error during prompt execution\n)
                                  %
    print                         % print message string above
                                  %
    exit                          % exit executive loop
                                  %
  } if                            % if prompt error
                                  %
                                  % place this procedure on the stack
  {                               % begin procedure
    mark                          % push mark on stack
    idleproc                      % if first time through loop in
                                  % special switch, cache fonts during
                                  % idle time
                                  %
    clearinterrupt                %
                                  %
                                  % create a file object called
                                  % stmtfile using %statementedit
                                  % this is the file object which is
                                  % used to input edited PostScript
                                  % statements to the executive
                                  % processing loop
                                  %
    /stmtfile                     % name of file object on stack
                                  % string length =  14
    (%statementedit)
                                  %
                                  % string length =  1
  (r)
                    %
  file              % create file object with read
                    % attribute
                    %
  def               % define file object as /stmtfile
                    %
  disableinterrupt  %
}                   % end procedure
                    %
stopped             % execute above procedure
                    %
{                   % if %statementedit open error
                    %
                    % this procedure traps problems with
                    % opening %statementedit or running
                    % idle time font caching procedures
                    %
                    % it is very unlikely this will
                    % happen, but it must be covered
                    %
  disableinterrupt  %
  cleartomark       % clear stack up to mark
                    %
  //$error          % $error
  /newerror         % check to see if PostScript
  get               % detected an error
                    %
  {                 % if newerror = true
    //$error        % $error
    /errorname      % fetch the name
    get             % of the error
                    %
    /undefinedfilename
    ne              % errorname ne undefinedfilename ?
                    %
    {               % if errorname ne undefinedfilename
                    %
      handleerror   % print any PostScript error message
                    % except undefinedfilename
                    %
    } if            % if errorname ne undefinedfilename
                    %
    //$error        % $error
    /newerror       %
    false           % reset newerror to false now that
    put             % the error has been taken care of
                    %
    exit            % exit the executive loop for this
                    % kind of error condition
                    %
    } if              % if newerror = true
                      %
  }                   % if %statementedit open error
                      %
  {                   % else if %statementedit open ok
                      %
    pop               % discard mark on stack
    stmtfile          % place filestream on stack
    end               % end execdict before
                      % executing statement
                      %
                      % place the procedure on the stack
                      % which executes the users
                      % input statement
                      %
    {                 % begin procedure
      clearinterrupt  %
      cvx             % convert filestream to executable
      exec            % execute the input statement
      disableinterrupt%
    }                 % end procedure
                      %
    stopped           % execute the above procedure
                      %
    //execdict        % execdict
    begin             % open execdict for the end of this
                      % loop and beginning of next loop
                      %
    {                 % if user input contained error
      disableinterrupt%
      handleerror     % report the error to the user
                      %
      stmtfile        % close the input file so only the
      closefile       % first error in the input statement
                      % is reported - the rest of
                      % the line is discarded
                      %
    } if              % if user input contained error
                      %
  } ifelse            % else if %statementedit open ok
                      %
  checkquit           % was quitflag set to true?
                      % if so exit loop
                      %
} loop                % end executive loop
                      %
/quitflag             % after exiting executive, be sure
false                 % quit flag is set back to
def                   % false
                      %
/execdepth            % decrement the nesting level of the
```

```
  execdepth             % executive by 1
  1                     %
  sub                   %
  def                   %
                        %
  end                   % end execdict
                        %
  //$error              % $error
  /doclose              % setting doclose to false tells
  false                 % execjob not to close stdin and
  put                   % stdout
                        %
  interrupt             % execute stop operator
                        %
} def                   % executive ends here
```

PostScript Interrupts

PostScript has the ability to respond to externally generated interrupts from at least two sources, a serial channel or a changing thumbwheel switch.

According to the *PostScript Language Reference Manual*, if the PostScript interpreter receives a control-c character from a serial communication channel an interrupt error is generated which aborts the currently executing PostScript program.

If the thumbwheel switch is changed during the execution of a program, the program is aborted in much the same manner. This allows the newly selected mode to start from an idle state.

In interactive mode, the response to a control-c interrupt is slightly different. In this case any user commands are immediately aborted and the executive procedure returns back to the next "PS>" prompt.

The executive procedure is able to control the effect of the control-c interrupt through the use of commands such as clearinterrupt and disableinterrupt so that the executive procedure is not aborted itself.

Tips on using executive

As mentioned above, the executive procedure, or interactive PostScript, may be used as a PostScript learning or debugging tool.

Learning PostScript Use of the executive procedure as a PostScript learning tool is very effective, especially when supported by an application such as Lasertalk on the Macintosh. On IBM PCs and compatible computers, a communications program such as Procomm or Lasertalk (IBM PC version) which can open up a two-way serial channel to the PostScript printer is needed.

Example of Interactive Session The following example shows the history of an interactive session in which some browsing through the specialswitch dictionary was done. Several key issues are pointed out.

First, in order to see if communications has been established, several status request (control-t) characters were sent to the printer to which it responded with the %%[status: idle]%% message. At this point the PostScript interpreter is in batch mode and communication is clearly established.

Batch mode has the characteristic of not echoing characters back to the host. Batch mode also does not recognize the backspace character, so any typing mistakes will cause an error.

Next, the command "executive" (with no typing errors) is entered followed by a carriage return. The word executive does not appear on your screen since PostScript is in the batch mode. The PostScript interpreter responds by executing the executive procedure (as described above) and printing the copyright banner and "PS>" prompt.

At this point, the user can enter PostScript commands and interact directly with the interpreter. Commands such as pstack, stack, =, and == are routinely used in interactive mode to view the operand stack as shown in the example below.

If the command entered contains a PostScript error, the user is informed when the line is entered as shown below. Interactive mode allows the user to use the backspace key to correct typing mistakes on the current line.

While in executive mode the response to a status request (control-t) is always waiting for input. Remember the wait timeout is set to infinity so the interpreter is always waiting for user input at the "PS>" prompt. In the case of the example below the status message reads :
%%[status: waiting; source: serial 25]%%

The quit procedure allows the user to return to batch mode as demonstrated by the "%%[status: idle]%%" message returned by the interpreter in response to a status request (control-t).

```
%%[ status: idle ]%%
%%[ status: idle ]%%
%%[ status: idle ]%%

PostScript(tm) Version 38.0
Copyright (c) 1985 Adobe Systems Incorporated.
PS>serverdict begin
PS>specialswitch {} forall
PS>pstack
{-dictionary- --begin--/print {-dictionary- --begin--altflag
 {altout 1 --index----writestring--altout --flushfile--}--if
--stdout --exch----writestring--stdout --flushfile----end--}
--def--/idleproc {/idleproc {}--def--/print -dictionary- /
print --get----def---dictionary- --begin--/watchstreams --
load--UseIdleTime setrealdevice 0 --defaulttimeouts----pop--
--exch----pop--0 settimeouts protect --end--}--def----end--
executive }
1
{/printpageflag false --def--{serverdict --begin--/
watchstreams --load--UseIdleTime setrealdevice --
defaulttimeouts--settimeouts 62 --eescratch--fontname 61 --
eescratch----dup--0 --eq--{--pop--1 }--if--fontname 60 --
eescratch--59 --eescratch----end----clearinterrupt----diablo
--}--stopped----pop--printpageflag {--showpage--}--if--}
0
PS>moveto
%%[ Error: typecheck; OffendingCommand: moveto ]%%
PS>5 5 rlineto
%%[ Error: nocurrentpoint; OffendingCommand: rlineto ]%%
PS>
PS>%%[ status: waiting; source: serial 25 ]%%

PS>quit
%%[ status: idle ]%%
%%[ status: idle ]%%
%%[ status: idle ]%%
```

Programmer's Reference Card If interactive PostScript is being used as a learning tool, I highly recommend purchasing a PostScript Programmer's Instant Reference Card such as the one published by Micro Logic, (201) 342-6518.

Even seasoned PostScript programmers find this chart helpful with its command summary and PostScript code examples.

Debugging PostScript Programs The executive procedure can be used to speed up debugging of PostScript programs. As any PostScript programmer knows, the following error message is not very helpful.

```
%%[ Error: undefined; OffendingCommand: foo ]%%
%%[ Flushing: rest of job (to end-of-file) will be ignored ]%%
```

Since in the batch mode, the rest of the job is flushed (ignored) until the end-of-file, it may be difficult to track down the mysterious "foo" variable.

By entering the interactive mode, and then uploading the program to the printer, the errors can be monitored on a line by line basis. If the PostScript file is large, a capture file on the host computer can be used to record the echoed information. If the PostScript file is very large, the executive command can be inserted in the PostScript file in the code region suspected of having the error. This cuts down on the amount of echoed information up to the point where the executive command is issued.

The use of Multiple executive Levels Using more than one level of the executive can be useful if the user's program contains a control-d at the end, but the programmer desires to remain in the executive procedure after the end-of-job.

In the executive procedure, a control-d has the same effect as issuing a quit command and if the executive is only one level deep, the PostScript interpreter returns to batch mode. When batch mode is reentered, all the stack information from the interactive session is lost.

Using multiple executive levels, the user can remain in the executive procedure and preserve all the stack information from the previous level.

In the following example, the second level of the executive is entered and some strings placed on the operand stack. After the flush operator, an end-of-file (control-d) character is entered which executes the current line and returns to the next lower level of the executive. The pstack operator reveals that the operand stack has been preserved.

```
PS>executive

PostScript(tm) Version 38.0
Copyright (c) 1985 Adobe Systems Incorporated.
PS>>(Inside PostScript\n)
PS>>(What's on the stack?\n) print flushWhat's on the stack?
PS>>PS>pstack
(Inside PostScript
)
PS>
```

Just for Fun

If you get tired of the "PS>" prompt, you can redefine the prompt procedure as shown in the following example.

```
PostScript(tm) Version 38.0
Copyright (c) 1985 Adobe Systems Incorporated.
PS>/prompt
PS>{(Hello Frank!) print execdepth {(:) print} repeat flush }
PS>def
Hello Frank!:executive         % enter second level

PostScript(tm) Version 38.0
Copyright (c) 1985 Adobe Systems Incorporated.
Hello Frank!::
Hello Frank!::
```

Chapter 6

The ==dict

Introduction

The ==dict is one of the two self contained dictionaries (mydict is discussed in another chapter) in this implementation of PostScript. That is, the ==dict is only defined within itself, and furthermore, it is only referenced once as an immediately evaluated name, in the == procedure.

Using ==dict

The == procedure is most often used in the executive or interactive mode for displaying the syntactic representation of an object on top of the operand stack. The syntactic representation of a procedure for example could be: {2 3 add} verses the text representation displayed as --nostringval-- (as produced by the = procedure). The best way to understand the difference is to try displaying several different data type using both the = and == procedures. Usually == is the most helpful of the two.

Displaying the systactic representation of a PostScript object is what the procedures in ==dict do, and Adobe uses some interesting techniques in the process.

PostScript Datatypes In ==dict you find procedures named after every PostScript data type as follows:

1. arraytype
2. booleantype
3. dicttype
4. filetype
5. fonttype
6. integertype
7. marktype
8. nametype
9. nulltype
10. operatortype
11. packedarraytype
12. realtype
13. savetype
14. stringtype

Each of the above procedures is responsible for printing the object of that type. Arraytype and packedarraytype objects are printed using special techniques to print arrays within arrays.

Examples of Output An example of the use of the == procedure in the interactive mode is given below. (You don't have to decipher the procedure listed below since it is explained later.)

```
PS>==dict /arraytype get
PS>==
{dup rcheck {dup xcheck {({)tprint {typeprint }forall (})
tprint }{([)tprint {typeprint }forall (])tprint }ifelse }{
pop (-array- )tprint }ifelse }
```

Compare the == output above with the discussion of the arraytype procedure below. The output of == is very compact, and hard to follow, especially for long procedures such as the one below.

```
PS>$printerdict 0 get
PS>==
{300 3276 75 12 {88 148 {-dictionary- --begin--/jobstate (
printing)--def--0 --setblink----margins----exch--4 -1 --roll
----add--3 1 --roll----add--2 --div----round----cvi--
manualfeed {manualfeedtimeout }{0 }--ifelse--#copies --
redwrite--/jobstate (busy)--def--1 --setblink----end--}--
exec--}{-dictionary- --begin--/proc --exch----def--/yoffset
--exch----def--/xoffset --exch----def--/height --exch----def
--/width --exch----def--300 72 --div--0 0 -300 72 --div--
xoffset --neg--height yoffset --add--mtx --astore--width
height /proc --load----framedevice--60 45 {--abs----exch----
abs--2 --copy----add--1 --gt--{1 --sub----dup----mul----exch
--1 --sub----dup----mul----add--1 --sub--}{--dup----mul----
exch----dup----mul----add--1 --exch----sub--}--ifelse--}--
setscreen--{}--settransfer----initgraphics----erasepage----
end--}--exec--}
```

This is the procedure for the 0 (or letter) command which is documented in the $printerdict chapter.

Using the Adobe code examples below, you may wish to develop a different format for displaying packed arrays or other data types for debugging purposes.

Object Attributes If objects have executable or locked attributes, the procedures take this into account. Thus, ==dict not only define all PostScript data types, but its procedures define all the attributes of each type.

Welcome to the world of ==dict.

Exploring ==dict

The detailed examination of == and ==dict begins with a discussion of its control variables and control procedures followed by the documentation of each "type" procedure.

Control Variables

References are made to cp (character pointer), rmargin (right margin) and NL (newline). The cp integer is used to keep track of how many characters have been printed on a line of output to the reverse channel. The value of cp is compared to rmargin to determine if the output exceeds the right margin limit. NL simply places a newline on the stack.

Control Procedures

The following section documents each of the top level procedures involved in executing the == procedure.

The main control procedures ==, typeprint and tprint are explained first and then each of the type procedures are documented.

The == Procedure The == procedure displays the syntactic representation of the PostScript object on the top of the stack. The == procedure is the highest level procedure responsible for printing the syntactic representation. Thus it is used to open the ==dict, initialize some variables and make a call to typeprint. It finishes by printing a newline and closing ==dict. Everything centers around the call to typeprint.

```
%____________________________________________________________
% calling format
% anything ==
% referenced in pstack and Run
% found in:  systemdict
/==                        %
{                          %
  //==dict                 % ==dict
                           % only reference to ==dict
  begin                    %
  /cp                      %
  0                        % set the character pointer to 0
  def                      %
  typeprint                % begin printing using typeprint
  NL                       % place newline on the stack
  print                    % print newline
  end                      % close ==dict
} def                      % == ends here
```

The typeprint Procedure The typeprint procedure invokes the procedures which print each of the object types.

Adobe uses a clever technique to print the different types. The typeprint procedure contains three statements which appear rather cryptic at first.

First, the dup operator stores an extra copy of the object on the stack. Next the type operator returns the type of the object on the top of the stack. The object type is in the form of a name such as arraytype or integertype. The exec operator then executes the object's type name. Remember that == opened ==dict, so executing the object's type name has the effect of calling the ==dict procedure by that name. Each respective procedure is responsible for printing information about that type.

```
%________________________________________________________________
% calling format
% anything typeprint
% referenced in ==, arraytype
% and packedarraytype
% found in:  ==dict
/typeprint              %
{                       %
  dup                   % save a copy of the object
  type                  % get the type of the object
  exec                  % exec invokes one of the type
                        % procedures in ==dict
} def                   % typeprint ends here
```

The tprint Procedure The tprint procedure is responsible for printing the strings returned by the type procedures and checking to see if the right margin has been exceeded.

```
%________________________________________________________________
% calling format
% string tprint
% referenced in marktype, cvsprint,
% dicttype, filetype, fonttype,
% marktype, nametype, nulltype,
% operatortype, packedarraytype,
% savetype, stringtype
% found in:  ==dict
/tprint                 %
{                       %
  dup                   %
  length                % get the length of the next
                        % string to print
  cp                    %
  add                   % add the current position to
                        % the length
  rmargin               %
  gt                    % is the new length greater
                        % than the right margin?
  {                     % if new cp  right margin
    NL                  %
    print               % print a newline
    /cp                 %
    0                   % set the character pointer
    def                 % back to the left margin of 0
  } if                  % if new cp  right margin
                        %
                        % update the character pointer
  dup                   %
  length                % get the string length again
  cp                    %
  add                   % add the length to current cp
  /cp                   % put the name on the stack
  exch                  % prepare to define new cp value
  def                   % define cp
                        %
  print                 % print the string in its
                        % proper place
} def                   % tprint ends here
```

The cvsprint Procedure The cvsprint procedure converts the value on the top of the stack into a string and then prints it using tprint.

```
%____________________________________________________________
% calling format
% anything cvsprint
% referenced in booleantype,
% integertype, nametype,
% realtype
% found in:  ==dict
/cvsprint               %
{                       %
  =string               % put junk string from systemdict
                        % on the top of the stack
  cvs                   % convert object on stack to string
                        %
  tprint                % use tprint to print the string
                        % string length =  1
  ( )
                        %
  tprint                % put a space after each object
                        % printed
} def                   % cvsprint ends here
```

Type procedures

Each of the 14 PostScript data type procedures are listed below. Each procedure is designed to return a printable string representation of its respective type.

The following collection of type procedures represents the complete list of PostScript data types, and each procedure takes into account any special attributes for that type such as readable or executable.

The type procedures are documented as unreferenced procedures in the sense that they are not called by name by any other internal PostScript procedures. As we learned above, the type procedures are indirectly called by the typeprint procedure.

The arraytype Procedure The code for the arraytype shows us several characteristics of arrays.

First, a check is made to see if the array is locked with the rcheck operator. If the array is locked, it cannot be displayed and the string "-array-" is returned.

Next, xcheck is used to determine if the array is executable. If it is executable, curly braces ("{") are used, and if it is not executable, square brackets ("[") are used. This points out the close relationship between procedures and arrays. An executable array is a procedure or collection of objects that can be executed and a nonexecutable array is simply a collection of objects.

After the opening brace or bracket is printed, each element of the array is accessed using a forall loop, and the individual elements are printed using the typeprint procedure. After the elements are printed, the closing brace or bracket is printed.

This simple code structure allows typeprint to call itself (known as recursion) to print arrays within arrays or procedures within procedures.

```
%____________________________________________________________
% calling format
% array arraytype
% unreferenced
% found in:  ==dict
/arraytype              %
{                       %
  dup                   %
  rcheck                % is array locked?
  {                     % if array not locked
```

```
    dup                     %
    xcheck                  % is array executable?
    {                       % if array executable
                            %
                            % in this case the array is an
                            % executable procedure
                            % string length =  1
      ({)
                            %
      tprint                % print opening curly brace
                            %
                            % the following forall statement
                            % takes each element of the
                            % executable array and prints it
                            % this takes advantage of recursion
                            % in the event that arrays are
                            % nested within arrays
                            %
      {                     % begin forall
        typeprint           % print each object
      } forall              %
                            % string length =  1
      (})
                            %
      tprint                % print the closing curly brace
    }                       %
    {                       % if array not executable
                            %
                            % in this case, the array contains
                            % data objects
                            % string length =  1
      ([)
                            %
      tprint                % print the opening square bracket
                            %
                            % the following forall statement
                            % takes each element of the
                            % array and prints it
                            % this takes advantage of recursion
                            % in the event that arrays are
                            % nested within arrays
                            %
      {                     % begin forall
        typeprint           % print each object
      } forall              %
                            % string length =  1
      (])
                            %
      tprint                % print closing square bracket
    } ifelse                % if array executable/not executable
  }                         %
  {                       % if array locked
    pop                   % dump object
                          % string length =  8
    (-array- )
                          %
    tprint                % print "-array-" message since
                          % we can't look inside
  } ifelse                % if array unlocked/locked
} def                     % arraytype ends here
```

The booleantype Procedure For a booleantype, print the word "true" or "false".

```
%_______________________________________________________________
% calling format
% boolean booleantype
% unreferenced
% found in:  ==dict
/booleantype              %
{                         %
  cvsprint                % convert the boolean object
                          % to a string and print
} def                     % booleantype ends here
```

The dicttype Procedure Print "-dictionary-" for a dictionary object.

```
%_______________________________________________________________
% calling format
% dictionary dicttype
% unreferenced
% found in:  ==dict
/dicttype                 %
{                         %
  pop                     % discard the dictionary object
                          % string length =  13
  (-dictionary- )
                          %
  tprint                  % print "-dictionary-"
} def                     % dicttype ends here
```

The filetype Procedure Print "-filestream-" for a file object.

```
%________________________________________________________________
% calling format
% file filetype
% unreferenced
% found in:  ==dict
/filetype               %
{                       %
  pop                   % discard the file object
                        % string length =  13
  (-filestream- )
                        %
  tprint                % print "-filestream-"
} def                   % filetype ends here
```

The fonttype Procedure Print "-fontid-" for a font object.

```
%________________________________________________________________
% calling format
% font fonttype
% unreferenced
% found in:  ==dict
/fonttype               %
{                       %
  pop                   % discard the font object
                        % string length =  9
  (-fontid- )
                        %
  tprint                % print "-fontid-"
} def                   % fonttype ends here
```

The integertype Procedure Take the integer object, convert it to a string and print it.

```
%________________________________________________________________
% calling format
% integer integertype
% unreferenced
% found in:  ==dict
/integertype            %
{                       %
  cvsprint              % convert the integer to a string
                        % and print it
} def                   % integertype ends here
```

The marktype Procedure Print "-mark-" for a mark object. Remember a mark object in PostScript is the same as an opening square bracket "[".

```
%_______________________________________________________________
% calling format
% mark marktype
% unreferenced
% found in:  ==dict
/marktype                %
{                        %
  pop                    % discard the mark object
                         % string length =  7
  (-mark- )
                         %
  tprint                 % print "-mark-"
} def                    % marktype ends here
```

The nametype Procedure Print a nametype. If the name is not executable, put a "/" in front of the name.

```
%_______________________________________________________________
% calling format
% name nametype
% unreferenced
% found in:  ==dict
/nametype                %
{                        %
  dup                    %
  xcheck                 % is the name executable
  not                    %
  {                      % if not executable
                         % string length =  1
    (/)
                         %
    tprint               % print the "/"
  } if                   % if not executable
  cvsprint               % convert the name to a
                         % string and print it
} def                    % nametype ends here
```

The nulltype Procedure Print "-null-" for a null type.

```
%___________________________________________________________
% calling format
% null nulltype
% unreferenced
% found in:  ==dict
/nulltype              %
{                      %
  pop                  % discard the null object
                       % string length =  7
  (-null- )
                       %
  tprint               % print "-null-"
} def                  % nulltype ends here
```

The operatortype Procedure Print the name of a PostScript operator between pairs of dashes like "--add--".

```
%___________________________________________________________
% calling format
% operator operatortype
% unreferenced
% found in:  ==dict
/operatortype          %
{                      %
                       % string length =  2
  (--)
                       %
  tprint               % print leading "--"
  =string              % put junk string on stack
  cvs                  % convert operator name to string
  tprint               % print the name of the operator
                       % string length =  2
  (--)
                       %
  tprint               % print trailing "--"
} def                  % operatortype ends here
```

The packedarraytype Procedure The packedarraytype procedure is virtually the same as the arraytype procedure, except that it prints "-packedarray-" instead of "-array-" if the packed array is locked. A packedarraytype object differs from an arraytype in that it has read-only access and is more memory efficient.

First, packedarraytype makes a check to see if the array is locked with the rcheck operator. If the array is locked, its contents cannot be displayed and the string "-packedarray-" is returned.

Next, xcheck is used to determine if the packedarray is executable. If it is executable, curly braces ("{") are used, and if it is not executable square brackets ("[") are used.

After the opening brace or bracket is printed, each element of the packedarray is accessed using a forall loop, and the individual elements are printed using the typeprint procedure. After the elements are printed, the closing brace or bracket is printed.

This simple code structure allows typeprint to call itself (known as recursion) to print packedarrays within packedarrays or procedures within procedures.

```
%_____________________________________________________________
% calling format
% packedarray packedarraytype
% unreferenced
% found in:  ==dict
/packedarraytype        %
{                       %
  dup                   %
  rcheck                % is packed array locked?
  {                     % if packed array not locked
    dup                 %
    xcheck              % is packed array executable?
    {                   % if packed array executable
                        %
                        % in this case the packed array is
                        % an executable procedure
                        % string length =  1
      ({)
                        %
      tprint            % print opening curly brace
                        %
                        % the following forall statement
                        % takes each element of the
                        % executable packed array
                        % and prints it
                        % this takes advantage of recursion
                        % in the event that packed arrays
                        % are nested within arrays
                        %
      {                 % begin forall
        typeprint       % print each object
      } forall          %
```

```
                        % string length =  1
      (})
                        %
      tprint            % print the closing curly brace
    }                   %
    {                   % if packed array not executable
                        %
                        % in this case, the packed array
                        % contains data objects
                        % string length =  1
      ([)
                        %
      tprint            % print the opening square bracket
                        %
                        % the following forall statement
                        % takes each element of the
                        % packed array and prints it
                        % this takes advantage of recursion
                        % in the event that packed arrays
                        % are nested within packed arrays
                        %
      {                 % begin forall
        typeprint       % print each object
      } forall          %
                        % string length =  1
      (])
                        %
      tprint            % print closing square bracket
    } ifelse            % if array executable/not executable
  }                     %
  {                     % if packed array locked
    pop                 % dump object
                        % string length =  8
    (-packedarray- )
                        %
    tprint              % print "-packedarray-" message
                        % since we can't look inside
  } ifelse              % if packed array unlocked/locked
} def                   % packedarraytype ends here
```

The realtype Procedure Take the real object, convert it to a string and print it.

```
%______________________________________________________________
% calling format
% real realtype
% unreferenced
% found in:  ==dict
/realtype               %
{                       %
  cvsprint              % convert the real number to
                        % a string and print it
} def                   % realtype ends here
```

The savetype Procedure Print "-savelevel-" for a save object.

```
%______________________________________________________________
% calling format
% savelevel savetype
% unreferenced
% found in:  ==dict
/savetype               %
{                       %
  pop                   % discard save object
                        % string length =  12
  (-savelevel- )
                        %
  tprint                % print "-savelevel-"
} def                   % savetype ends here
```

The stringtype Procedure The stringtype procedure checks to see if a string is locked. If it is locked "-string-" is printed; if open, the string is printed enclosed in parenthesis.

```
%______________________________________________________________
% calling format
% string stringtype
% unreferenced
% found in:  ==dict
/stringtype             %
{                       %
  dup                   %
  rcheck                % is string locked?
  {                     % if string not locked
                        % string length =  1
    (\()
                        %
```

```
      tprint               % print "("
      tprint               % print contents of string
                           % string length =  1
      (\))
                           %
      tprint               % print ")"
    }                      %
    {                      % if string locked
      pop                  % discard the locked string
                           % string length =  9
      (-string- )
                           %
      tprint               % print "-string-"
    } ifelse               % if string unlocked/locked
  } def                    % stringtype ends here
```

Chapter 7

Miscellaneous Procedures

Introduction

This chapter documents procedures which have not logically fallen into the groups of procedures discussed in other chapters. Most of these procedures have to do with sending information back to the host on the reverse channel and would tend to be most used in interactive mode.

These procedures include =print, =, pstack, stack and Run. The last procedure to be discussed is findfont.

Also discussed in this chapter is a string in the systemdict called =string.

The = Procedure and =print

The "=" procedure is commonly used in interactive mode to display the text representation of an object. The object is fetched from the top of the stack and displayed. Several examples follow from a session in interactive mode.

```
PS>0 =
0
PS>/somename =
somename
PS>{(procedure)} =
--nostringval--
PS>[0 1 2 3] =
--nostringval--
```

There is usually some confusion between the use of = and ==. The == operator displays the syntactic representation of an object and is illustrated with some examples below.

```
PS>0 ==
0
PS>/somename ==
/somename
PS>{(procedure)} ==
{(procedure)}
PS>[0 1 2 3] ==
[0 1 2 3 ]
PS>
```

More information on the == procedure is in the chapter on the ==dict.

While the =print procedure has uses as a stand alone procedure, it is also embedded in the = procedure below.

= Documented The = and =print procedures check to see if the object on the top of the stack is a string. If the object is not a string, it is converted to a string and printed.

```
%______________________________________________________________
% calling format
% any =
% referenced in execjob,
% initprinter, printerstatus,
% setrealdevice, warmedup
% found in:  systemdict
                       %
                       % print a text representation
                       % of an object's value
/=                     %
{                      %
                       % //=print starts here
                       %
                       % place =print on stack
  {                    %
    dup                % first check to see if the
    type               % object is a string
    /stringtype        %
    ne                 % if not a string,
    {                  % make it one
                       % scratch string
                       % string length =  128
      (This is a 128 byte string known as =string.........)
                       %
      cvs              % convert object to string
    } if               %
    print              % send info back over
                       % the reverse channel
  }                    %
                       % //=print ends here
                       %
  exec                 % execute the =print
                       % procedure
                       % carriage return string
                       % string length =  1
  (\n)
                       %
  print                % send CR back
} def                  % = ends here
```

=print Documented As you can see from above, =print is embedded in = as an immediately evaluated name.

```
%_______________________________________________________________
% calling format
% any =print
% referenced in pstack, Run
% found in:  systemdict
                        %
                        % print whatever presented
/=print                 %
{                       %
  dup                   %
  type                  % if something is not equal to
  /stringtype           % a string then
  ne                    % convert it to a string
  {                     % use a scratch string
                        % string length =  128
    (This is a 128 byte string known as =string...........)
                        %
    cvs                 % convert to a string
  } if                  %
  print                 % send string back over the
                        % reverse channel
} def                   % =print ends here
```

The stack and pstack Procedures

These procedures are used to nondestructively view the contents of the operand stack. The stack and pstack procedures display the stack information using "=" and "==" respectively. The following examples illustrate the use of both procedures in an interactive mode session.

```
PS>0 /somename {(procedure)} [0 1 2 3]
PS>stack
--nostringval--
--nostringval--
somename
0
PS>pstack
[0 1 2 3 ]
{(procedure)}
/somename
0
PS>
```

In order to nondestructively print the stack, both procedures first make copies of the entire operand stack. Next, the =print or == procedure is applied in a loop until all items are printed.

stack Procedure After making a copy of the operand stack, the =print procedure is used to print each stack element.

```
%_______________________________________________________________
% calling format
% stack
% unreferenced
% found in:  systemdict
/stack                  %
{                       %
  count                 % count number of object on stack
  dup                   % add 1 to the number of objects
  1                     % so that when the stack is
  add                   % copied, the count is copied
  copy                  % for the benefit of the
                        % repeat operator
                        %
  {                     % begin repeat loop
                        % //=print starts here
                        %
                        % place =print on stack
    {                   %
      dup               % first check to see if the
      type              % object is a string
      /stringtype       %
      ne                % if not a string,
      {                 % make it one
                        % scratch string
                        % string length =  128
        (This is a 128 byte string known as =string........)
                        %
        cvs             % convert object to string
      } if              %
      print             % send info back over
                        % the reverse channel
    }                   %
                        % //=print ends here
    exec                %
                        % string length =  1
    (\n)
                        %
    print               % print CR
  } repeat              % end repeat loop
                        %
  pop                   % discard count value on stack
} def                   % stack ends here
```

pstack procedure After making a copy of the stack, the == procedure is used to print each element of the stack.

```
%____________________________________________________________
% calling format
% pstack
% unreferenced
% found in:  userdict
/pstack                 %
{                       %
  count                 % count number of object on stack
  dup                   % add 1 to the number of objects
  1                     % so that when the stack is
  add                   % copied, the count is copied
  copy                  % for the benefit of the
                        % repeat operator
                        %
  {                     % begin repeat loop
    ==                  % print object
  } repeat              % end repeat loop
  pop                   % discard count value on stack
} def                   % pstack ends here
```

The Run Procedure

This procedure is simply a variation of the run operator which additionally prints the name of the file being executed.

The run operator is short for the command sequence:

```
(r) file cvx exec
```

```
%____________________________________________________________
% calling format
% (filename) Run
% unreferenced
% found in:  systemdict
/Run                    %
{                       %
  dup                   % make a copy of the filename
  ==                    % print the filename
  flush                 % flush the output buffer
  run                   % now run the file
} def                   % Run ends here
```

The findfont Procedure

This procedure is responsible for determining if a requested font is actually in the FontDirectory dictionary.

First, the font name is checked to see if it is present in the FontDirectory. If it is known, the font information is fetched, otherwise the Courier font is substituted.

```
%_____________________________________________________________
% calling format
% fontname findfont fontdictionary
% unreferenced
% found in:  systemdict
/findfont                   %
{                           %
                            % determine if font name is
                            % known
                            %
  dup                       % copy font name
  //FontDirectory           % FontDirectory
  exch                      %
  known                     % FontDirectory fontname known
  {                         % if font known
    //FontDirectory         % FontDirectory
    exch                    % get font dictionary and
    get                     % leave on stack
  }                         % if font known
                            %
  {                         % else if font unknown
                            %
                            % use =print to print name
                            % not found in FontDirectory
                            %
                            % //=print starts here
                            %
                            % place =print on stack
    {                       %
      dup                   % first check to see if the
      type                  % object is a string
      /stringtype           %
      ne                    % if not a string,
      {                     % make it one
                            % scratch string
                            % string length =  128
       (This is a 128 byte string known as =string...........)
                            %
        cvs                 % convert object to string
      } if                  %
      print                 % send info back over
                            % the reverse channel
    }                       %
```

```
                            % //=print ends here
                            %
    exec                    % execute =print to print
                            % font name
                            % string length =  27
    ( not found, using Courier.\n)
                            %
    print                   % print above string
    flush                   % flush buffer
                            %
    //FontDirectory         % FontDirectory
    /Courier                % fetch Courier font and place
    get                     % it on the stack
  } ifelse                  % else if font unknown
                            %
} def                       % findfont ends here
```

The =string String

This simple 128 byte string plays an important role in the structure of the PostScript interpreter because we find it imbedded (//=string as an immediately evaluated expression) in every procedure that uses temporary strings.

The use of =string over and over is one of the methods Adobe uses to conserve VM since memory for =string is only allocated once when it is created.

```
%___________________________________________________________
% =string is found in systemdict
% not executable; length = 128
/=string            %
(This is a 128 byte string known as =string...........)
def                          % =string ends here
```

Several of the procedures above illustrate the use of =string.

Part IV

The Printer Control Group

Chapter 8

Idle Time Font Scan Conversion

Introduction

This section looks into the method used to cache fonts during idle time. Idle time font scan conversion uses the time between jobs to scan convert characters in anticipation of the characters which will be used for the next job. What this means is that during the time the printer waits for the next job to start, a selected group of characters are scan converted and placed into the font cache.

Scan conversion is the process used to convert a character from outline format into a bit representation. These bit maps represent the definition of a character as a specified font, scale or size, and rotation or orientation. Once the bit map is in the font cache, it can be used over and over each time the specific character is requested by the user program.

Perhaps you have noticed that the same page of data runs faster the second time it is printed. This is because the first time the page is printed, the work of converting character outlines and placing them in the font cache is going on, but when the page is run again, the characters are taken directly from the font cache and placed on the page since they have already been scan converted.

The concept of the font cache is a very powerful feature of PostScript, since many documents use the same characters over and over again. Once the core set of characters is cached, succeeding pages are rendered fairly quickly if they use the same fonts.

While idle time font scan conversion can be a powerful tool, it is unfortunate that its implementation is poorly documented and the setidlefonts command is difficult to understand and use. Thus, very few users take advantage of this feature.

This chapter looks at how the idle font information is used and how the concept is implemented in the resident PostScript software.

Chapter Organization

The topic is presented as two parts. First, the structure of the idle time fonts data structures and their storage and retrieval from the EEPROM is studied. Secondly, the method in which the character data is scan converted during idle time is looked at.

Welcome to the world of idle time.

Idle Time Data Structures

The $idleTimeDict dictionary contains all the data structures and procedures to access the data structures which contain idle time font scan conversion information. In this dictionary, an array of all the font names and an array of the font scan conversion data are the primary data structures. The other elements of the dictionary are either strings which are referenced in the font scan conversion data array, or procedures which access the array.

The ReadIdleFonts procedure in the userdict is responsible for building the font scan conversion data array and it is discussed in detail.

The ROMnames Font Name Array This array contains the PostScript names for the fonts. In this case thirteen fonts are defined. In certain instances (such as with the setidlefonts operator), PostScript fonts are referred to by number rather than name, and the number used is the index of the font name in the ROMnames array.

```
%____________________________________________________________
% found in:  $idleTimeDict
% referenced in ReadIdleFonts,
% fontname
/ROMnames               %
[                       % array begin
  /Courier              % index 0
  /Courier-Bold         % index 1
  /Courier-Oblique      % index 2
  /Courier-BoldOblique  % index 3
  /Times-Roman          % index 4
  /Times-Bold           % index 5
  /Times-Italic         % index 6
  /Times-BoldItalic     % index 7
  /Helvetica            % index 8
  /Helvetica-Bold       % index 9
  /Helvetica-Oblique    % index 10
  /Helvetica-BoldOblique% index 11
  /Symbol               % index 12
] def                   % ROMnames ends here
```

The Idle Time Data Array: idleArry The idleArry contains the information used by PostScript for idle time font scan conversion. The array can be broken down into groups containing 5 pieces of information each. This information includes the following:

- The name of the font (related to the font index, called font by setidlefonts).
- The scale of the font in the x direction (Sx).
- The scale of the font in the y direction (Sy).
- The rotation of the font on the page (rot).
- The group of characters within the font to be scan converted (derived from the number of characters, nchars).

The data in the idleArry is surprisingly easy to understand compared to the way these five pieces of information must be presented to the setidlefonts operator. In fact, it is seen later that the ReadIdleFonts procedure interprets the idlefonts information to build this array. The reason for the computations used in the setidlefonts operator is probably that it is required by the Adobe algorithm used to pack the data into the EEPROM. So much for user friendliness!

The idleArry presented below is the default used by PostScript when no data is presented to setidlefonts. Also note that pre-scanned characters which are permanently resident in ROM are not a part of this array.

```
%_______________________________________________________________
% found in:  $idleTimeDict
% referenced in UseIdleTime,
% bumpI, idlA
/idleArry               %
[                       % array begin

  /Courier              % font name
  10                    % x scale
  10                    % y scale
  0                     % rotation
                        % conversion characters
                        % string length =  94
  (abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ\
   0123456789.,;?:-()'"!+[]$%&*/_=@#`{}^~|\)
                        %

  /Times-Roman          % font name
  10                    % x scale
  10                    % y scale
```

```
0                       % rotation
                        % conversion characters
                        % string length =  81
(abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ\
 0123456789.,;?:-()'"!+[]$%&*/)
                        %

/Helvetica              % font name
10                      % x scale
10                      % y scale
0                       % rotation
                        % conversion characters
                        % string length =  81
(abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ\
 0123456789.,;?:-()'"!+[]$%&*/)
                        %

/Times-Bold             % font name
12                      % x scale
12                      % y scale
0                       % rotation
                        % conversion characters
                        % string length =  26
(abcdefghijklmnopqrstuvwxyz)
                        %

/Helvetica-Bold         % font name
12                      % x scale
12                      % y scale
0                       % rotation
                        % conversion characters
                        % string length =  26
(abcdefghijklmnopqrstuvwxyz)
                        %

/Times-Bold             % font name
10                      % x scale
10                      % y scale
0                       % rotation
                        % conversion characters
                        % string length =  26
(abcdefghijklmnopqrstuvwxyz)
                        %

/Helvetica-Bold         % font name
10                      % x scale
10                      % y scale
0                       % rotation
                        % conversion characters
                        % string length =  26
(abcdefghijklmnopqrstuvwxyz)
```

```
                          %

 /Courier-Bold            % font name
 10                       % x scale
 10                       % y scale
 0                        % rotation
                          % conversion characters
                          % string length =  26
 (abcdefghijklmnopqrstuvwxyz)
                          %

 /Times-Roman             % font name
 14                       % x scale
 14                       % y scale
 0                        % rotation
                          % conversion characters
                          % string length =  62
 (abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ\
  0123456789)
                          %

 /Helvetica               % font name
 14                       % x scale
 14                       % y scale
 0                        % rotation
                          % conversion characters
                          % string length =  62
 (abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ\
  0123456789)
                          %

 /Times-Bold              % font name
 14                       % x scale
 14                       % y scale
 0                        % rotation
                          % conversion characters
                          % string length =  26
 (abcdefghijklmnopqrstuvwxyz)
                          %

 /Helvetica-Bold          % font name
 14                       % x scale
 14                       % y scale
 0                        % rotation
                          % conversion characters
                          % string length =  26
 (abcdefghijklmnopqrstuvwxyz)
                          %
] def                     % idleArry ends here
```

Character Group Strings Though these strings are unreferenced by name, we see that they appear in the default idleArry data structure. These strings contain distinct character groups such as lower case only; lower case, upper case and numbers; and lower case, upper case, numbers, and punctuation.

It turns out that these would be the most likely groupings of characters a user would pick for scan converting during idle time. Also notice that the shorter strings are always substrings of the longer strings. This is because of the definition of the setidlefonts parameter which makes the scan characters the first "n" characters of the 94 character group in the ascii94 string.

ascii26 This string contains only lower case characters.

```
%______________________________________________________________
% ascii26 is found in $idleTimeDict
% not executable; length = 26
% unreferenced
/ascii26
(abcdefghijklmnopqrstuvwxyz)
def                        % ascii26 ends here
```

ascii62 This string contains lower and upper case letters, plus numbers.

```
%______________________________________________________________
% ascii62 is found in $idleTimeDict
% not executable; length = 62
% unreferenced
/ascii62
(abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ\
0123456789)
def                        % ascii62 ends here
```

ascii81 This string contains letters, numbers and some punctuation characters.

```
%______________________________________________________________
% ascii81 is found in $idleTimeDict
% not executable; length = 81
% unreferenced
/ascii81
(abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ\
0123456789.,;?:-()'"!+[]$%&*/)
def                     % ascii81 ends here
```

ascii94 This string contains letters, number, and additional punctuation characters not found in ascii81.

```
%______________________________________________________________
% ascii94 is found in $idleTimeDict
% not executable; length = 94
% referenced in ReadIdleFonts
/ascii94
(abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ\
0123456789.,;?:-()'"!+[]$%&*/_=@#`{}^~|\)
def                     % ascii94 ends here
```

Control Variables and Procedures

Before we get into the main control procedures of UseIdleTime and ReadIdleFonts, the control variables and procedures are explained.

The idleStrI and idleStr Variables The variable idleStrI is used in UseIdleTime as an index into the string idleStr as each character in the string is placed in the font cache.

```
%______________________________________________________________
% idleStr is found in $idleTimeDict
% not executable; length = 26
% referenced in UseIdleTime
/idleStr
(abcdefghijklmnopqrstuvwxyz)
def                     % idleStr ends here
```

idleStrI This is the index into idleStr.

```
%_____________________________________________________________
% idleStrI is found in $idleTimeDict
% type = integertype
% referenced in UseIdleTime
/idleStrI 0 def
```

The idleI Variable The variable idleI is used as an index into the idle-Arry data structure.

```
%_____________________________________________________________
% idleI is found in $idleTimeDict
% type = integertype
% referenced in UseIdleTime,
% ReadIdleFonts, idleA, bmpI
/idleI 20 def
```

The eeinfo Variable A variable which is unreferenced in any dictionary (because the default idleArry is used and eeinfo is never created) but appears in the following procedures is eeinfo. Only if idle time data is given to setidlefonts will eeinfo be used since its purpose is to retrieve the idlefonts information which it finds on the stack during the execution of ReadIdle-Fonts.

The boundsCheck Procedure The boundsCheck procedure checks to make sure that the font and nchars information returned by idlefonts is within the bounds of the ROMnames array and ascii94 string respectively. The font and nchars information has been placed in the eeinfo variable by the bmpI procedure.

If the EEPROM data returned by idlefonts is out of bounds, eeinfo is set to 0.

```
%_____________________________________________________________
% calling format
% array boundsCheck
% referenced in ReadIdleFonts
% found in:  $idleTimeDict
/boundsCheck            %
{                       %
  length                % get the length of ascii94
                        % or ROMnames
  eeinfo                  % put eeinfo on stack
  le                      % length  eeinfo ?
  {                       % if length  eeinfo
    /eeinfo               %
    0                     % set eeinfo to 0
    def                   %
  } if                    % if length  eeinfo
} def                     % boundsCheck ends here
```

The bmpI Procedure This procedure is used to set the eeinfo variable to the EEPROM value on the top of the stack and decrement the index idleI as it steps through the idleArry array.

On exiting the bmpI procedure, idleArry and idleI are left on the stack in preparation for a put operator.

```
%_____________________________________________________________
% calling format
% eprominfo bmpI idleArry idleI
% referenced in ReadIdleFonts
% found in:  $idleTimeDict
/bmpI                     %
{                         %
  /eeinfo                 % store the eerom value on the
  exch                    % top of the stack in the
  def                     % eeinfo variable
                          %
  /idleI                  % decrement the idleI
  idleI                   % index
  1                       %
  sub                     %
  def                     %
                          %
  idleArry                % leave idleArry on the stack
  idleI                   % leave idleI on the stack
} def                     % bmpI ends here
```

The idlA Procedure The idleI variable is used slightly differently in the UseIdleTime procedure in that it is incremented by 5 each time through the control loop and the index value presented to idlA is used as an offset (values 0 - 4) to idleI.

```
%________________________________________________________________
% calling format
% index idlA idleArry-value
% referenced in UseIdleTime
% found in:  $idleTimeDict
/idlA                   %
{                       %
  idleArry              %
  exch                  % exchange idleArry and index
  idleI                 %
  add                   % add index offset to idleI
  get                   % fetch info from idleArray
                        % and leave on top of stack
} def                   % idlA ends here
```

The stopPred Procedure This procedure is used to terminate the Use-IdleTime procedure. Except during printer start-up, stopPred is defined as the watchstreams procedure. Watchstreams is documented in detail in the chapter on printer communications and is not presented here. The only reference to the name stopPred is in the UseIdleTime procedure.

During start-up, stopPred terminates UseIdleTime if either the printer is warmed up, or a printer time-out occurs. This is explained in the chapter on the start procedure.

Once the printer is warmed up, watchstreams is used to terminate UseIdle-Time by watching for any incoming data on the input channels. Incoming data signals the beginning of a job and the end of idle time.

The ReadIdleFonts Procedure

This procedure is used to retrieve the idle fonts information from the EEPROM and build the idleArry array.

First, the idle fonts information is retrieved from the EEPROM using the idlefonts operator in statusdict. After that, some checks are made to be sure that the idleI index variable is within bounds and is a multiple of 5.

Then the loop is entered in which the idleArry is built. At the top of the loop, terminating conditions are checked, and then each of the five pieces of idle time font information is logged into the idleArry. First, the character string is created, then rotation, x and y scale, and finally the font number is loaded into the idleArry array.

At the end of the procedure, a setidlefonts procedure is defined and placed in the statusdict. The code to define the setidlefonts procedure can be confusing at first glance, since it contains a call to setidlefonts. This is not recursion (setidlefonts is not calling itself). Because the ReadIdleFonts procedures are created using the bind operator, this code is creating a setidlefonts ***procedure*** which calls the setidlefonts ***operator*** in statusdict.

```
%____________________________________________________________
% calling format
% ReadIdleFonts
% referenced in ReadIdleFonts,
% start, setidlefonts
% found in:  userdict
/ReadIdleFonts          %
{                       %
  $idleTimeDict         %
  begin                 %
  idlefonts             % retrieve idlefonts data from
                        % EEPROM,  assumes statusdict is on
                        % the dict stack
  counttomark           % find out how many elements are
                        % in the idlefonts data
  /idleI                % initialize idleI with the number
  exch                  % of elements
  def                   %
                        % next, check validity of idleI
  idleI                 %
  0                     %
  gt                    % idleI > 0 ?
  idleI                 %
  5                     %
  mod                   %
  0                     %
  eq                    % idleI mod 5 = 0 ?
  and                   % and together above booleans
                        % for idleI to be valid, it must be
                        % greater than 0 and it must be
                        % a multiple of 5
  {                     % if idleI valid
    /idleArry           % create the idleArry to be
    idleI               % idleI number of elements
    array               % long
    def                 %
                        %
    {                   % loop to build idleArry
      idleI             % check for loop termination
```

```
0                  %
le                 % idleI <= 0 ?
{                  % if idleI <= 0
  exit             % exit loop if true
} if               % if idleI <= 0
                   %
bmpI               % set eeinfo = nchars
                   %
ascii94            % make sure nchars is not
boundsCheck        % out of bounds
                   %
ascii94            % get substring of characters
0                  % from ascii94 starting at location
eeinfo             % 0 through location nchars
getinterval        %
                   %
put                % put substring in idleArry
                   % (remember bumpI leaves idleArry
                   %  and idleI on the top of the stack
                   %  for this put operation)
                   %
bmpI               % set eeinfo = rotation / 5
                   %
eeinfo             % remember idlefonts requires
                   % rotation / 5
5                  %
mul                % eeinfo X 5
                   %
put                % put rotation into idleArry
                   %
bmpI               % set eeinfo = Sy X 10
                   %
eeinfo             % idlefonts requires Sy X 10
10                 %
div                % eeinfo / 10
                   %
put                % put Sy (scale factor in y)
                   % in idleArry
                   %
bmpI               % set eeinfo = Sx X 10
                   %
eeinfo             % idlefonts requires Sy X 10
10                 %
div                % eeinfo / 10
                   %
put                % put Sx (scale factor in x)
                   % in idleArry
                   %
bmpI               % set eeinfo = font#
                   %
ROMnames           % make sure font# is not out
```

```
          boundsCheck         % of bounds
                              %
          ROMnames            %
          eeinfo              % get font name from
          get                 % ROMnames
                              %
          put                 % put font name in idleArry
                              %
        } loop                % loop to build idleArry
                              %
      } if                    % if idleI valid
                              %
      cleartomark             % clear any invalid idlefonts
                              % information from stack
                              %
      end                     % $idleTimeDict
                              %
      statusdict              % place setidlefonts PROCEDURE
                              % in statusdict
      /setidlefonts           %
      {                       %
        setidlefonts          % enter new idle font info.
                              % into EEPROM using the
                              % setidlefonts OPERATOR
        ReadIdleFonts         % build a new idleArry array from
                              % data just entered into the EEPROM
      }                       %
      put                     % make statusdict entry
                              %
    } def                     % ReadIdleFonts ends here
```

The UseIdleTime Procedure

Next, UseIdleTime is examined to see how characters are actually cached. Once this procedure is studied, the whole notion of idle time font scanning and caching becomes much less mysterious. As it turns out, the font caching is done by a simple call to the stringwidth operator. If you are not familiar with stringwidth, you need to look it up in the *PostScript Language Reference Manual.*

Procedures which call UseIdleTime are those procedures which prepare the interpreter for the next job, plus UseIdleTime is called by the start procedure while the printer is waiting for the engine to warm up. Thus, if the PostScript interpreter is waiting for any reason, it uses that time for caching fonts.

Discussion of Algorithm When the UseIdleTime procedure is called, a procedure (called stopPred by UseIdleTime) is placed on the stack which is used to determine when the idle time is over.

During the start procedure stopPred is a procedure which checks to see if the printer is warmed up. In all other cases, stopPred is equal to the watchstreams procedure. Watchstreams is documented in detail in the printer communications chapter.

Watchstreams (in this implementation) monitors the 9 and 25 pin channels looking for any incoming data. If it detects activity, it returns true, otherwise false is returned.

In order to cache characters, each character from the strings in the idleArry is individually presented to the stringwidth operator. Each time through the loop, stopPred is also checked to determine if idle time is over.

It is also interesting to note that when all the characters have been cached, the idleI index is set back to 0 and the whole process starts over again.

```
%_____________________________________________________________
% calling format
% stopPred UseIdleTime
% referenced in intidleproc,
% start,
% 0, 1 (in specialswitch)
% 0, 1, 3  (in serverdict)
% found in:  userdict
/UseIdleTime            %
{                       %
  $idleTimeDict         % open $idleTimeDict
  begin                 %
                        %
  /stopPred             % define procedure on
  exch                  % top of stack as
  def                   % stopPred
                        %
  /idleI                % initialize idleI
  0                     % (idleI is used to step
  def                   %  through idleArry)
                        %
  /idleStrI             % initialize idleStrI
  0                     % (idleStrI is used to step through
  def                   %  the characters in the strings)
                        %
  /idleStr              % initialize idleStr
  0                     % set idleStr to a 0 length
  string                % string
  def                   %
```

```
                          %
gsave                     % save the current graphics state
                          %
initmatrix                % initialize the CTM to the default
                          %
{                         % loop to cache characters
                          %
  stopPred                % check terminating condition
  {                       % if idle time over
                          %
    exit                  % exit caching loop
                          %
  } if                    % if idle time over
                          %
  idleStrI                % check bounds of character string
  idleStr                 % being cached
  length                  %
  ge                      % idleStrI >= length of idleStr?
  {                       % if idleStrI >= length of idleStr
                          %
                          % this if statement will fetch a new
                          % set of characters to cache from
                          % idleArry
                          %
    idleI                 % check bounds of index into
    idleArry              % the idleArry array
    length                %
    ge                    % idleI >= length of idleArry
    {                     % if idleI >= length of idleArry
                          %
      /idleI              % set idleI back to 0 and
      0                   % start over again
      def                 %
                          %
    } if                  % if idleI >= length of idleArry
                          %
    0                     % 0 is the font name entry
    idlA                  % fetch font name
    FontDirectory         %
    exch                  %
    known                 % is the font name known?
    {                     % if font name valid
      FontDirectory       % prepare to get font name
                          %
      0                   %
      idlA                % get font name from idleArry
                          %
      get                 % get font dictionary
                          %
      setfont             % set the current font
                          %
      initmatrix      % initialize the CTM for this font
                      %
      1               %
      idlA            % get Sx (the x scale value)
                      %
      2               %
      idlA            % get Sy (the y scale value)
                      %
      scale           % Sx Sy scale
                      %
      3               %
      idlA            % get the rotation value, rot
                      %
      rotate          % perform the rotation
                      %
      /idleStr        % prepare to define character string
                      %
      4               %
      idlA            % get string from idleArry
      def             % define string as idleStr
                      %
      /idleStrI       % set the string index
      0               % idleStrI to 0
      def             %
                      %
    } if              % if font name valid
                      %
    /idleI            % increment idleI to point
    idleI             % to the next set of idle time
    5                 % font definitions
    add               %
    def               % idleI = idleI + 5
                      %
  }                   % if idleStrI >= length of idleStr
                      %
  {                   % else if idleStrI  length idleStr
                      %
    idleStr           % prepare to get 1 character
                      % from idleStr
    idleStrI          % location of character to fetch
    1                 % get 1 character
    getinterval       % get character and leave
                      % it on stack
                      %
    stringwidth       % cache the character
                      %
    pop               % discard Wy stringwidth info
    pop               % discard Wx stringwidth info
                      %
    /idleStrI         % increment idleStr index by 1 to
    idleStrI          % point to the next character
```

```
        1                   % to be cached
        add                 %
        def                 %
                            %
      } ifelse              % else if idleStrI  length idleStr
                            %
    } loop                  % loop to cache characters
                            %
    grestore                % restore graphics state
                            %
    end                     % end $idleTimeDict
                            %
} def                       % UseIdleTime ends here
```

Chapter 9

Paper Sizes and $printerdict

Introduction

The $printerdict contains entries which define parameters for the current paper size. In the case of this printer the page parameters for letter, legal, a4 and b5 are defined.

Papersize Procedures

The procedures 0, 2, 8, 18, and 24 in $printerdict have exact duplicates in the userdict as follows:

userdict	$printerdict
letter	0
letter	8
a4	2
b5	18
legal	24

Also, each of these procedures are identical except for the information pushed onto the stack at the beginning of the procedures.

The use of numbers (0, 2, 8, 18, 24) for dictionary key entries seems to indicate that this is a return code from the Canon engine for the paper tray currently inserted. This technique of using integer keys is also used for the hardware thumbwheel switch for routines in the serverdict. Using an integer key simply saves having to look up the value in a calling table.

Other procedures in the $printerdict are dopage, proc, and setpage. Each of these procedures are subsets of the page set-up functions described, plus dopage is a subset of proc. This relationship is flagged in the documentation of the 0 procedure below. Since all the other procedures are so similar, only 0 is documented in detail.

From the cross reference table in Appendix III we find that none of these procedures are directly referenced by any other procedures. However, they are indirectly called from the setrealdevice procedure which is executed before the beginning of each job from the thumbwheel procedures and execjob.

Calculation of the CTM

Calculations from the page size procedures cause the Current Transformation Matrix (CTM) for each page size to be calculated and stored in the mtx array in $printerdict. No PostScript matrix operations change the contents of mtx and the initmatrix command causes the contents of mtx to be the new CTM. Calculations in the page setup routines for the CTM cause the matrix to be created as follows:

```
[300/72 0 0 -300/72 -xoffset height+yoffset]
```

This matrix is responsible for shifting the 0,0 origin from the point where the hardware ships out pixel 0,0 to the print engine to the user origin 0,0 which is at the bottom left of the printed page. In the case of this printer, pixel 0,0 is at the top left of the page.

What this means is that the CTM as set up in the mtx array takes into account the device dependencies at the lowest level so the user need not be concerned with the number of dots per inch of the printer or any offsets of the imageable area.

On this particular printer, the resolution is 300 dots per inch, and the scale factors are calculated in terms of points or 300 dots per inch divided by 72 points per inch. Also notice that the y scale value is negative which means that the positive y direction is from the bottom of the page moving toward the top. The positive x direction is from left to right.

The offsets take into account the device dependent positioning of each paper size. Since PostScript defines location 0,0 at the bottom left of the page, it is necessary to translate the imageable region -xoffset pixels in the x direction and height+yoffset pixels in the y direction.

Papersize Procedures

Below are listings of the four procedures (using immediately evaluated expressions) which list the four values (width, height, xoffset and yoffset) used in the calculation of the CTM for each specific paper size.

Later, the fully expanded letter (or 0) procedure will be documented.

Letter Size Page Computed matrix = [4.1667 0 0 -4.1667 -75 3288]

```
%______________________________________________________________
% calling format
% 0 or 8
% found in:  $printerdict
                              %
8                             %
{                             %
                              % setpage parameters
  300                         % width
  3276                        % height
  75                          % xoffset
  12                          % yoffset
                              %
                              % proc procedure starts here
                              % proc is also placed on stack as a
                              % parameter for setpage
  {                           %
    88                        % x margin offset
    148                       % y margin offset
    //dopage                  % immediately evaluate dopage
    exec                      % execute dopage procedure
  }                           %
  //setpage                   % immediately evaluate proc
  exec                        % execute proc procedure
} bind def                    % 0 or 8 ends here
```

Legal Size Page Computed matrix = [4.16667 0 0 -4.16667 -75 4125]

```
%______________________________________________________________
% calling format
% 24
% found in:  $printerdict
                              %
24                            %
{                             %
                              % setpage parameters
  300                         % width
  4050                        % height
  75                          % xoffset
  75                          % yoffset
                              %
                              % proc procedure starts here
                              % proc is also placed on stack as a
                              % parameter for setpage
  {                           %
    151                       % x margin offset
    148                       % y margin offset
    //dopage            % immediately evaluate dopage
    exec                % execute dopage procedure
  }                     %
  //setpage             % immediately evaluate proc
  exec                  % execute proc procedure
} bind def              % 24 ends here
```

a4 Size Page Computed matrix = [4.16667 0 0 -4.16667 -40 3393]

```
%________________________________________________________________
% calling format
% 2
% found in:  $printerdict
                        %
2                       %
{                       %
                        % setpage parameters
  300                   % width
  3365                  % height
  40                    % xoffset
  28                    % yoffset
                        %
                        % proc procedure starts here
                        % proc is also placed on stack as a
                        % parameter for setpage
  {                     %
    104                 % x margin offset
    224                 % y margin offset
    //dopage            % immediately evaluate dopage
    exec                % execute dopage procedure
  }                     %
  //setpage             % immediately evaluate proc
  exec                  % execute proc procedure
} bind def              % 2 ends here
```

b5 Size Page Computed matrix = [4.16667 0 0 -4.16667 -11 2940]

```
%________________________________________________________________
% calling format
% 18
% found in:  $printerdict
                        %
18                      %
{                       %
                        % setpage parameters
  258                   % width
  2928                  % height
  11                    % xoffset
  12                    % yoffset
                        %
                        % proc procedure starts here
                        % proc is also placed on stack as a
                        % parameter for setpage
  {                     %
    88                  % x margin offset
    512                 % y margin offset
    //dopage            % immediately evaluate dopage
    exec                % execute dopage procedure
  }                     %
  //setpage             % immediately evaluate proc
  exec                  % execute proc procedure
} bind def              % 18 ends here
```

The Call to redwrite

The proc procedure, which is executed as a part of showpage, sets up a call to an undocumented command redwrite which performs the actual printing of the page. Redwrite probably stands for Redstone write, since this printer hardware was known as Redstone.

The call to redwrite is set up as follows:

```
xm+88 (ym+148)/2 mf #copies redwrite
```

xm and ym are the margin values returned by the margins command and mf is manualfeedtimeout or 0 depending if the manualfeed boolean is true or false respectively.

Redwrite takes into account further device dependencies of the page positioning for all page size types, and allows the margin adjustments provided by the setmargins command to be added in. From the setmargins command we find that the xm+88 calculation is an offset from the left of the page, and (ym+148)/2 calculation is an offset from the top of the page.

The letter (or 0) Procedure

The letter (or 0) procedure makes interesting use of the stack in its execution. First, the numbers width, height, xoffset and yoffset are placed on the stack, along with the procedure proc. Then another lengthy procedure, setpage, is placed on the stack. At the very end setpage is executed with an exec operator. In the process of its execution, setpage consumes the first five items on the stack and places them as key-value pairs in $printerdict. Finally, setpage defines setscreen and settransfer. A detailed technical discussion of the setscreen operator, settransfer operator, spot functions, screen angles and frequencies is beyond the scope of this book. The best discussion of these issues is contained in a book entitled "Real World PostScript". The *PostScript Language Reference Manual* and *PostScript Language Tutorial and Cookbook* also contain references to these operators and concepts.

Setpage then goes on to set up a call to framedevice. This installs the frame buffer in raster memory for the particular page size. In the set up for framedevice, the current transformation matrix (CTM) is calculated. The call to framedevice also establishes proc as the procedure to be executed as part of showpage or copypage (see framedevice discussion in the *PostScript Language Reference Manual*).

```
%___________________________________________________________
% calling format
% 0
% found in:  $printerdict
                        %
0                       %
{                       %
                        % setpage parameters
  300                   % width
  3276                  % height
  75                    % xoffset
  12                    % yoffset
                        %
                        % proc procedure starts here
                        % proc is also placed on stack as a
                        % parameter for setpage
  {                     %
    88                  % x margin offset
    148                 % y margin offset
                        %
                        % dopage procedure starts here
                        %
    {                   % place dopage proc on stack for
                        % execution
      //statusdict      % statusdict
      begin             %
```

```
                  %
/jobstate         % set jobstate to printing
                  % string length =  8
(printing)
                  %
def               %
                  %
0                 % 0 turns off blinking during
setblink          % printing
                  %
                  % begin set up for call to
                  % redwrite
                  %
margins           % get the margin values
exch              % begin the computations
4                 % as described above
-1                %
roll              %
add               %
3                 %
1                 %
roll              %
add               %
2                 %
div               %
round             %
cvi               %
                  % proper x and y offsets
                  % are now on the stack
                  %
manualfeed        % get manualfeed flag
{                 % if manualfeed true
                  % put this on stack
  manualfeedtimeout
}                 %
{                 % else if manualfeed false
  0               % put 0 on stack
} ifelse          %
                  %
#copies           % put #copies on stack
                  %
redwrite          % print the page
                  %
/jobstate         % set the jobstate to busy
                  % string length =  4
(busy)
                  %
def               %
                  %
1                 % set the blinking to
setblink          % single blinks
```

```
                          %
      end                 % close statusdict
    }                     %
                          %
                          %  dopage procedure ends here
                          %
    exec                  % execute dopage
  }                       %
                          %
                          % proc procedure ends here
                          %
                          % setpage procedure starts here
                          % setpage is placed on the stack
                          % for exec operation
  {                       %
    //$printerdict        % $printerdict
    begin                 %
                          %
    /proc                 % get proc from stack and
    exch                  % enter in $printerdict
    def                   %
                          %
    /yoffset              % get yoffset from stack and
    exch                  % enter in $printerdict
    def                   %
                          %
    /xoffset              % get xoffset from stack and
    exch                  % enter in $printerdict
    def                   %
                          %
    /height               % get height from stack and
    exch                  % enter in $printerdict
    def                   %
                          %
    /width                % get width from stack and
    exch                  % enter in $printerdict
    def                   %
                          %
                          % begin setup for framedevice
    300                   % calculate CTM
    72                    %
    div                   %
    0                     %
    0                     %
    -300                  %
    72                    %
    div                   %
    xoffset               %
    neg                   %
    height                %
    yoffset               %
```

```
add                 %
mtx                 %
astore              % put CTM on stack
                    %
width               % put width on stack
                    %
height              % put height on stack
                    %
/proc               % put proc on stack
load                %
                    %
framedevice         %
                    % begin setup for setscreen
60                  % 60 lines/inch frequency
45                  % 45 degree screen angle
                    %
{                   % place spot function on stack
  abs               % see "Real World PostScript"
  exch              % page 170 for discussion of how
  abs               % the spot function works
  2                 %
  copy              %
  add               %
  1                 %
  gt                %
  {                 %
    1               %
    sub             %
    dup             %
    mul             %
    exch            %
    1               %
    sub             %
    dup             %
    mul             %
    add             %
    1               %
    sub             %
  }                 %
  {                 %
    dup             %
    mul             %
    exch            %
    dup             %
    mul             %
    add             %
    1               %
    exch            %
    sub             %
  } ifelse          %
}                   %
```

```
        setscreen          %
                           %
        {                  % place transfer function on stack
        }                  %
        settransfer        %
                           %
        initgraphics       % reset graphics state values
        erasepage          % blank the raster memory
                           %
        end                % close $printerdict
      }                    %
                           %
                           % setpage procedure ends here
                           %
    exec                   % execute the above procedure
                           %
  } bind def               % 0 ends here
```

Chapter 10

Print Engine Status

Introduction

Four procedures (printerstatus, warmedup, initprinter, and setrealdevice) deal directly with the print engine. Their purpose is to detect and report conditions of the print engine itself.

Chapter Overview

The printerstatus procedure is discussed first since it is actually a part of the other three procedures and a result of the "//printerstatus" substitution. The warmedup and initprinter procedures are discussed next since they are both used by the start procedure during the PostScript start-up sequence. Finally, we will see how setrealdevice is used to fetch the paper size from the engine and execute the proper procedure in $printerdict.

Welcome to the world of the print engine!

The printerstatus Procedure

The printerstatus procedure is never referenced directly as a procedure call, but rather is imbedded in the warmedup, initprinter and setrealdevice procedures as an immediately evaluated expression in the form "//printerstatus".

It is important to note that there are printerstatus keys in both statusdict and serverdict. In statusdict, printerstatus is an **operator** which returns the status of the Canon print engine. The printerstatus **procedure** documented below is the one found in serverdict. Within this printerstatus procedure, the printerstatus operator is used, not to be confused with another (recursive) call to the printerstatus procedure in serverdict.

The printerstatus procedure first fetches the status of the print engine and checks to see if it is valid. If a failure is detected, a recovery is attempted. First, a delay of 6 seconds is executed after which two attempts are made to reset the print engine with the resetprinter operator. If the failure is still detected a message is sent back over the reverse channel.

```
%____________________________________________________________
% calling format
% printerstatus status-boolean
% unreferenced
% found in:  serverdict
/printerstatus          %
{                       %
  printerstatus         % fetch the engine status using the
                        % printerstatus operator in
                        % statusdict
  dup                   % save a copy for the calling
                        % procedure
```

```
                      %
-1                    % -1 indicates no engine response
eq                    % printerstatus = -1
{                     % if no engine response
                      %
  pop                 % discard -1 on top of stack
                      %
                      % this next piece of code performs
                      % a 6 second delay
                      %
  usertime            % get the current time
                      % (in milliseconds)
  6000                % 6000 milliseconds = 6 seconds
  add                 % add 6 seconds to the current time
                      % for the ending time
                      %
  {                   % begin delay loop
                      %
    dup               % dup the ending time
    usertime          % fetch the current time
    le                % ending time le current time
    {                 % if time to exit loop
                      %
      pop             % dump ending time from top of stack
      exit            % exit the loop
                      %
    } if              % if time to exit loop
                      %
  } loop              % end delay loop
                      %
                      % this next code attempts to
                      % reset the print engine twice
                      %
  2                   %
  {                   % begin repeat loop
                      %
    resetprinter      % resetprinter returns true if reset
                      % is successful
                      %
    {                 % if reset successful
                      %
      exit            % exit loop is reset successful
                      %
    } if              % if reset successful
                      %
  } repeat            % end repeat loop
                      %
                      % the engine status is requested one
                      % more time, and if a failure is
                      % still detected a message is
                      % returned over the reverse channel
```

```
                             %
   printerstatus             % get engine status one more time
   dup                       % save a copy for the calling
                             % procedure
                             %
   -1                        % -1 indicates no engine response
   eq                        % printerstatus = -1
                             %
   {                         % if no engine response after delay
                             % and reset
                             %
                             % string length =  47
     (%%[ PrinterError: controller not responding ]%%)
                             %
                             %
     =                       % send message back over the
                             % reverse channel
     flush                   % flush the output buffer
                             %
   } if                      % if no engine response after delay
                             % and reset
                             %
  } if                       % if no engine response
                             %
} def                        % printerstatus ends here
```

The warmedup Procedure

The warmedup procedure tells the calling procedure if the print engine is warmed up and ready to print. The boolean returned by warmedup is the result of a series of computations based on the print engine status.

```
%_______________________________________________________________
% calling format
% warmedup ready-boolean
% referenced in start
% found in:  serverdict
/warmedup                    %
{                            %
  //printerstatus            % immediately evaluate the
                             % printerstatus procedure
                             %
                             % this effectively places the
                             % the printerstatus procedure
                             % on the stack
                             %
  exec                       % execute printerstatus
                             %
  dup                        % a copy of the Canon print engine
                             % status is used later
                             %
  0                     %
  lt                    % printerstatus lt 0
  exch                  % place printerstatus back on
                        % top of stack
                        %
  134217728             % magic number = 8000000 Hex
                        %
  and                   % printerstatus and 134217728
  0                     %
  eq                    % (printerstatus and 134217728) eq 0
                        %
                        % the entire expression is below
                        %
  or                    % [(printerstatus and 134217728)
                        % eq 0] or [printerstatus lt 0]
                        %
                        % leave result on stack
                        %
} def                   % warmedup ends here
```

The initprinter Procedure

The initprinter procedure is the first attempt that the start procedure (the start procedure is discussed in another chapter) makes at communicating with the print engine during the power-up sequence.

The procedure consists of a "for" loop inside of which communication with the printer is attempted. If the printer does not respond, the printerstatus procedure is called as a last resort.

```
%________________________________________________________________
% calling format
% initprinter
% referenced in start
% found in:  serverdict
/initprinter            %
{                       %
                        % set up for loop
                        %
  1                     % beginning index
  1                     % increment value
  4                     % ending value
                        %
  {                     % begin for loop
                        %
                        % the ifelse statement is set up so
                        % that printerstatus is only called
                        % if the "for" loop reaches 4
                        %
    4                   %
    ne                  % index ne 4
```

```
                                %
      {                         % if not last index value
                                %
                                % in this next piece of code, a
                                % command is sent to the engine
                                % requesting information
                                %
                                % if the engine responds with any
                                % value greater than or equal to 0
                                % then the engine is alive and well
                                %
        64                      % 64 = 40 Hex
        sendpcmd                % 64 sendpcmd
                                % (request engine information)
        0                       %
        ge                      % engine-info ge 0
                                %
        {                       % if engine responding
                                %
          exit                  % exit for loop if engine responds
                                %
        } if                    % if engine responding
                                %
      }                         % if not last index value
                                %
      {                         % else if last index value
                                %
        //printerstatus         % immediately evaluate the
                                % printerstatus procedure
                                %
                                % this effectively places the
                                % the printerstatus procedure
                                % on the stack
                                %
        exec                    % execute printerstatus
                                %
        pop                     % discard status value
                                %
      } ifelse                  % else if last index value
                                %
    } for                       % end for loop
                                %
  } def                         % initprinter ends here
```

The setrealdevice Procedure

This procedure is used to call the proper page size procedure (0, 2, 8, 18, 24) in $printerdict based on the tray size information returned by the print engine. Refer back to Chapter 9 on the $printerdict for detailed information on the function of the page size procedures. The calls to setrealdevice are documented in Chapter 12 on the thumbwheel switch.

```
%____________________________________________________________
% calling format
% setrealdevice
% referenced in start, intidleproc,
% 0,1,3 (in serverdict)
% 0,1 (in specialswitch)
% found in:  serverdict
/setrealdevice          %
{                       %
  //printerstatus       % immediately evaluate the
                        % printerstatus procedure
                        %
                        % this effectively places the
                        % the printerstatus procedure
                        % on the stack
                        %
  exec                  % execute printerstatus
                        %
  126                   % magic number = 7E Hex
                        %
  and                   % printerstatus and 126
                        %
  dup                   % save an extra copy
                        %
                        % in this next section, a check
                        % is made to see if the result of
                        % printerstatus and 126
                        % results in an integer which
                        % matches one of the integer keys
                        % in $printerdict
                        %
                        % if no match is found,
                        % 8 (letter tray)
                        % is substituted for the bad value
                        %
  //$printerdict        % $printerdict
  exch                  % set up stack for known command
                        %
  known                 % $printerdict
                        % [printerstatus and 126] known
                        %
  not                   % not known
                        %
  {                     % if returned information not valid
```

```
                                    %
        pop                         % pop invalid information
                                    %
        8                           % substitute letter tray information
                                    %
      } if                          % if returned information not valid
                                    %
      //$printerdict                % $printerdict
      exch                          % set up for get command
                                    %
                                    % get the procedure from
                                    % $printerdict using the integer key
                                    %
      get                           % $printerdict tray-info get
                                    %
      exec                          % execute the $printerdict
                                    % procedures (0, 2, 8, 18, 24)
                                    %
    } def                           % setrealdevice ends here
```

Chapter 11

Print Engine Error Reporting and mydict

Introduction

The job of reporting print engine problems and malfunctions is done with the procedures, variables and strings found in the mydict dictionary. Mydict is one of the two self-contained dictionaries (==dict is discussed in another chapter) in this implementation of PostScript. This means that mydict is only defined within itself, and it is only referenced once as an immediately evaluated name, in the printererror procedure which is explained in detail below.

Among other things, a study of the mydict procedures reveals interesting characteristics and usages of PostScript strings.

Chapter Overview

Control variables are explained first, followed by error message strings and procedures. The error printing procedures eprint and eflush are discussed next and finally, the printererror procedure is documented.

Control Variables

These variables are used by the procedures in mydict.

The abort Boolean The abort boolean is only set true if a manual feed time out occurs in procedure 528.

```
%______________________________________________________
% abort is found in mydict
% type = booleantype
% referenced in printererror,
% 528 (in mydict)
/abort false def
```

The ntrys Integer The ntrys integer is the number of times the PostScript controller has called printererror during the current showpage or copypage.

```
%______________________________________________________
% ntrys is found in mydict
% type = integertype
% referenced in printererror
/ntrys 0 def
```

The report Boolean The report Boolean is used during a call to printererror to determine if the "cover open" message has been printed yet.

```
%_____________________________________________________________
% report is found in mydict
% type = booleantype
% referenced in printererror
/report true def
```

The bits Integer The bits integer is defined in printererror as the result of a calculation used to determine which error has occurred.

```
%_____________________________________________________________
% bits is found in mydict
% type = integertype
% referenced in printererror
/bits 0 def
```

The stat and laststat Integers This pair of integers contain the current error code (stat) and the previous error code (laststat).

The printer error procedures use stat and laststat to see if there was any change in the error status.

The stat integer This integer contains the current error code of the print engine.

```
%_____________________________________________________________
% stat is found in mydict
% type = integertype
% referenced in printererror,
% 528, 4616 (in mydict)
/stat 0 def
```

The laststat Integer This integer contains the previous error code.

```
%_____________________________________________________________
% laststat is found in mydict
% type = integertype
% referenced in printererror
/laststat 0 def
```

The eindex and errstr Variables The eindex integer is used to point to a location in the 60 character errstr string. These two objects are used to build error messages which in some cases require several strings to be joined. This is explained in the discussion of eprint.

```
%_____________________________________________________________
% eindex is found in mydict
% type = integertype
% referenced in printererror,
% eprint, eflush
/eindex 0 def
```

```
%_____________________________________________________________
% errstr is found in mydict
% not executable; length = 60
% referenced in eprint, eflush
/errstr                 %
(.........1.........2.........3.........4.........\
5.........6)
def                           % errstr ends here
```

Error Messages and Print Engine Error Codes

The mydict dictionary contains eight entries with integer keys. These integer keys point to either procedures or strings, and are indirectly referenced and used in the printererror procedure. The integer keys correspond to a variety of print engine error codes and their purpose is to leave a string on the stack which describes the error.

String 0 This string contains the "timeout, clearing printer" message.

```
%_____________________________________________________________
% 0 (0 Hex) is found in mydict
% not executable; length = 25
0
(timeout, clearing printer)
def                           % 0 ends here
```

String 512 This string contains the "service call" message.

```
%_____________________________________________________________
% 512 (200 Hex) is found in mydict
% not executable; length = 12
512
(service call)
def                           % 512 ends here
```

String 520 This string contains the "paper entry misfeed" message.

```
%_____________________________________________________________
% 520 (208 Hex) is found in mydict
% not executable; length = 19
520
(paper entry misfeed)
def                       % 520 ends here
```

Procedure 528 This procedure can indicate one of three different printer conditions, "manual feed timeout", "no paper tray", or "out of paper".

```
%_____________________________________________________________
% calling format
% 528 message
% found in:  mydict
528                       %(210 Hex)
{                         %
                          %
  //statusdict            % statusdict
  /manualfeed             %
  get                     % fetch manualfeed boolean
                          %
  {                       % if manualfeed true
                          % string length =  19
    (manual feed timeout)
                          %
                          % leave message on stack for eprint
                          %
    /abort                % the abort flag is only used
    true                  % in this one case
    def                   %
                          %
  }                       % if manualfeed true
                          %
  {                       % else if manualfeed false
                          %
                          % check certain bits to distinguish
                          % between "no paper tray" or
                          % "out of paper"
    stat                  %
    126                   % magic number = 7E Hex
    and                   % stat & 126
    0                     %
    eq                    % 0 = (stat & 126) ?
                          %
    {                     % if 0 = (stat & 126)
                          % string length =  13
      (no paper tray)
```

```
                                %
                                % leave message on stack for eprint
    }                           % if 0 = (stat & 126)
                                %
    {                           % else if 0 ne (stat & 126)
                                % string length =  12
      (out of paper)
                                %
                                % leave message on stack for eprint
                                %
    } ifelse                    % else if 0 ne (stat & 126)
                                %
  } ifelse                      % else if manualfeed false
                                %
} def                           % 528 ends here
```

String 536 This string contains the "paper entry misfeed" message.

```
%___________________________________________________________
% 536 (218 Hex) is found in mydict
% not executable; length = 19
536
(paper entry misfeed)
def                         % 536 ends here
```

String 576 This string contains the "no toner cartridge" message.

```
%___________________________________________________________
% 576 (240 Hex) is found in mydict
% not executable; length = 18
576
(no toner cartridge)
def                         % 576 ends here
```

String 2048 This string contains the "warming up" message.

```
%___________________________________________________________
% 2048 (800 Hex) is found in mydict
% not executable; length = 10
2048
(warming up)
def                         % 2048 ends here
```

Procedure 4616 This procedure takes care of paper jams which occur during the paper's exit from the printer. In this case pages can be lost since the page has already been completely printed and cannot be recovered by PostScript.

```
%_______________________________________________________________
% calling format
% 4616 message
% found in:  mydict
4616                    % (1208 Hex)
{                       %
                        % string length =  20
  (paper exit misfeed, )
                        %
                        % leave message on stack for eprint
  eprint                % add string to message string
                        %
                        % next determine how many pages
                        % have been lost
                        %
  stat                  % current printer status
                        %
  32256                 % magic number = 7E00 Hex
  and                   % stat & 32256
  -9                    %
                        % shift right by 9 bits
  bitshift              % (stat & 32256) > 9
                        % string length =  128
  (junk string used for conversion)
                        %
  cvs                   % convert number of pages lost
                        % to string
                        %
  eprint                % add string to message string
                        % string length =  18
  ( pages may be lost)
                        %
                        % leave last part of message
                        % on stack
                        % for eprint
} def                   % 4616 ends here
```

Error Message Reporting

Error messages issued by printererror are built from several strings which are joined together. When the message is completely built, it is printed on the reverse channel, and assigned to the jobstate variable in statusdict.

The eprint Procedure The job of joining the messages is done by eprint. In order to do this, advantage is taken of the fact that substrings are still part of the parent string. The parent string is errstr, and eindex is used to point to the last character in the error message contained in errstr. The job of copying the message into the parent string is done by the cvs operator.

To fully understand how this works try portions of the following code sequence in interactive mode.

```
%______________________________________________________________
% calling format
% message eprint
% referenced in printererror,
% 4616 (in mydict)
% found in:  mydict
/eprint                 %
{                       %
                        % getinterval is used to
                        % fetch the tail end of the parent
                        % string - this is where the message
                        % is placed
                        %
  errstr                % place parent string on stack
                        % in preparation for getinterval
                        %
  eindex                % place the beginning index of the
                        % parent string on the stack
                        % in preparation for getinterval
                        %
                        % next figure out number of bytes
                        % to fetch from the parent string
                        %
  errstr                %
  length                % get the total length of errstr
  eindex                %
  sub                   % length errstr - eindex
                        % = bytes left
                        % at the tail of errstr for message
                        %
  getinterval           % fetch the tail of errstr not yet
                        % filled with a message and leave it
                        % on the stack
                        %
  cvs                   % cvs takes message and places it
                        % in the tail of errstr
                        %
                        % the result string after cvs
                        % will contain only the message
                        %
                        % this last section updates
```

```
                         % eindex to point to the new end
                         % of the error message since
                         % appending this portion of the
                         % message to errstr
                         %
  length                 % get length of the message
                         % (even though length consumed the
                         %  substring on the stack, the
                         %  message still remains in errstr)
                         %
  eindex                 %
  add                    % length message + eindex
  /eindex                % prepare to redefine eindex
  exch                   %
  def                    % create the new eindex
                         %
} def                    % eprint ends here
```

The eflush Procedure The eflush procedure is responsible for printing the message on the reverse channel and assigning the message to the jobstate variable in statusdict. If a status inquiry is made at a later time by the host computer, then jobstate is used to send the error message back on the reverse channel.

```
%____________________________________________________________
% calling format
% eflush
% referenced in printererror
% found in:  mydict
/eflush                  %
{                        %
  //statusdict           % statusdict
                         % place dict on stack for put
                         %
  /jobstate              % place key on stack for put
                         %
                         % the next few lines fetch the
                         % error message as a substring
                         % of errstr
                         %
  errstr                 % parent string
  0                      % beginning index
  eindex                 % number of characters to fetch
  getinterval            % fetch substring
                         %
  dup                    % save a copy for put
                         %
                         % string length =  4
  (%%[ )
```

```
                                 %
  print                          % print start of error message "%%["
                                 %
  print                          % print message
                                 %
                                 % string length =  5
  ( ]%%\n)
                                 %
  print                          % print end of message "]%%"
  flush                          % flush output buffer
                                 %
  put                            % assign message to jobstate
                                 % in statusdict
                                 %
} def                            % eflush ends here
```

The printererror Procedure This procedure is used to report print engine errors to the user. According to the *PostScript Language Reference Manual,* it is called during the execution of showpage or copypage. This explains why it is unreferenced in any of the accessible procedures in the PostScript interpreter, except for the start procedure. In this instance printererror is defined to be equal to the stop operator in the statusdict, but after the start-up sequence, we find printererror to be defined as listed below.

The printererror procedure starts by defining key control variables. Next, the current printer status is checked. If the status is -1, ten attempts are made to reset the printer. A -1 status means the printer is not responding, possibly because the printer cover is open.

If the status is not -1, the procedure tries to determine which error occurred. A check is also made to be sure that ntrys is not 0 and that the same error message is not issued twice in a row.

At the end of printererror, the abort boolean is checked to see if a manual-feed timeout has occurred. If the timeout has occurred, an interrupt (or stop) is issued and the job is aborted.

```
%_______________________________________________________________
% calling format
% status tries printererror
% referenced in start
% found in:  statusdict
                                 %
/printererror                    %
{                                %
  //mydict                       % mydict
```

```
                      % this is the only reference
                      % to mydict
begin                 % open mydict
                      %
enableinterrupt       % turn on interrupts
                      %
/ntrys                %
exch                  %
def                   % define number of tryies
                      %
/stat                 %
exch                  %
def                   % define printer status
                      %
/eindex               %
0                     %
def                   % set index to errstr to 0
                      %
/abort                %
false                 %
def                   % only true if manualfeed timeout
                      % occurs
                      %
stat                  %
-1                    %
eq                    % stat = -1 ?
                      % (no print engine response)
                      %
{                     % if stat = -1
                      % string length =  31
  (PrinterError: resetting printer)
                      %
                      %
  eprint              % add message to errstr
  eflush              % send message
                      %
  /report             % report is used to
  true                % control printing of the
  def                 % "cover open" message
                      %
  10                  % loop count (try 10 times)
  {                   % repeat loop
                      %
    resetprinter      % try to reset print engine
                      %
    {                 % if reset successful
                      %
      exit            % exit repeat loop
                      %
    } if              % if reset successful
                      %
    report             % report "cover open"?
                       %
    {                  % if reporting "cover open"
                       %
      /eindex          %
      0                %
      def              % set errstr index to 0
                       %
                       % string length =  24
      (PrinterError: cover open)
                       %
                       %
      eprint           % add message to errstr
      eflush           % print "cover open" message
                       %
      /report          % set report to false to
      false            % avoid printing the
      def              % "cover open" message again
                       %
    } if               % if reporting "cover open"
                       %
  } repeat             % repeat 10 times
                       %
}                      % if stat = -1
                       %
{                      % else if stat ne -1
                       % (print engine responding)
                       %
  ntrys                %
  0                    %
  eq                   % ntrys = 0 ?
                       %
  stat                 %
  laststat             %
  ne                   % stat ne laststat ?
                       %
  or                   % (ntrys = 0) or (stat ne laststat)
                       % is this the first try, or has the
                       % print engine status changed since
                       % the last try?
                       %
  {                    % if (ntrys = 0)or(stat ne laststat)
                       %
                       % string length =  14
    (PrinterError: )
                       %
                       %
    eprint             % add "PrinterError:" to errstr
                       %
                       % compute the error number and
                       % place the result in /bits
```

```
                          %
     /bits                % prepare for def
     stat                 % get raw engine status
     -16                  %
     bitshift             % shift right by 16 bits
     6744                 % 6744 = 1A58 Hex
     and                  % and with 6744
                          %
                          % (stat > 16) & 6744
     def                  % define /bits
                          %
     currentdict          % mydict
                          %
     bits                 %
     known                % is /bits one of the integer keys
                          % in mydict?
                          %
     {                    % if bits known
                          %
       bits               % place integer key on stack
       load               % load procedure
       exec               % execute procedure
                          % (a string is left on the
                          %  stack containing the
                          %  error message)
                          %
     }                    % if bits known
                          %
     {                    % else if bits unknown
                          %
                          % in this case convert raw engine
                          % error code to a hex string and
                          % leave the string on the stack
                          %
       stat               % raw engine code
       16                 % radix 16 (for cvrs)
                          % string length =  128
       (junk string for cvrs conversion)
                          %
                          %
       cvrs               % convert number to string
                          % (radix 16)
                          %
     } ifelse             % else if bits unknown
                          %
     eprint               % add message on stack to errstr
     eflush               % print complete errstr message
                          %
   } if                   % if (ntrys = 0)or(stat ne laststat)
                          %
 } ifelse                 % else if stat ne -1
```

```
                                        %
   /laststat                            %
   stat                                 %
   def                                  % define laststat = stat
                                        %
   abort                                % place abort flag on stack
                                        %
   end                                  % end mydict
                                        %
   {                                    % if abort true
                                        %
     1                                  %
     setblink                           % set LED to single blinks
                                        %
     interrupt                          % interrupt = stop
                                        %
   } if                                 % if abort true
                                        %
 } def                                  % printererror ends here
```

Chapter 12

The Thumbwheel Switch and PostScript Operating Modes

Introduction

Procedures for looking at the settings of the thumbwheel switch, the test page and job execution control procedures are found in serverdict. Only three of the procedures in serverdict are locked (exitserver, server, and protect) while the rest are open for study. Much of what will be found are the procedures which control the allocation of PostScript file resources.

The serverdict is undocumented by Adobe except for one command: exit-server. Those who have even casually browsed through this dictionary know that in the execution of PostScript, this dictionary is where the action is.

This chapter focuses on procedures related to the settings of the thumbwheel switch which selects different operating modes.

NOTE: On other printers, DIP switches or front panels are available to change operating modes.

The Thumbwheel Switch

The procedures 0, 1, 2 and 3 in serverdict are executed based on the setting of the thumbwheel switch. These procedures are never accessed directly by name, but instead are called indirectly from execjob based on the setting of the variable saveswitch (see execjob discussion in the Job Execution chapter). Adobe uses integer dictionary keys so calls to these procedures can be made simply by using the switch reading.

The four switch settings represent the following modes:

Setting	Mode	25 and 9 Pin Channel
0	Serial Batch Mode	1200 Baud
1	Serial Batch Mode	User Defined Default = 9600 Baud
2	Diablo Emulation (Special Switch = 0)	User Defined Default = 9600 Baud
2	Executive Mode (Special Switch = 1)	User Defined Default = 9600 Baud
3	AppleTalk	25 pin = Unused 9 pin = AppleTalk

Examination of the four procedures shows that 0, 1, and 3 are identical, so only procedures 0 and 2 are documented below.

These procedures are important because it is here that the user program is executed. This ties in with the stopped sequences in execjob, and shows how PostScript handles error recovery. Going a little deeper in the Job Execution and Start Procedure chapters a better understanding of how the basic server loop works is gained.

Chapter Overview

There are several utility procedures used in the main control procedures. The procedures 0, 1, and 3 from the serverdict, which are related to the respective thumbwheel settings are documented. Procedure 2 from serverdict is then covered including its relationship to the special switch. Finally, the special switch procedures 0 and 1 are documented.

Utility Procedures

The utility procedures exchdef, settimeouts and fontname are documented below.

exchdef Procedure The exchdef procedure is a handy procedure which combines the common "exch def" sequence.

```
%________________________________________________________________
% calling format
% value key exchdef
% calling format
% found in:  serverdict
                        %
/exchdef                %
{                       %
  exch                  %
  def                   %
} def                   % exchdef ends here
```

settimeouts Procedure Note that the settimeouts procedure is used in several procedures below. Its purpose is to set the job, manualfeed and wait timeouts for the job which is about to be executed.

```
%________________________________________________________________
% calling format
% job manualfeed wait settimeouts
% found in:  serverdict
                        %
/settimeouts            %
{                       %
```

```
  //statusdict          % statusdict
  begin                 % begin statusdict
                        %
  /waittimeout          % take the wait timeout value from
  exchdef               % the stack and define it
                        %
  /manualfeedtimeout    % take the manual feed timeout from
  exchdef               % the stack and define it
                        %
  setjobtimeout         % take job timeout from stack and
                        % define it
                        %
  end                   % end statusdict
} def                   % settimeouts ends here
```

fontname Procedure The fontname procedure is used to fetch the name of a font from the ROMnames array in $idleTimeDict based on an integer index value.

```
%____________________________________________________________
% calling format
% index fontname name
% found in:  serverdict
                        %
/fontname               %
{                       %
  $idleTimeDict         %
  begin                 % begin $idleTimeDict
                        %
                        % check to be sure that the index
                        % is not out of bounds
                        %
  dup                   % duplicate index for later use
                        %
  ROMnames              %
  length                % get the length of ROMnames
                        %
  ge                    % index = length of ROMnames
                        %
                        % if the index is out of bounds
                        % replace it with 0
                        %
  {                     % if index = length
                        %
    pop                 % discard bad index
    0                   % replace it with 0
                        %
  } if                  % if index = length
                        %
  ROMnames               % prepare to fetch name
  exch                   % exchange index and ROMnames
  get                    % get name from ROMnames array
                         %
  end                    % end $idleTimeDict
} def                    % fontname ends here
```

Switch Settings 0, 1, and 3

These thumbwheel switch settings are responsible for executing PostScript programs on either of the two serial channels, or using AppleTalk communication.

The 0, 1, or 3 procedures are called by execjob (see Chapter 14 on Job Execution) after the end of one job, but before the beginning of the next job.

If there is any idle time between jobs, the UseIdleTime procedure performs idle time font caching. When input data is detected on one of the filestreams as the beginning of the next job, idle time font caching stops and setrealdevice is called to check the print engine status and set up the frame buffer for the proper paper size. The idleproc procedure is defined next and then the default job, manualfeed, and wait timeouts are established for the upcoming job. Finally at the end of the procedure, the incoming user job is executed.

```
%________________________________________________________________
% calling format
% serverdict 0 get exec
% called from execjob
% found in:  serverdict
0                        %
{                        %
  clearinterrupt         % clean up old interrupts
  disableinterrupt       % disable interrupts during
                         % critical code sequence
  //serverdict           % serverdict
  begin                  %
                         %
  /watchstreams          % watchstreams is used by
                         % UseIdleTime to watch for input
                         % data which indicates
                         % the beginning of a job,
                         % and the end of idle time
  load                   % place watchstreams on stack for
                         % UseIdleTime
                         %
  UseIdleTime            % perform idle time font caching
                         %
  setrealdevice          % check status of printer and set up
```

```
                                        % for proper paper size
                                        %
//execdict                              % execdict
begin                                   %
                                        %
                                        % prepare idleproc definition
                                        % for executive mode using
                                        % batchidleproc
                                        %
                                        % (complete discussion and
                                        % documentation of
                                        % batchidleproc in chapter on
                                        % executive mode)
                                        %
                                        % batchidleproc begins here
                                        %
/idleproc                               %
{                                       %
  /idleproc                             %
  {                                     %
  }                                     %
  def                                   %
  //serverdict                          % serverdict
  begin                                 %
  0                                     %
  defaulttimeouts                       %
  pop                                   %
  exch                                  %
  pop                                   %
  0                                     %
  settimeouts                           %
  end                                   %
}                                       %
def                                     %
                                        %
                                        % batchidleproc ends here
                                        %
end                                     % end execdict
                                        %
protect                                 % protect font cache
                                        %
defaulttimeouts                         % fetch default job, manualfeed and
                                        % wait timeouts from EEPROM
                                        %
settimeouts                             % use default timeouts to set the
                                        % job, manualfeed and wait timeouts
                                        % for this job
                                        %
enableinterrupt                         %
stdin                                   % place input filestream on stack
end                                     % end serverdict
```

```
  cvx                  % convert filestream to executable
  exec                 % execute input data on
                       % stdin filestream
  disableinterrupt     %
} def                  % 0 ends here
```

Switch Setting 2 and the Special Switch

Switch setting 2 may be used to select either Diablo mode or PostScript interactive mode, depending on the value of location 58 in the eescratch array. Location 58 is referred to as the "special switch". A value of 0 tells PostScript to use Diablo mode, while a value of 1 invokes interactive mode.

We will later see that the dictionary called specialswitch contains two entries, 0 and 1. Guess what these two entries are responsible for! Procedure 2 described below indirectly calls procedures 0 or 1 in specialswitch based on the value it retrieves from location 58 of eescratch.

```
%______________________________________________________________
% calling format
% serverdict 2 get exec
% found in:  serverdict
                        %
2                       %
{                       %
  58                    % fetch the value of the special
  eescratch             % switch from location 58
                        % of eescratch
                        %
  dup                   % save a copy of the switch setting
                        %
                        % next a check is made to determine
                        % if the switch value is valid
                        %
  //specialswitch       % specialswitch
  exch                  %
  known                 % is the switch value known in the
                        % specialswitch dictionary
                        %
  {                     % if known in specialswitch
                        %
                        % do nothing
                        %
  }                     % if known in specialswitch
                        %
  {                     % else if not known in specialswitch
                        %
    pop                 % pop bogus value
    0                   % make mode Diablo
                        %
    } ifelse               % else if not known in specialswitch
                           %
                           % next fetch procedure 0 or 1 from
                           % the specialswitch dictionary
                           %
    //specialswitch        % specialswitch
    exch                   %
    get                    % get procedure 0 or 1 from
                           % specialswitch
                           %
    exec                   % execute procedure 0 or 1
                           %
} def                      % 2 ends here
```

The Special Switch Procedures 0 and 1

These two procedures are called indirectly from procedure 2 in serverdict. The specialswitch procedure 0 invokes the Diablo 630 emulator and procedure 1 invokes the executive procedure.

Special Switch Procedure 0 Procedure 0 in the specialswitch dictionary is responsible for setting up and executing a Diablo 630 print job. The call to the diablo operator is set up as follows:

```
normalfont boldfont pitch autoLF diablo
```

The diablo parameters come from EEPROM settings as described below.

The diablo operator is responsible for internally setting printpageflag. This determines if the Diablo 630 page description should be printed at the end of procedure 0.

Note: Lower case "diablo" refers to the PostScript operator while upper case "Diablo" refers to the Diablo 630 printer language.

Values stored in the EEPROM in the eescratch array which are accessed by procedure 0 are defined as follows.

Location 59, Auto-linefeed - the Diablo auto-linefeed feature is enabled by a value of 1, any other value disables it.

Location 60, Pitch - selects the number of characters per inch or pitch. Common values are 10, 12 or 15, with the default value of 0 selecting 10 characters per inch.

Location 61, Bold Font - selects the bold font by using a number which indexes into the ROMnames array. If 0 (the default) is used as the index, then 1 is substituted thereby selecting Courier Bold.

Location 62, Normal Font - selects the normal font from the ROMnames array. The default is 0 or Courier.

Procedure 0 Outline The procedure starts by initializing printpageflag to false. If there is idle time before the diablo job, UseIdleTime is called to perform idle time font caching and watch for input job data. When the job data begins arriving, setrealdevice checks the print engine status and sets up the frame buffer for the proper paper size. The default timeouts are established next after which the EEPROM locations are accessed in preparation for the call to the diablo procedure. The call to diablo is done within a stopped context in order to trap any errors during execution of the diablo job. After the call to diablo, if the printpageflag is true, a showpage is done.

```
%____________________________________________________________
% calling format
% specialswitch 0 get exec
% found in:  specialswitch
                        %
0                       %
{                       %
                        %
  /printpageflag        % the only reference to
                        % printpageflag
  false                 % is in this procedure.
                        % It is initialized to false,
  def                   % and can only be set true in
                        % the call to the diablo procedure
                        %
  {                     % begin stopped procedure
                        %
    serverdict          % open serverdict
    begin               %
                        %
    /watchstreams       % prepare for idle time font
                        % caching while
    load                % waiting for Diablo job to arrive
    UseIdleTime         %
                        %
    setrealdevice       % make sure print engine is ready,
                        % and set up for correct paper size.
                        %
    defaulttimeouts     % make the default job, manualfeed
                        % and wait timeouts the current
```

```
      settimeouts        % timeouts for this job
                         %
      62                 % get the normal font ROMnames
      eescratch          % index value from the EEPROM
      fontname           % fetch the font name
                         % using the index
                         %
      61                 % get the bold font ROMnames
      eescratch          % index value from the EEPROM
                         %
                         % in this next section of code,
                         % if we find the default value of 0
                         % in the EEPROM, a value of 1 is
                         % substituted so that Courier
                         % Bold is selected as the bold font
                         %
      dup                % save a copy of the index
      0                  %
      eq                 % index = 0 (Courier)
      {                  % if index = 0
                         %
        pop              % discard 0
        1                % put 1 in the place of 0
                         %
      } if               % if index = 0
                         %
      fontname           % fetch the font name using
                         % the index
                         %
      60                 % get the pitch from the EEPROM
      eescratch          % a value of 0 (the default)
                         % selects 10 pitch
                         %
      59                 % get auto-linefeed information
      eescratch          % 1 = select auto-linefeed,
                         % else disable it
                         %
      end                % end serverdict
                         %
      clearinterrupt     %
      diablo             % execute Diablo job
                         %
    }                    % end stopped procedure
                         %
    stopped              % executed above procedure
                         %
    pop                  % dump boolean returned by stopped
                         %
                         % instead of checking to see if the
                         % stopped procedure ran by checking
                         % the returned boolean,
                        % printpageflag is used to
                        % determine whether the
                        % diablo procedure executed properly
                        %
  printpageflag         %
  {                     % if printpageflag = true
    showpage            % print Diablo page
  } if                  % if printpageflag = true
                        %
} def                   % 0 ends here
```

Procedure 1 Outline "Special switch" procedure 1 is responsible for setting up and executing the executive procedure and entering PostScript interactive mode.

Procedure 1 is composed of two major sections. First, the print command is redefined in execdict and output is sent to both stdout and altout (if altout is available). Secondly, idleproc is defined for use by the executive procedure. It turns out that this section of code is identical to the intidleproc procedure. Finally, the executive procedure itself is executed.

```
%_______________________________________________________
% calling format
% specialswitch 1 get exec
% found in:  specialswitch
                        %
1                       %
{                       %
  //execdict            % execdict
  begin                 % begin execdict
                        %
                        % this section of code redefines the
                        % print procedure such that output
                        % is sent to both stdio and
                        % altio channels
                        %
                        % it also turns out that this print
                        % code is identical to the
                        % altprint procedure
                        %
                        % altprint begins here
                        %
  /print                %
  {                     %
    //serverdict        % serverdict
    begin               % begin serverdict
                        %
    altflag             % is alternate channel available?
  {                     % if altflag = true
                        %
    altout              % altout is channel to write to
                        %
    1                   % fetch output string from
    index               % stack using index command
                        %
    writestring         % write string to altout
                        %
    altout              % flush any remaining
    flushfile           % characters from altout buffer
                        %
  } if                  % if altflag = true
                        %
  stdout                % stdout is channel to write to
  exch                  % put output string in correct
                        % stack location
  writestring           % write string to stdout
                        %
  stdout                % flush any remaining
  flushfile             % characters from stdout buffer
                        %
  end                   % end serverdict
}                       %
def                     %
                        % altprint ends here
                        %
                        % this section of code defines
                        % idleproc for the benefit of
                        % the executive procedure
                        % it is identical to the
                        % intidleproc procedure
                        %
                        % start intidleproc procedure
                        % (documented in executive chapter)
/idleproc               %
{                       %
  /idleproc             %
  {                     %
  }                     %
  def                   %
  /print                %
  //systemdict          % systemdict
  /print                %
  get                   %
  def                   %
  //serverdict          % serverdict
  begin                 %
  /watchstreams         %
  load                  %
  UseIdleTime           %
```

```
            setrealdevice       %
            0                   %
            defaulttimeouts     %
            pop                 %
            exch                %
            pop                 %
            0                   %
            settimeouts         %
            protect             %
            end                 %
          }                     %
          def                   %
                                % end intidleproc procedure
                                %
                                %
          end                   % end execdict
                                %
          executive             % invoke executive procedure
                                %
        } def                   % 1 ends here
```

Chapter 13

Printer Communications

Introduction

This chapter documents how the PostScript printer communicates with the outside world through its communication channels. PostScript communication activity is fairly complex with many procedures and variables involved. Most of the activity centers around the procedures setstreams and setsccstreams in the serverdict.

Communication Control Procedures

In the control loop portion of the start procedure (explained in the start procedure chapter) we find the sole reference to setstreams. It purpose is to check the thumbwheel setting prior to the beginning of a new job, and set the communication channels to the proper settings.

None of the printer communications operators or procedures are documented by Adobe except for sccbatch, setsccbatch, sccinteractive, and setsccinteractive. Have you ever wondered about the differences between these two pairs of operators?

In this chapter you will see exactly where the documented procedures fit into the undocumented world of PostScript printer communications. You will learn where stdio and altio channels are assigned and used, how Adobe opens a transparent channel for Diablo 630 emulation, how AppleTalk is opened and closed, plus other interesting communications issues.

Chapter Outline

This topic is presented in several sections. First, the system variables which define communication settings are discussed, followed by two utility procedures: exchdef and dexch. Next, AppleTalk procedures are documented, followed by the major communication procedures watchstreams, setsccstreams and setstreams.

Welcome to the world of PostScript communications!

System Variables

The variables used by system communication procedures are presented first.

Jobsource Variable The jobsource variable can take on the values of "serial 25" or "serial 9" to identify the source of the incoming job data. The 9 or 25 refers to the 9 pin or 25 pin hardware connector on the printer where the host computer is connected.

```
%_______________________________________________________________
% jobsource is found in statusdict
% not executable; length = 9
/jobsource              % referenced in setstreams,
                        % stopPred, watchstreams
(serial 25)             %
def                     % jobsource ends here
```

stdname Variable The stdname variable indicates the source of the standard input/output filestreams and can be either "serial 25", "serial 9" or "AppleTalk".

```
%_______________________________________________________________
% stdname is found in serverdict
% not executable; length = 9
/stdname                % referenced in setsccstreams
                        % appletalkopen, stopPred,
                        % watchstreams
(serial 25)             %
def                     % stdname ends here
```

altname Variable The altname variable indicates the source of the alternate input/output filestreams and can be either "serial 25" or "serial 9" or "AppleTalk".

```
%_______________________________________________________________
% altname is found in serverdict
% not executable; length = 8
/altname                % referenced in setsccstreams,
                        % stopPred, watchstreams
(serial 9)              %
def                     % altname ends here
```

appletalktype Variable The appletalktype variable is used in establishing communications with a Macintosh computer and is always "LaserWriter".

```
%_______________________________________________________________
% appletalktype is found in statusdict
% not executable; length = 11
/appletalktype          % referenced in appletalkopen
(LaserWriter)           %
def                     % appletalktype ends here
```

baud25 and baud9 Variables The baud25 and baud9 variables hold the current values of the serial 25 and serial 9 baud rates.

```
%______________________________________________________________
% baud25 is found in serverdict
% type = integertype
/baud25 9600 def        % referenced in setsccstreams
```

```
%______________________________________________________________
% baud9 is found in serverdict
% type = integertype
/baud9 9600 def         % referenced in setsccstreams
```

parity25 and parity9 Variables The parity25 and parity9 variables hold the parity information for the serial 25 and serial 9 communication channels as described in the setsccbatch and setsccinteractive documentation from Adobe.

```
%______________________________________________________________
% parity25 is found in serverdict
% type = integertype
/parity25 3 def         % referenced in setsccstreams
```

```
%______________________________________________________________
% parity9 is found in serverdict
% type = integertype
/parity9 0 def          % referenced in setsccstreams
```

commhash Variable The commhash variable holds a combination of switch and communication parameters as defined by hashcommparams. The hashcommparams procedure is documented in the chapter on job execution.

```
%______________________________________________________________
% commhash is found in serverdict
% type = integertype
/commhash -1022600191   % referenced in setsccstreams,
def                     % appletalkopen, execjob
```

saveswitch Variable The saveswitch variable holds the last recorded value of the thumbwheel switch setting.

```
%________________________________________________________________
% saveswitch is found in serverdict
% type = integertype
/saveswitch 1 def       % referenced in setstreams,
                        % setsccstreams, start
                        % execjob, checkquit
```

altflag Boolean The altflag boolean indicates whether the alternate communication channel is available for use.

```
%________________________________________________________________
% altflag is found in serverdict
% type = booleantype
/altflag true def       % referenced in setsccstreams,
                        % altprint, stopPred, watchstreams,
                        % appletalkopen,
                        % 1 (in specialswitch)
```

sccok Boolean The sccok boolean determines if any changes to the communications settings have been made since the last call to setsccstreams.

```
%________________________________________________________________
% sccok is found in serverdict
% type = booleantype
/sccok true def         % referenced in setsccstreams
```

sendctrld Boolean The sendctrld boolean decides if a control-d should be sent back over the reverse channel by PostScript. It is set to false in AppleTalk mode; true in RS 232 and RS 422 serial modes.

```
%________________________________________________________________
% sendctrld is found in serverdict
% type = booleantype
/sendctrld true def     % referenced in setsccstreams,
                        % appletalkopen, execjob
```

transparent Boolean The transparent boolean decides whether to open the serial channel in transparent mode where the normal control codes (control-c and control-d) are not interpreted and the 8th bit is used as data.

```
%______________________________________________________________
% transparent is found in serverdict
% type = booleantype
/transparent false def  % referenced in setsccstreams
```

debugmode Integer The debugmode integer is used to decide whether to enable or disable the serial 25 channel. It is normally defined to be 0 which enables the serial 25 channel.

```
%______________________________________________________________
% debugmode is found in statusdict
% type = integertype
/debugmode 0 def        % referenced in setsccstreams
```

switchclose Array The switchclose array is an array of four procedures which correspond to communication channel shutdown procedures for each of the four respective thumbwheel switch settings. The corresponding switchopen array which opens the channels is imbedded in the setstreams procedure which is documented later in this chapter.

```
%______________________________________________________________
% found in:  serverdict
% referenced in setstreams
/switchclose            %
[                       % array begin
  {                     % thumbwheel setting 0
    9                   %
    closescc            % shutdown the serial 9 filestreams
    25                  %
    closescc            % shutdown the serial 25 filestreams
  }                     %
  {                     % thumbwheel setting 1
    9                   %
    closescc            % shutdown the serial 9 filestreams
    25                  %
    closescc            % shutdown the serial 25 filestreams
  }                     %
  {                     % thumbwheel setting 2
    9                   %
    closescc            % shutdown the serial 9 filestreams
    25                  %
    closescc            % shutdown the serial 25 filestreams
  }                     %
  {                     % thumbwheel setting 3
    appletalkclose      % shutdown AppleTalk
  }                     %
] def                   % switchclose ends here
```

Utility Procedures

General purpose utility procedures dexch and exchdef are presented next.

dexch Procedure The dexch or double exchange procedure takes two dictionary keys and exchange the values between them. It may be helpful to draw a snapshot of the stack through each step of the procedure to see more clearly how the exchange is done.

```
%_____________________________________________________________
% calling format
% key1 key2 dexch
% referenced in stopPred, watchstreams
% found in:  serverdict
/dexch                  %
{                       %
  dup                   % dup key2
  load                  % get value2
  exch                  % put key2 back on top of stack
                        %
  2                     % fetch a copy of key1
  index                 % using index command
                        %
  load                  % get value1
                        %
  store                 % key2 value1 store
  store                 % key1 value2 store
                        %
} def                   % dexch ends here
```

exchdef Procedure The exchdef procedure is a handy procedure which combines the common "exch def" sequence.

```
%_____________________________________________________________
% calling format
% value key exchdef
% referenced in setsccstreams, settimeouts
% found in:  serverdict
/exchdef                %
{                       %
  exch                  % exchange key and value
  def                   % key value def
                        %
} def                   % exchdef ends here
```

AppleTalk Procedures

AppleTalk mode uses its own set of procedures for setting up the input and output filestreams and establishing communications. It is a bit more involved since AppleTalk is a network, and not just a simple serial channel.

appletalkopen Procedure The appletalkopen procedure has several distinct parts. First the serial 9 and serial 25 channels are shut down with closescc commands. A long piece of code builds the string "PS 800:LaserWriter" according to the AppleTalk Name Binding Protocol. It identifies the type of printer to the AppleTalk network. (Studying this code shows a technique for joining strings.)

Adobe briefly discusses the AppleTalk Name Binding Protocol in its PostScript printer supplements and also references two Apple documents: *Inside LaserWriter* and *Inside AppleTalk.* Refer to these documents for additional information on AppleTalk.

The value from the undocumented location 63 of the eescratch array is used to uniquely identify the printer to the network, and if location 63 is not a unique identifier the software creates one. After this, all the important system control variables are defined such as stdin, stdout, sendctrld, altflag, stdname, and commhash.

The definition of sendctrld indicates that control-d's are not sent over the AppleTalk network. The altflag definition indicates that the serial 25 channel is never used while AppleTalk is active.

```
%_______________________________________________________________
% calling format
% appletalkopen
% referenced in setstreams, switchopen
% found in:  serverdict
/appletalkopen          %
{                       %
  //statusdict          % statusdict
  begin                 % begin statusdict
                        %
  9                     % shut down the
  closescc              % serial 9 channel
                        %
  25                    % shut down the
  closescc              % serial 25 channel
                        %
                        % this next section of code
                        % builds a string used by
                        % setappletalkname
                        %
                        % this string is designed to conform
                        % to the AppleTalk Name Binding
                        % protocol
                        %
                        % the first part of the string is
                        % called the object and is the
                        % printer's individual name
                        %
                        % the second part is called the type
                        % and is always "LaserWriter"
                        %
                        % string length =  128
  (this string will eventually contain "PS 800:LaserWriter")
                        %
                        %
  printername           % place the printer name in
                        % above string
  length                % get the length of the printer name
  dup                   % save a copy of length on stack
                        % to use as index for putinterval
                        %
                        % string length =  128
  (this string will eventually contain "PS 800:LaserWriter")
                        %
                        %
  exch                  % place index in proper place
                        % on stack
                        %
                        % string length =  1
  (:)
                        %
```

```
putinterval          % place ":" after printer name
                     %
1                    % add 1 to the printer name length
add                  % for the new index value
dup                  % save new index value on stack
                     %
                     % string length =  128
(this string will eventually contain "PS 800:LaserWriter")
                     %
                     %
exch                 % place index in proper place
                     % on stack
                     %
appletalktype        % place (LaserWriter) string
                     % on stack
                     %
putinterval          % place "LaserWriter" on end of
                     % the string
                     %
appletalktype        % get the length of
length               % the "LaserWriter" string
                     %
add                  % add to previous length
                     %
                     % string length =  128
(this string will eventually contain "PS 800:LaserWriter")
                     %
                     %
exch                 % place interval end in proper place
                     %
0                    %
exch                 % place interval begin in
                     % proper place
                     %
getinterval          % extract the "PS 800:LaserWriter"
                     % string
                     %
setappletalkname     % send string to the Macintosh
                     %
                     % the next section of code uses
                     % location 63 of the eescratch
                     % array to uniquely identify this
                     % printer on an AppleTalk
                     % network
                     %
                     % if the integer returned by
                     % initappletalk is different,
                     % then location 63 is changed
                     % to the new value
                     %
63                   %
```

```
eescratch           % fetch location 63
                    %
initappletalk       % send value to the Macintosh
                    %
dup                 % save a copy of the returned value
                    %
63                  %
eescratch           % fetch location 63 again
ne                  % returned-value ne location-63
                    %
{                   % if values different
  63                %
  exch              %
  seteescratch      % set location 63 to returned value
                    %
}                   % if values different
                    %
{                   % else if values same
                    %
  pop               % discard returned value
                    %
} ifelse            % else if values same
                    %
                    % the next piece of code defines
                    % the variables stdin and stdout
                    %
//$error            % $error
/initializing       %
get                 % fetch initializing boolean
not                 % not initializing
                    %
{                   % if not initializing
                    %
  openappletalk     % openappletalk returns input and
                    % output filestreams
                    %
  /stdout           % define the stdout variable
  exch              % as the output filestream
  store             %
                    %
  /stdin            % define the stdin variable
  exch              % as the input filestream
  store             %
                    %
  stdin             % use setstdio operator to define
  stdout            % the standard input and output
  setstdio          % filestreams
                    %
} if                % if not initializing
                    %
                    % the final section of the procedure
```

```
                            % defines several system variables
                            %
  /sendctrld                % control-d is not be sent in
  false                     % AppleTalk mode
  store                     %
                            %
  /altflag                  % the serial 25 channel is not
  false                     % used as an alternate channel in
  store                     % AppleTalk mode
                            %
  /stdname                  % define stdname as "AppleTalk"
                            %
                            % string length =  9
  (AppleTalk)
                            %
  store                     %
                            %
  /commhash                 % replace the old comm and switch
  hashcommparams            % settings with the current settings
  store                     %
                            %
  end                       % end statusdict
                            %
} def                       % appletalkopen ends here
```

appletalkclose Procedure The appletalkclose procedure shows that the value of -1 detaches a printer from the AppleTalk network.

```
%___________________________________________________________
% calling format
% appletalkclose
% referenced in switchclose
% found in:  serverdict
/appletalkclose             %
{                           %
  -1                        %
  initappletalk             % a -1 detaches the printer from
                            % the appletalk network
                            %
  pop                       % discard the result
                            %
} def                       % appletalkclose ends here
```

The watchstreams Procedure

This procedure is used during the printer idle time to watch for input on one or both (if the altflag is true) of the serial input filestreams. A boolean left on the stack by watchstreams is used to signal the calling procedure if any input activity was detected.

Remember, the watchstreams procedure is the identical to the stopPred procedure in $idleTimeDict. A detailed discussion of how stopPred is used by the UseIdleTime procedure is in Chapter 8 on Idle Time Font Scan Conversion.

```
%____________________________________________________________
% calling format
% watchstreams boolean
% referenced in 0,1 (in specialswitch),
% 0,1,3 (in serverdict), intidleproc
% found in:  serverdict
/watchstreams           %
{                       %
  //serverdict          % serverdict
  begin                 % begin serverdict
                        %
  altflag               % true if alternate channel
                        % available
                        %
  {                     % if alternate channel available
                        %
                        % in the next piece of code, if
                        % input is sensed on the alternate
                        % channel, then the alternate and
                        % standard channel definitions are
                        % switched such that the alternate
                        % channel becomes the standard
                        % channel
                        %
    altin               % check to see if the alternate
    bytesavailable      % input filestream has any bytes
    0                   % available for reading yet
    ne                  %
    {                   % if bytes available
                        %
      /stdin            % switch the alternate and standard
      /altin            % input filestream definitions
      dexch             %
                        %
      /stdout           % switch the alternate and standard
      /altout           % output filestream definitions
      dexch             %
                        %
      /stdname          % switch the alternate and standard
      /altname          % name definitions
      dexch             %
                        %
    } if                % if bytes available
  } if                  % if alternate channel available
                        %
                        % in the next piece of code,
                                % standard input is checked for any
                                % signs of activity
                                %
    stdin                       % check to see if the standard input
    bytesavailable              % filestream has any bytes available
    0                           % for reading yet
    ne                          %
                                %
    dup                         % save a copy of the resulting
                                % boolean to pass back to the
                                % calling procedure
                                %
    {                           % if bytes available
      stdin                     %
      stdout                    %
      setstdio                  % define the stdin, stdout
                                % filestreams as standard input
                                % and output
                                %
      //statusdict              % statusdict
      begin                     % begin statusdict
                                %
      /jobstate                 % change the jobsource to "busy"
                                %
                                % string length =  4
      (busy)
                                %
      def                       %
                                %
      /jobsource                % define the jobsource as
      stdname                   % "serial 25" or "serial 9"
                                % depending on where the source
      def                       % of the input
                                %
      end                       % end statusdict
                                %
      1                         % tell the front panel light to
      setblink                  % single-blink
                                %
    } if                        % if bytes available
                                %
    end                         % end serverdict
                                %
    enableinterrupt             % enable interrupts momentarily
    disableinterrupt            % disable interrupts
                                %
  } def                         % watchstreams ends here
```

The setsccstreams Procedure

The setsccstreams procedure is the most involved of all the communication procedures. It is responsible for setting up all the file streams and system control variables for the one or two (if the alternate channel is available) serial channels.

Its first task is to define the baud rate and parity for each channel and check for a situation where baud25 and baud9 have both been defined as 0. If both channels are set to 0 baud, the printer would be deaf and dumb with no communications possible.

Next, the system variables altflag, sendctrld, transparent, and commhash are defined. The definition of altflag shows that both serial channels are available unless the user defines the baud rate of one channel to be 0. In normal serial mode (not AppleTalk) the sendctrld boolean is true so that control-d characters can be echoed over the reverse channel.

The transparent boolean is true only if switches are set to invoke the Diablo 630 emulator. In transparent mode, control codes are not interpreted and high ascii values (values greater than 127) may be transmitted to the Diablo emulator.

A complex boolean calculation is performed next to define the sccok boolean. It detects any switch changes, communications parameter changes, or changes in the status of any of the active filestreams.

The sccok variable is then used to create the serial 25 and serial 9 filestreams. If sccok is true, meaning that no changes have been detected, then the current filestreams are fetched using the sccfiles operator. If changes are detected, then new filestreams are created using the openscc operator.

Finally, the system variables for stdname, stdout, stdin, altname, altout, and altin are created and stdin/stdout are defined as the standard input/output filestreams.

```
%___________________________________________________________
% calling format
% baud9 parity9 baud25 parity25 setsccstreams
% referenced in setstreams and
% switchopen
% found in:  serverdict
/setsccstreams          %
{                       %
  //serverdict          % serverdict
  begin                 % begin serverdict
                        %
  /parity25             % get parity25 value from the
```

```
exchdef                 % stack and define it
                        %
/baud25                 % get baud25 value from the
exchdef                 % stack and define it
                        %
                        % debugmode is normally 0, but if
                        % it is nonzero, the serial 25
                        % channel is turned off by setting
                        % its baud rate to 0
                        %
//statusdict            % statusdict
/debugmode              % this is the only reference
                        % to debugmode
get                     % fetch value of debugmode
                        %
0                       %
ne                      % debugmode ne 0?
{                       % if debugmode nonzero
                        %
  /baud25               % define the serial 25 channel
                        % baud rate
  0                     % to be 0 (turn off the serial 25
  def                   % channel)
                        %
} if                    % if debugmode nonzero
                        %
/parity9                % get parity9 value from the
exchdef                 % stack and define it
                        %
/baud9                  % get the baud9 value from the
exchdef                 % stack and define it
                        %
                        % this next piece of code checks to
                        % make sure that at least one of the
                        % serial channels is active
                        %
                        % if both of the channels are set to
                        % a baud rate of 0 then the
                        % channels are reset to 1200 baud
                        %
baud9                   % get baud9
baud25                  % get baud25
or                      % bitwise or
0                       %
eq                      % baud9 or baud25 = 0
                        %
{                       % if both baud rates = 0
                        %
  /baud9                % reset baud9 to 1200 baud
  1200                  %
  def                   %
```

```
                                 %
  /baud25                        % reset baud25 to 1200 baud
  1200                           %
  def                            %
                                 %
} if                             % if both baud rates = 0
                                 %
                                 % the altflag is used later
                                 % to determine if an alternate
                                 % channel is available
                                 %
                                 % the alternate channel is not
                                 % available if one of the baud
                                 % rates is set to 0
                                 %
/altflag                         %  prepare to define altflag
                                 %
baud9                            %
0                                %
ne                               % baud9 ne 0
                                 %
baud25                           %
0                                %
ne                               % baud25 ne 0
                                 %
and                              % (baud9 ne 0) and (baud25 ne 0)
                                 %
def                              % define altflag
                                 %
/sendctrld                       % sendctrld is used in execjob to
true                             % decide if a control-d is to be
def                              % sent back to the host over the
                                 % reverse channel
                                 %
                                 % the transparent boolean is used
                                 % with the openscc operator to
                                 % place the communications channels
                                 % in transparent mode such that the
                                 % high order bit is not interpreted
                                 % by PostScript
                                 %
                                 % transparent is only true for the
                                 % Diablo mode of operation
                                 %
/transparent                     % prepare to define transparent
saveswitch                       %
2                                %
eq                               % saveswitch = 2
                                 %
58                               %
eescratch                        % get specialswitch value
```

```
0                     %
eq                    % specialswitch = 0
                      %
and                   % (saveswitch = 2) and
                      % (specialswitch = 0)
                      %
                      % Diablo mode is only active if the
                      % thumbwheel switch = 2 and
                      % specialswitch = 0
                      %
def                   % define transparent
                      %
                      % this next piece of code computes a
                      % boolean which is assigned
                      % to sccok
                      %
                      % sccok is true if
                      % no switches, comm parameters,
                      % or file status' have been
                      % changed
                      %
                      % clever use of the "and" function
                      % requires close study of the
                      % following code
                      %
commhash              % prepare to compare the previous
                      % commhash settings with the
                      % current settings
hashcommparams        % provided by hashcommparams
                      %
dup                   % save a copy of the current
/commhash             % settings and reassign them
                      % to commhash
exchdef               %
                      %
eq                    % previous settings =
                      % current settings ?
                      %
baud9                 % fetch baud rate of serial 9
0                     %
ne                    % baud9 ne 0 ?  (serial 9 active?)
{                     % if serial 9 active
                      %
  9                   % fetch the input and output
  sccfiles            % filestreams from serial 9
                      %
  status              % get output filestream status
  exch                %
  status              % get input filestream status
                      %
  and                 % (output status) and (input status)
```

```
                          %
  and                     % (comm settings) and
                          % (filestream status)
                          % leave boolean result on stack
                          %
} if                      % if serial 9 active
                          %
baud25                    % fetch baud rate of serial 25
0                         %
ne                        % baud25 ne 0 ? (serial 25 active?)
{                         % if serial 25 active
                          %
  25                      % fetch input and output filestreams
  sccfiles                % from serial 25
                          %
  status                  % get output filestream status
  exch                    %
  status                  % get input filestream status
                          %
  and                     % (output status) and (input status)
                          %
  and                     % (boolean from above) and
                          % (filestream status)
                          %
                          % the "boolean from above" is
                          % either the boolean left on the
                          % stack from the serial 25 active
                          % procedure, or the comm settings
                          % boolean if serial 25 was inactive
                          %
                          % leave boolean result on stack
                          % for sccok
                          %
} if                      % if serial 25 active
                          %
/sccok                    % it's finally time to
exchdef                   % define sccok
                          %
                          % sccok is only referenced
                          % in setsccstreams
                          %
                          % the following section of code
                          % contains two similar "if"
                          % procedures which open, close
                          % or maintain files in their
                          % current status
                          %
25                        % prepare to operate on serial 25
                          %
baud25                    % fetch baud rate of serial 25
0                         %
```

```
ne                      % baud25 ne 0? (serial 25 active?)
                        %
{                       % if serial 25 active
                        %
  sccok                 % sccok = true if no changes in comm
                        % parameters or filestreams
                        %
  {                     % if sccok = true
                        %
    sccfiles            % 25 sccfiles returns
                        % input-filestream output-filestream
                        % where output-filestream is on
                        % top of the stack
                        %
  }                     % if sccok = true
                        %
  {                     % else if sccok = false
                        %
    baud25              % fetch baud25
    parity25            % fetch parity25
    transparent         % fetch transparent boolean
                        % (remember transparent is only
                        %  true for Diablo mode)
                        %
    openscc             % 25 baud parity boolean openscc
                        % returns
                        % input-filestream output-filestream
                        % where output-filestream is on
                        % top of the stack
                        %
  } ifelse              % else if sccok = false
                        %
                        % the following string is
                        % assigned to altname or stdname
                        % later
                        %
                        % string length =  9
  (serial 25)
                        %
}                       % if serial 25 active
                        %
{                       % else if serial 25 inactive
                        %
  closescc              % 25 closescc
                        % shuts down serial 25 input and
                        % output filestreams
                        %
} ifelse                % else if serial 25 inactive
                        %
                        %
9                       % prepare to operate on serial 9
```

```
                                %
baud9                           % fetch baud rate of serial 9
0                               %
ne                              % baud25 ne 0? (serial 25 active?)
                                %
{                               % if serial 9 active
                                %
  sccok                         % sccok = true if no changes in comm
                                % parameters or filestreams
                                %
  {                             % if sccok = true
                                %
    sccfiles                    % 9 sccfiles returns
                                % input-filestream output-filestream
                                % where output-filestream is on
                                % top of the stack
                                %
  }                             % if sccok = true
                                %
  {                             % else if sccok = false
                                %
    baud9                       % fetch baud9
    parity9                     % fetch parity9
    transparent                 % fetch transparent boolean
                                % (remember transparent is only
                                %  true for Diablo mode)
                                %
    openscc                     % 9 baud parity boolean openscc
                                % return
                                % input-filestream output-filestream
                                % where output-filestream is on
                                % top of the stack
                                %
  } ifelse                      % else if sccok = false
                                %
                                % the following string is
                                % assigned to altname or stdname
                                % later
                                %
                                % string length =  8
  (serial 9)
                                %
}                               % if serial 9 active
                                %
{                               % else if serial 9 inactive
                                %
  closescc                      % 9 closescc
                                % shuts down serial 9 input and
                                % output filestreams
                                %
} ifelse                        % else if serial 9 inactive
```

```
                      %
                      % at this point, the names and
                      % filestreams are on the stack
                      % prepared for the following
                      % assignments
                      %
 /stdname             % assign stdname to either
 exchdef              % "serial 9" or "serial 25"
                      %
 /stdout              % assign stdout to the
 exchdef              % output-filestream on top of stack
                      %
 /stdin               % assign stdin to the
 exchdef              % input-filestream on top of stack
                      %
 altflag              % altflag is true only if both
                      % channels are available
                      %
 {                    % if both channels available
                      %
   /altname           % assign altname to
   exchdef            % "serial 9"
                      %
   /altout            % assign altout to the
   exchdef            % output-filestream on top of stack
                      %
   /altin             % assign altin to the
   exchdef            % input-filestream on top of stack
                      %
 } if                 % if both channels available
                      %
 stdin                % fetch stdin filestream
 stdout               % fetch stdout filestream
                      %
                      % setstdio defines which set of
                      % filestreams will be treated as the
                      % standard input and standard output
                      %
 setstdio             % input-file output-file setstdio
                      %
 end                  % end serverdict
                      %
} def                 % setsccstreams ends here
```

The setstreams Procedure

setstreams is the controlling procedure for setting up and shutting down PostScript host communications. It is only referenced once in the control loop portion of the start procedure. The start procedure is discussed in a later chapter.

The function of setstreams is to shut down the current communication channels if a thumbwheel switch change has been detected. It does this by fetching the appropriate procedure from the switchclose array and executing it. Next, the proper channels are opened by executing a selected procedure from the imbedded switchopen array.

Study the switchopen array in setstreams and see the distinction between sccbatch and sccinteractive. It turns out that sccbatch is used for switch setting 1, which is for PostScript batch mode processing. sccinteractive is used for switch setting 2, which is used for PostScript interactive mode (or Diablo mode) processing.

```
%____________________________________________________________
% calling format
% setstreams
% referenced only in start
% found in:  serverdict
/setstreams             %
{                       %
  //statusdict          % statusdict
  /jobsource            % the jobsource is defined as null
  null                  % at this point, until data is
  put                   % later sensed on one of the
                        % communication channels by the
                        % watchstreams procedure
                        %
                        % jobsource is then assigned
                        % the string "serial 25" or
                        % "serial 9"
                        %
                        % in this first section of code,
                        % determine if the thumbwheel
                        % switch has been changed
                        %
                        % if changed, close channels from
                        % previous setting
                        %
  saveswitch            % fetch previous setting of switch
  switchsetting         % fetch current setting of switch
  ne                    %
  {                     % if current ne previous setting
                        %
                        % close the previously open channels
                        % by fetching the proper procedure
```

```
                                  % from the switchclose array and
                                  % executing it
                                  %
  switchclose                     % put switchclose array on stack
  saveswitch                      % put index into switchclose
                                  % on stack
  get                             % fetch procedure from switchclose
  exec                            % execute procedure to close
                                  % channels
                                  %
} if                              % if current ne previous setting
                                  %
/saveswitch                       % redefine saveswitch to be
switchsetting                     % the current switch setting
def                               %
                                  % the following array is equivalent
                                  % to the switchopen array in
                                  % serverdict
                                  %
                                  % this array also contains the only
                                  % references to setsccstreams
                                  %
                                  % switchopen begins here
                                  %
[                                 % array begin
                                  %
                                  % the first procedure corresponds
                                  % to switch setting 0 where both
                                  % channels are set to 1200 baud
                                  %
  {                               %
    1200                          % serial 9 baud rate
    0                             % serial 9 options
    1200                          % serial 25 baud rate
    0                             % serial 9 options
    setsccstreams                 % set up both channels
  }                               %
                                  % the second procedure corresponds
                                  % to switch setting 1 where the
                                  % channels are programmed to the
                                  % settings defined by sccbatch
  {                               %
    9                             % fetch the serial 9 baud rate
    sccbatch                      % and options
    25                            % fetch the serial 25 baud rate
    sccbatch                      % and options
    setsccstreams                 % set up both channels
  }                               %
                                  % the third procedure corresponds to
                                  % switch setting 2 where the
                                  % channels are programmed to the
```

```
                                   % settings defined by sccinteractive
      {                            %
        9                          % fetch the serial 9 baud rate
        sccinteractive             % and options
        25                         % fetch the serial 25 baud rate
        sccinteractive             % and options
        setsccstreams              % set up both options
      }                            %
                                   % the fourth procedure corresponds
                                   % to switch setting 3 where the
                                   % serial 9 channel is programmed
                                   % for AppleTalk
                                   %
      {                            %
        appletalkopen              % set up AppleTalk
      }                            %
    ]                              % array end
                                   %
                                   % switchopen ends here
                                   %
    saveswitch                     % index for the above array
    get                            % fetch the proper procedure
    exec                           % execute the procedure to open the
                                   % proper communication channels
                                   %
  } def                            % setstreams ends here
```

Chapter 14

Job Execution and execjob

Introduction

All the procedures and variables involved in preparing for user job execution in the server loop, and cleaning up afterwards are covered in this chapter. The server loop is discussed in detail in Chapter 15 on the start procedure. All of the action takes place in the procedure called execjob, and in this procedure we find some of the following interesting items:

- Several different uses of the stopped operator.
- PostScript error recovery and call to handleerror.
- The source of the dreaded error message:
  `%%[ Flushing: rest of job (to end-of-file) will be ignored ]%%`
- How PostScript prepares for job execution.
- How PostScript cleans up after job execution.

Before the execjob procedure is documented, all of the procedures, variables, and operators it references are discussed briefly.

Welcome to the world of execjob!

Procedures used by execjob

The execjob procedure directly references only two simple procedures: cleardictstack and hashcommparams.

The cleardictstack Procedure This procedure clears all but the bottom two dictionaries off of the dictionary stack. The systemdict and userdict are left since they cannot be cleared. The cleardictstack procedure is referenced as an embedded block of code, plus it is called by name.

```
%___________________________________________________________
% calling format
% cleardictstack
% referenced in start, execjob
% found in:  userdict
/cleardictstack          %
{                        %
  countdictstack         % get total number of dicts
                         %
  2                      % sub 2 to account for userdict
  sub                    % and systemdict
                         %
  {                      % start repeat loop
                         %
    end                  % close dictionary
```

```
                       %
  } repeat             % end repeat loop
} def                  % cleardictstack ends here
```

The hashcommparams Procedure This procedure combines the all the communication settings and switch settings into one number which is assigned to the variable called commhash. The execjob procedure uses the commhash number to determine if any of the settings have been changed since the last time commhash was computed.

```
%____________________________________________________________
% calling format
% hashcommparams
% referenced in appletalkopen,
% setsccstreams, execjob
% found in:  serverdict
/hashcommparams        %
{                      %
  switchsetting        % fetch the thumbwheel setting
                       %
  9                    % fetch the baud and parity settings
  sccbatch             % from the 9 pin channel
                       %
  25                   % fetch the baud and parity settings
  sccbatch             % from the 25 pin channel
                       %
  9                    % fetch the baud and parity settings
  sccinteractive       % from the 9 pin channel (special
                       % switch setting)
                       %
  25                   % fetch the baud and parity settings
  sccinteractive       % from the 25 pin channel (special
                       % switch setting)
                       %
  58                   % fetch the value of the special
  eescratch            % switch
                       %
  9                    % 9 times through the loop adds
                       % the above 10 items together
                       %
  {                    % start repeat loop
                       %
    3                  % shift each value by 3 bits
    bitshift           % and then add to the
    add                % previous result
                       %
  } repeat             % end repeat loop
} def                  % hashcommparams ends here
```

Variables used by execjob

The variables used by execjob are explained below.

commhash Integer The commhash variable is the last known value computed by hashcommparams.

```
%_______________________________________________________
% commhash is found in serverdict
% type = integertype
/commhash -1022600191 def
```

saveswitch Integer The saveswitch variable is the last known value of the thumbwheel switch.

```
%_______________________________________________________
% saveswitch is found in serverdict
% type = integertype
/saveswitch 1 def
```

quitflag Boolean The quitflag variable is the terminating boolean for the executive procedure and is set to false by execjob in case executive is invoked.

```
%_______________________________________________________
% quitflag is found in execdict
% type = booleantype
/quitflag false def
```

jobsource String The jobsource string indicates whether the incoming job source is the 25 pin channel ("serial 25") or the 9 pin channel ("serial 9").

```
%_______________________________________________________
% jobsource is found in statusdict
% not executable; length = 9
/jobsource              %
(serial 25)             %
def                     % jobsource ends here
```

jobstate String The jobstate string can take on values such as "busy", "idle", "printing test page", "printing", plus a variety of engine error messages.

```
%________________________________________________________
% jobstate is found in statusdict
% not executable; length = 4
/jobstate               %
(busy)                  %
def                     % jobstate ends here
```

doclose Boolean If the doclose boolean is true, execjob closes the stdin and stdout file streams as part of its clean-up procedure.

```
%________________________________________________________
% doclose is found in $error
% type = booleantype
/doclose true def
```

The execjob Procedure

It is finally time to discuss execjob. As this procedure is studied, it is important to realize that this is a crucial control procedure which cannot under any circumstances fail. A study of the start procedure in another chapter shows the overall importance of execjob and where it fits into the server loop. In short, if execjob fails, the PostScript interpreter will crash. Because of this, execjob makes extensive use of the stopped operator to trap every possible type of error which could happen during its execution.

The stopped operator is first used to trap user program errors. After that, the error handling section of execjob is executed using stopped in case it should fail. That means there is error recovery for the error recovery code!

execjob Outline

In order to make the execjob procedure more understandable, its outline is presented as follows:

- Several control variables are initialized.
- The thumbwheel switch is read and the appropriate procedure 0, 1, 2 or 3 (from serverdict) is executed. These procedures are documented in detail in the chapter on the thumbwheel switch and are responsible for actually executing the user's job.
- Error conditions are dealt with.
- General clean-up is performed such as closing filestreams, setting timeouts, initializing the graphics state, clearing stacks, and erasing the page memory.

```
%_____________________________________________________________
% calling format
% execjob
% referenced in start
% found in:  serverdict
/execjob                    %
{                           %
  //statusdict              % statusdict
  begin                     % open statusdict
                            %
  /jobstate                 %
                            % string length =  4
  (idle)
                            %
  def                       % jobstate = "idle"
                            %
  /jobname                  %
  null                      %
  def                       % jobname = null
                            %
  end                       % end statusdict
                            %
  0                         % turn off
  setblink                  % blinking light
                            %
  //$error                  % $error
  /doclose                  % doclose is used to decide to
  true                      % close stdin and stdout at the
  put                       % end of execjob
                            %
  //execdict                % execdict
  /quitflag                 % initialize quitflag in case
  false                     % executive is invoked as the
  put                       % user job
```

```
                            %
                            % the next step is to fetch the
                            % procedures which executes the
                            % user's program
                            %
                            % the procedures are the 0,1,2,3
                            % which are found in the serverdict
                            % and correspond to the settings on
                            % the thumbwheel switch
                            %
//serverdict                % serverdict
                            % prepare for get
                            %
//serverdict                % serverdict
/saveswitch                 % fetch the current setting of the
get                         % thumbwheel switch (a value of
                            % 0,1,2,3 will be returned)
                            %
get                         % use the above numeric value from
                            % the previous get to fetch the
                            % procedures 0 or 1 or 2 or 3 from
                            % serverdict
                            %
stopped                     % execute the thumbwheel procedure
                            % which in turn executes the user
                            % program
                            %
                            % the stopped operator places
                            % true on the stack if an error
                            % occurred
                            %
                            % false is placed on the stack
                            % if the job runs to completion
                            %
disableinterrupt            % no interrupts should occur during
                            % the next section of code
                            %
                            % the next section of code contains
                            % the error handling control code
                            %
                            % the following procedure is
                            % executed using a stopped operator
                            % rather than with an if operator
                            % as would normally be the case
                            % after a stopped.
                            %
                            % the boolean on the stack here is
                            % actually used later within this
                            % procedure to handle the error
                            % if one occurred
                            %
```

```
                              %
{                             % begin stopped procedure
  defaulttimeouts             %
  //serverdict                % serverdict
  begin                       %
  settimeouts                 % set the timeouts to default values
  end                         % end serverdict
                              %
  clearinterrupt              %
                              %
                              % this next procedure uses the
                              % error boolean from the stopped
                              % operator above
                              %
                              % remember the boolean is true if
                              % an error occurred
                              %
  {                           % if user error
                              %
                              % when an error occurs, clear the
                              % dictionary and operand stacks
                              %
    clear                     % clear the operand stack
                              %
                              % next, clear the dictionary stack
                              %
                              % cleardictstack code begins here
                              %
    {                         %
      countdictstack          % get total number of dicts
                              %
      2                       % sub 2 to account for userdict
      sub                     % and systemdict
                              %
      {                       % start repeat loop
                              %
        end                   % close dictionary
                              %
      } repeat                % end repeat loop
    }                         %
                              %
                              % cleardictstack code ends here
                              %
    exec                      % execute cleardictstack code
                              %
    //$error                  % $error
    begin                     % open $error
                              %
    //serverdict              % serverdict
    begin                     % open serverdict
                              %
```

```
newerror            % has error been taken care of yet?
{                   % if error not taken care of
                    %
                    % take care of error by executing
                    % the handleerror procedure
                    %
  //errordict       % errordict
  /handleerror      %
  get               % fetch handleerror
                    %
  exec              % execute handleerror to print the
                    % error message
                    %
} if                % if error not taken care of
                    %
                    % this next section uses a complex
                    % boolean statement to decide if
                    % the "flushing" message should
                    % be printed
                    %
                    % first see if any switches or comm
                    % values have changed
hashcommparams      % compute the current hash value
commhash            % fetch the stored hash value
eq                  % are they eq?
                    %
stdin               % check to see if stdin is open
status              % status = true if open
                    %
and                 % and above booleans
                    %
doclose             % fetch doclose boolean
                    %
and                 % and above booleans
                    %
errorname           % fetch errorname
/timeout            % check to see if errorname is
eq                  % eq to the timeout error
                    %
stdin               % check to see if the number of
bytesavailable      % bytes available on stdin
0                   % is eq to 0
eq                  %
                    %
and                 % and the above two booleans
                    %
not                 % invert the boolean
                    %
and                 % finally, and the two booleans
                    % together
                    %
```

```
                                 % in summary here is the equation:
                                 %
                                 % [(switches unchanged?
                                 % and stdin open?)
                                 %  and doclose] and
                                 % [(timeout error? and
                                 % 0 bytes avail?) not]
                                 %
        {                        % if complex condition true
                                 %
                                 % this is my favorite message!
                                 % string length =  62
          (%%[ Flushing: rest of job (to end-of-file) will be\
           ignored ]%%)
                                 %
          =                      % print string
          flush                  % flush stdio buffer
                                 %
          stdin                  % flush stdin so that no additional
          flushfile              % information will be processed
                                 %
        } if                     % if complex condition true
                                 %
        end                      % end serverdict
        end                      % end $error
                                 %
      } if                       % if user error
                                 %
      //serverdict               % serverdict
      begin                      % begin serverdict
                                 %
      //$error                   % $error
      /doclose                   % fetch doclose
      get                        % from $error
                                 %
      sendctrld                  % fetch sendctrld boolean
                                 %
      and                        % doclose and sendctrld
                                  %
      {                          % if time to print control-d
                                 % string length =  1
        (\004)
                                 %
        print                    % print control-d
      } if                       % if time to print control-d
                                 %
      flush                      % flush stdout
                                 %
      end                        % end serverdict
                                 %
    }                            % end stopped procedure
```

```
                      %
stopped               % execute above procedure
                      %
{                     % if error control procedure crashes
                      %
  stdin               % reset stdin and
  resetfile           % flush stdin buffers
                      %
  stdout              % reset stdout and
  resetfile           % flush stdout buffers
                      %
  //$error            % $error
  /doclose            % be sure that
  true                % the doclose boolean
  put                 % is set to true
                      %
} if                  % if error control procedure crashes
                      %
//serverdict          % serverdict
begin                 % begin serverdict
                      %
defaulttimeouts       % set the current timeouts to the
settimeouts           % default timeouts
                      %
                      % in the next section, if doclose
                      % is true then close the std files
                      %
                      % the stopped operator is used to
                      % execute the close procedures in
                      % case there is an error trying to
                      % execute closefile - the booleans
                      % from stopped are later discarded
                      %
//$error              % $error
/doclose              %
get                   % fetch the doclose boolean
                      %
{                     % if doclose = true
  {                   %
    stdout            %
    closefile         %
  }                   %
  stopped             % close stdout
                      %
  {                   %
    stdin             %
    closefile         %
  }                   %
  stopped             % close stdin
                      %
} if                  % if doclose = true
```

```
                                  %
                                  % perform final cleanup after
                                  % end of job
                                  %
    disableinterrupt              %
                                  %
    0                             %
    0                             %
    0                             %
    settimeouts                   % set all timeouts to infinity
                                  %
    clear                         % clear operand stack
                                  %
    cleardictstack                % clear dictionary stack
                                  %
    initgraphics                  % initialize graphics state
                                  %
    erasepage                     % clear page memory
                                  %
  } def                           % execjob ends here
```

Chapter 15

The start Procedure and Server Loop

Introduction

The start procedure ties together all the PostScript control procedures discussed up until now. At the end of the start procedure is the control loop which invokes the server procedure. The server procedure is responsible for building the famous PostScript server loop which Adobe has shrouded in mystery. It's no wonder it's a mystery with the many chapters of information needed to bring us here.

Even though we can view portions of the server loop on the execution stack, it is not possible to completely analyze it as we have done with all the procedures up to this point. We have come as far as we can come without going into Adobe proprietary-land because the last two pieces of the puzzle (the server and exitserver procedures) are not readable by mere mortal programmers.

Chapter Outline

Only one control procedure has not been documented up to this point: setnulldevice. After this short procedure is discussed, the start procedure is described and documented. At the end of the chapter the execution stack is examined to see what happens when a PostScript program is executed inside and outside the server loop.

Welcome to the world of the server loop, the final frontier!

The setnulldevice Procedure

The setnulldevice procedure is responsible for initializing the current transformation matrix (CTM) before the beginning of each PostScript job.

```
%________________________________________________________________
% calling format
% setnulldevice
% referenced in start
% found in:  serverdict
/setnulldevice          %
{                       %
  nulldevice            % this is a documented operator
                        % which replaces the current CTM
                        % with the identity matrix
                        % [1 0 0 1 0 0]
                        %
  $printerdict          % fetch the device matrix from
  /mtx                  % the "mtx" variable in
  get                   % the $printerdict dictionary
                        %
  setmatrix              % establish "mtx" as the default
                         % CTM which each program begins
                         % with
                         %
} def                    % setnulldevice ends here
```

The start Procedure Discussion

The start procedure is the first PostScript program executed by the controller upon power-up. It is responsible for printing the test page, executing all user programs, and is executing when the printer is turned off. The control loop at the end of start is designed to be invulnerable, however if you should find a way to break out of this loop by accident or design you will crash the PostScript controller. Should this happen, the message "start procedure lost control" is printed, and the printer restarts as if you had just turned on the power.

start Procedure Outline

The procedure begins by defining several system variables and establishing print engine communication with the initprinter procedure. If the test page is to be printed, it is first imaged into memory and then idle time font caching is performed until the print engine is warmed-up. If there is any problem printing the test page, the attempt is aborted.

After the test page code, the control loop is entered. It is amazingly short, containing only 12 lines of code and calls to four procedures. The four procedures are server, setstreams, setnulldevice and execjob which have been studied (with the exception of server which is locked) in earlier material. These procedures are responsible for all the functions of the printer and execution of user jobs.

```
%___________________________________________________________
% calling format
% start
% found in:  userdict
/start                  %
{                       %
  disableinterrupt      % turn off interrupts
                        %
  execdict              %
  /execdepth            % define the execdepth
  0                     % variable to be 0
  put                   % for the executive procedure
                        % should it be invoked by the user
                        %
  ReadIdleFonts         % build the idleArry with the
                        % idle time font cache information
```

```
                                        %
//serverdict                            % serverdict
begin                                   % begin serverdict
                                        %
/saveswitch                             % define saveswitch to be the
switchsetting                           % current thumbwheel switch
def                                     % setting
                                        %
setstreams                              % establish the file streams for the
                                        % first time
                                        %
initprinter                             % establish print engine
                                        % communications for the first time
                                        %
dostartpage                             % if dostartpage is true, the
                                        % test page is printed
                                        %
{                                       % if dostartpage = true
                                        %
                                        % for the printing of the
                                        % test page the current save level
                                        % is preserved with the next few
                                        % statements
                                        % svlv = savelevel
                                        %
  userdict                              % prepare to place the save object
                                        % in the userdict
                                        %
  save                                  % place a save object on the stack
                                        %
  /svlv                                 % prepare to name the save object
  exch                                  %
  put                                   % userdict /svlv save-object put
                                        %
  printerstatus                         % this is the printerstatus operator
                                        % in statusdict
                                        %
  -1                                    % -1 indicated print engine problems
                                        %
  ne                                    % printerstatus ne -1
                                        %
  {                                     % if print engine ok
                                        %
    statusdict                          % define the jobstate in statusdict
    /jobstate                           %
                                        % string length =  18
    (printing test page)
                                        %
    put                                 % define jobstate =
                                        % "printing test page"
                                        %
```

```
1                   % tell the status light to
setblink            % single-blink
                    %
statusdict          % define printererror in statusdict
/printererror       %
/stop               %
load                % get the stop operator
put                 % define printererror = stop
                    %
setrealdevice       % set-up proper page size
                    % definitions
                    %
{                   % begin stopped procedure
                    %
                    % the wtimeout variable is
                    % defined to be 3 minutes from the
                    % current user time
                    %
  /wtimeout         % prepare to define wtimeout
  usertime          % fetch current time
  180000            % 180000 ms = 180 sec = 3 min
  add               % current-time + 180000
  def               % define wtimeout
                    %
                    % the next few lines of code
                    % image the test page in the
                    % raster memory without issuing
                    % the showpage
                    %
                    % the startpage string is executed
                    % with userdict on the top of the
                    % dictionary stack because of the
                    % many definitions it creates
                    %
  userdict          %
  begin             % userdict begin
  startpage         % image test page
  end               % end userdict
                    %
                    % the next lines of code
                    % cause idle time font caching to
                    % occur while waiting for the print
                    % engine to warm up.
                    %
                    % the following procedure is used by
                    % the procedure UseIdleTime as the
                    % terminating condition for
                    % idle time
                    %
                    % in this case idle time ends
                    % when either the printer is
```

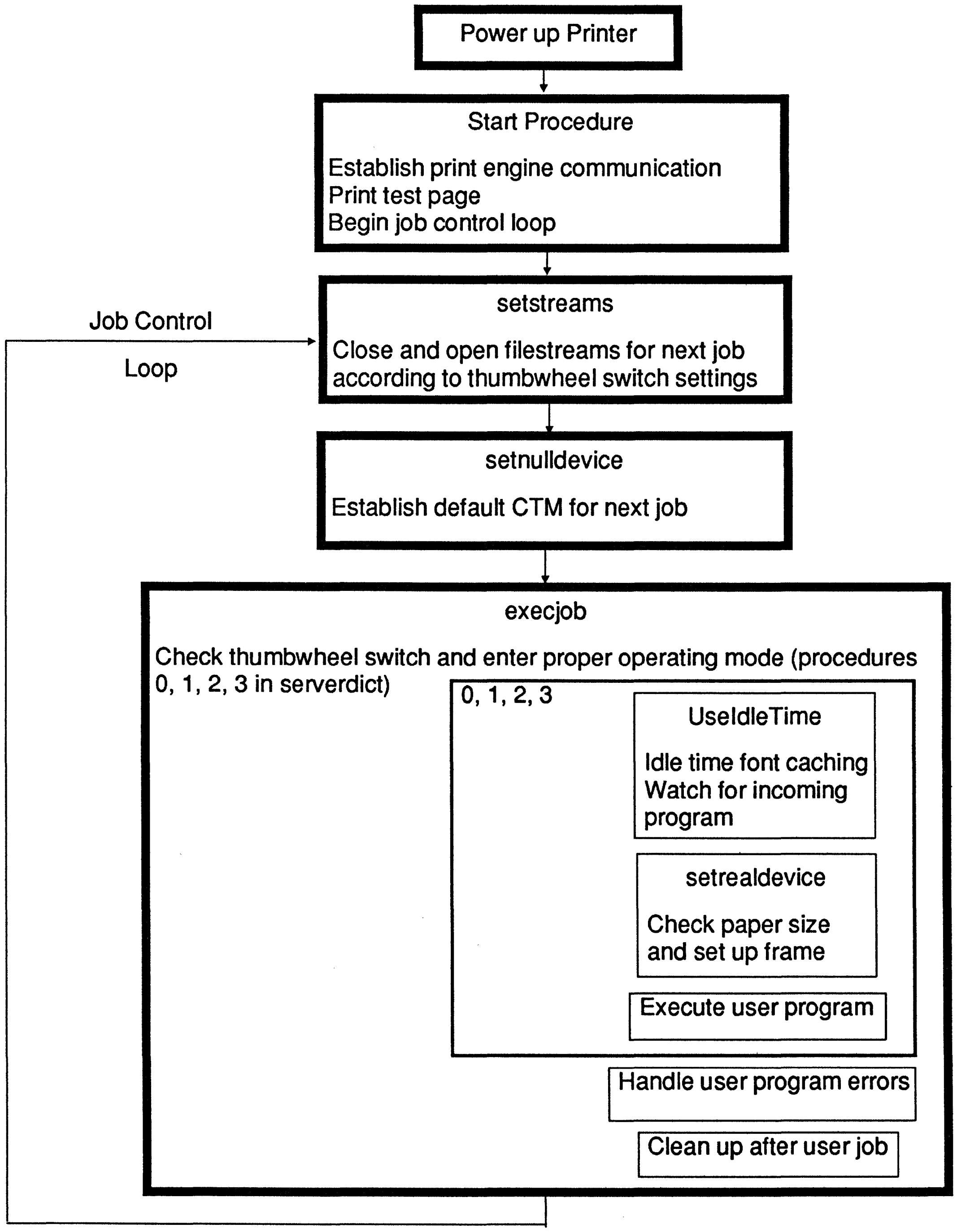
```
                                   % warmed up, or if wtimeout = 3 min
                                   % is exceeded
                                   %
      {                            % begin procedure
                                   %
        warmedup                   % warmedup returns true if printer
                                   % is ready
                                   %
        usertime                   %
        wtimeout                   %
        gt                         % usertime gt wtimeout
                                   % (3 minutes yet?)
                                   %
        or                         % warmedup or [usertime gt wtimeout]
                                   %
      }                            % end procedure
                                   %
      UseIdleTime                  % begin idle time font caching
                                   %
                                   % in the next few statements, a
                                   % check is made to see if the 3 min
                                   % expired before the printer has
                                   % warmed up
                                   %
                                   % if the printer has not warmed up
                                   % in three minutes, the test page
                                   % is not printed
                                   %
      usertime                     %
      wtimeout                     %
      le                           % usertime le wtimeout
                                   %
      {                            % if print engine warmed up ok
                                   %
        showpage                   % print the test page
                                   %
      } if                         % if print engine warmed up ok
                                   %
    }                              % end stopped procedure
                                   %
    stopped                        % execute above procedure
                                   %
    pop                            % dump the stopped boolean
                                   % (it doesn't matter if the
                                   % test page didn't print)
                                   % continue anyway
                                   %
    disableinterrupt               % disable interrupts
                                   %
  } if                             % if print engine ok
                                   %
```

```
                      % clean up after printing the
                      % test page by clearing the stacks
                      % and restoring to the previous
                      % save level
                      %
  clear               % clear the operand stack
                      %
  cleardictstack      % clear the dictionary stack
                      %
  svlv                % fetch the previous save object
  restore             % restore to previous save level
                      %
} if                  % if dostartpage = true
                      %
                      % just two final details are left
                      % before entering the server loop
                      %
cleardictstack        % clear the dictionary stack
                      %
$error                % tell PostScript initialization
/initializing         % is complete
false                 %
put                   % $error /initializing false put
                      %
                      % THE CONTROL LOOP
                      %
{                     % begin control loop
                      %
                      % step 1: execute the server
                      % procedure
                      %
  //serverdict        % serverdict
  /server             % fetch the server procedure
  get                 % from the serverdict dictionary
                      %
  exec                % execute the server procedure
                      %
                      % this places the server
                      % control loop on the execution
                      % stack
                      %
  //serverdict        % serverdict
  begin               % begin serverdict
                      %
  setstreams          % step 2: execute setstreams to
                      %         define filestreams
                      %
                      %
  setnulldevice       % step 3: execute setnulldevice to
                      %         establish CTM
                      %
```

```
                                    %
                                    %
    /execjob                        % step 4: execute execjob to control
                                    %         execution of user programs
                                    %
                                    %
    load                            % prepare to execute execjob
                                    %
    end                             % end serverdict prior to user job
                                    % execution
                                    %
    exec                            % execute execjob
                                    %
  } loop                            % end control loop
                                    %
} def                               % start ends here
```

Block Diagram of PostScript Interpreter

The following block diagram gives a general idea of the interconnection of the major portions of the PostScript interpreter.


Power up Printer
Start Procedure
Establish print engine communication
Print test page
Begin job control loop
Job Control
Loop
setstreams
Close and open filestreams for next job according to thumbwheel switch settings
setnulldevice
Establish default CTM for next job
execjob
Check thumbwheel switch and enter proper operating mode (procedures 0, 1, 2, 3 in serverdict)
0, 1, 2, 3
UseIdleTime
Idle time font caching
Watch for incoming program
setrealdevice
Check paper size and set up frame
Execute user program
Handle user program errors
Clean up after user job

Inside and Outside the Server Loop

Now that the control loop has been studied, what happens when the

```
serverdict begin 0 exitserver
```

command sequence is executed? By looking at the execution stack some general observations can be made, however several key procedures are locked prohibiting a complete study.

The execution stack is presented in compressed form as if the == procedure had been applied to estack in the $error dictionary. As the execution stack examples are studied, key portions of the execjob procedure and the control loop portion of the start procedure are recognized.

The key to knowing whether you are inside or outside the server loop is the structure of the execution stack. The control loop portion of the start procedure documented above is always at the bottom of the execution stack, however the server procedure itself places additional items on the stack depending on whether exitserver has been executed successfully.

The Execution Stack from Inside the Server Loop If a job is executing inside the server loop, several additional control procedures are placed on the stack between the control loop portion of the start procedure and execjob. These locked procedures are responsible for recovering VM consumed by the user program through the use of special save levels. Once VM is cleaned up, execjob is called again for the next user job.

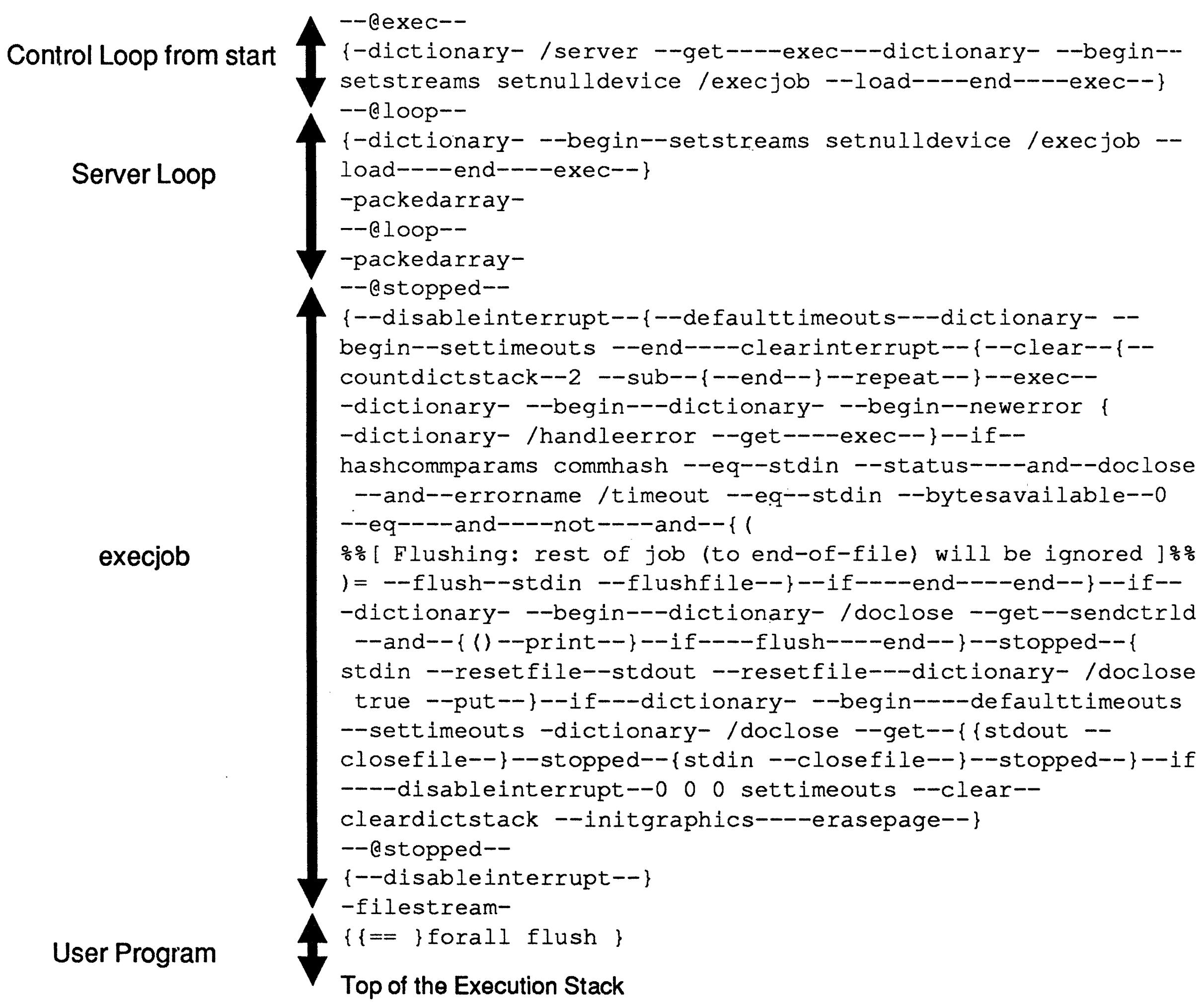
The special save level variables and other internal procedures and variables are no doubt stored in hidden dictionaries inaccessible to mortal programmers. The implication of allowing access to these inaccessible variables is that PostScript programmers could potentially destroy the job control structure established by Adobe. At best, control of VM would be lost and at worst, a user program could easily crash the printer.

By not allowing access to critical system variables and procedures, PostScript is able to maintain control of the printer no matter what the user program does.

It takes some practice to pick through the execution stack, but in the future it will greatly help your debugging efforts to be able to quickly find the different system and user procedures.

Execution Stack Example from Inside the Server Loop Spend a few minutes studying the first execution stack example below which shows the normal state of the execution stack.

Bottom of the Execution Stack



```
--@exec--
{-dictionary- /server --get----exec---dictionary- --begin--
setstreams setnulldevice /execjob --load----end----exec--}
--@loop--
{-dictionary- --begin--setstreams setnulldevice /execjob --
load----end----exec--}
-packedarray-
--@loop--
-packedarray-
--@stopped--
{--disableinterrupt--{--defaulttimeouts---dictionary- --
begin--settimeouts --end----clearinterrupt--{--clear--{--
countdictstack--2 --sub--{--end--}--repeat--}--exec--
-dictionary- --begin---dictionary- --begin--newerror {
-dictionary- /handleerror --get----exec--}--if--
hashcommparams commhash --eq--stdin --status----and--doclose
 --and--errorname /timeout --eq--stdin --bytesavailable--0
--eq----and----not----and--{(
%%[ Flushing: rest of job (to end-of-file) will be ignored ]%%
)= --flush--stdin --flushfile--}--if----end----end--}--if--
-dictionary- --begin---dictionary- /doclose --get--sendctrld
 --and--{()--print--}--if----flush----end--}--stopped--{
stdin --resetfile--stdout --resetfile---dictionary- /doclose
 true --put--}--if---dictionary- --begin----defaulttimeouts
--settimeouts -dictionary- /doclose --get--{{stdout --
closefile--}--stopped--{stdin --closefile--}--stopped--}--if
----disableinterrupt--0 0 0 settimeouts --clear--
cleardictstack --initgraphics----erasepage--}
--@stopped--
{--disableinterrupt--}
-filestream-
{{== }forall flush }
```

Top of the Execution Stack

The Execution Stack from Outside the Server Loop When exit-server is successfully executed, the execution stack shows that the control loop portion of the start procedure is just before execjob. The intermediate procedures observed in the previous example are gone. This is because jobs are now executing at save level 0 and any VM consumed will not be recovered until the printer is turned off.

Bottom of the Execution Stack

```
%%[ exitserver: permanent state may be changed ]%%
--@exec--
```

Control Loop from start

```
{-dictionary- /server --get----exec---dictionary- --begin--
setstreams setnulldevice /execjob --load----end----exec--}
```

execjob

```
--@loop--
{--disableinterrupt--{--defaulttimeouts---dictionary- --
begin--settimeouts --end----clearinterrupt--{--clear--{--
countdictstack--2 --sub--{--end--}--repeat--}--exec--
-dictionary- --begin---dictionary- --begin--newerror {
-dictionary- /handleerror --get----exec--}--if--
hashcommparams commhash --eq--stdin --status----and--doclose
 --and--errorname /timeout --eq--stdin --bytesavailable--0
--eq----and----not----and--{(
%%[ Flushing: rest of job (to end-of-file) will be ignored ]%%
)= --flush--stdin --flushfile--}--if----end----end--}--if--
-dictionary- --begin---dictionary- /doclose --get--sendctrld
 --and--{()--print--}--if----flush----end--}--stopped--{
stdin --resetfile--stdout --resetfile---dictionary- /doclose
 true --put--}--if---dictionary- --begin----defaulttimeouts
--settimeouts -dictionary- /doclose --get--{{stdout --
closefile--}--stopped--{stdin --closefile--}--stopped--}--if
----disableinterrupt--0 0 0 settimeouts --clear--
cleardictstack --initgraphics----erasepage--}
--@stopped--
{--disableinterrupt--}
```

User Program

```
-filestream-
{{== }forall flush }
```

Top of the Execution Stack

Server Loop Discussion

When the user job executing outside the server loop is ended, save levels 1 and 2 are reestablished again by the server procedure, protecting the changes made in save level 0. Under normal conditions, when a user program begins it is always at save level 2.

Look again closely at the save/restore procedures placed on the stack by the server procedure above the control loop. They are controlled by another loop (displayed as --@loop--). This is the server loop. That is, it is a looping structure placed on the stack by the server procedure. Exiting the server loop means exiting or breaking out of this looping structure.

These execution stack examples show how the server procedure is used to control save levels and sheds light on the concept of exiting the server loop. Exiting the server loop simply means that the special save/restore procedures protecting save level 0 are removed from the execution stack.

Server Loop Tips

According to Adobe documentation, work outside the server loop is for variables which the programmer intends to persist from job to job. This means a change to PostScript virtual memory (VM) outside the server loop persists until the printer is turned off.

By looking at the results of the vmstatus operator inside and outside the server loop, we find that inside the server loop, the number of save levels is 2, and outside the server loop the number of save levels is 0. PostScript uses save level 2 for the user jobs, save level 1 is used for clean-up between user jobs, and save level 0 is used when the user exits the server loop. This is why the user never finds himself at save level 1.

Being able to permanently modify VM in the form of adding dictionaries and definitions outside the server loop is the most common reason for exiting the server loop, however another benefit is being able to modify the procedures within the serverdict itself.

If an attempt is made to redefine or add a definition to the serverdict at save level 2 or higher, an "invalidaccess" error occurrs, however at save level 0 the user has free access to serverdict definitions. The implication of this is that procedures such as server, execjob, setstreams, and etc. may be redefined. However, the danger of tampering with the interpreter at this level is that the start procedure may lose control.

Given this capability to redefine the Adobe PostScript interpreter, perhaps someone will design a more user friendly or error tolerant interpreter.

Happy computing!

Part V

The Test Page Group

Chapter 16

The Test Page

Introduction

The test page code is completely different in nature than any of the other PostScript programs studied in this book. This is because the test page code is self-contained and actually produces an image in the frame buffer, where the other programs are very interdependent and are geared toward control functions rather than actual imaging of pages.

There are several differences worth pointing out. First, the test page code is the only (visible or accessible) executable string resident in the printer. The control procedures on the other hand, are all arrays or packed arrays and all make use of binding (using the bind operator) and immediately evaluated names (using "//"). Since the test page is self-contained, extensive redefinition of PostScript operators is used to produce very compact (and cryptic) code.

startpage Organization

The test page code is organized as a definitions section followed by an execution section at the very end. The definitions start as very simple abbreviations for operators and later involve more complex sequences for printing the different sections of the test page. These procedures are later invoked in the execution section to render the page.

```
%___________________________________________________________________
% calling format
% startpage
% referenced in start
% found in serverdict
/startpage               %
(                        % startpage string starts here
                         %
                         % establish some simple definitions
  /D  { def } def        %
  /ED { exch D } D       %
  /RL { rlineto } D      %
  /MT { moveto } D       %
  /RMT { rmoveto } D     %
  /CP { closepath } D    %
  /SG { setgray } D      %
  /GS { gsave } D        %
  /GR { grestore } D     %
  /SB { statusdict       %
        begin } D        %
  /SL { setlinewidth } D%
  /ST { stroke } D       %
  /FS { GS SG fill GR    % fill and stroke a path
        ST } D           %
                         %
  /Str 100 string D      %
```

```
/TBuf 100 string D      % buffer for building strings
                        %
/ILin                   % initialize Tbuf to null string
{                       %
  /TLin                 % call new string TLin
  TBuf                  %
  0                     %
  0                     %
  getinterval           % get null string from TBuf
  D                     %
} D                     %
                        %
/C 306 D                % center of page is 306 points
                        %
/AS                     % string AS (calling format)
                        % AS (append string)
                        % the idea is to append STr
                        % on the end of TLin
                        % TBuf is used as a work string
{                       %
  /STr                  % fetch string from top of stack
  ED                    % and call it /STr
                        %
                        % TBuf is the work string
                        % TLin is a subset of TBuf
                        %
  TBuf                  % put STr in TBuf after
  TLin                  % the end of TLin
  length                %
  STr                   %
  putinterval           %
                        %
                        % next, redefine TLin by fetching
                        % the complete string from TBuf
                        %
  /TLin                 % prepare to redefine TLin
                        %
  TBuf                  % get the string from TBuf
  0                     % by figuring out the sum
  TLin                  % of the lengths of the two
  length                % strings STr and TLin
  STr                   %
  length                %
  add                   % add the lengths
  getinterval           % get the string
                        %
  D                     % redefine TLin
} D                     %
                        %
                        % more definitions
                        %
```

```
/LT { lineto } D       %
/SH { show } D         %
/SP2D { stringwidth    % given a string, return
        pop 2 div } D  % 1/2 the width
                       %
          .            % font related definitions
/SS { scalefont        %
      setfont } D      %
/MS { makefont         %
      setfont } D      %
/FF { findfont } D     %
/TB  /Times-Bold FF D  %
/TR /Times-Roman FF D  %
/TR7 { TR 7 SS } D     %
/TR12 {TR 12 SS } D    %
/Sym8 { /Symbol FF     %
        8 SS } D       %
/TBX { TB [42 0 0      %
       37 0 0] MS } D  % redefine the matrix for Times-Bold
/TB12 { TB 12 SS } D   %
                       %
/CCR                   % string CCR  (calling format)
                       % CCR (center and carriage return)
                       % this routine centers a string
                       % and then moves -18 points down
                       % to the next line
{                      %
  dup                  % dup the string
  SP2D                 % get half the width
  neg                  % negate the distance
                       %
  dup                  % move left half the distance in
  0                    % the x direction
  RMT                  %
                       %
  exch                 % put the string on top of stack
  SH                   % and show it
                       %
                       % move back half the distance from
  -18                  % the end of the string where we are
                       % now and then move down -18
  RMT                  % points
} D                    %
                       %
/Title                 % Title (calling format)
                       % print the product name and
                       % version number on the top
                       % of the test page
{                      %
  TBX                  % us the special Times-Bold font
  C 698 MT             % move to the upper center of page
  SB product end      % get the product string
  CCR                 % center it and move down
  TR12                % change to Times-Roman
  ILin                % initialize the line buffer TLin
                      % string length =  8
  (Version )
                      %
  AS                  % put "Version " in line buffer TLin
  SB version end      % get the version number
  AS                  % append it to the line buffer TLin
  TLin                %
  CCR                 % print TLin in the center of page
} D                   %
                      %
/Fonts                % Fonts (calling format)
                      % print all the font name
                      % information
{                     %
  TB12                %
  C 640 MT            %
                      % string length =  20
  (De\256ned Font Outlines)
                      %
                      % \256 is the "fi" ligature
  CCR                 %
                      %
  209.5  601 MT       % position to print Times
                      % family information
                      %
                      % in this section, the length
                      % of the string Times-Roman with
                      % the special "registered" character
                      % is calculated and adjusted for
                      % before the string is printed
                      % CCR cannot be used because of the
                      % font change in the string
  TR12                %
                      % string length =  5
  (Times)
                      %
  SP2D                %
  Sym8                %
                      % string length =  1
  (\342)
                      %
  SP2D                %
  add                 %
  TR12                %
                      % string length =  6
  (-Roman)
                      %
```

```
SP2D                %
add                 %
neg                 %
0                   %
RMT                 %
                    % string length =  5
(Times)
                    %
SH                  %
0 4 RMT             %
Sym8                %
                    % string length =  1
(\342)
                    %
SH                  %
0 -4 RMT            %
TR12                %
                    % string length =  6
(-Roman)
                    %
SH                  %
                    %
209.5  583 MT       % position for printing the
                    % rest of the Times typefaces
                    %
                    % string length =  10
(Times-Bold)
                    %
CCR                 %
                    % string length =  12
(Times-Italic)
                    %
CCR                 %
                    % string length =  16
(Times-BoldItalic)
                    %
CCR                 %
                    %
402.5 601 MT        % position for printing the
                    % Helvetica typefaces
                    %
                    % in this section, the length
                    % of the string Helvetica with
                    % the special "registered" character
                    % is calculated and adjusted for
                    % before the string is printed
                    % CCR cannot be used because of the
                    % font change in the string
                    %
                    % string length =  9
(Helvetica)
```

```
                    %
SP2D                %
Sym8                %
                    % string length =  1
(/342)
                    %
SP2D                %
add                 %
neg                 %
0                   %
RMT                 %
TR12                %
                    % string length =  9
(Helvetica)
                    %
SH                  %
0 4 RMT             %
Sym8                %
                    % string length =  1
(\342)
                    %
SH                  %
TR12                %
                    %
402.5 583 MT        % position for printing the rest of
                    % the Helvetica typefaces
                    %
                    % string length =  14
(Helvetica-Bold)
                    %
CCR                 %
                    % string length =  17
(Helvetica-Oblique)
                    %
CCR                 %
                    % string length =  21
(Helvetica-BoldOblique)
                    %
CCR                 %
                    %
209.5 509 MT        % position for printing the
                    % Courier typefaces
                    %
                    % string length =  7
(Courier)
                    %
CCR                 %
                    % string length =  12
(Courier-Bold)
                    %
CCR                 %
```

```
                             % string length =  15
  (Courier-Oblique)
                             %
  CCR                        %
                             % string length =  19
  (Courier-BoldOblique)
                             %
  CCR                        %
                             %
  402.5 509 MT               % position for printing Symbols Set
                             %
                             % string length =  11
  (Symbols Set)
                             %
  CCR                        %
} D                          %
                             %
                             % the next section defines a number
                             % of string related to the
                             % communications settings
                             %
/PC                          % pin channel string
                             % string length =  14
( pin channel: )
                             %
D                            %
/BD                          % baud string
                             % string length =  7
( baud, )
                             %
D                            %
/P                           % parity string
                             % string length =  7
(parity )
                             %
D                            %
/PAr                         % parity strings array
[                            %
                             % string length =  7
(ignored)
                             %
                             % string length =  3
(odd)
                             %
                             % string length =  4
(even)
                             %
                             % string length =  4
(none)
                             %
] D                          %
```

```
/If                    % interface string
                       % string length =  11
(Interface: )
                       %
D                      %
/RS                    % RS 232 string
                       % string length =  15
(RS232C Serial, )
                       %
D                      %
/Ba                    % batch string
                       % string length =  5
(Batch)
                       %
D                      %
                       % procedures S1 through S8
                       % are called from the switch
                       % array procedures below
                       %
                       % they are responsible for
                       % printing the communications
                       % interface information
                       %
/S1 { S2 RS AS AS      % add "RS232C Serial, " to string
      TLin CCR } D     %
                       %
/S2 { C 288 MT ILin    % start building "Interface: " line
      If AS } D        %
                       %
/S3 { S2 AS TLin       % add info to interface line
      CCR } D          %
                       %
/S4                    % this procedure builds the lines
                       % which list the serial
                       % communication parameters
{                      %
  C exch MT            % fetch y location from stack and
                       % moveto center of page
  ILin                 %
  AS                   % get (9) or (25) string from stack
                       % and begin building line
  PC                   %
  AS                   % add (pin channel) to line
  Str                  % get baud rate from the top of
  cvs                  % stack and convert to string
  AS                   % add baud rate to line
  BD                   %
  AS                   % add (baud) to line
  P                    %
  AS                   % add (parity) to line
  PAr                  % the next section determines
```

```
                      % the parity setting based on the
                      % option number on the top of the
                      % stack
  exch                %
  dup                 %
  4                   % options  4 are XON/XOFF
                      % options = 4 are DTR
  lt                  %
  {                   % if options  4
    get               % get parity string from PAr
  }                   % if options  4
                      %
  {                   % else if options >= 4
    4                 %
    sub               %
    get               % get parity string from PAr
    AS                % add parity info to line
                      % string length =  14
    (, DTR protocol)
                      %
  } ifelse            % else if options >= 4
  AS                  % add parity info to line
  TLin                %
  CCR                 % print line centered on page
} D                   %
                      %
/S6 { SB sccbatch     % get sccbatch info
      end exch } D    %
                      %
/S7 { SB              % get sccinteractive info
      sccinteractive  %
      end exch } D    %
                      %
/S8                   % S8 is called for thumbwheel
                      % setting 3
{                     %
  S1                  % print interface line
  9                   %
  S7                  % fetch 9 pin channel info
                      % string length =  1
  (9)
                      %
  267                 %
  S4                  % print 9 pin channel info
  25                  %
  S7                  % fetch 25 pin channel info
                      % string length =  2
  (25)
                      %
  246                 %
  S4                  % print 25 pin channel info
```

```
} D                       %
                          %
/SwAr                     % SwAr (switch array)
                          % this array prints the proper
                          % setting for the thumbwheel
                          % positions 0,1,2,3
                          %
[                         % begin array
                          %
{                         % begin procedure for position 0
  Ba                      %
  S1                      % print interface line
  0                       %
  1200                    %
                          % string length =  1
  (9)
                          %
  267                     %
  S4                      % print 9 pin channel info
  0                       %
  1200                    %
                          % string length =  2
  (25)
                          %
  246                     %
  S4                      % print 25 pin channel info
}                         % end procedure for position 0
                          %
{                         % begin procedure for position 1
  Ba                      %
  S1                      % print interface line
  9                       %
  S6                      % fetch 9 pin channel info
                          % string length =  1
  (9)
                          %
  267                     %
  S4                      % print 9 pin channel info
  25                      %
  S6                      % fetch 25 pin channel info
                          % string length =  2
  (25)
                          %
  246                     %
  S4                      % print 25 pin channel info
}                         % end procedure for position 1
                          %
{                         % begin procedure for position 2
  SB 58 eescratch end % determine special switch setting
  0                       %
  eq                      %
  {                       % if special switch = 0
                          % string length =  11
    (Diablo Mode)
                          %
  }                       % if special switch = 0
                          %
  {                       % else if special switch = 1
                          % string length =  11
    (Interactive)
                          %
  } ifelse                % else if special switch = 1
  S8                      % print interface and channel info
}                         % end procedure for position 2
                          %
{                         % begin procedure for position 3
                          % string length =  9
  (AppleTalk)
                          %
  S3                      % print "Interface: AppleTalk"
}                         % end procedure for position 3
                          %
] D                       % end array
                          %
/PgFr                     % PgFr (page frame)
                          % this procedure produces the frames
{                         %
  GS                      % gsave
  0.3                     %
  SL                      % 0.3 setlinewidth
  30 30 MT                % begin path for lower box
  0 367 RL                % draw box
  552 0 RL                %
  0 -367 RL               %
  CP                      % closepath
  0.93 FS                 % fill and stroke
                          %
  30 417 MT               % begin path for upper box
  0 344 RL                % draw box
  552 0 RL                %
  0 -344 RL               %
  CP                      % closepath
  0.99 FS                 % fill and stroke
  305 397 MT              % draw notch in bottom box
  2 0 RL                  %
  1 SG                    % 1 setgray (white line)
  ST                      % stroke with white
  GR                      % grestore
} D                       %
                          %
/PN                       % PN (printer name)
                          % this procedure prints the printer
```

```
                           % name at the bottom of the page
  {                        %
    TBX                    %
    C 72 MT                %
    Str                    %
    SB printername end     %
    CCR                    %
  } D                      %
                           % this is the end of the definitions
                           % and the start of the main
                           % program
                           %
  PgFr                     % first place the frames
  Title                    % write the product name and version
  PN                       % write the printer name at bottom
                           %
  TB12                     % this next section prints the
  C 342 MT                 % number of pages printed
  ILin                     %
  SB pagecount end         %
  Str                      %
  cvs                      %
  AS                       %
                           % string length =  14
  ( pages printed)
                           %
  AS                       %
  TLin                     %
  CCR                      % print number of pages printed
                           %
  SwAr                     % prepare to fetch procedure from
                           % SwAr
  ILin                     %
  SB switchsetting end     % fetch thumbwheel setting
  get                      % get procedure
  exec                     % print interface and communication
                           % setting information
                           %
  Fonts                    % print font info
                           %
  TR7                      % finally print the trademark info
  C 424 MT                 %
                           % string length =  68
  (Times and Helvetica are registered trademarks\
   of Allied Corporation.)
                           %
  CCR                      %
) cvx def                  % startpage ends here
```

Some Comments About the Test Page

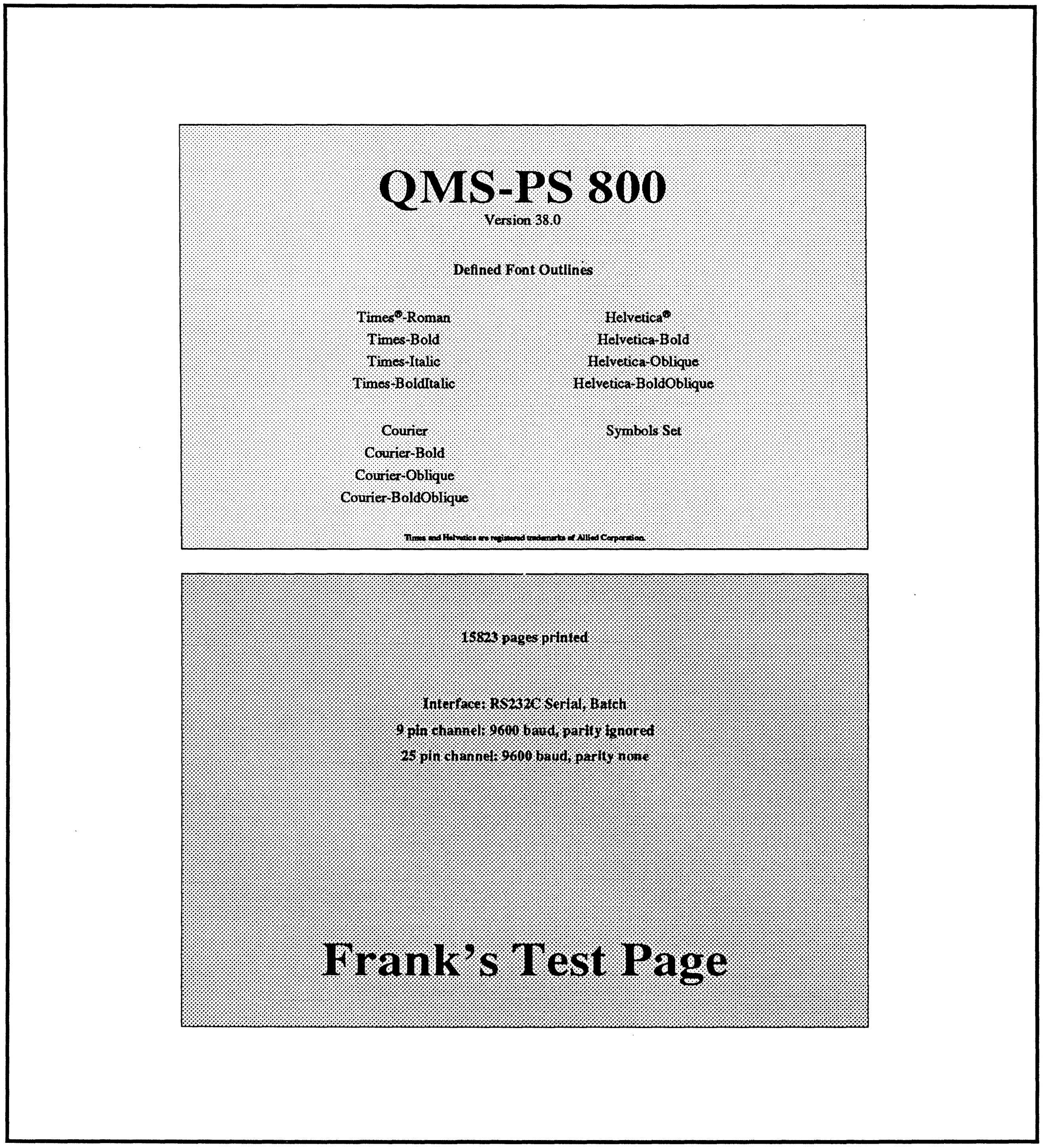
Notice that showpage is never issued in the test page code. This executable string is only responsible for rendering the page in the frame buffer. Chapter 15 on the start procedure documents how the printing of the test page is handled upon power-up.

If you want to print a test page as part of a user program the following code could be used.

```
serverdict begin        % open serverdict
                        % home of startpage
userdict begin          % open userdict for startpage
                        % definitions
                        % this is necessary since PostScript
                        % will not allow the user to define
                        % variables in serverdict
startpage               % execute startpage string
showpage                % print the results
end                     % close userdict
end                     % close serverdict
```

QMS-PS 800 Test Page

This is the image produced by the startpage code.



QMS-PS 800

Version 38.0

Defined Font Outlines

Times®-Roman	Helvetica®
Times-Bold	Helvetica-Bold
Times-Italic	Helvetica-Oblique
Times-BoldItalic	Helvetica-BoldOblique
Courier	Symbols Set
Courier-Bold	
Courier-Oblique	
Courier-BoldOblique	

Times and Helvetica are registered trademarks of Allied Corporation.

15823 pages printed

Interface: RS232C Serial, Batch
9 pin channel: 9600 baud, parity ignored
25 pin channel: 9600 baud, parity none

Frank's Test Page

Appendices

Appendix I

Dictionary Analysis

Introduction

The dictionary analysis table in this appendix contains information about every PostScript dictionary visible to the programmer. A block diagram of the PostScript interpreter dictionary structure is also presented.

This analysis applies specifically to the QMS-PS 800, version 38.0, revision 0. The QMS-PS 800, which is very similar to the Apple LaserWriter, is based on the Canon CX print engine and contains 13 resident fonts.

Dictionary Block Diagram

The block diagram shows the relationships between all the resident dictionaries. The dictionary interconnections depart from traditional "tree" structure in that a dictionary can contain a reference to itself. Dictionaries systemdict, mydict and ==dict are examples of this. Most all of the major dictionaries reside in systemdict and userdict.

How to Use the Table

The names of the 60 dictionaries are listed in the left column. The "Found In" column indicates where each dictionary is found.

The "Access Privilege" column was determined by the rcheck operator. The only dictionaries which are not accessible are those which are associated with font dictionaries.

The "Length" and "Maxlength" columns were determined by the length and maxlength operators respectively. This shows which dictionaries can be added to by the PostScript programmer.

The mydict and ==dict dictionaries (entries 11 and 12) have some strange entries in the "Found In" column. The significance of mydict being found in mydict and ==dict being found in ==dict is that these two dictionaries are only found as entries within themselves and are not referenced in any other dictionary. Chapters 6 and 11 cover these two special dictionaries in detail.

The entries for each dictionary in this table are listed in Appendix II.

Block Diagram

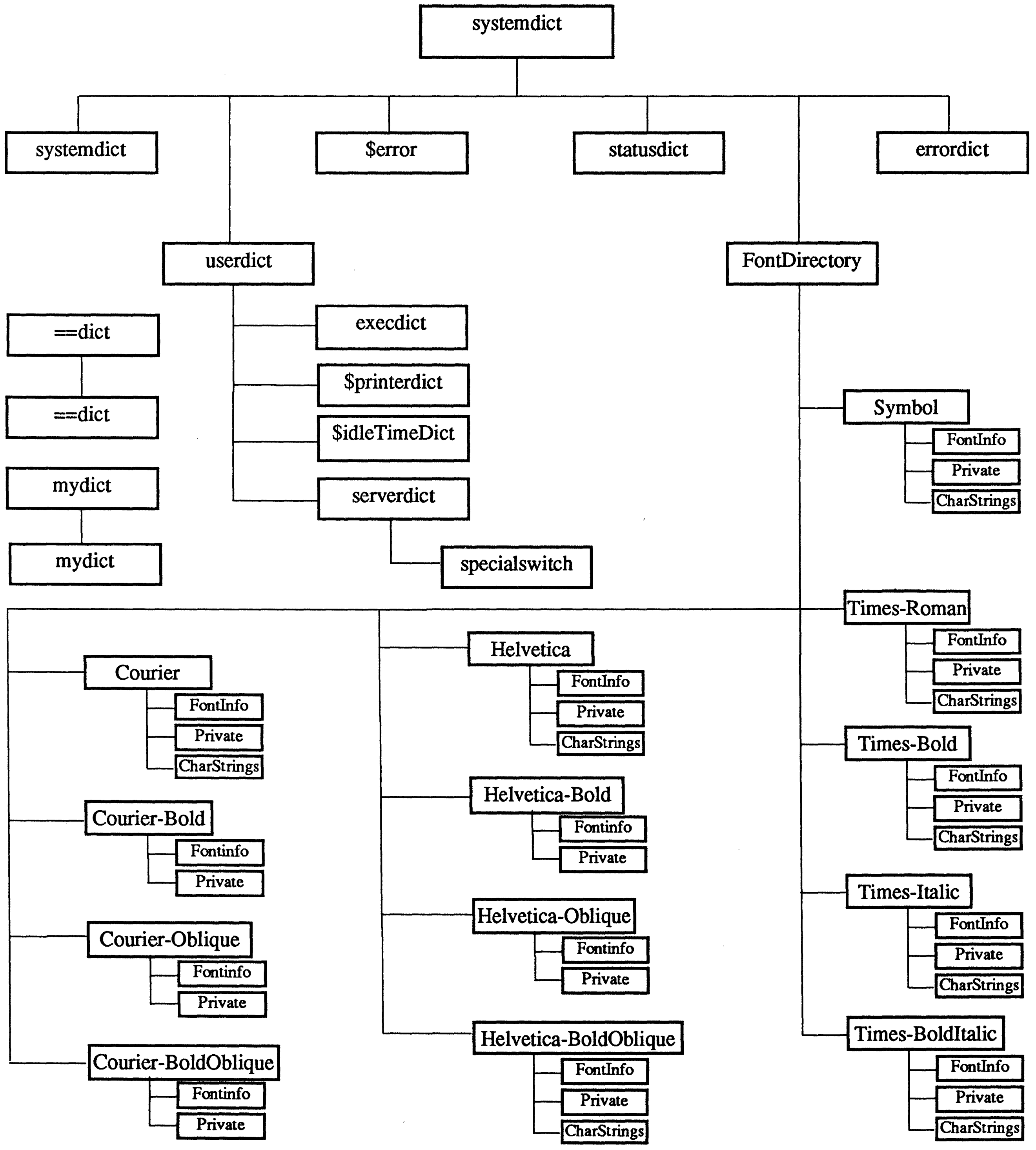


Table of Dictionaries

Product: QMS-PS 800 **Version: 38.0** **Revision: 0**

#	*Dictionary Name*	*Found In*	*Access Priviledge*	*Length*	*Max. Length*
0.	FontDirectory	systemdict	Accessible	13	250
1.	$error	systemdict	Accessible	11	13
2.	systemdict	systemdict	Accessible	246	256
3.	statusdict	systemdict	Accessible	52	65
4.	errordict	systemdict	Accessible	26	28
5.	userdict	systemdict	Accessible	17	200
6.	execdict	userdict	Accessible	6	6
7.	$printerdict	userdict	Accessible	13	20
8.	$idleTimeDict	userdict	Accessible	13	15
9.	serverdict	userdict	Accessible	48	50
10.	specialswitch	serverdict	Accessible	2	5
11.	mydict	mydict	Accessible	19	20
12.	==dict	==dict	Accessible	21	24
13.	Helvetica-Bold	FontDirectory	Accessible	11	11
14.	Courier-Oblique	FontDirectory	Accessible	12	12
15.	Courier-BoldOblique	FontDirectory	Accessible	12	12
16.	Courier-Bold	FontDirectory	Accessible	12	12
17.	Symbol	FontDirectory	Accessible	11	11
18.	Helvetica-BoldOblique	FontDirectory	Accessible	11	11
19.	Courier	FontDirectory	Accessible	12	12
20.	Times-Roman	FontDirectory	Accessible	11	11
21.	Times-BoldItalic	FontDirectory	Accessible	11	11
22.	Helvetica-Oblique	FontDirectory	Accessible	11	11
23.	Helvetica	FontDirectory	Accessible	11	11
24.	Times-Bold	FontDirectory	Accessible	11	11
25.	Times-Italic	FontDirectory	Accessible	11	11
26.	FontInfo	Times-Italic	Accessible	9	9
27.	Private	Times-Italic	No Access	n/a	n/a
28.	CharStrings	Times-Italic	Accessible	211	211
29.	FontInfo	Times-Bold	Accessible	9	9
30.	Private	Times-Bold	No Access	n/a	n/a
31.	CharStrings	Times-Bold	Accessible	211	211
32.	FontInfo	Helvetica	Accessible	9	9
33.	Private	Helvetica	No Access	n/a	n/a
34.	CharStrings	Helvetica	Accessible	211	211
35.	FontInfo	Helvetica-Oblique	Accessible	9	9
36.	Private	Helvetica-Oblique	No Access	n/a	n/a
37.	FontInfo	Times-BoldItalic	Accessible	9	9
38.	Private	Times-BoldItalic	No Access	n/a	n/a
39.	CharStrings	Times-BoldItalic	Accessible	211	211
40.	FontInfo	Times-Roman	Accessible	9	9

Product: QMS-PS 800 **Version: 38.0** **Revision: 0**

#	*Dictionary Name*	*Found In*	*Access Priviledge*	*Length*	*Max. Length*
41.	Private	Times-Roman	No Access	n/a	n/a
42.	CharStrings	Times-Roman	Accessible	211	211
43.	FontInfo	Courier	Accessible	8	8
44.	Private	Courier	No Access	n/a	n/a
45.	CharStrings	Courier	Accessible	199	199
46.	FontInfo	Helvetica-BoldOblique	Accessible	9	9
47.	Private	Helvetica-BoldOblique	No Access	n/a	n/a
48.	CharStrings	Helvetica-BoldOblique	Accessible	211	211
49.	FontInfo	Symbol	Accessible	8	8
50.	Private	Symbol	No Access	n/a	n/a
51.	CharStrings	Symbol	Accessible	190	190
52.	FontInfo	Courier-Bold	Accessible	8	8
53.	Private	Courier-Bold	No Access	n/a	n/a
54.	FontInfo	Courier-BoldOblique	Accessible	8	8
55.	Private	Courier-BoldOblique	No Access	n/a	n/a
56.	FontInfo	Courier-Oblique	Accessible	8	8
57.	Private	Courier-Oblique	No Access	n/a	n/a
58.	FontInfo	Helvetica-Bold	Accessible	9	9
59.	Private	Helvetica-Bold	No Access	n/a	n/a

Appendix II

Dictionary Key/Value Analysis

Introduction

This table contains an alphabetized list of every entry of every dictionary visible to the PostScript programmer. With this information, the contents of a dictionary can be studied at a glance.

How to Use the Table

The name of each dictionary is in a bold face next to a number which ties back into the "Dictionary Analysis" table in Appendix I. After the dictionary name are several lines of basic information about the dictionary followed by a list of each entry in the dictionary.

The entries are numbered and presented in alphabetical order. The data type of each entry is also listed. Entries followed by an "*" are documented by Adobe in the *PostScript Language Reference Manual.* Most of the documented entries are in the systemdict dictionary.

Structure of PostScript

The overall structure of PostScript can be studied from the display of the contents of each dictionary.

Most dictionaries contain groups of operators or executable arrays which address a specific aspect of the language. For example, "errordict" contains a list of every error procedure and "$idleTimeDict" contains all the procedures and variables involved in idle time font caching.

Another observation is that font dictionaries make up the bulk of the dictionaries. Only the first fourteen dictionaries deal with non-font related operations such as printer control and general PostScript language operations.

The systemdict contains almost all of the documented portion of the language with a few of them contained in userdict. Since systemdict and userdict are always on the dictionary stack, this means that no additional dictionaries are required for the user to have access to the documented part of the language.

Reading and writing to the EEPROM is handled by operators in the statusdict.

Most of the actual printer control is handled by operators and procedures in the serverdict and specialswitch dictionaries, with a few operations handled by entries in statusdict. See the Printer Control Group chapters for detailed information about entries in these dictionaries.

Key/Value Table

Product: QMS-PS 800 Version: 38.0 Revision: 0

0 FontDirectory
found in: systemdict
access priviledge: Accessible
length: 13
max length: 250

entry	type	name
1.	dictionary	Courier
2.	dictionary	Courier-Bold
3.	dictionary	Courier-BoldOblique
4.	dictionary	Courier-Oblique
5.	dictionary	Helvetica
6.	dictionary	Helvetica-Bold
7.	dictionary	Helvetica-BoldOblique
8.	dictionary	Helvetica-Oblique
9.	dictionary	Symbol
10.	dictionary	Times-Bold
11.	dictionary	Times-BoldItalic
12.	dictionary	Times-Italic
13.	dictionary	Times-Roman

1 $error
found in: systemdict
access priviledge: Accessible
length: 11
max length: 13

entry	type	name
1.	operator	command
2.	boolean	doclose
3.	array	dstack
4.	array	dstackarray
5.	name	errorname
6.	array	estack
7.	array	estackarray
8.	boolean	initializing
9.	boolean	newerror
10.	array	ostack
11.	array	ostackarray

2 systemdict
found in: systemdict
access priviledge: Accessible
length: 246
max length: 256

entry	type	name
1.	dictionary	$error
2.	packed array	.error
3.	packed array	- *
4.	packed array	-- *
5.	packed array	-print
6.	string	-string
7.	dictionary	FontDirectory *
8.	packed array	Run
9.	array	StandardEncoding *
10.	operator	[*
11.	operator	] *
12.	operator	abs *
13.	operator	add *
14.	operator	aload *
15.	operator	anchorsearch *
16.	operator	and *
17.	operator	arc *
18.	operator	arcn *
19.	operator	arcto *
20.	operator	array *
21.	operator	ashow *
22.	operator	astore *
23.	operator	atan *
24.	operator	awidthshow *
25.	operator	begin *
26.	operator	bind *
27.	operator	bitshift *
28.	operator	bytesavailable *
29.	operator	cachestatus *
30.	operator	ceiling *
31.	operator	cexec
32.	operator	charpath *
33.	operator	clear *
34.	operator	clearinterrupt
35.	operator	cleartomark *
36.	operator	clip *
37.	operator	clippath *
38.	operator	closefile *
39.	operator	closepath *
40.	operator	concat *
41.	operator	concatmatrix *
42.	operator	copy *
43.	operator	copypage *
44.	operator	cos *
45.	operator	count *
46.	operator	countdictstack *
47.	operator	countexecstack *
48.	operator	counttomark *
49.	operator	currentcacheparams
50.	operator	currentdash *
51.	operator	currentdict *
52.	operator	currentfile *
53.	operator	currentflat *
54.	operator	currentfont *
55.	operator	currentgray *
56.	operator	currenthsbcolor *
57.	operator	currentlinecap *
58.	operator	currentlinejoin *
59.	operator	currentlinewidth *
60.	operator	currentmatrix *
61.	operator	currentmiterlimit *
62.	operator	currentpacking
63.	operator	currentpoint *
64.	operator	currentrgbcolor *
65.	operator	currentscreen *
66.	operator	currenttransfer *
67.	operator	curveto *
68.	operator	cvi *
69.	operator	cvlit *
70.	operator	cvn *
71.	operator	cvr *
72.	operator	cvrs *
73.	operator	cvs *
74.	operator	cvx *
75.	operator	daytime
76.	operator	def *
77.	operator	defaultmatrix *
78.	operator	definefont *
79.	operator	dict *
80.	operator	dictstack *
81.	operator	disableinterrupt
82.	operator	div *
83.	operator	dtransform *
84.	operator	dup *
85.	operator	echo *
86.	operator	eexec
87.	operator	enableinterrupt

Product: QMS-PS 800 Version: 38.0 Revision: 0

entry	*type*	*name*	*entry*	*type*	*name*
88.	operator	end *	161.	operator	pathbbox *
89.	operator	eoclip *	162.	operator	pathforall *
90.	operator	eofill *	163.	operator	pop *
91.	operator	eq *	164.	operator	print *
92.	operator	erasepage *	165.	operator	psdevice
93.	dictionary	errordict *	166.	operator	put *
94.	operator	exch *	167.	operator	putinterval *
95.	operator	exec *	168.	operator	quit *
96.	operator	execstack *	169.	operator	rand *
97.	operator	executeonly *	170.	operator	rcheck *
98.	operator	exit *	171.	operator	rcurveto *
99.	operator	exp *	172.	operator	read *
100.	boolean	false *	173.	operator	readhexstring *
101.	operator	file *	174.	operator	readline *
102.	operator	fill *	175.	operator	readonly *
103.	packed array	findfont *	176.	operator	readstring *
104.	operator	flattenpath *	177.	operator	repeat *
105.	operator	floor *	178.	operator	resetfile *
106.	operator	flush *	179.	operator	restore *
107.	operator	flushfile *	180.	operator	reversepath *
108.	operator	for *	181.	operator	rlineto *
109.	operator	forall *	182.	operator	rmoveto *
110.	operator	framedevice *	183.	operator	roll *
111.	operator	ge *	184.	operator	rotate *
112.	operator	get *	185.	operator	round *
113.	operator	getinterval *	186.	operator	rrand *
114.	operator	grestore *	187.	operator	run *
115.	operator	grestoreall *	188.	operator	save *
116.	operator	gsave *	189.	operator	scale *
117.	operator	gt *	190.	operator	scalefont *
118.	packed array	handleerror	191.	operator	search *
119.	operator	identmatrix *	192.	operator	setcachedevice *
120.	operator	idiv *	193.	operator	setcachelimit *
121.	operator	idtransform *	194.	operator	setcacheparams
122.	operator	if *	195.	operator	setcharwidth *
123.	operator	ifelse *	196.	operator	setdash *
124.	operator	image *	197.	operator	setflat *
125.	operator	imagemask *	198.	operator	setfont *
126.	operator	index *	199.	operator	setgray *
127.	operator	initclip *	200.	operator	sethsbcolor *
128.	operator	initgraphics *	201.	operator	setlinecap *
129.	integer	initialized	202.	operator	setlinejoin *
130.	operator	initmatrix *	203.	operator	setlinewidth *
131.	operator	internaldict	204.	operator	setmatrix *
132.	operator	invertmatrix *	205.	operator	setmiterlimit *
133.	operator	itransform *	206.	operator	setpacking
134.	operator	known *	207.	operator	setram
135.	operator	kshow *	208.	operator	setrgbcolor *
136.	operator	le *	209.	operator	setrom
137.	operator	length *	210.	operator	setscreen *
138.	operator	lineto *	211.	operator	settransfer *
139.	operator	ln *	212.	operator	show *
140.	operator	load *	213.	operator	showpage *
141.	operator	log *	214.	operator	sin *
142.	operator	loop *	215.	operator	sqrt *
143.	operator	lt *	216.	operator	srand *
144.	operator	makefont *	217.	packed array	stack *
145.	operator	makevm	218.	operator	status *
146.	operator	mark *	219.	dictionary	statusdict
147.	operator	matrix *	220.	operator	stop *
148.	operator	maxlength *	221.	operator	stopped *
149.	operator	mod *	222.	operator	store *
150.	operator	moveto *	223.	operator	string *
151.	operator	mul *	224.	operator	stringwidth *
152.	operator	ne *	225.	operator	stroke *
153.	operator	neg *	226.	operator	strokepath *
154.	operator	newpath *	227.	operator	sub *
155.	operator	noaccess *	228.	dictionary	systemdict *
156.	operator	not *	229.	operator	token *
157.	null	null *	230.	operator	transform *
158.	operator	nulldevice *	231.	operator	translate *
159.	operator	or *	232.	boolean	true *
160.	operator	packedarray	233.	operator	truncate *

Product: QMS-PS 800 Version: 38.0 Revision: 0

entry	type	name
234.	operator	type *
235.	dictionary	userdict *
236.	operator	usertime *
237.	string	version *
238.	operator	vmstatus *
239.	operator	wcheck *
240.	operator	where *
241.	operator	widthshow *
242.	operator	write *
243.	operator	writehexstring *
244.	operator	writestring *
245.	operator	xcheck *
246.	operator	xor *

3 statusdict
found in: systemdict
access priviledge: Accessible
length: 52
max length: 65

entry	type	name
1.	string	appletalktype
2.	operator	blink
3.	operator	checkpassword
4.	operator	closescc
5.	integer	debugmode
6.	operator	defaulttimeouts
7.	operator	diablo
8.	operator	dostartpage
9.	boolean	eerom
10.	operator	eescratch
11.	operator	idlefonts
12.	operator	initappletalk
13.	null	jobname
14.	string	jobsource
15.	string	jobstate
16.	operator	jobtimeout
17.	boolean	manualfeed
18.	integer	manualfeedtimeout
19.	operator	margins
20.	operator	openappletalk
21.	operator	openscc
22.	operator	pagecount
23.	operator	pagestackorder
24.	operator	pagetype
25.	packed array	printererror
26.	operator	printername
27.	operator	printerstatus
28.	string	product
29.	operator	redclose
30.	operator	redwrite
31.	operator	resetprinter
32.	integer	revision
33.	operator	sccbatch
34.	operator	sccfiles
35.	operator	sccinteractive
36.	operator	sendpcmd
37.	operator	setappletalkname
38.	operator	setblink
39.	operator	setdefaulttimeouts
40.	operator	setdostartpage
41.	operator	seteescratch
42.	packed array	setidlefonts
43.	operator	setjobtimeout
44.	operator	setmargins
45.	operator	setpagetype
46.	operator	setpassword
47.	operator	setprintername
48.	operator	setsccbatch
49.	operator	setsccinteractive
50.	operator	setstdio
51.	operator	switchsetting
52.	integer	waittimeout

4 errordict
found in: systemdict
access priviledge: Accessible
length: 26
max length: 28

entry	type	name
1.	packed array	VMerror
2.	packed array	dictfull
3.	packed array	dictstackoverflow
4.	packed array	dictstackunderflow
5.	packed array	execstackoverflow
6.	packed array	handleerror
7.	operator	interrupt
8.	packed array	invalidaccess
9.	packed array	invalidexit
10.	packed array	invalidfileaccess
11.	packed array	invalidfont
12.	packed array	invalidrestore
13.	packed array	ioerror
14.	packed array	limitcheck
15.	packed array	nocurrentpoint
16.	packed array	rangecheck
17.	packed array	stackoverflow
18.	packed array	stackunderflow
19.	packed array	syntaxerror
20.	packed array	timeout
21.	packed array	typecheck
22.	packed array	undefined
23.	packed array	undefinedfilename
24.	packed array	undefinedresult
25.	packed array	unmatchedmark
26.	packed array	unregistered

5 userdict
found in: systemdict
access priviledge: Accessible
length: 17
max length: 200

entry	type	name
1.	integer	#copies
2.	dictionary	$idleTimeDict
3.	dictionary	$printerdict
4.	packed array	ReadIdleFonts
5.	packed array	UseIdleTime
6.	packed array	a4
7.	packed array	b5
8.	packed array	cleardictstack
9.	dictionary	execdict
10.	packed array	executive
11.	packed array	legal
12.	packed array	letter
13.	packed array	prompt *
14.	packed array	pstack *
15.	packed array	quit
16.	dictionary	serverdict
17.	packed array	start *

Product: QMS-PS 800 Version: 38.0 Revision: 0

6 execdict
found in: userdict
access priviledge: Accessible
length: 6
max length: 6

entry	type	name
1.	packed array	checkquit
2.	integer	execdepth
3.	packed array	idleproc
4.	operator	print
5.	boolean	quitflag
6.	file	stmtfile

7 $printerdict
found in: userdict
access priviledge: Accessible
length: 13
max length: 20

entry	type	name
1.	packed array	0
2.	packed array	18
3.	packed array	2
4.	packed array	24
5.	packed array	8
6.	packed array	dopage
7.	integer	height
8.	array	mtx
9.	packed array	proc
10.	packed array	setpage
11.	integer	width
12.	integer	xoffset
13.	integer	yoffset

8 $idleTimeDict
found in: userdict
access priviledge: Accessible
length: 13
max length: 15

entry	type	name
1.	array	ROMnames
2.	string	ascii26
3.	string	ascii62
4.	string	ascii81
5.	string	ascii94
6.	packed array	bmpI
7.	packed array	boundsCheck
8.	packed array	idlA
9.	array	idleArry
10.	integer	idleI
11.	string	idleStr
12.	integer	idleStrI
13.	packed array	stopPred

9 serverdict
found in: userdict
access priviledge: Accessible
length: 48
max length: 50

entry	type	name
1.	packed array	0
2.	packed array	1
3.	packed array	2
4.	packed array	3
5.	boolean	altflag
6.	file	altin
7.	string	altname
8.	file	altout
9.	packed array	altprint
10.	packed array	appletalkclose
11.	packed array	appletalkopen
12.	packed array	batchidleproc
13.	integer	baud25
14.	integer	baud9
15.	integer	commhash
16.	packed array	dexch
17.	packed array	exchdef
18.	packed array	execjob
19.	array	exitserver
20.	packed array	fontname
21.	packed array	hashcommparams
22.	packed array	initprinter
23.	packed array	intidleproc
24.	integer	parity25
25.	integer	parity9
26.	packed array	printerstatus
27.	packed array	protect
28.	integer	saveswitch
29.	boolean	sccok
30.	integer	secretdict
31.	boolean	sendctrld
32.	array	server
33.	packed array	setnulldevice
34.	packed array	setrealdevice
35.	packed array	setsccstreams
36.	packed array	setstreams
37.	packed array	settimeouts
38.	dictionary	specialswitch
39.	string	startpage
40.	file	stdin
41.	string	stdname
42.	file	stdout
43.	array	switchclose
44.	array	switchopen
45.	boolean	transparent
46.	packed array	warmedup
47.	packed array	watchstreams
48.	integer	wtimeout

10 specialswitch
found in: serverdict
access priviledge: Accessible
length: 2
max length: 5

entry	type	name
1.	packed array	0
2.	packed array	1

11 mydict
found in: 12
access priviledge: Accessible
length: 19
max length: 20

entry	type	name
1.	string	0
2.	string	2048
3.	packed array	4616
4.	string	512
5.	string	520
6.	packed array	528
7.	string	536

Product: QMS-PS 800 Version: 38.0 Revision: 0

entry	type	name
8.	string	576
9.	boolean	abort
10.	integer	bits
11.	packed array	eflush
12.	integer	eindex
13.	packed array	eprint
14.	string	errstr
15.	integer	laststat
16.	dictionary	mydict
17.	integer	ntrys
18.	boolean	report
19.	integer	stat

12 ==dict
found in: 13
access priviledge: Accessible
length: 21
max length: 24

entry	type	name
1.	dictionary	--dict
2.	string	NL
3.	packed array	arraytype
4.	packed array	booleantype
5.	integer	cp
6.	packed array	cvsprint
7.	packed array	dicttype
8.	packed array	filetype
9.	packed array	fonttype
10.	packed array	integertype
11.	packed array	marktype
12.	packed array	nametype
13.	packed array	nulltype
14.	packed array	operatortype
15.	packed array	packedarraytype
16.	packed array	realtype
17.	integer	rmargin
18.	packed array	savetype
19.	packed array	stringtype
20.	packed array	tprint
21.	packed array	typeprint

13 Helvetica-Bold
found in: FontDirectory
access priviledge: Accessible
length: 11
max length: 11

entry	type	name
1.	dictionary	CharStrings
2.	array	Encoding
3.	font	FID
4.	packed array	FontBBox
5.	dictionary	FontInfo
6.	array	FontMatrix
7.	name	FontName
8.	integer	FontType
9.	integer	PaintType
10.	dictionary	Private
11.	integer	UniqueID

14 Courier-Oblique
found in: FontDirectory
access priviledge: Accessible
length: 12
max length: 12

entry	type	name
1.	dictionary	CharStrings
2.	array	Encoding
3.	font	FID
4.	packed array	FontBBox
5.	dictionary	FontInfo
6.	array	FontMatrix
7.	name	FontName
8.	integer	FontType
9.	integer	PaintType
10.	dictionary	Private
11.	integer	StrokeWidth
12.	integer	UniqueID

15 Courier-BoldOblique
found in: FontDirectory
access priviledge: Accessible
length: 12
max length: 12

entry	type	name
1.	dictionary	CharStrings
2.	array	Encoding
3.	font	FID
4.	packed array	FontBBox
5.	dictionary	FontInfo
6.	array	FontMatrix
7.	name	FontName
8.	integer	FontType
9.	integer	PaintType
10.	dictionary	Private
11.	integer	StrokeWidth
12.	integer	UniqueID

16 Courier-Bold
found in: FontDirectory
access priviledge: Accessible
length: 12
max length: 12

entry	type	name
1.	dictionary	CharStrings
2.	array	Encoding
3.	font	FID
4.	packed array	FontBBox
5.	dictionary	FontInfo
6.	array	FontMatrix
7.	name	FontName
8.	integer	FontType
9.	integer	PaintType
10.	dictionary	Private
11.	integer	StrokeWidth
12.	integer	UniqueID

Product: QMS-PS 800 Version: 38.0 Revision: 0

17 Symbol
found in: FontDirectory
access priviledge: Accessible
length: 11
max length: 11

entry	type	name
1.	dictionary	CharStrings
2.	array	Encoding
3.	font	FID
4.	packed array	FontBBox
5.	dictionary	FontInfo
6.	array	FontMatrix
7.	name	FontName
8.	integer	FontType
9.	integer	PaintType
10.	dictionary	Private
11.	integer	UniqueID

18 Helvetica-BoldOblique
found in: FontDirectory
access priviledge: Accessible
length: 11
max length: 11

entry	type	name
1.	dictionary	CharStrings
2.	array	Encoding
3.	font	FID
4.	packed array	FontBBox
5.	dictionary	FontInfo
6.	array	FontMatrix
7.	name	FontName
8.	integer	FontType
9.	integer	PaintType
10.	dictionary	Private
11.	integer	UniqueID

19 Courier
found in: FontDirectory
access priviledge: Accessible
length: 12
max length: 12

entry	type	name
1.	dictionary	CharStrings
2.	array	Encoding
3.	font	FID
4.	packed array	FontBBox
5.	dictionary	FontInfo
6.	array	FontMatrix
7.	name	FontName
8.	integer	FontType
9.	integer	PaintType
10.	dictionary	Private
11.	integer	StrokeWidth
12.	integer	UniqueID

20 Times-Roman
found in: FontDirectory
access priviledge: Accessible
length: 11
max length: 11

entry	type	name
1.	dictionary	CharStrings
2.	array	Encoding
3.	font	FID
4.	packed array	FontBBox
5.	dictionary	FontInfo
6.	array	FontMatrix
7.	name	FontName
8.	integer	FontType
9.	integer	PaintType
10.	dictionary	Private
11.	integer	UniqueID

21 Times-BoldItalic
found in: FontDirectory
access priviledge: Accessible
length: 11
max length: 11

entry	type	name
1.	dictionary	CharStrings
2.	array	Encoding
3.	font	FID
4.	packed array	FontBBox
5.	dictionary	FontInfo
6.	array	FontMatrix
7.	name	FontName
8.	integer	FontType
9.	integer	PaintType
10.	dictionary	Private
11.	integer	UniqueID

22 Helvetica-Oblique
found in: FontDirectory
access priviledge: Accessible
length: 11
max length: 11

entry	type	name
1.	dictionary	CharStrings
2.	array	Encoding
3.	font	FID
4.	packed array	FontBBox
5.	dictionary	FontInfo
6.	array	FontMatrix
7.	name	FontName
8.	integer	FontType
9.	integer	PaintType
10.	dictionary	Private
11.	integer	UniqueID

23 Helvetica
found in: FontDirectory
access priviledge: Accessible
length: 11
max length: 11

entry	type	name
1.	dictionary	CharStrings
2.	array	Encoding
3.	font	FID
4.	packed array	FontBBox
5.	dictionary	FontInfo
6.	array	FontMatrix
7.	name	FontName
8.	integer	FontType
9.	integer	PaintType
10.	dictionary	Private
11.	integer	UniqueID

Product: QMS-PS 800 Version: 38.0 Revision: 0

24 Times-Bold
found in: FontDirectory
access priviledge: Accessible
length: 11
max length: 11

entry	type	name
1.	dictionary	CharStrings
2.	array	Encoding
3.	font	FID
4.	packed array	FontBBox
5.	dictionary	FontInfo
6.	array	FontMatrix
7.	name	FontName
8.	integer	FontType
9.	integer	PaintType
10.	dictionary	Private
11.	integer	UniqueID

25 Times-Italic
found in: FontDirectory
access priviledge: Accessible
length: 11
max length: 11

entry	type	name
1.	dictionary	CharStrings
2.	array	Encoding
3.	font	FID
4.	packed array	FontBBox
5.	dictionary	FontInfo
6.	array	FontMatrix
7.	name	FontName
8.	integer	FontType
9.	integer	PaintType
10.	dictionary	Private
11.	integer	UniqueID

26 FontInfo
found in: Times-Italic
access priviledge: Accessible
length: 9
max length: 9

entry	type	name
1.	string	FamilyName
2.	string	FullName
3.	real	ItalicAngle
4.	string	Notice
5.	integer	UnderlinePosition
6.	integer	UnderlineThickness
7.	string	Weight
8.	boolean	isFixedPitch
9.	string	version

27 Private
found in: Times-Italic
access priviledge: No Access

28 CharStrings
found in: Times-Italic
access priviledge: Accessible
length: 211
max length: 211

entry	type	name
1.	string	.notdef
2.	string	A
3.	string	AE
4.	string	Aacute
5.	string	Acircumflex
6.	string	Adieresis
7.	string	Agrave
8.	string	Aring
9.	string	Atilde
10.	string	B
11.	string	C
12.	string	Ccedilla
13.	string	D
14.	string	E
15.	string	Eacute
16.	string	Ecircumflex
17.	string	Edieresis
18.	string	Egrave
19.	string	F
20.	string	G
21.	string	H
22.	string	I
23.	string	Iacute
24.	string	Icircumflex
25.	string	Idieresis
26.	string	Igrave
27.	string	J
28.	string	K
29.	string	L
30.	string	Lslash
31.	string	M
32.	string	N
33.	string	Ntilde
34.	string	O
35.	string	OE
36.	string	Oacute
37.	string	Ocircumflex
38.	string	Odieresis
39.	string	Ograve
40.	string	Oslash
41.	string	Otilde
42.	string	P
43.	string	Q
44.	string	R
45.	string	S
46.	string	Scaron
47.	string	T
48.	string	U
49.	string	Uacute
50.	string	Ucircumflex
51.	string	Udieresis
52.	string	Ugrave
53.	string	V
54.	string	W
55.	string	X
56.	string	Y
57.	string	Ydieresis
58.	string	Z
59.	string	Zcaron
60.	string	a
61.	string	aacute
62.	string	acircumflex
63.	string	acute
64.	string	adieresis
65.	string	ae

Product: QMS-PS 800 Version: 38.0 Revision: 0

entry	*type*	*name*	*entry*	*type*	*name*
66.	string	agrave	139.	string	logicalnot
67.	string	ampersand	140.	string	lslash
68.	string	aring	141.	string	m
69.	string	asciicircum	142.	string	macron
70.	string	asciitilde	143.	string	minus
71.	string	asterisk	144.	string	n
72.	string	at	145.	string	nine
73.	string	atilde	146.	string	ntilde
74.	string	b	147.	string	numbersign
75.	string	backslash	148.	string	o
76.	string	bar	149.	string	oacute
77.	string	braceleft	150.	string	ocircumflex
78.	string	braceright	151.	string	odieresis
79.	string	bracketleft	152.	string	oe
80.	string	bracketright	153.	string	ogonek
81.	string	breve	154.	string	ograve
82.	string	bullet	155.	string	one
83.	string	c	156.	string	ordfeminine
84.	string	caron	157.	string	ordmasculine
85.	string	ccedilla	158.	string	oslash
86.	string	cedilla	159.	string	otilde
87.	string	cent	160.	string	p
88.	string	circumflex	161.	string	paragraph
89.	string	colon	162.	string	parenleft
90.	string	comma	163.	string	parenright
91.	string	copyright	164.	string	percent
92.	string	currency	165.	string	period
93.	string	d	166.	string	periodcentered
94.	string	dagger	167.	string	perthousand
95.	string	daggerdbl	168.	string	plus
96.	string	dieresis	169.	string	q
97.	string	dollar	170.	string	question
98.	string	dotaccent	171.	string	questiondown
99.	string	dotlessi	172.	string	quotedbl
100.	string	e	173.	string	quotedblbase
101.	string	eacute	174.	string	quotedblleft
102.	string	ecircumflex	175.	string	quotedblright
103.	string	edieresis	176.	string	quoteleft
104.	string	egrave	177.	string	quoteright
105.	string	eight	178.	string	quotesinglbase
106.	string	ellipsis	179.	string	quotesingle
107.	string	emdash	180.	string	r
108.	string	endash	181.	string	registered
109.	string	equal	182.	string	ring
110.	string	exclam	183.	string	s
111.	string	exclamdown	184.	string	scaron
112.	string	f	185.	string	section
113.	string	fi	186.	string	semicolon
114.	string	five	187.	string	seven
115.	string	fl	188.	string	six
116.	string	florin	189.	string	slash
117.	string	four	190.	string	space
118.	string	fraction	191.	string	sterling
119.	string	g	192.	string	t
120.	string	germandbls	193.	string	three
121.	string	grave	194.	string	tilde
122.	string	greater	195.	string	trademark
123.	string	guillemotleft	196.	string	two
124.	string	guillemotright	197.	string	u
125.	string	guilsinglleft	198.	string	uacute
126.	string	guilsinglright	199.	string	ucircumflex
127.	string	h	200.	string	udieresis
128.	string	hungarumlaut	201.	string	ugrave
129.	string	hyphen	202.	string	underscore
130.	string	i	203.	string	v
131.	string	iacute	204.	string	w
132.	string	icircumflex	205.	string	x
133.	string	idieresis	206.	string	y
134.	string	igrave	207.	string	ydieresis
135.	string	j	208.	string	yen
136.	string	k	209.	string	z
137.	string	l	210.	string	zcaron
138.	string	less	211.	string	zero

Product: QMS-PS 800 Version: 38.0 Revision: 0

29 FontInfo
found in: Times-Bold
access priviledge: Accessible
length: 9
max length: 9

entry	type	name
1.	string	FamilyName
2.	string	FullName
3.	integer	ItalicAngle
4.	string	Notice
5.	integer	UnderlinePosition
6.	integer	UnderlineThickness
7.	string	Weight
8.	boolean	isFixedPitch
9.	string	version

30 Private
found in: Times-Bold
access priviledge: No Access

31 CharStrings
found in: Times-Bold
access priviledge: Accessible
length: 211
max length: 211

entry	type	name
1.	string	.notdef
2.	string	A
3.	string	AE
4.	string	Aacute
5.	string	Acircumflex
6.	string	Adieresis
7.	string	Agrave
8.	string	Aring
9.	string	Atilde
10.	string	B
11.	string	C
12.	string	Ccedilla
13.	string	D
14.	string	E
15.	string	Eacute
16.	string	Ecircumflex
17.	string	Edieresis
18.	string	Egrave
19.	string	F
20.	string	G
21.	string	H
22.	string	I
23.	string	Iacute
24.	string	Icircumflex
25.	string	Idieresis
26.	string	Igrave
27.	string	J
28.	string	K
29.	string	L
30.	string	Lslash
31.	string	M
32.	string	N
33.	string	Ntilde
34.	string	O
35.	string	OE
36.	string	Oacute
37.	string	Ocircumflex
38.	string	Odieresis
39.	string	Ograve
40.	string	Oslash
41.	string	Otilde
42.	string	P
43.	string	Q
44.	string	R
45.	string	S
46.	string	Scaron
47.	string	T
48.	string	U
49.	string	Uacute
50.	string	Ucircumflex
51.	string	Udieresis
52.	string	Ugrave
53.	string	V
54.	string	W
55.	string	X
56.	string	Y
57.	string	Ydieresis
58.	string	Z
59.	string	Zcaron
60.	string	a
61.	string	aacute
62.	string	acircumflex
63.	string	acute
64.	string	adieresis
65.	string	ae
66.	string	agrave
67.	string	ampersand
68.	string	aring
69.	string	asciicircum
70.	string	asciitilde
71.	string	asterisk
72.	string	at
73.	string	atilde
74.	string	b
75.	string	backslash
76.	string	bar
77.	string	braceleft
78.	string	braceright
79.	string	bracketleft
80.	string	bracketright
81.	string	breve
82.	string	bullet
83.	string	c
84.	string	caron
85.	string	ccedilla
86.	string	cedilla
87.	string	cent
88.	string	circumflex
89.	string	colon
90.	string	comma
91.	string	copyright
92.	string	currency
93.	string	d
94.	string	dagger
95.	string	daggerdbl
96.	string	dieresis
97.	string	dollar
98.	string	dotaccent
99.	string	dotlessi
100.	string	e
101.	string	eacute
102.	string	ecircumflex
103.	string	edieresis
104.	string	egrave
105.	string	eight
106.	string	ellipsis
107.	string	emdash
108.	string	endash
109.	string	equal
110.	string	exclam
111.	string	exclamdown
112.	string	f
113.	string	fi

Product: QMS-PS 800 Version: 38.0 Revision: 0

entry	*type*	*name*
114.	string	five
115.	string	fl
116.	string	florin
117.	string	four
118.	string	fraction
119.	string	g
120.	string	germandbls
121.	string	grave
122.	string	greater
123.	string	guillemotleft
124.	string	guillemotright
125.	string	guilsinglleft
126.	string	guilsinglright
127.	string	h
128.	string	hungarumlaut
129.	string	hyphen
130.	string	i
131.	string	iacute
132.	string	icircumflex
133.	string	idieresis
134.	string	igrave
135.	string	j
136.	string	k
137.	string	l
138.	string	less
139.	string	logicalnot
140.	string	lslash
141.	string	m
142.	string	macron
143.	string	minus
144.	string	n
145.	string	nine
146.	string	ntilde
147.	string	numbersign
148.	string	o
149.	string	oacute
150.	string	ocircumflex
151.	string	odieresis
152.	string	oe
153.	string	ogonek
154.	string	ograve
155.	string	one
156.	string	ordfeminine
157.	string	ordmasculine
158.	string	oslash
159.	string	otilde
160.	string	p
161.	string	paragraph
162.	string	parenleft
163.	string	parenright
164.	string	percent
165.	string	period
166.	string	periodcentered
167.	string	perthousand
168.	string	plus
169.	string	q
170.	string	question
171.	string	questiondown
172.	string	quotedbl
173.	string	quotedblbase
174.	string	quotedblleft
175.	string	quotedblright
176.	string	quoteleft
177.	string	quoteright
178.	string	quotesinglbase
179.	string	quotesingle
180.	string	r
181.	string	registered
182.	string	ring
183.	string	s
184.	string	scaron
185.	string	section
186.	string	semicolon
187.	string	seven
188.	string	six
189.	string	slash
190.	string	space
191.	string	sterling
192.	string	t
193.	string	three
194.	string	tilde
195.	string	trademark
196.	string	two
197.	string	u
198.	string	uacute
199.	string	ucircumflex
200.	string	udieresis
201.	string	ugrave
202.	string	underscore
203.	string	v
204.	string	w
205.	string	x
206.	string	y
207.	string	ydieresis
208.	string	yen
209.	string	z
210.	string	zcaron
211.	string	zero

32 FontInfo
found in: Helvetica
access priviledge: Accessible
length: 9
max length: 9

1.	string	FamilyName
2.	string	FullName
3.	integer	ItalicAngle
4.	string	Notice
5.	integer	UnderlinePosition
6.	integer	UnderlineThickness
7.	string	Weight
8.	boolean	isFixedPitch
9.	string	version

33 Private
found in: Helvetica
access priviledge: No Access

34 CharStrings
found in: Helvetica
access priviledge: Accessible
length: 211
max length: 211

1.	string	.notdef
2.	string	A
3.	string	AE
4.	string	Aacute
5.	string	Acircumflex
6.	string	Adieresis
7.	string	Agrave
8.	string	Aring
9.	string	Atilde
10.	string	B
11.	string	C
12.	string	Ccedilla
13.	string	D

Product: QMS-PS 800 Version: 38.0 Revision: 0

entry	*type*	*name*
14.	string	E
15.	string	Eacute
16.	string	Ecircumflex
17.	string	Edieresis
18.	string	Egrave
19.	string	F
20.	string	G
21.	string	H
22.	string	I
23.	string	Iacute
24.	string	Icircumflex
25.	string	Idieresis
26.	string	Igrave
27.	string	J
28.	string	K
29.	string	L
30.	string	Lslash
31.	string	M
32.	string	N
33.	string	Ntilde
34.	string	O
35.	string	OE
36.	string	Oacute
37.	string	Ocircumflex
38.	string	Odieresis
39.	string	Ograve
40.	string	Oslash
41.	string	Otilde
42.	string	P
43.	string	Q
44.	string	R
45.	string	S
46.	string	Scaron
47.	string	T
48.	string	U
49.	string	Uacute
50.	string	Ucircumflex
51.	string	Udieresis
52.	string	Ugrave
53.	string	V
54.	string	W
55.	string	X
56.	string	Y
57.	string	Ydieresis
58.	string	Z
59.	string	Zcaron
60.	string	a
61.	string	aacute
62.	string	acircumflex
63.	string	acute
64.	string	adieresis
65.	string	ae
66.	string	agrave
67.	string	ampersand
68.	string	aring
69.	string	asciicircum
70.	string	asciitilde
71.	string	asterisk
72.	string	at
73.	string	atilde
74.	string	b
75.	string	backslash
76.	string	bar
77.	string	braceleft
78.	string	braceright
79.	string	bracketleft
80.	string	bracketright
81.	string	breve
82.	string	bullet
83.	string	c
84.	string	caron
85.	string	ccedilla
86.	string	cedilla
87.	string	cent
88.	string	circumflex
89.	string	colon
90.	string	comma
91.	string	copyright
92.	string	currency
93.	string	d
94.	string	dagger
95.	string	daggerdbl
96.	string	dieresis
97.	string	dollar
98.	string	dotaccent
99.	string	dotlessi
100.	string	e
101.	string	eacute
102.	string	ecircumflex
103.	string	edieresis
104.	string	egrave
105.	string	eight
106.	string	ellipsis
107.	string	emdash
108.	string	endash
109.	string	equal
110.	string	exclam
111.	string	exclamdown
112.	string	f
113.	string	fi
114.	string	five
115.	string	fl
116.	string	florin
117.	string	four
118.	string	fraction
119.	string	g
120.	string	germandbls
121.	string	grave
122.	string	greater
123.	string	guillemotleft
124.	string	guillemotright
125.	string	guilsinglleft
126.	string	guilsinglright
127.	string	h
128.	string	hungarumlaut
129.	string	hyphen
130.	string	i
131.	string	iacute
132.	string	icircumflex
133.	string	idieresis
134.	string	igrave
135.	string	j
136.	string	k
137.	string	l
138.	string	less
139.	string	logicalnot
140.	string	lslash
141.	string	m
142.	string	macron
143.	string	minus
144.	string	n
145.	string	nine
146.	string	ntilde
147.	string	numbersign
148.	string	o
149.	string	oacute
150.	string	ocircumflex
151.	string	odieresis
152.	string	oe
153.	string	ogonek
154.	string	ograve
155.	string	one
156.	string	ordfeminine
157.	string	ordmasculine
158.	string	oslash
159.	string	otilde

Product: QMS-PS 800 Version: 38.0 Revision: 0

entry	*type*	*name*
160.	string	p
161.	string	paragraph
162.	string	parenleft
163.	string	parenright
164.	string	percent
165.	string	period
166.	string	periodcentered
167.	string	perthousand
168.	string	plus
169.	string	q
170.	string	question
171.	string	questiondown
172.	string	quotedbl
173.	string	quotedblbase
174.	string	quotedblleft
175.	string	quotedblright
176.	string	quoteleft
177.	string	quoteright
178.	string	quotesinglbase
179.	string	quotesingle
180.	string	r
181.	string	registered
182.	string	ring
183.	string	s
184.	string	scaron
185.	string	section
186.	string	semicolon
187.	string	seven
188.	string	six
189.	string	slash
190.	string	space
191.	string	sterling
192.	string	t
193.	string	three
194.	string	tilde
195.	string	trademark
196.	string	two
197.	string	u
198.	string	uacute
199.	string	ucircumflex
200.	string	udieresis
201.	string	ugrave
202.	string	underscore
203.	string	v
204.	string	w
205.	string	x
206.	string	y
207.	string	ydieresis
208.	string	yen
209.	string	z
210.	string	zcaron
211.	string	zero

35 FontInfo
found in: Helvetica-Oblique
access priviledge: Accessible
length: 9
max length: 9

1.	string	FamilyName
2.	string	FullName
3.	integer	ItalicAngle
4.	string	Notice
5.	integer	UnderlinePosition
6.	integer	UnderlineThickness
7.	string	Weight
8.	boolean	isFixedPitch
9.	string	version

36 Private
found in: Helvetica-Oblique
access priviledge: No Access

37 FontInfo
found in: Times-BoldItalic
access priviledge: Accessible
length: 9
max length: 9

1.	string	FamilyName
2.	string	FullName
3.	integer	ItalicAngle
4.	string	Notice
5.	integer	UnderlinePosition
6.	integer	UnderlineThickness
7.	string	Weight
8.	boolean	isFixedPitch
9.	string	version

38 Private
found in: Times-BoldItalic
access priviledge: No Access

39 CharStrings
found in: Times-BoldItalic
access priviledge: Accessible
length: 211
max length: 211

1.	string	.notdef
2.	string	A
3.	string	AE
4.	string	Aacute
5.	string	Acircumflex
6.	string	Adieresis
7.	string	Agrave
8.	string	Aring
9.	string	Atilde
10.	string	B
11.	string	C
12.	string	Ccedilla
13.	string	D
14.	string	E
15.	string	Eacute
16.	string	Ecircumflex
17.	string	Edieresis
18.	string	Egrave
19.	string	F
20.	string	G
21.	string	H
22.	string	I
23.	string	Iacute
24.	string	Icircumflex
25.	string	Idieresis
26.	string	Igrave
27.	string	J
28.	string	K
29.	string	L
30.	string	Lslash
31.	string	M
32.	string	N
33.	string	Ntilde
34.	string	O

Product: QMS-PS 800 Version: 38.0 Revision: 0

entry	*type*	*name*	*entry*	*type*	*name*
35.	string	OE	108.	string	endash
36.	string	Oacute	109.	string	equal
37.	string	Ocircumflex	110.	string	exclam
38.	string	Odieresis	111.	string	exclamdown
39.	string	Ograve	112.	string	f
40.	string	Oslash	113.	string	fi
41.	string	Otilde	114.	string	five
42.	string	P	115.	string	fl
43.	string	Q	116.	string	florin
44.	string	R	117.	string	four
45.	string	S	118.	string	fraction
46.	string	Scaron	119.	string	g
47.	string	T	120.	string	germandbls
48.	string	U	121.	string	grave
49.	string	Uacute	122.	string	greater
50.	string	Ucircumflex	123.	string	guillemotleft
51.	string	Udieresis	124.	string	guillemotright
52.	string	Ugrave	125.	string	guilsinglleft
53.	string	V	126.	string	guilsinglright
54.	string	W	127.	string	h
55.	string	X	128.	string	hungarumlaut
56.	string	Y	129.	string	hyphen
57.	string	Ydieresis	130.	string	i
58.	string	Z	131.	string	iacute
59.	string	Zcaron	132.	string	icircumflex
60.	string	a	133.	string	idieresis
61.	string	aacute	134.	string	igrave
62.	string	acircumflex	135.	string	j
63.	string	acute	136.	string	k
64.	string	adieresis	137.	string	l
65.	string	ae	138.	string	less
66.	string	agrave	139.	string	logicalnot
67.	string	ampersand	140.	string	lslash
68.	string	aring	141.	string	m
69.	string	asciicircum	142.	string	macron
70.	string	asciitilde	143.	string	minus
71.	string	asterisk	144.	string	n
72.	string	at	145.	string	nine
73.	string	atilde	146.	string	ntilde
74.	string	b	147.	string	numbersign
75.	string	backslash	148.	string	o
76.	string	bar	149.	string	oacute
77.	string	braceleft	150.	string	ocircumflex
78.	string	braceright	151.	string	odieresis
79.	string	bracketleft	152.	string	oe
80.	string	bracketright	153.	string	ogonek
81.	string	breve	154.	string	ograve
82.	string	bullet	155.	string	one
83.	string	c	156.	string	ordfeminine
84.	string	caron	157.	string	ordmasculine
85.	string	ccedilla	158.	string	oslash
86.	string	cedilla	159.	string	otilde
87.	string	cent	160.	string	p
88.	string	circumflex	161.	string	paragraph
89.	string	colon	162.	string	parenleft
90.	string	comma	163.	string	parenright
91.	string	copyright	164.	string	percent
92.	string	currency	165.	string	period
93.	string	d	166.	string	periodcentered
94.	string	dagger	167.	string	perthousand
95.	string	daggerdbl	168.	string	plus
96.	string	dieresis	169.	string	q
97.	string	dollar	170.	string	question
98.	string	dotaccent	171.	string	questiondown
99.	string	dotlessi	172.	string	quotedbl
100.	string	e	173.	string	quotedblbase
101.	string	eacute	174.	string	quotedblleft
102.	string	ecircumflex	175.	string	quotedblright
103.	string	edieresis	176.	string	quoteleft
104.	string	egrave	177.	string	quoteright
105.	string	eight	178.	string	quotesinglbase
106.	string	ellipsis	179.	string	quotesingle
107.	string	emdash	180.	string	r

Product: QMS-PS 800 Version: 38.0 Revision: 0

entry	*type*	*name*
181.	string	registered
182.	string	ring
183.	string	s
184.	string	scaron
185.	string	section
186.	string	semicolon
187.	string	seven
188.	string	six
189.	string	slash
190.	string	space
191.	string	sterling
192.	string	t
193.	string	three
194.	string	tilde
195.	string	trademark
196.	string	two
197.	string	u
198.	string	uacute
199.	string	ucircumflex
200.	string	udieresis
201.	string	ugrave
202.	string	underscore
203.	string	v
204.	string	w
205.	string	x
206.	string	y
207.	string	ydieresis
208.	string	yen
209.	string	z
210.	string	zcaron
211.	string	zero

40 FontInfo
found in: Times-Roman
access priviledge: Accessible
length: 9
max length: 9

1.	string	FamilyName
2.	string	FullName
3.	integer	ItalicAngle
4.	string	Notice
5.	integer	UnderlinePosition
6.	integer	UnderlineThickness
7.	string	Weight
8.	boolean	isFixedPitch
9.	string	version

41 Private
found in: Times-Roman
access priviledge: No Access

42 CharStrings
found in: Times-Roman
access priviledge: Accessible
length: 211
max length: 211

1.	string	.notdef
2.	string	A
3.	string	AE
4.	string	Aacute
5.	string	Acircumflex
6.	string	Adieresis
7.	string	Agrave
8.	string	Aring
9.	string	Atilde
10.	string	B
11.	string	C
12.	string	Ccedilla
13.	string	D
14.	string	E
15.	string	Eacute
16.	string	Ecircumflex
17.	string	Edieresis
18.	string	Egrave
19.	string	F
20.	string	G
21.	string	H
22.	string	I
23.	string	Iacute
24.	string	Icircumflex
25.	string	Idieresis
26.	string	Igrave
27.	string	J
28.	string	K
29.	string	L
30.	string	Lslash
31.	string	M
32.	string	N
33.	string	Ntilde
34.	string	O
35.	string	OE
36.	string	Oacute
37.	string	Ocircumflex
38.	string	Odieresis
39.	string	Ograve
40.	string	Oslash
41.	string	Otilde
42.	string	P
43.	string	Q
44.	string	R
45.	string	S
46.	string	Scaron
47.	string	T
48.	string	U
49.	string	Uacute
50.	string	Ucircumflex
51.	string	Udieresis
52.	string	Ugrave
53.	string	V
54.	string	W
55.	string	X
56.	string	Y
57.	string	Ydieresis
58.	string	Z
59.	string	Zcaron
60.	string	a
61.	string	aacute
62.	string	acircumflex
63.	string	acute
64.	string	adieresis
65.	string	ae
66.	string	agrave
67.	string	ampersand
68.	string	aring
69.	string	asciicircum
70.	string	asciitilde
71.	string	asterisk
72.	string	at
73.	string	atilde
74.	string	b
75.	string	backslash
76.	string	bar
77.	string	braceleft
78.	string	braceright
79.	string	bracketleft
80.	string	bracketright

Product: QMS-PS 800 Version: 38.0 Revision: 0

entry	*type*	*name*
81.	string	breve
82.	string	bullet
83.	string	c
84.	string	caron
85.	string	ccedilla
86.	string	cedilla
87.	string	cent
88.	string	circumflex
89.	string	colon
90.	string	comma
91.	string	copyright
92.	string	currency
93.	string	d
94.	string	dagger
95.	string	daggerdbl
96.	string	dieresis
97.	string	dollar
98.	string	dotaccent
99.	string	dotlessi
100.	string	e
101.	string	eacute
102.	string	ecircumflex
103.	string	edieresis
104.	string	egrave
105.	string	eight
106.	string	ellipsis
107.	string	emdash
108.	string	endash
109.	string	equal
110.	string	exclam
111.	string	exclamdown
112.	string	f
113.	string	fi
114.	string	five
115.	string	fl
116.	string	florin
117.	string	four
118.	string	fraction
119.	string	g
120.	string	germandbls
121.	string	grave
122.	string	greater
123.	string	guillemotleft
124.	string	guillemotright
125.	string	guilsinglleft
126.	string	guilsinglright
127.	string	h
128.	string	hungarumlaut
129.	string	hyphen
130.	string	i
131.	string	iacute
132.	string	icircumflex
133.	string	idieresis
134.	string	igrave
135.	string	j
136.	string	k
137.	string	l
138.	string	less
139.	string	logicalnot
140.	string	lslash
141.	string	m
142.	string	macron
143.	string	minus
144.	string	n
145.	string	nine
146.	string	ntilde
147.	string	numbersign
148.	string	o
149.	string	oacute
150.	string	ocircumflex
151.	string	odieresis
152.	string	oe
153.	string	ogonek
154.	string	ograve
155.	string	one
156.	string	ordfeminine
157.	string	ordmasculine
158.	string	oslash
159.	string	otilde
160.	string	p
161.	string	paragraph
162.	string	parenleft
163.	string	parenright
164.	string	percent
165.	string	period
166.	string	periodcentered
167.	string	perthousand
168.	string	plus
169.	string	q
170.	string	question
171.	string	questiondown
172.	string	quotedbl
173.	string	quotedblbase
174.	string	quotedblleft
175.	string	quotedblright
176.	string	quoteleft
177.	string	quoteright
178.	string	quotesinglbase
179.	string	quotesingle
180.	string	r
181.	string	registered
182.	string	ring
183.	string	s
184.	string	scaron
185.	string	section
186.	string	semicolon
187.	string	seven
188.	string	six
189.	string	slash
190.	string	space
191.	string	sterling
192.	string	t
193.	string	three
194.	string	tilde
195.	string	trademark
196.	string	two
197.	string	u
198.	string	uacute
199.	string	ucircumflex
200.	string	udieresis
201.	string	ugrave
202.	string	underscore
203.	string	v
204.	string	w
205.	string	x
206.	string	y
207.	string	ydieresis
208.	string	yen
209.	string	z
210.	string	zcaron
211.	string	zero

43 FontInfo
found in: Courier
access priviledge: Accessible
length: 8
max length: 8

1.	string	FamilyName
2.	string	FullName
3.	integer	ItalicAngle
4.	integer	UnderlinePosition
5.	integer	UnderlineThickness

Product: QMS-PS 800 Version: 38.0 Revision: 0

entry	*type*	*name*
6.	string	Weight
7.	boolean	isFixedPitch
8.	string	version

44 Private
found in: Courier
access priviledge: No Access

45 CharStrings
found in: Courier
access priviledge: Accessible
length: 199
max length: 199

entry	*type*	*name*
1.	string	.notdef
2.	string	A
3.	string	Aacute
4.	string	Acircumflex
5.	string	Adieresis
6.	string	Agrave
7.	string	Aring
8.	string	Atilde
9.	string	B
10.	string	C
11.	string	Ccedilla
12.	string	D
13.	string	E
14.	string	Eacute
15.	string	Ecircumflex
16.	string	Edieresis
17.	string	Egrave
18.	string	F
19.	string	G
20.	string	H
21.	string	I
22.	string	Iacute
23.	string	Icircumflex
24.	string	Idieresis
25.	string	Igrave
26.	string	J
27.	string	K
28.	string	L
29.	string	Lslash
30.	string	M
31.	string	N
32.	string	Ntilde
33.	string	O
34.	string	Oacute
35.	string	Ocircumflex
36.	string	Odieresis
37.	string	Ograve
38.	string	Oslash
39.	string	Otilde
40.	string	P
41.	string	Q
42.	string	R
43.	string	S
44.	string	Scaron
45.	string	T
46.	string	U
47.	string	Uacute
48.	string	Ucircumflex
49.	string	Udieresis
50.	string	Ugrave
51.	string	V
52.	string	W
53.	string	X
54.	string	Y
55.	string	Ydieresis
56.	string	Z
57.	string	Zcaron
58.	string	a
59.	string	aacute
60.	string	acircumflex
61.	string	acute
62.	string	adieresis
63.	string	agrave
64.	string	ampersand
65.	string	aring
66.	string	asciicircum
67.	string	asciitilde
68.	string	asterisk
69.	string	at
70.	string	atilde
71.	string	b
72.	string	backslash
73.	string	bar
74.	string	braceleft
75.	string	braceright
76.	string	bracketleft
77.	string	bracketright
78.	string	breve
79.	string	bullet
80.	string	c
81.	string	caron
82.	string	ccedilla
83.	string	cedilla
84.	string	cent
85.	string	circumflex
86.	string	colon
87.	string	comma
88.	string	currency
89.	string	d
90.	string	dagger
91.	string	daggerdbl
92.	string	dieresis
93.	string	dollar
94.	string	dotaccent
95.	string	dotlessi
96.	string	e
97.	string	eacute
98.	string	ecircumflex
99.	string	edieresis
100.	string	egrave
101.	string	eight
102.	string	ellipsis
103.	string	emdash
104.	string	endash
105.	string	equal
106.	string	exclam
107.	string	exclamdown
108.	string	f
109.	string	five
110.	string	florin
111.	string	four
112.	string	fraction
113.	string	g
114.	string	germandbls
115.	string	grave
116.	string	greater
117.	string	guillemotleft
118.	string	guillemotright
119.	string	guilsinglleft
120.	string	guilsinglright
121.	string	h
122.	string	hungarumlaut
123.	string	hyphen
124.	string	i
125.	string	iacute
126.	string	icircumflex
127.	string	idieresis

Product: QMS-PS 800 Version: 38.0 Revision: 0

entry	*type*	*name*
128.	string	igrave
129.	string	j
130.	string	k
131.	string	l
132.	string	less
133.	string	lslash
134.	string	m
135.	string	macron
136.	string	n
137.	string	nine
138.	string	ntilde
139.	string	numbersign
140.	string	o
141.	string	oacute
142.	string	ocircumflex
143.	string	odieresis
144.	string	ogonek
145.	string	ograve
146.	string	one
147.	string	ordfeminine
148.	string	ordmasculine
149.	string	oslash
150.	string	otilde
151.	string	p
152.	string	paragraph
153.	string	parenleft
154.	string	parenright
155.	string	percent
156.	string	period
157.	string	periodcentered
158.	string	plus
159.	string	q
160.	string	question
161.	string	questiondown
162.	string	quotedbl
163.	string	quotedblbase
164.	string	quotedblleft
165.	string	quotedblright
166.	string	quoteleft
167.	string	quoteright
168.	string	quotesinglbase
169.	string	quotesingle
170.	string	r
171.	string	ring
172.	string	s
173.	string	scaron
174.	string	section
175.	string	semicolon
176.	string	seven
177.	string	six
178.	string	slash
179.	string	space
180.	string	sterling
181.	string	t
182.	string	three
183.	string	tilde
184.	string	two
185.	string	u
186.	string	uacute
187.	string	ucircumflex
188.	string	udieresis
189.	string	ugrave
190.	string	underscore
191.	string	v
192.	string	w
193.	string	x
194.	string	y
195.	string	ydieresis
196.	string	yen
197.	string	z
198.	string	zcaron
199.	string	zero

46 FontInfo
found in: Helvetica-BoldOblique
access priviledge: Accessible
length: 9
max length: 9

entry	*type*	*name*
1.	string	FamilyName
2.	string	FullName
3.	integer	ItalicAngle
4.	string	Notice
5.	integer	UnderlinePosition
6.	integer	UnderlineThickness
7.	string	Weight
8.	boolean	isFixedPitch
9.	string	version

47 Private
found in: Helvetica-BoldOblique
access priviledge: No Access

48 CharStrings
found in: Helvetica-BoldOblique
access priviledge: Accessible
length: 211
max length: 211

entry	*type*	*name*
1.	string	.notdef
2.	string	A
3.	string	AE
4.	string	Aacute
5.	string	Acircumflex
6.	string	Adieresis
7.	string	Agrave
8.	string	Aring
9.	string	Atilde
10.	string	B
11.	string	C
12.	string	Ccedilla
13.	string	D
14.	string	E
15.	string	Eacute
16.	string	Ecircumflex
17.	string	Edieresis
18.	string	Egrave
19.	string	F
20.	string	G
21.	string	H
22.	string	I
23.	string	Iacute
24.	string	Icircumflex
25.	string	Idieresis
26.	string	Igrave
27.	string	J
28.	string	K
29.	string	L
30.	string	Lslash
31.	string	M
32.	string	N
33.	string	Ntilde
34.	string	O
35.	string	OE
36.	string	Oacute
37.	string	Ocircumflex
38.	string	Odieresis
39.	string	Ograve
40.	string	Oslash

Product: QMS-PS 800 Version: 38.0 Revision: 0

entry	*type*	*name*	*entry*	*type*	*name*
41.	string	Otilde	114.	string	five
42.	string	P	115.	string	fl
43.	string	Q	116.	string	florin
44.	string	R	117.	string	four
45.	string	S	118.	string	fraction
46.	string	Scaron	119.	string	g
47.	string	T	120.	string	germandbls
48.	string	U	121.	string	grave
49.	string	Uacute	122.	string	greater
50.	string	Ucircumflex	123.	string	guillemotleft
51.	string	Udieresis	124.	string	guillemotright
52.	string	Ugrave	125.	string	guilsinglleft
53.	string	V	126.	string	guilsinglright
54.	string	W	127.	string	h
55.	string	X	128.	string	hungarumlaut
56.	string	Y	129.	string	hyphen
57.	string	Ydieresis	130.	string	i
58.	string	Z	131.	string	iacute
59.	string	Zcaron	132.	string	icircumflex
60.	string	a	133.	string	idieresis
61.	string	aacute	134.	string	igrave
62.	string	acircumflex	135.	string	j
63.	string	acute	136.	string	k
64.	string	adieresis	137.	string	l
65.	string	ae	138.	string	less
66.	string	agrave	139.	string	logicalnot
67.	string	ampersand	140.	string	lslash
68.	string	aring	141.	string	m
69.	string	asciicircum	142.	string	macron
70.	string	asciitilde	143.	string	minus
71.	string	asterisk	144.	string	n
72.	string	at	145.	string	nine
73.	string	atilde	146.	string	ntilde
74.	string	b	147.	string	numbersign
75.	string	backslash	148.	string	o
76.	string	bar	149.	string	oacute
77.	string	braceleft	150.	string	ocircumflex
78.	string	braceright	151.	string	odieresis
79.	string	bracketleft	152.	string	oe
80.	string	bracketright	153.	string	ogonek
81.	string	breve	154.	string	ograve
82.	string	bullet	155.	string	one
83.	string	c	156.	string	ordfeminine
84.	string	caron	157.	string	ordmasculine
85.	string	ccedilla	158.	string	oslash
86.	string	cedilla	159.	string	otilde
87.	string	cent	160.	string	p
88.	string	circumflex	161.	string	paragraph
89.	string	colon	162.	string	parenleft
90.	string	comma	163.	string	parenright
91.	string	copyright	164.	string	percent
92.	string	currency	165.	string	period
93.	string	d	166.	string	periodcentered
94.	string	dagger	167.	string	perthousand
95.	string	daggerdbl	168.	string	plus
96.	string	dieresis	169.	string	q
97.	string	dollar	170.	string	question
98.	string	dotaccent	171.	string	questiondown
99.	string	dotlessi	172.	string	quotedbl
100.	string	e	173.	string	quotedblbase
101.	string	eacute	174.	string	quotedblleft
102.	string	ecircumflex	175.	string	quotedblright
103.	string	edieresis	176.	string	quoteleft
104.	string	egrave	177.	string	quoteright
105.	string	eight	178.	string	quotesinglbase
106.	string	ellipsis	179.	string	quotesingle
107.	string	emdash	180.	string	r
108.	string	endash	181.	string	registered
109.	string	equal	182.	string	ring
110.	string	exclam	183.	string	s
111.	string	exclamdown	184.	string	scaron
112.	string	f	185.	string	section
113.	string	fi	186.	string	semicolon

Product: QMS-PS 800 Version: 38.0 Revision: 0

entry	type	name
187.	string	seven
188.	string	six
189.	string	slash
190.	string	space
191.	string	sterling
192.	string	t
193.	string	three
194.	string	tilde
195.	string	trademark
196.	string	two
197.	string	u
198.	string	uacute
199.	string	ucircumflex
200.	string	udieresis
201.	string	ugrave
202.	string	underscore
203.	string	v
204.	string	w
205.	string	x
206.	string	y
207.	string	ydieresis
208.	string	yen
209.	string	z
210.	string	zcaron
211.	string	zero

49 FontInfo
found in: Symbol
access priviledge: Accessible
length: 8
max length: 8

1.	string	FamilyName
2.	string	FullName
3.	integer	ItalicAngle
4.	integer	UnderlinePosition
5.	integer	UnderlineThickness
6.	string	Weight
7.	boolean	isFixedPitch
8.	string	version

50 Private
found in: Symbol
access priviledge: No Access

51 CharStrings
found in: Symbol
access priviledge: Accessible
length: 190
max length: 190

entry	type	name
1.	string	.notdef
2.	string	Alpha
3.	string	Beta
4.	string	Chi
5.	string	Delta
6.	string	Epsilon
7.	string	Eta
8.	string	Gamma
9.	string	Ifraktur
10.	string	Iota
11.	string	Kappa
12.	string	Lambda
13.	string	Mu
14.	string	Nu
15.	string	Omega
16.	string	Omicron
17.	string	Phi
18.	string	Pi
19.	string	Psi
20.	string	Rfraktur
21.	string	Rho
22.	string	Sigma
23.	string	Tau
24.	string	Theta
25.	string	Upsilon
26.	string	Upsilon1
27.	string	Xi
28.	string	Zeta
29.	string	aleph
30.	string	alpha
31.	string	ampersand
32.	string	angle
33.	string	angleleft
34.	string	angleright
35.	string	apple
36.	string	approxequal
37.	string	arrowboth
38.	string	arrowdblboth
39.	string	arrowdbldown
40.	string	arrowdblleft
41.	string	arrowdblright
42.	string	arrowdblup
43.	string	arrowdown
44.	string	arrowhorizex
45.	string	arrowleft
46.	string	arrowright
47.	string	arrowup
48.	string	arrowvertex
49.	string	asteriskmath
50.	string	bar
51.	string	beta
52.	string	braceex
53.	string	braceleft
54.	string	braceleftbt
55.	string	braceleftmid
56.	string	bracelefttp
57.	string	braceright
58.	string	bracerightbt
59.	string	bracerightmid
60.	string	bracerighttp
61.	string	bracketleft
62.	string	bracketleftbt
63.	string	bracketleftex
64.	string	bracketlefttp
65.	string	bracketright
66.	string	bracketrightbt
67.	string	bracketrightex
68.	string	bracketrighttp
69.	string	bullet
70.	string	carriagereturn
71.	string	chi
72.	string	circlemultiply
73.	string	circleplus
74.	string	club
75.	string	colon
76.	string	comma
77.	string	congruent
78.	string	copyrightsans
79.	string	copyrightserif
80.	string	degree
81.	string	delta
82.	string	diamond
83.	string	divide
84.	string	dotmath
85.	string	eight
86.	string	element
87.	string	ellipsis

Product: QMS-PS 800 Version: 38.0 Revision: 0

entry	*type*	*name*
88.	string	emptyset
89.	string	epsilon
90.	string	equal
91.	string	equivalence
92.	string	eta
93.	string	exclam
94.	string	existential
95.	string	five
96.	string	florin
97.	string	four
98.	string	fraction
99.	string	gamma
100.	string	gradient
101.	string	greater
102.	string	greaterequal
103.	string	heart
104.	string	infinity
105.	string	integral
106.	string	integralbt
107.	string	integralex
108.	string	integraltp
109.	string	intersection
110.	string	iota
111.	string	kappa
112.	string	lambda
113.	string	less
114.	string	lessequal
115.	string	logicaland
116.	string	logicalnot
117.	string	logicalor
118.	string	lozenge
119.	string	minus
120.	string	minute
121.	string	mu
122.	string	multiply
123.	string	nine
124.	string	notelement
125.	string	notequal
126.	string	notsubset
127.	string	nu
128.	string	numbersign
129.	string	omega
130.	string	omega1
131.	string	omicron
132.	string	one
133.	string	parenleft
134.	string	parenleftbt
135.	string	parenleftex
136.	string	parenlefttp
137.	string	parenright
138.	string	parenrightbt
139.	string	parenrightex
140.	string	parenrighttp
141.	string	partialdiff
142.	string	percent
143.	string	period
144.	string	perpendicular
145.	string	phi
146.	string	phi1
147.	string	pi
148.	string	plus
149.	string	plusminus
150.	string	product
151.	string	propersubset
152.	string	propersuperset
153.	string	proportional
154.	string	psi
155.	string	question
156.	string	radical
157.	string	radicalex
158.	string	reflexsubset
159.	string	reflexsuperset
160.	string	registersans
161.	string	registerserif
162.	string	rho
163.	string	second
164.	string	semicolon
165.	string	seven
166.	string	sigma
167.	string	sigma1
168.	string	similar
169.	string	six
170.	string	slash
171.	string	space
172.	string	spade
173.	string	suchthat
174.	string	summation
175.	string	tau
176.	string	therefore
177.	string	theta
178.	string	theta1
179.	string	three
180.	string	trademarksans
181.	string	trademarkserif
182.	string	two
183.	string	underscore
184.	string	union
185.	string	universal
186.	string	upsilon
187.	string	weierstrass
188.	string	xi
189.	string	zero
190.	string	zeta

52 FontInfo
found in: Courier-Bold
access priviledge: Accessible
length: 8
max length: 8

1.	string	FamilyName
2.	string	FullName
3.	integer	ItalicAngle
4.	integer	UnderlinePosition
5.	integer	UnderlineThickness
6.	string	Weight
7.	boolean	isFixedPitch
8.	string	version

53 Private
found in: Courier-Bold
access priviledge: No Access

54 FontInfo
found in: Courier-BoldOblique
access priviledge: Accessible
length: 8
max length: 8

1.	string	FamilyName
2.	string	FullName
3.	integer	ItalicAngle
4.	integer	UnderlinePosition
5.	integer	UnderlineThickness
6.	string	Weight
7.	boolean	isFixedPitch
8.	string	version

Product: QMS-PS 800 Version: 38.0 Revision: 0

entry type name

55 Private
found in: Courier-BoldOblique
access priviledge: No Access

56 FontInfo
found in: Courier-Oblique
access priviledge: Accessible
length: 8
max length: 8

1.	string	FamilyName
2.	string	FullName
3.	integer	ItalicAngle
4.	integer	UnderlinePosition
5.	integer	UnderlineThickness
6.	string	Weight
7.	boolean	isFixedPitch
8.	string	version

57 Private
found in: Courier-Oblique
access priviledge: No Access

58 FontInfo
found in: Helvetica-Bold
access priviledge: Accessible
length: 9
max length: 9

1.	string	FamilyName
2.	string	FullName
3.	integer	ItalicAngle
4.	string	Notice
5.	integer	UnderlinePosition
6.	integer	UnderlineThickness
7.	string	Weight
8.	boolean	isFixedPitch
9.	string	version

59 Private
found in: Helvetica-Bold
access priviledge: No Access

Commands with * are documented commands.

Appendix III

Exhaustive Cross Reference Table of PostScript Names

Introduction

This cross reference table is the key to unlocking the relationships between all of the internal procedures. In it can be found every name which occurs in the resident PostScript interpreter.

Perhaps you have come across a mysterious system variable such as an entry in the serverdict. This table will help locate all of its occurances in the interpreter which will then lead you to its discussion in this book through the *Inside PostScript* index in the next section. The names can occur within a dictionary in which case they are marked ***dict entry***, or within a procedure in which case the procedure name and the dictionary where the procedure is found are listed.

How to Use the Table

The entries themselves tell quite a bit about the structure of the interpreter. For instance the numeric entries at the beginning of the table are never directly referenced from within a procedure since they are all marked as ***dict entry***. Instead, as the interpreter is studied, these procedures are found to be indirectly referenced and executed as engine status and thumbwheel switches are read.

Suppose the integer 0 is placed on the stack and executed. With four procedures named 0 how does the interpreter know which one to execute. The table shows that a procedure called 0 exists in four separate dictionaries.

If all occurences of appletalkclose must be determined, the table shows that the dictionary entry (***dict entry***) is in serverdict and that it is called from the procedure switchclose.

Most interesting is the searching out of the locations of system procedures and variables. For instance, where is the startpage string executed? The table shows that startpage is located in the serverdict and that it is referenced only once in the start procedure (start is located in userdict).

This list is exhaustive of all dictionaries and procedures. If a procedure contains two references to a single name, only one entry is made in the table. The list does not contain entries in the font dictionaries and their sub dictionaries since these entries are repetative and would unnecessarily clutter the table. Also, the font dictionaries are not discussed in this book since they are not involved in any of the printer control procedures discussed herein.

The last portion of the table contains names which are preceeded by a "/". If you are looking for a PostScript name variable, both the slash and non-slash entries should be checked.

Cross Reference Table

name	*where found*	*dict found in*
0	***dict entry***	specialswitch
0	***dict entry***	mydict
0	***dict entry***	$printerdict
0	***dict entry***	serverdict
1	***dict entry***	specialswitch
1	***dict entry***	serverdict
18	***dict entry***	$printerdict
2	***dict entry***	serverdict
2	***dict entry***	$printerdict
2048	***dict entry***	mydict
24	***dict entry***	$printerdict
3	***dict entry***	serverdict
4616	***dict entry***	mydict
512	***dict entry***	mydict
520	***dict entry***	mydict
528	***dict entry***	mydict
536	***dict entry***	mydict
576	***dict entry***	mydict
8	***dict entry***	$printerdict
a4	***dict entry***	userdict
abort	printererror	statusdict
abort	***dict entry***	mydict
abs	0	$printerdict
abs	18	$printerdict
abs	2	$printerdict
abs	24	$printerdict
abs	8	$printerdict
abs	a4	userdict
abs	b5	userdict
abs	legal	userdict
abs	letter	userdict
abs	setpage	$printerdict
abs	***dict entry***	systemdict
add	0	$printerdict
add	18	$printerdict
add	2	$printerdict
add	24	$printerdict
add	8	$printerdict
add	a4	userdict
add	appletalkopen	serverdict
add	b5	userdict
add	dopage	$printerdict
add	eprint	mydict
add	executive	userdict
add	hashcommparams	serverdict
add	idlA	$idleTimeDict
add	initprinter	serverdict
add	legal	userdict
add	letter	userdict
add	printerstatus	serverdict
add	proc	$printerdict
add	pstack	userdict
add	setpage	$printerdict
add	setrealdevice	serverdict
add	stack	systemdict
add	start	userdict
add	tprint	--dict
add	UseIdleTime	userdict
add	warmedup	serverdict
add	***dict entry***	systemdict
aload	dictfull	errordict
aload	dictstackoverflow	errordict
aload	dictstackunderflow	errordict
aload	execstackoverflow	errordict
aload	invalidaccess	errordict
aload	invalidexit	errordict
aload	invalidfileaccess	errordict
aload	invalidfont	errordict
aload	invalidrestore	errordict
aload	ioerror	errordict
aload	limitcheck	errordict
aload	nocurrentpoint	errordict
aload	rangecheck	errordict
aload	stackoverflow	errordict
aload	stackunderflow	errordict
aload	syntaxerror	errordict
aload	timeout	errordict
aload	typecheck	errordict
aload	undefined	errordict
aload	undefinedfilename	errordict
aload	undefinedresult	errordict
aload	unmatchedmark	errordict
aload	unregistered	errordict
aload	VMerror	errordict
aload	***dict entry***	systemdict
aload	.error	systemdict
altflag	1	specialswitch
altflag	altprint	serverdict
altflag	setsccstreams	serverdict
altflag	stopPred	$idleTimeDict
altflag	watchstreams	serverdict
altflag	***dict entry***	serverdict
altin	stopPred	$idleTimeDict
altin	watchstreams	serverdict
altin	***dict entry***	serverdict
altname	***dict entry***	serverdict
altout	1	specialswitch
altout	altprint	serverdict
altout	***dict entry***	serverdict
altprint	***dict entry***	serverdict
anchorsearch	***dict entry***	systemdict
and	4616	mydict
and	528	mydict
and	execjob	serverdict
and	printererror	statusdict
and	ReadIdleFonts	userdict
and	setrealdevice	serverdict
and	setsccstreams	serverdict
and	warmedup	serverdict
and	***dict entry***	systemdict
appletalkclose	switchclose	serverdict
appletalkclose	***dict entry***	serverdict
appletalkopen	setstreams	serverdict
appletalkopen	switchopen	serverdict
appletalkopen	***dict entry***	serverdict
appletalktype	appletalkopen	serverdict
appletalktype	***dict entry***	statusdict
arc	***dict entry***	systemdict
arcn	***dict entry***	systemdict
arcto	***dict entry***	systemdict
array	dictfull	errordict
array	dictstackoverflow	errordict
array	dictstackunderflow	errordict
array	execstackoverflow	errordict
array	invalidaccess	errordict
array	invalidexit	errordict
array	invalidfileaccess	errordict
array	invalidfont	errordict
array	invalidrestore	errordict
array	ioerror	errordict
array	limitcheck	errordict
array	nocurrentpoint	errordict
array	rangecheck	errordict
array	ReadIdleFonts	userdict
array	stackoverflow	errordict
array	stackunderflow	errordict
array	syntaxerror	errordict
array	timeout	errordict
array	typecheck	errordict
array	undefined	errordict
array	undefinedfilename	errordict
array	undefinedresult	errordict

name	*where found*	*dict found in*
array	unmatchedmark	errordict
array	unregistered	errordict
array	VMerror	errordict
array	***dict entry***	systemdict
array	.error	systemdict
arraytype	***dict entry***	--dict
ascii26	***dict entry***	$idleTimeDict
ascii62	***dict entry***	$idleTimeDict
ascii81	***dict entry***	$idleTimeDict
ascii94	ReadIdleFonts	userdict
ascii94	***dict entry***	$idleTimeDict
ashow	***dict entry***	systemdict
astore	0	$printerdict
astore	18	$printerdict
astore	2	$printerdict
astore	24	$printerdict
astore	8	$printerdict
astore	a4	userdict
astore	b5	userdict
astore	dictfull	errordict
astore	dictstackoverflow	errordict
astore	dictstackunderflow	errordict
astore	execstackoverflow	errordict
astore	invalidaccess	errordict
astore	invalidexit	errordict
astore	invalidfileaccess	errordict
astore	invalidfont	errordict
astore	invalidrestore	errordict
astore	ioerror	errordict
astore	legal	userdict
astore	letter	userdict
astore	limitcheck	errordict
astore	nocurrentpoint	errordict
astore	rangecheck	errordict
astore	setpage	$printerdict
astore	stackoverflow	errordict
astore	stackunderflow	errordict
astore	syntaxerror	errordict
astore	timeout	errordict
astore	typecheck	errordict
astore	undefined	errordict
astore	undefinedfilename	errordict
astore	undefinedresult	errordict
astore	unmatchedmark	errordict
astore	unregistered	errordict
astore	VMerror	errordict
astore	***dict entry***	systemdict
astore	.error	systemdict
atan	***dict entry***	systemdict
awidthshow	***dict entry***	systemdict
b5	***dict entry***	userdict
batchidleproc	***dict entry***	serverdict
baud25	setsccstreams	serverdict
baud25	***dict entry***	serverdict
baud9	setsccstreams	serverdict
baud9	***dict entry***	serverdict
begin	0	serverdict
begin	0	$printerdict
begin	0	specialswitch
begin	1	serverdict
begin	1	specialswitch
begin	18	$printerdict
begin	2	$printerdict
begin	24	$printerdict
begin	3	serverdict
begin	8	$printerdict
begin	a4	userdict
begin	altprint	serverdict
begin	appletalkopen	serverdict
begin	b5	userdict
begin	batchidleproc	serverdict
begin	dopage	$printerdict
begin	execjob	serverdict
begin	executive	userdict
begin	fontname	serverdict
begin	handleerror	errordict
begin	intidleproc	serverdict
begin	legal	userdict
begin	letter	userdict
begin	printererror	statusdict
begin	proc	$printerdict
begin	ReadIdleFonts	userdict
begin	setpage	$printerdict
begin	setsccstreams	serverdict
begin	settimeouts	serverdict
begin	start	userdict
begin	stopPred	$idleTimeDict
begin	UseIdleTime	userdict
begin	watchstreams	serverdict
begin	***dict entry***	systemdict
begin	--	systemdict
bind	***dict entry***	systemdict
bits	printererror	statusdict
bits	***dict entry***	mydict
bitshift	4616	mydict
bitshift	hashcommparams	serverdict
bitshift	printererror	statusdict
bitshift	***dict entry***	systemdict
blink	***dict entry***	statusdict
bmpI	ReadIdleFonts	userdict
bmpI	***dict entry***	$idleTimeDict
booleantype	***dict entry***	--dict
boundsCheck	ReadIdleFonts	userdict
boundsCheck	***dict entry***	$idleTimeDict
bytesavailable	execjob	serverdict
bytesavailable	stopPred	$idleTimeDict
bytesavailable	watchstreams	serverdict
bytesavailable	***dict entry***	systemdict
cachestatus	***dict entry***	systemdict
ceiling	***dict entry***	systemdict
cexec	***dict entry***	systemdict
charpath	***dict entry***	systemdict
checkpassword	***dict entry***	statusdict
checkquit	executive	userdict
checkquit	***dict entry***	execdict
clear	execjob	serverdict
clear	start	userdict
clear	***dict entry***	systemdict
cleardictstack	execjob	serverdict
cleardictstack	start	userdict
cleardictstack	***dict entry***	userdict
clearinterrupt	0	serverdict
clearinterrupt	0	specialswitch
clearinterrupt	1	serverdict
clearinterrupt	3	serverdict
clearinterrupt	execjob	serverdict
clearinterrupt	executive	userdict
clearinterrupt	***dict entry***	systemdict
cleartomark	executive	userdict
cleartomark	ReadIdleFonts	userdict
cleartomark	***dict entry***	systemdict
clip	***dict entry***	systemdict
clippath	***dict entry***	systemdict
closefile	execjob	serverdict
closefile	executive	userdict
closefile	***dict entry***	systemdict
closepath	***dict entry***	systemdict
closescc	appletalkopen	serverdict
closescc	setsccstreams	serverdict
closescc	switchclose	serverdict
closescc	***dict entry***	statusdict
command	***dict entry***	$error
commhash	execjob	serverdict
commhash	setsccstreams	serverdict
commhash	***dict entry***	serverdict
concat	***dict entry***	systemdict
concatmatrix	***dict entry***	systemdict
copy	0	$printerdict
copy	18	$printerdict
copy	2	$printerdict

name	*where found*	*dict found in*
copy	24	$printerdict
copy	8	$printerdict
copy	a4	userdict
copy	b5	userdict
copy	legal	userdict
copy	letter	userdict
copy	pstack	userdict
copy	setpage	$printerdict
copy	stack	systemdict
copy	***dict entry***	systemdict
copypage	***dict entry***	systemdict
cos	***dict entry***	systemdict
count	dictfull	errordict
count	dictstackoverflow	errordict
count	dictstackunderflow	errordict
count	execstackoverflow	errordict
count	invalidaccess	errordict
count	invalidexit	errordict
count	invalidfileaccess	errordict
count	invalidfont	errordict
count	invalidrestore	errordict
count	ioerror	errordict
count	limitcheck	errordict
count	nocurrentpoint	errordict
count	pstack	userdict
count	rangecheck	errordict
count	stack	systemdict
count	stackoverflow	errordict
count	stackunderflow	errordict
count	syntaxerror	errordict
count	timeout	errordict
count	typecheck	errordict
count	undefined	errordict
count	undefinedfilename	errordict
count	undefinedresult	errordict
count	unmatchedmark	errordict
count	unregistered	errordict
count	VMerror	errordict
count	***dict entry***	systemdict
count	.error	systemdict
countdictstack	cleardictstack	userdict
countdictstack	execjob	serverdict
countdictstack	***dict entry***	systemdict
countexecstack	***dict entry***	systemdict
counttomark	ReadIdleFonts	userdict
counttomark	***dict entry***	systemdict
Courier	***dict entry***	FontDirectory
Courier-Bold	***dict entry***	FontDirectory
Courier-BoldOblique	***dict entry***	FontDirectory
Courier-Oblique	***dict entry***	FontDirectory
cp	tprint	--dict
cp	***dict entry***	--dict
currentcacheparams	***dict entry***	systemdict
currentdash	***dict entry***	systemdict
currentdict	printererror	statusdict
currentdict	***dict entry***	systemdict
currentfile	***dict entry***	systemdict
currentflat	***dict entry***	systemdict
currentfont	***dict entry***	systemdict
currentgray	***dict entry***	systemdict
currenthsbcolor	***dict entry***	systemdict
currentlinecap	***dict entry***	systemdict
currentlinejoin	***dict entry***	systemdict
currentlinewidth	***dict entry***	systemdict
currentmatrix	***dict entry***	systemdict
currentmiterlimit	***dict entry***	systemdict
currentpacking	***dict entry***	systemdict
currentpoint	***dict entry***	systemdict
currentrgbcolor	***dict entry***	systemdict
currentscreen	***dict entry***	systemdict
currenttransfer	***dict entry***	systemdict
curveto	***dict entry***	systemdict
cvi	0	$printerdict
cvi	18	$printerdict
cvi	2	$printerdict
cvi	24	$printerdict
cvi	8	$printerdict
cvi	a4	userdict
cvi	b5	userdict
cvi	dopage	$printerdict
cvi	legal	userdict
cvi	letter	userdict
cvi	proc	$printerdict
cvi	***dict entry***	systemdict
cvlit	***dict entry***	systemdict
cvn	***dict entry***	systemdict
cvr	***dict entry***	systemdict
cvrs	printererror	statusdict
cvrs	***dict entry***	systemdict
cvs	4616	mydict
cvs	cvsprint	--dict
cvs	eprint	mydict
cvs	findfont	systemdict
cvs	handleerror	errordict
cvs	operatortype	--dict
cvs	stack	systemdict
cvs	***dict entry***	systemdict
cvs	=	systemdict
cvs	=print	systemdict
cvsprint	booleantype	--dict
cvsprint	integertype	--dict
cvsprint	nametype	--dict
cvsprint	realtype	--dict
cvsprint	***dict entry***	--dict
cvx	0	serverdict
cvx	1	serverdict
cvx	3	serverdict
cvx	executive	userdict
cvx	***dict entry***	systemdict
daytime	***dict entry***	systemdict
debugmode	***dict entry***	statusdict
def	0	serverdict
def	0	$printerdict
def	0	specialswitch
def	1	serverdict
def	1	specialswitch
def	18	$printerdict
def	2	$printerdict
def	24	$printerdict
def	3	serverdict
def	528	mydict
def	8	$printerdict
def	a4	userdict
def	b5	userdict
def	batchidleproc	serverdict
def	bmpI	$idleTimeDict
def	boundsCheck	$idleTimeDict
def	dopage	$printerdict
def	eprint	mydict
def	exchdef	serverdict
def	execjob	serverdict
def	executive	userdict
def	handleerror	errordict
def	intidleproc	serverdict
def	legal	userdict
def	letter	userdict
def	printererror	statusdict
def	proc	$printerdict
def	ReadIdleFonts	userdict
def	setpage	$printerdict
def	setsccstreams	serverdict
def	setstreams	serverdict
def	start	userdict
def	stopPred	$idleTimeDict
def	tprint	--dict
def	UseIdleTime	userdict
def	watchstreams	serverdict
def	***dict entry***	systemdict
def	==	systemdict
defaultmatrix	***dict entry***	systemdict

name	*where found*	*dict found in*	*name*	*where found*	*dict found in*
defaulttimeouts	0	serverdict	dstackarray	***dict entry***	$error
defaulttimeouts	0	specialswitch	dtransform	***dict entry***	systemdict
defaulttimeouts	1	serverdict	dup	0	$printerdict
defaulttimeouts	1	specialswitch	dup	0	specialswitch
defaulttimeouts	3	serverdict	dup	18	$printerdict
defaulttimeouts	batchidleproc	serverdict	dup	2	$printerdict
defaulttimeouts	execjob	serverdict	dup	2	serverdict
defaulttimeouts	intidleproc	serverdict	dup	24	$printerdict
defaulttimeouts	***dict entry***	statusdict	dup	8	$printerdict
definefont	***dict entry***	systemdict	dup	a4	userdict
dexch	stopPred	$idleTimeDict	dup	appletalkopen	serverdict
dexch	watchstreams	serverdict	dup	arraytype	--dict
dexch	***dict entry***	serverdict	dup	b5	userdict
diablo	0	specialswitch	dup	dexch	serverdict
diablo	***dict entry***	statusdict	dup	dictfull	errordict
dict	***dict entry***	systemdict	dup	dictstackoverflow	errordict
dictfull	***dict entry***	errordict	dup	dictstackunderflow	errordict
dictstack	dictfull	errordict	dup	eflush	mydict
dictstack	dictstackoverflow	errordict	dup	execstackoverflow	errordict
dictstack	dictstackunderflow	errordict	dup	findfont	systemdict
dictstack	execstackoverflow	errordict	dup	fontname	serverdict
dictstack	invalidaccess	errordict	dup	handleerror	errordict
dictstack	invalidexit	errordict	dup	initprinter	serverdict
dictstack	invalidfileaccess	errordict	dup	invalidaccess	errordict
dictstack	invalidfont	errordict	dup	invalidexit	errordict
dictstack	invalidrestore	errordict	dup	invalidfileaccess	errordict
dictstack	ioerror	errordict	dup	invalidfont	errordict
dictstack	limitcheck	errordict	dup	invalidrestore	errordict
dictstack	nocurrentpoint	errordict	dup	ioerror	errordict
dictstack	rangecheck	errordict	dup	legal	userdict
dictstack	stackoverflow	errordict	dup	letter	userdict
dictstack	stackunderflow	errordict	dup	limitcheck	errordict
dictstack	syntaxerror	errordict	dup	nametype	--dict
dictstack	timeout	errordict	dup	nocurrentpoint	errordict
dictstack	typecheck	errordict	dup	packedarraytype	--dict
dictstack	undefined	errordict	dup	printerstatus	serverdict
dictstack	undefinedfilename	errordict	dup	pstack	userdict
dictstack	undefinedresult	errordict	dup	rangecheck	errordict
dictstack	unmatchedmark	errordict	dup	Run	systemdict
dictstack	unregistered	errordict	dup	setpage	$printerdict
dictstack	VMerror	errordict	dup	setrealdevice	serverdict
dictstack	***dict entry***	systemdict	dup	setsccstreams	serverdict
dictstack	.error	systemdict	dup	stack	systemdict
dictstackoverflow	***dict entry***	errordict	dup	stackoverflow	errordict
dictstackunderflow	***dict entry***	errordict	dup	stackunderflow	errordict
dicttype	***dict entry***	--dict	dup	stopPred	$idleTimeDict
disableinterrupt	0	serverdict	dup	stringtype	--dict
disableinterrupt	1	serverdict	dup	syntaxerror	errordict
disableinterrupt	3	serverdict	dup	timeout	errordict
disableinterrupt	execjob	serverdict	dup	tprint	--dict
disableinterrupt	executive	userdict	dup	typecheck	errordict
disableinterrupt	start	userdict	dup	typeprint	--dict
disableinterrupt	stopPred	$idleTimeDict	dup	undefined	errordict
disableinterrupt	watchstreams	serverdict	dup	undefinedfilename	errordict
disableinterrupt	***dict entry***	systemdict	dup	undefinedresult	errordict
div	0	$printerdict	dup	unmatchedmark	errordict
div	18	$printerdict	dup	unregistered	errordict
div	2	$printerdict	dup	VMerror	errordict
div	24	$printerdict	dup	warmedup	serverdict
div	8	$printerdict	dup	watchstreams	serverdict
div	a4	userdict	dup	***dict entry***	systemdict
div	b5	userdict	dup	.error	systemdict
div	dopage	$printerdict	dup	-	systemdict
div	legal	userdict	dup	-print	systemdict
div	letter	userdict	echo	***dict entry***	systemdict
div	proc	$printerdict	eeinfo	boundsCheck	$idleTimeDict
div	ReadIdleFonts	userdict	eeinfo	ReadIdleFonts	userdict
div	setpage	$printerdict	eerom	***dict entry***	statusdict
div	***dict entry***	systemdict	eescratch	0	specialswitch
doclose	execjob	serverdict	eescratch	2	serverdict
doclose	***dict entry***	$error	eescratch	appletalkopen	serverdict
dopage	***dict entry***	$printerdict	eescratch	hashcommparams	serverdict
dostartpage	start	userdict	eescratch	setsccstreams	serverdict
dostartpage	***dict entry***	statusdict	eescratch	***dict entry***	statusdict
dstack	***dict entry***	$error	eexec	***dict entry***	systemdict

name	*where found*	*dict found in*
eflush	printererror	statusdict
eflush	***dict entry***	mydict
eindex	eflush	mydict
eindex	eprint	mydict
eindex	***dict entry***	mydict
enableinterrupt	0	serverdict
enableinterrupt	1	serverdict
enableinterrupt	3	serverdict
enableinterrupt	printererror	statusdict
enableinterrupt	stopPred	$idleTimeDict
enableinterrupt	watchstreams	serverdict
enableinterrupt	***dict entry***	systemdict
end	0	$printerdict
end	0	serverdict
end	0	specialswitch
end	1	specialswitch
end	1	serverdict
end	18	$printerdict
end	2	$printerdict
end	24	$printerdict
end	3	serverdict
end	8	$printerdict
end	a4	userdict
end	altprint	serverdict
end	appletalkopen	serverdict
end	b5	userdict
end	batchidleproc	serverdict
end	cleardictstack	userdict
end	dopage	$printerdict
end	execjob	serverdict
end	executive	userdict
end	fontname	serverdict
end	handleerror	errordict
end	intidleproc	serverdict
end	legal	userdict
end	letter	userdict
end	printererror	statusdict
end	proc	$printerdict
end	ReadIdleFonts	userdict
end	setpage	$printerdict
end	setsccstreams	serverdict
end	settimeouts	serverdict
end	start	userdict
end	stopPred	$idleTimeDict
end	UseIdleTime	userdict
end	watchstreams	serverdict
end	***dict entry***	systemdict
end	--	systemdict
eoclip	***dict entry***	systemdict
eofill	***dict entry***	systemdict
eprint	4616	mydict
eprint	printererror	statusdict
eprint	***dict entry***	mydict
eq	0	specialswitch
eq	528	mydict
eq	dictfull	errordict
eq	dictstackoverflow	errordict
eq	dictstackunderflow	errordict
eq	execjob	serverdict
eq	execstackoverflow	errordict
eq	initprinter	serverdict
eq	invalidaccess	errordict
eq	invalidexit	errordict
eq	invalidfileaccess	errordict
eq	invalidfont	errordict
eq	invalidrestore	errordict
eq	ioerror	errordict
eq	limitcheck	errordict
eq	nocurrentpoint	errordict
eq	printererror	statusdict
eq	printerstatus	serverdict
eq	rangecheck	errordict
eq	ReadIdleFonts	userdict
eq	setrealdevice	serverdict
eq	setsccstreams	serverdict
eq	stackoverflow	errordict
eq	stackunderflow	errordict
eq	syntaxerror	errordict
eq	timeout	errordict
eq	typecheck	errordict
eq	undefined	errordict
eq	undefinedfilename	errordict
eq	undefinedresult	errordict
eq	unmatchedmark	errordict
eq	unregistered	errordict
eq	VMerror	errordict
eq	warmedup	serverdict
eq	***dict entry***	systemdict
eq	.error	systemdict
erasepage	0	$printerdict
erasepage	18	$printerdict
erasepage	2	$printerdict
erasepage	24	$printerdict
erasepage	8	$printerdict
erasepage	a4	userdict
erasepage	b5	userdict
erasepage	execjob	serverdict
erasepage	legal	userdict
erasepage	letter	userdict
erasepage	setpage	$printerdict
erasepage	***dict entry***	systemdict
errordict	execjob	serverdict
errordict	handleerror	systemdict
errordict	***dict entry***	systemdict
errorname	execjob	serverdict
errorname	handleerror	errordict
errorname	***dict entry***	$error
errstr	eflush	mydict
errstr	eprint	mydict
errstr	***dict entry***	mydict
estack	***dict entry***	$error
estackarray	***dict entry***	$error
exch	0	serverdict
exch	0	$printerdict
exch	1	specialswitch
exch	1	serverdict
exch	18	$printerdict
exch	2	$printerdict
exch	2	serverdict
exch	24	$printerdict
exch	3	serverdict
exch	8	$printerdict
exch	a4	userdict
exch	altprint	serverdict
exch	appletalkopen	serverdict
exch	b5	userdict
exch	batchidleproc	serverdict
exch	bmpI	$idleTimeDict
exch	dexch	serverdict
exch	dictfull	errordict
exch	dictstackoverflow	errordict
exch	dictstackunderflow	errordict
exch	dopage	$printerdict
exch	eprint	mydict
exch	exchdef	serverdict
exch	execstackoverflow	errordict
exch	findfont	systemdict
exch	fontname	serverdict
exch	idlA	$idleTimeDict
exch	intidleproc	serverdict
exch	invalidaccess	errordict
exch	invalidexit	errordict
exch	invalidfileaccess	errordict
exch	invalidfont	errordict
exch	invalidrestore	errordict
exch	ioerror	errordict
exch	legal	userdict
exch	letter	userdict
exch	limitcheck	errordict
exch	nocurrentpoint	errordict

name	*where found*	*dict found in*
exch	printererror	statusdict
exch	proc	$printerdict
exch	rangecheck	errordict
exch	ReadIdleFonts	userdict
exch	setpage	$printerdict
exch	setrealdevice	serverdict
exch	setsccstreams	serverdict
exch	stackoverflow	errordict
exch	stackunderflow	errordict
exch	start	userdict
exch	syntaxerror	errordict
exch	timeout	errordict
exch	tprint	--dict
exch	typecheck	errordict
exch	undefined	errordict
exch	undefinedfilename	errordict
exch	undefinedresult	errordict
exch	unmatchedmark	errordict
exch	unregistered	errordict
exch	UseIdleTime	userdict
exch	VMerror	errordict
exch	warmedup	serverdict
exch	***dict entry***	systemdict
exch	.error	systemdict
exchdef	setsccstreams	serverdict
exchdef	settimeouts	serverdict
exchdef	***dict entry***	serverdict
exec	0	$printerdict
exec	0	serverdict
exec	1	serverdict
exec	18	$printerdict
exec	2	serverdict
exec	2	$printerdict
exec	24	$printerdict
exec	3	serverdict
exec	8	$printerdict
exec	a4	userdict
exec	b5	userdict
exec	dictfull	errordict
exec	dictstackoverflow	errordict
exec	dictstackunderflow	errordict
exec	execjob	serverdict
exec	execstackoverflow	errordict
exec	executive	userdict
exec	findfont	systemdict
exec	handleerror	errordict
exec	handleerror	systemdict
exec	initprinter	serverdict
exec	invalidaccess	errordict
exec	invalidexit	errordict
exec	invalidfileaccess	errordict
exec	invalidfont	errordict
exec	invalidrestore	errordict
exec	ioerror	errordict
exec	legal	userdict
exec	letter	userdict
exec	limitcheck	errordict
exec	nocurrentpoint	errordict
exec	printererror	statusdict
exec	proc	$printerdict
exec	rangecheck	errordict
exec	setrealdevice	serverdict
exec	setstreams	serverdict
exec	stack	systemdict
exec	stackoverflow	errordict
exec	stackunderflow	errordict
exec	start	userdict
exec	syntaxerror	errordict
exec	timeout	errordict
exec	typecheck	errordict
exec	typeprint	--dict
exec	undefined	errordict
exec	undefinedfilename	errordict
exec	undefinedresult	errordict
exec	unmatchedmark	errordict
exec	unregistered	errordict
exec	VMerror	errordict
exec	warmedup	serverdict
exec	***dict entry***	systemdict
exec	-	systemdict
execdepth	executive	userdict
execdepth	prompt	userdict
execdepth	***dict entry***	execdict
execdict	0	serverdict
execdict	1	serverdict
execdict	1	specialswitch
execdict	3	serverdict
execdict	execjob	serverdict
execdict	executive	userdict
execdict	quit	userdict
execdict	start	userdict
execdict	***dict entry***	userdict
execjob	***dict entry***	serverdict
execstack	dictfull	errordict
execstack	dictstackoverflow	errordict
execstack	dictstackunderflow	errordict
execstack	execstackoverflow	errordict
execstack	invalidaccess	errordict
execstack	invalidexit	errordict
execstack	invalidfileaccess	errordict
execstack	invalidfont	errordict
execstack	invalidrestore	errordict
execstack	ioerror	errordict
execstack	limitcheck	errordict
execstack	nocurrentpoint	errordict
execstack	rangecheck	errordict
execstack	stackoverflow	errordict
execstack	stackunderflow	errordict
execstack	syntaxerror	errordict
execstack	timeout	errordict
execstack	typecheck	errordict
execstack	undefined	errordict
execstack	undefinedfilename	errordict
execstack	undefinedresult	errordict
execstack	unmatchedmark	errordict
execstack	unregistered	errordict
execstack	VMerror	errordict
execstack	***dict entry***	systemdict
execstack	.error	systemdict
execstackoverflow	***dict entry***	errordict
executeonly	***dict entry***	systemdict
executive	1	specialswitch
executive	***dict entry***	userdict
exit	checkquit	execdict
exit	executive	userdict
exit	initprinter	serverdict
exit	printererror	statusdict
exit	printerstatus	serverdict
exit	ReadIdleFonts	userdict
exit	setrealdevice	serverdict
exit	UseIdleTime	userdict
exit	warmedup	serverdict
exit	***dict entry***	systemdict
exitserver	***dict entry***	serverdict
exp	***dict entry***	systemdict
false	0	specialswitch
false	appletalkopen	serverdict
false	***dict entry***	systemdict
file	executive	userdict
file	***dict entry***	systemdict
filetype	***dict entry***	--dict
fill	***dict entry***	systemdict
findfont	***dict entry***	systemdict
flattenpath	***dict entry***	systemdict
floor	***dict entry***	systemdict
flush	eflush	mydict
flush	execjob	serverdict
flush	findfont	systemdict
flush	handleerror	errordict
flush	initprinter	serverdict

name	where found	dict found in
flush	printerstatus	serverdict
flush	prompt	userdict
flush	Run	systemdict
flush	setrealdevice	serverdict
flush	warmedup	serverdict
flush	***dict entry***	systemdict
flushfile	1	specialswitch
flushfile	altprint	serverdict
flushfile	execjob	serverdict
flushfile	***dict entry***	systemdict
FontDirectory	findfont	systemdict
FontDirectory	UseIdleTime	userdict
FontDirectory	***dict entry***	systemdict
fontname	0	specialswitch
fontname	***dict entry***	serverdict
fonttype	***dict entry***	--dict
for	initprinter	serverdict
for	***dict entry***	systemdict
forall	arraytype	--dict
forall	packedarraytype	--dict
forall	***dict entry***	systemdict
framedevice	0	$printerdict
framedevice	18	$printerdict
framedevice	2	$printerdict
framedevice	24	$printerdict
framedevice	8	$printerdict
framedevice	a4	userdict
framedevice	b5	userdict
framedevice	legal	userdict
framedevice	letter	userdict
framedevice	setpage	$printerdict
framedevice	***dict entry***	systemdict
franksdict	***dict entry***	userdict
ge	fontname	serverdict
ge	initprinter	serverdict
ge	UseIdleTime	userdict
ge	***dict entry***	systemdict
get	1	specialswitch
get	2	serverdict
get	528	mydict
get	appletalkopen	serverdict
get	checkquit	execdict
get	dictfull	errordict
get	dictstackoverflow	errordict
get	dictstackunderflow	errordict
get	execjob	serverdict
get	execstackoverflow	errordict
get	executive	userdict
get	findfont	systemdict
get	fontname	serverdict
get	handleerror	systemdict
get	idlA	$idleTimeDict
get	intidleproc	serverdict
get	invalidaccess	errordict
get	invalidexit	errordict
get	invalidfileaccess	errordict
get	invalidfont	errordict
get	invalidrestore	errordict
get	ioerror	errordict
get	limitcheck	errordict
get	nocurrentpoint	errordict
get	rangecheck	errordict
get	ReadIdleFonts	userdict
get	setnulldevice	serverdict
get	setrealdevice	serverdict
get	setsccstreams	serverdict
get	setstreams	serverdict
get	stackoverflow	errordict
get	stackunderflow	errordict
get	start	userdict
get	syntaxerror	errordict
get	timeout	errordict
get	typecheck	errordict
get	undefined	errordict
get	undefinedfilename	errordict
get	undefinedresult	errordict
get	unmatchedmark	errordict
get	unregistered	errordict
get	UseIdleTime	userdict
get	VMerror	errordict
get	***dict entry***	systemdict
get	.error	systemdict
getinterval	appletalkopen	serverdict
getinterval	dictfull	errordict
getinterval	dictstackoverflow	errordict
getinterval	dictstackunderflow	errordict
getinterval	eflush	mydict
getinterval	eprint	mydict
getinterval	execstackoverflow	errordict
getinterval	invalidaccess	errordict
getinterval	invalidexit	errordict
getinterval	invalidfileaccess	errordict
getinterval	invalidfont	errordict
getinterval	invalidrestore	errordict
getinterval	ioerror	errordict
getinterval	limitcheck	errordict
getinterval	nocurrentpoint	errordict
getinterval	rangecheck	errordict
getinterval	ReadIdleFonts	userdict
getinterval	stackoverflow	errordict
getinterval	stackunderflow	errordict
getinterval	syntaxerror	errordict
getinterval	timeout	errordict
getinterval	typecheck	errordict
getinterval	undefined	errordict
getinterval	undefinedfilename	errordict
getinterval	undefinedresult	errordict
getinterval	unmatchedmark	errordict
getinterval	unregistered	errordict
getinterval	UseIdleTime	userdict
getinterval	VMerror	errordict
getinterval	***dict entry***	systemdict
getinterval	.error	systemdict
grestore	UseIdleTime	userdict
grestore	***dict entry***	systemdict
grestoreall	***dict entry***	systemdict
gsave	UseIdleTime	userdict
gsave	***dict entry***	systemdict
gt	0	$printerdict
gt	18	$printerdict
gt	2	$printerdict
gt	24	$printerdict
gt	8	$printerdict
gt	a4	userdict
gt	b5	userdict
gt	legal	userdict
gt	letter	userdict
gt	ReadIdleFonts	userdict
gt	setpage	$printerdict
gt	start	userdict
gt	tprint	--dict
gt	***dict entry***	systemdict
handleerror	dictfull	errordict
handleerror	dictstackoverflow	errordict
handleerror	dictstackunderflow	errordict
handleerror	execstackoverflow	errordict
handleerror	executive	userdict
handleerror	invalidaccess	errordict
handleerror	invalidexit	errordict
handleerror	invalidfileaccess	errordict
handleerror	invalidfont	errordict
handleerror	invalidrestore	errordict
handleerror	ioerror	errordict
handleerror	limitcheck	errordict
handleerror	nocurrentpoint	errordict
handleerror	rangecheck	errordict
handleerror	stackoverflow	errordict
handleerror	stackunderflow	errordict
handleerror	syntaxerror	errordict
handleerror	timeout	errordict

name	*where found*	*dict found in*	*name*	*where found*	*dict found in*
handleerror	typecheck	errordict	if	nocurrentpoint	errordict
handleerror	undefined	errordict	if	printererror	statusdict
handleerror	undefinedfilename	errordict	if	printerstatus	serverdict
handleerror	undefinedresult	errordict	if	rangecheck	errordict
handleerror	unmatchedmark	errordict	if	ReadIdleFonts	userdict
handleerror	unregistered	errordict	if	setrealdevice	serverdict
handleerror	VMerror	errordict	if	setsccstreams	serverdict
handleerror	***dict entry***	errordict	if	setstreams	serverdict
handleerror	***dict entry***	systemdict	if	stack	systemdict
handleerror	.error	systemdict	if	stackoverflow	errordict
hashcommparams	appletalkopen	serverdict	if	stackunderflow	errordict
hashcommparams	execjob	serverdict	if	start	userdict
hashcommparams	setsccstreams	serverdict	if	stopPred	$idleTimeDict
hashcommparams	***dict entry***	serverdict	if	syntaxerror	errordict
height	0	$printerdict	if	timeout	errordict
height	18	$printerdict	if	tprint	==dict
height	2	$printerdict	if	typecheck	errordict
height	24	$printerdict	if	undefined	errordict
height	8	$printerdict	if	undefinedfilename	errordict
height	a4	userdict	if	undefinedresult	errordict
height	b5	userdict	if	unmatchedmark	errordict
height	legal	userdict	if	unregistered	errordict
height	letter	userdict	if	UseIdleTime	userdict
height	setpage	$printerdict	if	VMerror	errordict
height	***dict entry***	$printerdict	if	warmedup	serverdict
Helvetica	***dict entry***	FontDirectory	if	watchstreams	serverdict
Helvetica-Bold	***dict entry***	FontDirectory	if	***dict entry***	systemdict
Helvetica-BoldOblique	***dict entry***	FontDirectory	if	.error	systemdict
Helvetica-Oblique	***dict entry***	FontDirectory	if	-	systemdict
identmatrix	***dict entry***	systemdict	if	=print	systemdict
idiv	***dict entry***	systemdict	ifelse	0	$printerdict
idlA	UseIdleTime	userdict	ifelse	18	$printerdict
idlA	***dict entry***	$idleTimeDict	ifelse	2	$printerdict
idleArry	bmpI	$idleTimeDict	ifelse	2	serverdict
idleArry	idlA	$idleTimeDict	ifelse	24	$printerdict
idleArry	UseIdleTime	userdict	ifelse	528	mydict
idleArry	***dict entry***	$idleTimeDict	ifelse	8	$printerdict
idlefonts	ReadIdleFonts	userdict	ifelse	a4	userdict
idlefonts	***dict entry***	statusdict	ifelse	appletalkopen	serverdict
idleI	bmpI	$idleTimeDict	ifelse	arraytype	--dict
idleI	idlA	$idleTimeDict	ifelse	b5	userdict
idleI	ReadIdleFonts	userdict	ifelse	dopage	$printerdict
idleI	UseIdleTime	userdict	ifelse	executive	userdict
idleI	***dict entry***	$idleTimeDict	ifelse	findfont	systemdict
idleproc	executive	userdict	ifelse	initprinter	serverdict
idleproc	***dict entry***	execdict	ifelse	legal	userdict
idleStr	UseIdleTime	userdict	ifelse	letter	userdict
idleStr	***dict entry***	$idleTimeDict	ifelse	packedarraytype	--dict
idleStrI	UseIdleTime	userdict	ifelse	printererror	statusdict
idleStrI	***dict entry***	$idleTimeDict	ifelse	proc	$printerdict
idtransform	***dict entry***	systemdict	ifelse	setpage	$printerdict
if	0	specialswitch	ifelse	setsccstreams	serverdict
if	1	specialswitch	ifelse	stringtype	--dict
if	altprint	serverdict	ifelse	UseIdleTime	userdict
if	appletalkopen	serverdict	ifelse	***dict entry***	systemdict
if	boundsCheck	$idleTimeDict	image	***dict entry***	systemdict
if	checkquit	execdict	imagemask	***dict entry***	systemdict
if	dictfull	errordict	index	1	specialswitch
if	dictstackoverflow	errordict	index	altprint	serverdict
if	dictstackunderflow	errordict	index	dexch	serverdict
if	execjob	serverdict	index	***dict entry***	systemdict
if	execstackoverflow	errordict	initappletalk	appletalkclose	serverdict
if	executive	userdict	initappletalk	appletalkopen	serverdict
if	findfont	systemdict	initappletalk	***dict entry***	statusdict
if	fontname	serverdict	initclip	***dict entry***	systemdict
if	handleerror	errordict	initgraphics	0	$printerdict
if	initprinter	serverdict	initgraphics	18	$printerdict
if	invalidaccess	errordict	initgraphics	2	$printerdict
if	invalidexit	errordict	initgraphics	24	$printerdict
if	invalidfileaccess	errordict	initgraphics	8	$printerdict
if	invalidfont	errordict	initgraphics	a4	userdict
if	invalidrestore	errordict	initgraphics	b5	userdict
if	ioerror	errordict	initgraphics	execjob	serverdict
if	limitcheck	errordict	initgraphics	legal	userdict
if	nametype	--dict	initgraphics	letter	userdict

name	*where found*	*dict found in*	*name*	*where found*	*dict found in*
initgraphics	setpage	$printerdict	length	eprint	mydict
initgraphics	***dict entry***	systemdict	length	execstackoverflow	errordict
initialized	***dict entry***	systemdict	length	fontname	serverdict
initializing	***dict entry***	$error	length	invalidaccess	errordict
initmatrix	UseIdleTime	userdict	length	invalidexit	errordict
initmatrix	***dict entry***	systemdict	length	invalidfileaccess	errordict
initprinter	start	userdict	length	invalidfont	errordict
initprinter	***dict entry***	serverdict	length	invalidrestore	errordict
integertype	***dict entry***	--dict	length	ioerror	errordict
internaldict	***dict entry***	systemdict	length	limitcheck	errordict
interrupt	dictfull	errordict	length	nocurrentpoint	errordict
interrupt	dictstackoverflow	errordict	length	rangecheck	errordict
interrupt	dictstackunderflow	errordict	length	stackoverflow	errordict
interrupt	execstackoverflow	errordict	length	stackunderflow	errordict
interrupt	executive	userdict	length	syntaxerror	errordict
interrupt	invalidaccess	errordict	length	timeout	errordict
interrupt	invalidexit	errordict	length	tprint	--dict
interrupt	invalidfileaccess	errordict	length	typecheck	errordict
interrupt	invalidfont	errordict	length	undefined	errordict
interrupt	invalidrestore	errordict	length	undefinedfilename	errordict
interrupt	ioerror	errordict	length	undefinedresult	errordict
interrupt	limitcheck	errordict	length	unmatchedmark	errordict
interrupt	nocurrentpoint	errordict	length	unregistered	errordict
interrupt	printererror	statusdict	length	UseIdleTime	userdict
interrupt	quit	userdict	length	VMerror	errordict
interrupt	rangecheck	errordict	length	***dict entry***	systemdict
interrupt	stackoverflow	errordict	length	.error	systemdict
interrupt	stackunderflow	errordict	letter	***dict entry***	userdict
interrupt	syntaxerror	errordict	limitcheck	***dict entry***	errordict
interrupt	timeout	errordict	lineto	***dict entry***	systemdict
interrupt	typecheck	errordict	ln	***dict entry***	systemdict
interrupt	undefined	errordict	load	0	specialswitch
interrupt	undefinedfilename	errordict	load	0	$printerdict
interrupt	undefinedresult	errordict	load	0	serverdict
interrupt	unmatchedmark	errordict	load	1	specialswitch
interrupt	unregistered	errordict	load	1	serverdict
interrupt	VMerror	errordict	load	18	$printerdict
interrupt	***dict entry***	errordict	load	2	$printerdict
interrupt	.error	systemdict	load	24	$printerdict
intidleproc	***dict entry***	serverdict	load	3	serverdict
invalidaccess	***dict entry***	errordict	load	8	$printerdict
invalidexit	***dict entry***	errordict	load	a4	userdict
invalidfileaccess	***dict entry***	errordict	load	b5	userdict
invalidfont	***dict entry***	errordict	load	dexch	serverdict
invalidrestore	***dict entry***	errordict	load	executive	userdict
invertmatrix	***dict entry***	systemdict	load	handleerror	errordict
ioerror	***dict entry***	errordict	load	intidleproc	serverdict
itransform	***dict entry***	systemdict	load	legal	userdict
jobname	***dict entry***	statusdict	load	letter	userdict
jobsource	***dict entry***	statusdict	load	printererror	statusdict
jobstate	***dict entry***	statusdict	load	setpage	$printerdict
jobtimeout	***dict entry***	statusdict	load	start	userdict
known	2	serverdict	load	***dict entry***	systemdict
known	findfont	systemdict	locked	exitserver	serverdict
known	printererror	statusdict	locked	protect	serverdict
known	setrealdevice	serverdict	locked	server	serverdict
known	UseIdleTime	userdict	log	***dict entry***	systemdict
known	***dict entry***	systemdict	loop	executive	userdict
kshow	***dict entry***	systemdict	loop	initprinter	serverdict
laststat	printererror	statusdict	loop	printerstatus	serverdict
laststat	***dict entry***	mydict	loop	ReadIdleFonts	userdict
le	boundsCheck	$idleTimeDict	loop	setrealdevice	serverdict
le	initprinter	serverdict	loop	start	userdict
le	printerstatus	serverdict	loop	UseIdleTime	userdict
le	ReadIdleFonts	userdict	loop	warmedup	serverdict
le	setrealdevice	serverdict	loop	***dict entry***	systemdict
le	start	userdict	lt	warmedup	serverdict
le	warmedup	serverdict	lt	***dict entry***	systemdict
le	***dict entry***	systemdict	makefont	***dict entry***	systemdict
legal	***dict entry***	userdict	makevm	***dict entry***	systemdict
length	appletalkopen	serverdict	manualfeed	0	$printerdict
length	boundsCheck	$idleTimeDict	manualfeed	18	$printerdict
length	dictfull	errordict	manualfeed	2	$printerdict
length	dictstackoverflow	errordict	manualfeed	24	$printerdict
length	dictstackunderflow	errordict	manualfeed	8	$printerdict

name	*where found*	*dict found in*
manualfeed	a4	userdict
manualfeed	b5	userdict
manualfeed	dopage	$printerdict
manualfeed	legal	userdict
manualfeed	letter	userdict
manualfeed	proc	$printerdict
manualfeed	***dict entry***	statusdict
manualfeedtimeout	0	$printerdict
manualfeedtimeout	18	$printerdict
manualfeedtimeout	2	$printerdict
manualfeedtimeout	24	$printerdict
manualfeedtimeout	8	$printerdict
manualfeedtimeout	a4	userdict
manualfeedtimeout	b5	userdict
manualfeedtimeout	dopage	$printerdict
manualfeedtimeout	legal	userdict
manualfeedtimeout	letter	userdict
manualfeedtimeout	proc	$printerdict
manualfeedtimeout	***dict entry***	statusdict
margins	0	$printerdict
margins	18	$printerdict
margins	2	$printerdict
margins	24	$printerdict
margins	8	$printerdict
margins	a4	userdict
margins	b5	userdict
margins	dopage	$printerdict
margins	legal	userdict
margins	letter	userdict
margins	proc	$printerdict
margins	***dict entry***	statusdict
mark	executive	userdict
mark	***dict entry***	systemdict
marktype	***dict entry***	==dict
matrix	***dict entry***	systemdict
maxlength	***dict entry***	systemdict
mod	ReadIdleFonts	userdict
mod	***dict entry***	systemdict
moveto	***dict entry***	systemdict
mtx	0	$printerdict
mtx	18	$printerdict
mtx	2	$printerdict
mtx	24	$printerdict
mtx	8	$printerdict
mtx	a4	userdict
mtx	b5	userdict
mtx	legal	userdict
mtx	letter	userdict
mtx	setpage	$printerdict
mtx	***dict entry***	$printerdict
mul	0	$printerdict
mul	18	$printerdict
mul	2	$printerdict
mul	24	$printerdict
mul	8	$printerdict
mul	a4	userdict
mul	b5	userdict
mul	legal	userdict
mul	letter	userdict
mul	ReadIdleFonts	userdict
mul	setpage	$printerdict
mul	***dict entry***	systemdict
mydict	printererror	statusdict
mydict	***dict entry***	mydict
nametype	***dict entry***	==dict
ne	appletalkopen	serverdict
ne	checkquit	execdict
ne	dictfull	errordict
ne	dictstackoverflow	errordict
ne	dictstackunderflow	errordict
ne	execstackoverflow	errordict
ne	executive	userdict
ne	findfont	systemdict
ne	handleerror	errordict
ne	initprinter	serverdict
ne	invalidaccess	errordict
ne	invalidexit	errordict
ne	invalidfileaccess	errordict
ne	invalidfont	errordict
ne	invalidrestore	errordict
ne	ioerror	errordict
ne	limitcheck	errordict
ne	nocurrentpoint	errordict
ne	printererror	statusdict
ne	rangecheck	errordict
ne	setsccstreams	serverdict
ne	setstreams	serverdict
ne	stack	systemdict
ne	stackoverflow	errordict
ne	stackunderflow	errordict
ne	start	userdict
ne	stopPred	$idleTimeDict
ne	syntaxerror	errordict
ne	timeout	errordict
ne	typecheck	errordict
ne	undefined	errordict
ne	undefinedfilename	errordict
ne	undefinedresult	errordict
ne	unmatchedmark	errordict
ne	unregistered	errordict
ne	VMerror	errordict
ne	watchstreams	serverdict
ne	***dict entry***	systemdict
ne	.error	systemdict
ne	=	systemdict
ne	=print	systemdict
neg	0	$printerdict
neg	18	$printerdict
neg	2	$printerdict
neg	24	$printerdict
neg	8	$printerdict
neg	a4	userdict
neg	b5	userdict
neg	legal	userdict
neg	letter	userdict
neg	setpage	$printerdict
neg	***dict entry***	systemdict
newerror	execjob	serverdict
newerror	handleerror	errordict
newerror	***dict entry***	$error
newpath	***dict entry***	systemdict
NL	tprint	==dict
NL	***dict entry***	==dict
NL	==	systemdict
noaccess	***dict entry***	systemdict
nocurrentpoint	***dict entry***	errordict
not	appletalkopen	serverdict
not	execjob	serverdict
not	nametype	==dict
not	setrealdevice	serverdict
not	***dict entry***	systemdict
ntrys	printererror	statusdict
ntrys	***dict entry***	mydict
null	dictfull	errordict
null	dictstackoverflow	errordict
null	dictstackunderflow	errordict
null	execjob	serverdict
null	execstackoverflow	errordict
null	invalidaccess	errordict
null	invalidexit	errordict
null	invalidfileaccess	errordict
null	invalidfont	errordict
null	invalidrestore	errordict
null	ioerror	errordict
null	limitcheck	errordict
null	nocurrentpoint	errordict
null	rangecheck	errordict
null	setstreams	serverdict
null	stackoverflow	errordict
null	stackunderflow	errordict

name	*where found*	*dict found in*
null	syntaxerror	errordict
null	timeout	errordict
null	typecheck	errordict
null	undefined	errordict
null	undefinedfilename	errordict
null	undefinedresult	errordict
null	unmatchedmark	errordict
null	unregistered	errordict
null	VMerror	errordict
null	***dict entry***	systemdict
null	.error	systemdict
nulldevice	setnulldevice	serverdict
nulldevice	***dict entry***	systemdict
nulltype	***dict entry***	--dict
openappletalk	appletalkopen	serverdict
openappletalk	***dict entry***	statusdict
openscc	setsccstreams	serverdict
openscc	***dict entry***	statusdict
operatortype	***dict entry***	--dict
or	checkquit	execdict
or	printererror	statusdict
or	setsccstreams	serverdict
or	start	userdict
or	warmedup	serverdict
or	***dict entry***	systemdict
ostack	***dict entry***	$error
ostackarray	***dict entry***	$error
packedarray	***dict entry***	systemdict
packedarraytype	***dict entry***	--dict
pagecount	***dict entry***	statusdict
pagestackorder	***dict entry***	statusdict
pagetype	***dict entry***	statusdict
parity25	setsccstreams	serverdict
parity25	***dict entry***	serverdict
parity9	setsccstreams	serverdict
parity9	***dict entry***	serverdict
pathbbox	***dict entry***	systemdict
pathforall	***dict entry***	systemdict
pop	0	serverdict
pop	0	specialswitch
pop	1	serverdict
pop	1	specialswitch
pop	2	serverdict
pop	3	serverdict
pop	appletalkclose	serverdict
pop	appletalkopen	serverdict
pop	arraytype	--dict
pop	batchidleproc	serverdict
pop	dictfull	errordict
pop	dictstackoverflow	errordict
pop	dictstackunderflow	errordict
pop	dicttype	--dict
pop	execstackoverflow	errordict
pop	executive	userdict
pop	filetype	--dict
pop	fontname	serverdict
pop	fonttype	--dict
pop	initprinter	serverdict
pop	intidleproc	serverdict
pop	invalidaccess	errordict
pop	invalidexit	errordict
pop	invalidfileaccess	errordict
pop	invalidfont	errordict
pop	invalidrestore	errordict
pop	ioerror	errordict
pop	limitcheck	errordict
pop	marktype	--dict
pop	nocurrentpoint	errordict
pop	nulltype	--dict
pop	packedarraytype	--dict
pop	printerstatus	serverdict
pop	pstack	userdict
pop	rangecheck	errordict
pop	savetype	--dict
pop	setrealdevice	serverdict
pop	stack	systemdict
pop	stackoverflow	errordict
pop	stackunderflow	errordict
pop	start	userdict
pop	stringtype	--dict
pop	syntaxerror	errordict
pop	timeout	errordict
pop	typecheck	errordict
pop	undefined	errordict
pop	undefinedfilename	errordict
pop	undefinedresult	errordict
pop	unmatchedmark	errordict
pop	unregistered	errordict
pop	UseIdleTime	userdict
pop	VMerror	errordict
pop	warmedup	serverdict
pop	***dict entry***	systemdict
pop	.error	systemdict
print	eflush	mydict
print	execjob	serverdict
print	executive	userdict
print	findfont	systemdict
print	handleerror	errordict
print	prompt	userdict
print	stack	systemdict
print	tprint	--dict
print	***dict entry***	systemdict
print	***dict entry***	execdict
print	=	systemdict
print	=print	systemdict
print	==	systemdict
printererror	***dict entry***	statusdict
printername	appletalkopen	serverdict
printername	***dict entry***	statusdict
printerstatus	initprinter	serverdict
printerstatus	printerstatus	serverdict
printerstatus	setrealdevice	serverdict
printerstatus	start	userdict
printerstatus	warmedup	serverdict
printerstatus	***dict entry***	statusdict
printerstatus	***dict entry***	serverdict
printpageflag	0	specialswitch
proc	***dict entry***	$printerdict
product	***dict entry***	statusdict
prompt	***dict entry***	userdict
protect	0	serverdict
protect	1	serverdict
protect	1	specialswitch
protect	3	serverdict
protect	intidleproc	serverdict
protect	***dict entry***	serverdict
psdevice	***dict entry***	systemdict
pstack	***dict entry***	userdict
put	dictfull	errordict
put	dictstackoverflow	errordict
put	dictstackunderflow	errordict
put	eflush	mydict
put	execjob	serverdict
put	execstackoverflow	errordict
put	executive	userdict
put	invalidaccess	errordict
put	invalidexit	errordict
put	invalidfileaccess	errordict
put	invalidfont	errordict
put	invalidrestore	errordict
put	ioerror	errordict
put	limitcheck	errordict
put	nocurrentpoint	errordict
put	quit	userdict
put	rangecheck	errordict
put	ReadIdleFonts	userdict
put	setstreams	serverdict
put	stackoverflow	errordict
put	stackunderflow	errordict
put	start	userdict

name	*where found*	*dict found in*
put	syntaxerror	errordict
put	timeout	errordict
put	typecheck	errordict
put	undefined	errordict
put	undefinedfilename	errordict
put	undefinedresult	errordict
put	unmatchedmark	errordict
put	unregistered	errordict
put	VMerror	errordict
put	***dict entry***	systemdict
put	.error	systemdict
putinterval	appletalkopen	serverdict
putinterval	***dict entry***	systemdict
quit	***dict entry***	userdict
quit	***dict entry***	systemdict
quitflag	checkquit	execdict
quitflag	***dict entry***	execdict
rand	***dict entry***	systemdict
rangecheck	***dict entry***	errordict
rcheck	arraytype	--dict
rcheck	packedarraytype	--dict
rcheck	stringtype	--dict
rcheck	***dict entry***	systemdict
rcurveto	***dict entry***	systemdict
read	***dict entry***	systemdict
readhexstring	***dict entry***	systemdict
ReadIdleFonts	ReadIdleFonts	userdict
ReadIdleFonts	setidlefonts	statusdict
ReadIdleFonts	start	userdict
ReadIdleFonts	***dict entry***	userdict
readline	***dict entry***	systemdict
readonly	***dict entry***	systemdict
readstring	***dict entry***	systemdict
realtype	***dict entry***	--dict
redclose	***dict entry***	statusdict
redwrite	0	$printerdict
redwrite	18	$printerdict
redwrite	2	$printerdict
redwrite	24	$printerdict
redwrite	8	$printerdict
redwrite	a4	userdict
redwrite	b5	userdict
redwrite	dopage	$printerdict
redwrite	legal	userdict
redwrite	letter	userdict
redwrite	proc	$printerdict
redwrite	***dict entry***	statusdict
repeat	cleardictstack	userdict
repeat	execjob	serverdict
repeat	hashcommparams	serverdict
repeat	initprinter	serverdict
repeat	printererror	statusdict
repeat	printerstatus	serverdict
repeat	prompt	userdict
repeat	pstack	userdict
repeat	setrealdevice	serverdict
repeat	stack	systemdict
repeat	warmedup	serverdict
repeat	***dict entry***	systemdict
report	printererror	statusdict
report	***dict entry***	mydict
resetfile	execjob	serverdict
resetfile	***dict entry***	systemdict
resetprinter	initprinter	serverdict
resetprinter	printererror	statusdict
resetprinter	printerstatus	serverdict
resetprinter	setrealdevice	serverdict
resetprinter	warmedup	serverdict
resetprinter	***dict entry***	statusdict
restore	start	userdict
restore	***dict entry***	systemdict
reversepath	***dict entry***	systemdict
revision	***dict entry***	statusdict
rlineto	***dict entry***	systemdict
rmargin	tprint	--dict
rmargin	***dict entry***	--dict
rmoveto	***dict entry***	systemdict
roll	0	$printerdict
roll	18	$printerdict
roll	2	$printerdict
roll	24	$printerdict
roll	8	$printerdict
roll	a4	userdict
roll	b5	userdict
roll	dopage	$printerdict
roll	legal	userdict
roll	letter	userdict
roll	proc	$printerdict
roll	***dict entry***	systemdict
ROMnames	fontname	serverdict
ROMnames	ReadIdleFonts	userdict
ROMnames	***dict entry***	$idleTimeDict
rotate	UseIdleTime	userdict
rotate	***dict entry***	systemdict
round	0	$printerdict
round	18	$printerdict
round	2	$printerdict
round	24	$printerdict
round	8	$printerdict
round	a4	userdict
round	b5	userdict
round	dopage	$printerdict
round	legal	userdict
round	letter	userdict
round	proc	$printerdict
round	***dict entry***	systemdict
rrand	***dict entry***	systemdict
run	Run	systemdict
Run	***dict entry***	systemdict
run	***dict entry***	systemdict
save	start	userdict
save	***dict entry***	systemdict
saveswitch	setsccstreams	serverdict
saveswitch	setstreams	serverdict
saveswitch	***dict entry***	serverdict
savetype	***dict entry***	--dict
scale	UseIdleTime	userdict
scale	***dict entry***	systemdict
scalefont	***dict entry***	systemdict
sccbatch	hashcommparams	serverdict
sccbatch	setstreams	serverdict
sccbatch	switchopen	serverdict
sccbatch	***dict entry***	statusdict
sccfiles	setsccstreams	serverdict
sccfiles	***dict entry***	statusdict
sccinteractive	hashcommparams	serverdict
sccinteractive	setstreams	serverdict
sccinteractive	switchopen	serverdict
sccinteractive	***dict entry***	statusdict
sccok	setsccstreams	serverdict
sccok	***dict entry***	serverdict
search	***dict entry***	systemdict
secretdict	***dict entry***	serverdict
sendctrld	execjob	serverdict
sendctrld	***dict entry***	serverdict
sendpcmd	initprinter	serverdict
sendpcmd	***dict entry***	statusdict
server	***dict entry***	serverdict
serverdict	0	specialswitch
serverdict	0	serverdict
serverdict	1	serverdict
serverdict	1	specialswitch
serverdict	3	serverdict
serverdict	altprint	serverdict
serverdict	batchidleproc	serverdict
serverdict	checkquit	execdict
serverdict	execjob	serverdict
serverdict	intidleproc	serverdict
serverdict	setsccstreams	serverdict
serverdict	start	userdict

name	*where found*	*dict found in*
serverdict	stopPred	$idleTimeDict
serverdict	watchstreams	serverdict
serverdict	***dict entry***	userdict
setappletalkname	appletalkopen	serverdict
setappletalkname	***dict entry***	statusdict
setblink	0	$printerdict
setblink	18	$printerdict
setblink	2	$printerdict
setblink	24	$printerdict
setblink	8	$printerdict
setblink	a4	userdict
setblink	b5	userdict
setblink	dopage	$printerdict
setblink	execjob	serverdict
setblink	legal	userdict
setblink	letter	userdict
setblink	printererror	statusdict
setblink	proc	$printerdict
setblink	start	userdict
setblink	stopPred	$idleTimeDict
setblink	watchstreams	serverdict
setblink	***dict entry***	statusdict
setcachedevice	***dict entry***	systemdict
setcachelimit	***dict entry***	systemdict
setcacheparams	***dict entry***	systemdict
setcharwidth	***dict entry***	systemdict
setdash	***dict entry***	systemdict
setdefaulttimeouts	***dict entry***	statusdict
setdostartpage	***dict entry***	statusdict
seteescratch	appletalkopen	serverdict
seteescratch	***dict entry***	statusdict
setflat	***dict entry***	systemdict
setfont	UseIdleTime	userdict
setfont	***dict entry***	systemdict
setgray	***dict entry***	systemdict
sethsbcolor	***dict entry***	systemdict
setidlefonts	ReadIdleFonts	userdict
setidlefonts	setidlefonts	statusdict
setidlefonts	***dict entry***	statusdict
setjobtimeout	settimeouts	serverdict
setjobtimeout	***dict entry***	statusdict
setlinecap	***dict entry***	systemdict
setlinejoin	***dict entry***	systemdict
setlinewidth	***dict entry***	systemdict
setmargins	***dict entry***	statusdict
setmatrix	setnulldevice	serverdict
setmatrix	***dict entry***	systemdict
setmiterlimit	***dict entry***	systemdict
setnulldevice	start	userdict
setnulldevice	***dict entry***	serverdict
setpacking	***dict entry***	systemdict
setpage	***dict entry***	$printerdict
setpagetype	***dict entry***	statusdict
setpassword	***dict entry***	statusdict
setprintername	***dict entry***	statusdict
setram	***dict entry***	systemdict
setrealdevice	0	serverdict
setrealdevice	0	specialswitch
setrealdevice	1	specialswitch
setrealdevice	1	serverdict
setrealdevice	3	serverdict
setrealdevice	intidleproc	serverdict
setrealdevice	start	userdict
setrealdevice	***dict entry***	serverdict
setrgbcolor	***dict entry***	systemdict
setrom	***dict entry***	systemdict
setsccbatch	***dict entry***	statusdict
setsccinteractive	***dict entry***	statusdict
setsccstreams	setstreams	serverdict
setsccstreams	switchopen	serverdict
setsccstreams	***dict entry***	serverdict
setscreen	0	$printerdict
setscreen	18	$printerdict
setscreen	2	$printerdict
setscreen.	24	$printerdict
setscreen	8	$printerdict
setscreen	a4	userdict
setscreen	b5	userdict
setscreen	legal	userdict
setscreen	letter	userdict
setscreen	setpage	$printerdict
setscreen	***dict entry***	systemdict
setstdio	appletalkopen	serverdict
setstdio	setsccstreams	serverdict
setstdio	stopPred	$idleTimeDict
setstdio	watchstreams	serverdict
setstdio	***dict entry***	statusdict
setstreams	start	userdict
setstreams	***dict entry***	serverdict
settimeouts	0	serverdict
settimeouts	0	specialswitch
settimeouts	1	serverdict
settimeouts	1	specialswitch
settimeouts	3	serverdict
settimeouts	batchidleproc	serverdict
settimeouts	execjob	serverdict
settimeouts	intidleproc	serverdict
settimeouts	***dict entry***	serverdict
settransfer	0	$printerdict
settransfer	18	$printerdict
settransfer	2	$printerdict
settransfer	24	$printerdict
settransfer	8	$printerdict
settransfer	a4	userdict
settransfer	b5	userdict
settransfer	legal	userdict
settransfer	letter	userdict
settransfer	setpage	$printerdict
settransfer	***dict entry***	systemdict
show	***dict entry***	systemdict
showpage	0	specialswitch
showpage	start	userdict
showpage	***dict entry***	systemdict
sin	***dict entry***	systemdict
specialswitch	2	serverdict
specialswitch	***dict entry***	serverdict
sqrt	***dict entry***	systemdict
srand	***dict entry***	systemdict
stack	***dict entry***	systemdict
stackoverflow	***dict entry***	errordict
stackunderflow	***dict entry***	errordict
StandardEncoding	***dict entry***	systemdict
start	***dict entry***	userdict
startpage	start	userdict
startpage	***dict entry***	serverdict
stat	4616	mydict
stat	528	mydict
stat	printererror	statusdict
stat	***dict entry***	mydict
status	execjob	serverdict
status	setsccstreams	serverdict
status	***dict entry***	systemdict
statusdict	0	$printerdict
statusdict	18	$printerdict
statusdict	2	$printerdict
statusdict	24	$printerdict
statusdict	528	mydict
statusdict	8	$printerdict
statusdict	a4	userdict
statusdict	appletalkopen	serverdict
statusdict	b5	userdict
statusdict	dopage	$printerdict
statusdict	eflush	mydict
statusdict	execjob	serverdict
statusdict	legal	userdict
statusdict	letter	userdict
statusdict	proc	$printerdict
statusdict	ReadIdleFonts	userdict
statusdict	setsccstreams	serverdict
statusdict	setstreams	serverdict

name	*where found*	*dict found in*
statusdict	settimeouts	serverdict
statusdict	start	userdict
statusdict	stopPred	$idleTimeDict
statusdict	watchstreams	serverdict
statusdict	***dict entry***	systemdict
stdin	0	serverdict
stdin	1	serverdict
stdin	3	serverdict
stdin	appletalkopen	serverdict
stdin	execjob	serverdict
stdin	setsccstreams	serverdict
stdin	stopPred	$idleTimeDict
stdin	watchstreams	serverdict
stdin	***dict entry***	serverdict
stdname	stopPred	$idleTimeDict
stdname	watchstreams	serverdict
stdname	***dict entry***	serverdict
stdout	1	specialswitch
stdout	altprint	serverdict
stdout	appletalkopen	serverdict
stdout	execjob	serverdict
stdout	setsccstreams	serverdict
stdout	stopPred	$idleTimeDict
stdout	watchstreams	serverdict
stdout	***dict entry***	serverdict
stmtfile	executive	userdict
stmtfile	***dict entry***	execdict
stop	***dict entry***	systemdict
stopped	0	specialswitch
stopped	execjob	serverdict
stopped	executive	userdict
stopped	start	userdict
stopped	***dict entry***	systemdict
stopPred	UseIdleTime	userdict
stopPred	***dict entry***	$idleTimeDict
store	appletalkopen	serverdict
store	dexch	serverdict
store	***dict entry***	systemdict
string	UseIdleTime	userdict
string	***dict entry***	systemdict
stringtype	***dict entry***	--dict
stringwidth	UseIdleTime	userdict
stringwidth	***dict entry***	systemdict
stroke	***dict entry***	systemdict
strokepath	***dict entry***	systemdict
sub	0	$printerdict
sub	18	$printerdict
sub	2	$printerdict
sub	24	$printerdict
sub	8	$printerdict
sub	a4	userdict
sub	b5	userdict
sub	bmpI	$idleTimeDict
sub	cleardictstack	userdict
sub	dictfull	errordict
sub	dictstackoverflow	errordict
sub	dictstackunderflow	errordict
sub	eprint	mydict
sub	execjob	serverdict
sub	execstackoverflow	errordict
sub	executive	userdict
sub	invalidaccess	errordict
sub	invalidexit	errordict
sub	invalidfileaccess	errordict
sub	invalidfont	errordict
sub	invalidrestore	errordict
sub	ioerror	errordict
sub	legal	userdict
sub	letter	userdict
sub	limitcheck	errordict
sub	nocurrentpoint	errordict
sub	rangecheck	errordict
sub	setpage	$printerdict
sub	stackoverflow	errordict
sub	stackunderflow	errordict
sub	syntaxerror	errordict
sub	timeout	errordict
sub	typecheck	errordict
sub	undefined	errordict
sub	undefinedfilename	errordict
sub	undefinedresult	errordict
sub	unmatchedmark	errordict
sub	unregistered	errordict
sub	VMerror	errordict
sub	***dict entry***	systemdict
sub	.error	systemdict
svlv	start	userdict
switchclose	setstreams	serverdict
switchclose	***dict entry***	serverdict
switchopen	***dict entry***	serverdict
switchsetting	checkquit	execdict
switchsetting	hashcommparams	serverdict
switchsetting	setstreams	serverdict
switchsetting	start	userdict
switchsetting	***dict entry***	statusdict
Symbol	***dict entry***	FontDirectory
syntaxerror	***dict entry***	errordict
systemdict	1	specialswitch
systemdict	intidleproc	serverdict
systemdict	***dict entry***	systemdict
timeout	***dict entry***	errordict
Times-Bold	***dict entry***	FontDirectory
Times-BoldItalic	***dict entry***	FontDirectory
Times-Italic	***dict entry***	FontDirectory
Times-Roman	***dict entry***	FontDirectory
token	***dict entry***	systemdict
tprint	arraytype	--dict
tprint	cvsprint	--dict
tprint	dicttype	--dict
tprint	filetype	--dict
tprint	fonttype	--dict
tprint	marktype	--dict
tprint	nametype	--dict
tprint	nulltype	--dict
tprint	operatortype	--dict
tprint	packedarraytype	--dict
tprint	savetype	--dict
tprint	stringtype	--dict
tprint	***dict entry***	--dict
transform	***dict entry***	systemdict
translate	***dict entry***	systemdict
transparent	setsccstreams	serverdict
transparent	***dict entry***	serverdict
true	setsccstreams	serverdict
true	***dict entry***	systemdict
truncate	***dict entry***	systemdict
type	findfont	systemdict
type	handleerror	errordict
type	stack	systemdict
type	typeprint	--dict
type	***dict entry***	systemdict
type	-	systemdict
type	-print	systemdict
typecheck	***dict entry***	errordict
typeprint	arraytype	--dict
typeprint	packedarraytype	--dict
typeprint	***dict entry***	--dict
typeprint	--	systemdict
undefined	***dict entry***	errordict
undefinedfilename	***dict entry***	errordict
undefinedresult	***dict entry***	errordict
unmatchedmark	***dict entry***	errordict
unregistered	***dict entry***	errordict
UseIdleTime	0	specialswitch
UseIdleTime	0	serverdict
UseIdleTime	1	serverdict
UseIdleTime	1	specialswitch
UseIdleTime	3	serverdict
UseIdleTime	intidleproc	serverdict
UseIdleTime	start	userdict

name	*where found*	*dict found in*	*name*	*where found*	*dict found in*
UseIdleTime	***dict entry***	userdict	#copies	letter	userdict
userdict	start	userdict	#copies	proc	$printerdict
userdict	***dict entry***	systemdict	#copies	***dict entry***	userdict
usertime	initprinter	serverdict	$error	appletalkopen	serverdict
usertime	printerstatus	serverdict	$error	dictfull	errordict
usertime	setrealdevice	serverdict	$error	dictstackoverflow	errordict
usertime	start	userdict	$error	dictstackunderflow	errordict
usertime	warmedup	serverdict	$error	execjob	serverdict
usertime	***dict entry***	systemdict	$error	execstackoverflow	errordict
version	executive	userdict	$error	executive	userdict
version	***dict entry***	systemdict	$error	handleerror	errordict
VMerror	***dict entry***	errordict	$error	invalidaccess	errordict
vmstatus	***dict entry***	systemdict	$error	invalidexit	errordict
waittimeout	***dict entry***	statusdict	$error	invalidfileaccess	errordict
warmedup	start	userdict	$error	invalidfont	errordict
warmedup	***dict entry***	serverdict	$error	invalidrestore	errordict
watchstreams	***dict entry***	serverdict	$error	ioerror	errordict
wcheck	***dict entry***	systemdict	$error	limitcheck	errordict
where	***dict entry***	systemdict	$error	nocurrentpoint	errordict
width	0	$printerdict	$error	rangecheck	errordict
width	18	$printerdict	$error	stackoverflow	errordict
width	2	$printerdict	$error	stackunderflow	errordict
width	24	$printerdict	$error	start	userdict
width	8	$printerdict	$error	syntaxerror	errordict
width	a4	userdict	$error	timeout	errordict
width	b5	userdict	$error	typecheck	errordict
width	legal	userdict	$error	undefined	errordict
width	letter	userdict	$error	undefinedfilename	errordict
width	setpage	$printerdict	$error	undefinedresult	errordict
width	***dict entry***	$printerdict	$error	unmatchedmark	errordict
widthshow	***dict entry***	systemdict	$error	unregistered	errordict
write	***dict entry***	systemdict	$error	VMerror	errordict
writehexstring	***dict entry***	systemdict	$error	***dict entry***	systemdict
writestring	1	specialswitch	$error	.error	systemdict
writestring	altprint	serverdict	$idleTimeDict	fontname	serverdict
writestring	***dict entry***	systemdict	$idleTimeDict	ReadIdleFonts	userdict
wtimeout	start	userdict	$idleTimeDict	UseIdleTime	userdict
wtimeout	***dict entry***	serverdict	$idleTimeDict	***dict entry***	userdict
xcheck	arraytype	==dict	$printerdict	0	$printerdict
xcheck	nametype	==dict	$printerdict	18	$printerdict
xcheck	packedarraytype	==dict	$printerdict	2	$printerdict
xcheck	***dict entry***	systemdict	$printerdict	24	$printerdict
xoffset	0	$printerdict	$printerdict	8	$printerdict
xoffset	18	$printerdict	$printerdict	a4	userdict
xoffset	2	$printerdict	$printerdict	b5	userdict
xoffset	24	$printerdict	$printerdict	legal	userdict
xoffset	8	$printerdict	$printerdict	letter	userdict
xoffset	a4	userdict	$printerdict	setnulldevice	serverdict
xoffset	b5	userdict	$printerdict	setpage	$printerdict
xoffset	legal	userdict	$printerdict	setrealdevice	serverdict
xoffset	letter	userdict	$printerdict	***dict entry***	userdict
xoffset	setpage	$printerdict	.error	***dict entry***	systemdict
xoffset	***dict entry***	$printerdict	/a	StandardEncoding	systemdict
xor	***dict entry***	systemdict	/A	StandardEncoding	systemdict
yoffset	0	$printerdict	/abort	528	mydict
yoffset	18	$printerdict	/abort	printererror	statusdict
yoffset	2	$printerdict	/acute	StandardEncoding	systemdict
yoffset	24	$printerdict	/AE	StandardEncoding	systemdict
yoffset	8	$printerdict	/ae	StandardEncoding	systemdict
yoffset	a4	userdict	/altflag	appletalkopen	serverdict
yoffset	b5	userdict	/altflag	setsccstreams	serverdict
yoffset	legal	userdict	/altin	setsccstreams	serverdict
yoffset	letter	userdict	/altin	stopPred	$idleTimeDict
yoffset	setpage	$printerdict	/altin	watchstreams	serverdict
yoffset	***dict entry***	$printerdict	/altname	setsccstreams	serverdict
#copies	0	$printerdict	/altname	stopPred	$idleTimeDict
#copies	18	$printerdict	/altname	watchstreams	serverdict
#copies	2	$printerdict	/altout	setsccstreams	serverdict
#copies	24	$printerdict	/altout	stopPred	$idleTimeDict
#copies	8	$printerdict	/altout	watchstreams	serverdict
#copies	a4	userdict	/ampersand	StandardEncoding	systemdict
#copies	b5	userdict	/asciicircum	StandardEncoding	systemdict
#copies	dopage	$printerdict	/asciitilde	StandardEncoding	systemdict
#copies	legal	userdict	/asterisk	StandardEncoding	systemdict
			/at	StandardEncoding	systemdict

name	*where found*	*dict found in*
/B	StandardEncoding	systemdict
/b	StandardEncoding	systemdict
/backslash	StandardEncoding	systemdict
/bar	StandardEncoding	systemdict
/baud25	setsccstreams	serverdict
/baud9	setsccstreams	serverdict
/bits	printererror	statusdict
/braceleft	StandardEncoding	systemdict
/braceright	StandardEncoding	systemdict
/bracketleft	StandardEncoding	systemdict
/bracketright	StandardEncoding	systemdict
/breve	StandardEncoding	systemdict
/bullet	StandardEncoding	systemdict
/c	StandardEncoding	systemdict
/C	StandardEncoding	systemdict
/caron	StandardEncoding	systemdict
/cedilla	StandardEncoding	systemdict
/cent	StandardEncoding	systemdict
/circumflex	StandardEncoding	systemdict
/colon	StandardEncoding	systemdict
/comma	StandardEncoding	systemdict
/command	dictfull	errordict
/command	dictstackoverflow	errordict
/command	dictstackunderflow	errordict
/command	execstackoverflow	errordict
/command	handleerror	errordict
/command	invalidaccess	errordict
/command	invalidexit	errordict
/command	invalidfileaccess	errordict
/command	invalidfont	errordict
/command	invalidrestore	errordict
/command	ioerror	errordict
/command	limitcheck	errordict
/command	nocurrentpoint	errordict
/command	rangecheck	errordict
/command	stackoverflow	errordict
/command	stackunderflow	errordict
/command	syntaxerror	errordict
/command	timeout	errordict
/command	typecheck	errordict
/command	undefined	errordict
/command	undefinedfilename	errordict
/command	undefinedresult	errordict
/command	unmatchedmark	errordict
/command	unregistered	errordict
/command	VMerror	errordict
/command	.error	systemdict
/commhash	appletalkopen	serverdict
/commhash	setsccstreams	serverdict
/Courier	findfont	systemdict
/Courier	idleArry	$idleTimeDict
/Courier	ROMnames	$idleTimeDict
/Courier-Bold	idleArry	$idleTimeDict
/Courier-Bold	ROMnames	$idleTimeDict
/Courier-BoldOblique	ROMnames	$idleTimeDict
/Courier-Oblique	ROMnames	$idleTimeDict
/cp	tprint	--dict
/cp	--	systemdict
/currency	StandardEncoding	systemdict
/D	StandardEncoding	systemdict
/d	StandardEncoding	systemdict
/dagger	StandardEncoding	systemdict
/daggerdbl	StandardEncoding	systemdict
/debugmode	setsccstreams	serverdict
/dictfull	dictfull	errordict
/dictstackoverflow	dictstackoverflow	errordict
/dictstackunderflow	dictstackunderflow	errordict
/dieresis	StandardEncoding	systemdict
/doclose	execjob	serverdict
/doclose	executive	userdict
/dollar	StandardEncoding	systemdict
/dotaccent	StandardEncoding	systemdict
/dotlessi	StandardEncoding	systemdict
/dstack	dictfull	errordict
/dstack	dictstackoverflow	errordict
/dstack	dictstackunderflow	errordict
/dstack	execstackoverflow	errordict
/dstack	invalidaccess	errordict
/dstack	invalidexit	errordict
/dstack	invalidfileaccess	errordict
/dstack	invalidfont	errordict
/dstack	invalidrestore	errordict
/dstack	ioerror	errordict
/dstack	limitcheck	errordict
/dstack	nocurrentpoint	errordict
/dstack	rangecheck	errordict
/dstack	stackoverflow	errordict
/dstack	stackunderflow	errordict
/dstack	syntaxerror	errordict
/dstack	timeout	errordict
/dstack	typecheck	errordict
/dstack	undefined	errordict
/dstack	undefinedfilename	errordict
/dstack	undefinedresult	errordict
/dstack	unmatchedmark	errordict
/dstack	unregistered	errordict
/dstack	VMerror	errordict
/dstack	.error	systemdict
/dstackarray	dictfull	errordict
/dstackarray	dictstackoverflow	errordict
/dstackarray	dictstackunderflow	errordict
/dstackarray	execstackoverflow	errordict
/dstackarray	invalidaccess	errordict
/dstackarray	invalidexit	errordict
/dstackarray	invalidfileaccess	errordict
/dstackarray	invalidfont	errordict
/dstackarray	invalidrestore	errordict
/dstackarray	ioerror	errordict
/dstackarray	limitcheck	errordict
/dstackarray	nocurrentpoint	errordict
/dstackarray	rangecheck	errordict
/dstackarray	stackoverflow	errordict
/dstackarray	stackunderflow	errordict
/dstackarray	syntaxerror	errordict
/dstackarray	timeout	errordict
/dstackarray	typecheck	errordict
/dstackarray	undefined	errordict
/dstackarray	undefinedfilename	errordict
/dstackarray	undefinedresult	errordict
/dstackarray	unmatchedmark	errordict
/dstackarray	unregistered	errordict
/dstackarray	VMerror	errordict
/dstackarray	.error	systemdict
/E	StandardEncoding	systemdict
/e	StandardEncoding	systemdict
/eeinfo	bmpI	$idleTimeDict
/eeinfo	boundsCheck	$idleTimeDict
/eight	StandardEncoding	systemdict
/eindex	eprint	mydict
/eindex	printererror	statusdict
/ellipsis	StandardEncoding	systemdict
/emdash	StandardEncoding	systemdict
/endash	StandardEncoding	systemdict
/equal	StandardEncoding	systemdict
/errorname	dictfull	errordict
/errorname	dictstackoverflow	errordict
/errorname	dictstackunderflow	errordict
/errorname	execstackoverflow	errordict
/errorname	executive	userdict
/errorname	invalidaccess	errordict
/errorname	invalidexit	errordict
/errorname	invalidfileaccess	errordict
/errorname	invalidfont	errordict
/errorname	invalidrestore	errordict
/errorname	ioerror	errordict
/errorname	limitcheck	errordict
/errorname	nocurrentpoint	errordict
/errorname	rangecheck	errordict
/errorname	stackoverflow	errordict
/errorname	stackunderflow	errordict

name	*where found*	*dict found in*
/errorname	syntaxerror	errordict
/errorname	timeout	errordict
/errorname	typecheck	errordict
/errorname	undefined	errordict
/errorname	undefinedfilename	errordict
/errorname	undefinedresult	errordict
/errorname	unmatchedmark	errordict
/errorname	unregistered	errordict
/errorname	VMerror	errordict
/errorname	.error	systemdict
/estack	dictfull	errordict
/estack	dictstackoverflow	errordict
/estack	dictstackunderflow	errordict
/estack	execstackoverflow	errordict
/estack	invalidaccess	errordict
/estack	invalidexit	errordict
/estack	invalidfileaccess	errordict
/estack	invalidfont	errordict
/estack	invalidrestore	errordict
/estack	ioerror	errordict
/estack	limitcheck	errordict
/estack	nocurrentpoint	errordict
/estack	rangecheck	errordict
/estack	stackoverflow	errordict
/estack	stackunderflow	errordict
/estack	syntaxerror	errordict
/estack	timeout	errordict
/estack	typecheck	errordict
/estack	undefined	errordict
/estack	undefinedfilename	errordict
/estack	undefinedresult	errordict
/estack	unmatchedmark	errordict
/estack	unregistered	errordict
/estack	VMerror	errordict
/estack	.error	systemdict
/estackarray	dictfull	errordict
/estackarray	dictstackoverflow	errordict
/estackarray	dictstackunderflow	errordict
/estackarray	execstackoverflow	errordict
/estackarray	invalidaccess	errordict
/estackarray	invalidexit	errordict
/estackarray	invalidfileaccess	errordict
/estackarray	invalidfont	errordict
/estackarray	invalidrestore	errordict
/estackarray	ioerror	errordict
/estackarray	limitcheck	errordict
/estackarray	nocurrentpoint	errordict
/estackarray	rangecheck	errordict
/estackarray	stackoverflow	errordict
/estackarray	stackunderflow	errordict
/estackarray	syntaxerror	errordict
/estackarray	timeout	errordict
/estackarray	typecheck	errordict
/estackarray	undefined	errordict
/estackarray	undefinedfilename	errordict
/estackarray	undefinedresult	errordict
/estackarray	unmatchedmark	errordict
/estackarray	unregistered	errordict
/estackarray	VMerror	errordict
/estackarray	.error	systemdict
/exclam	StandardEncoding	systemdict
/exclamdown	StandardEncoding	systemdict
/execdepth	executive	userdict
/execdepth	start	userdict
/execjob	start	userdict
/execstackoverflow	execstackoverflow	errordict
/f	StandardEncoding	systemdict
/F	StandardEncoding	systemdict
/fi	StandardEncoding	systemdict
/five	StandardEncoding	systemdict
/fl	StandardEncoding	systemdict
/florin	StandardEncoding	systemdict
/four	StandardEncoding	systemdict
/fraction	StandardEncoding	systemdict
/g	StandardEncoding	systemdict
/G	StandardEncoding	systemdict
/germandbls	StandardEncoding	systemdict
/grave	StandardEncoding	systemdict
/greater	StandardEncoding	systemdict
/guillemotleft	StandardEncoding	systemdict
/guillemotright	StandardEncoding	systemdict
/guilsinglleft	StandardEncoding	systemdict
/guilsinglright	StandardEncoding	systemdict
/h	StandardEncoding	systemdict
/H	StandardEncoding	systemdict
/handleerror	execjob	serverdict
/handleerror	handleerror	systemdict
/height	0	$printerdict
/height	18	$printerdict
/height	2	$printerdict
/height	24	$printerdict
/height	8	$printerdict
/height	a4	userdict
/height	b5	userdict
/height	legal	userdict
/height	letter	userdict
/height	setpage	$printerdict
/Helvetica	idleArry	$idleTimeDict
/Helvetica	ROMnames	$idleTimeDict
/Helvetica-Bold	idleArry	$idleTimeDict
/Helvetica-Bold	ROMnames	$idleTimeDict
/Helvetica-BoldOblique	ROMnames	$idleTimeDict
/Helvetica-Oblique	ROMnames	$idleTimeDict
/hungarumlaut	StandardEncoding	systemdict
/hyphen	StandardEncoding	systemdict
/i	StandardEncoding	systemdict
/I	StandardEncoding	systemdict
/idleArry	ReadIdleFonts	userdict
/idleI	bmpI	$idleTimeDict
/idleI	ReadIdleFonts	userdict
/idleI	UseIdleTime	userdict
/idleproc	0	serverdict
/idleproc	1	specialswitch
/idleproc	1	serverdict
/idleproc	3	serverdict
/idleproc	batchidleproc	serverdict
/idleproc	intidleproc	serverdict
/idleStr	UseIdleTime	userdict
/idleStrI	UseIdleTime	userdict
/initializing	appletalkopen	serverdict
/initializing	dictfull	errordict
/initializing	dictstackoverflow	errordict
/initializing	dictstackunderflow	errordict
/initializing	execstackoverflow	errordict
/initializing	invalidaccess	errordict
/initializing	invalidexit	errordict
/initializing	invalidfileaccess	errordict
/initializing	invalidfont	errordict
/initializing	invalidrestore	errordict
/initializing	ioerror	errordict
/initializing	limitcheck	errordict
/initializing	nocurrentpoint	errordict
/initializing	rangecheck	errordict
/initializing	stackoverflow	errordict
/initializing	stackunderflow	errordict
/initializing	start	userdict
/initializing	syntaxerror	errordict
/initializing	timeout	errordict
/initializing	typecheck	errordict
/initializing	undefined	errordict
/initializing	undefinedfilename	errordict
/initializing	undefinedresult	errordict
/initializing	unmatchedmark	errordict
/initializing	unregistered	errordict
/initializing	VMerror	errordict
/initializing	.error	systemdict
/invalidaccess	invalidaccess	errordict
/invalidexit	invalidexit	errordict
/invalidfileaccess	invalidfileaccess	errordict
/invalidfont	invalidfont	errordict

name	*where found*	*dict found in*
/invalidrestore	invalidrestore	errordict
/ioerror	ioerror	errordict
/J	StandardEncoding	systemdict
/j	StandardEncoding	systemdict
/jobname	execjob	serverdict
/jobsource	setstreams	serverdict
/jobsource	stopPred	$idleTimeDict
/jobsource	watchstreams	serverdict
/jobstate	0	$printerdict
/jobstate	18	$printerdict
/jobstate	2	$printerdict
/jobstate	24	$printerdict
/jobstate	8	$printerdict
/jobstate	a4	userdict
/jobstate	b5	userdict
/jobstate	dopage	$printerdict
/jobstate	eflush	mydict
/jobstate	execjob	serverdict
/jobstate	legal	userdict
/jobstate	letter	userdict
/jobstate	proc	$printerdict
/jobstate	start	userdict
/jobstate	stopPred	$idleTimeDict
/jobstate	watchstreams	serverdict
/k	StandardEncoding	systemdict
/K	StandardEncoding	systemdict
/L	StandardEncoding	systemdict
/l	StandardEncoding	systemdict
/laststat	printererror	statusdict
/less	StandardEncoding	systemdict
/limitcheck	limitcheck	errordict
/Lslash	StandardEncoding	systemdict
/lslash	StandardEncoding	systemdict
/M	StandardEncoding	systemdict
/m	StandardEncoding	systemdict
/macron	StandardEncoding	systemdict
/manualfeed	528	mydict
/manualfeedtimeout	settimeouts	serverdict
/mtx	setnulldevice	serverdict
/n	StandardEncoding	systemdict
/N	StandardEncoding	systemdict
/newerror	dictfull	errordict
/newerror	dictstackoverflow	errordict
/newerror	dictstackunderflow	errordict
/newerror	execstackoverflow	errordict
/newerror	executive	userdict
/newerror	handleerror	errordict
/newerror	invalidaccess	errordict
/newerror	invalidexit	errordict
/newerror	invalidfileaccess	errordict
/newerror	invalidfont	errordict
/newerror	invalidrestore	errordict
/newerror	ioerror	errordict
/newerror	limitcheck	errordict
/newerror	nocurrentpoint	errordict
/newerror	rangecheck	errordict
/newerror	stackoverflow	errordict
/newerror	stackunderflow	errordict
/newerror	syntaxerror	errordict
/newerror	timeout	errordict
/newerror	typecheck	errordict
/newerror	undefined	errordict
/newerror	undefinedfilename	errordict
/newerror	undefinedresult	errordict
/newerror	unmatchedmark	errordict
/newerror	unregistered	errordict
/newerror	VMerror	errordict
/newerror	.error	systemdict
/nine	StandardEncoding	systemdict
/nocurrentpoint	nocurrentpoint	errordict
/ntrys	printererror	statusdict
/numbersign	StandardEncoding	systemdict
/O	StandardEncoding	systemdict
/o	StandardEncoding	systemdict
/OE	StandardEncoding	systemdict
/oe	StandardEncoding	systemdict
/ogonek	StandardEncoding	systemdict
/one	StandardEncoding	systemdict
/ordfeminine	StandardEncoding	systemdict
/ordmasculine	StandardEncoding	systemdict
/oslash	StandardEncoding	systemdict
/Oslash	StandardEncoding	systemdict
/ostack	dictfull	errordict
/ostack	dictstackoverflow	errordict
/ostack	dictstackunderflow	errordict
/ostack	execstackoverflow	errordict
/ostack	invalidaccess	errordict
/ostack	invalidexit	errordict
/ostack	invalidfileaccess	errordict
/ostack	invalidfont	errordict
/ostack	invalidrestore	errordict
/ostack	ioerror	errordict
/ostack	limitcheck	errordict
/ostack	nocurrentpoint	errordict
/ostack	rangecheck	errordict
/ostack	stackoverflow	errordict
/ostack	stackunderflow	errordict
/ostack	syntaxerror	errordict
/ostack	timeout	errordict
/ostack	typecheck	errordict
/ostack	undefined	errordict
/ostack	undefinedfilename	errordict
/ostack	undefinedresult	errordict
/ostack	unmatchedmark	errordict
/ostack	unregistered	errordict
/ostack	VMerror	errordict
/ostack	.error	systemdict
/ostackarray	dictfull	errordict
/ostackarray	dictstackoverflow	errordict
/ostackarray	dictstackunderflow	errordict
/ostackarray	execstackoverflow	errordict
/ostackarray	invalidaccess	errordict
/ostackarray	invalidexit	errordict
/ostackarray	invalidfileaccess	errordict
/ostackarray	invalidfont	errordict
/ostackarray	invalidrestore	errordict
/ostackarray	ioerror	errordict
/ostackarray	limitcheck	errordict
/ostackarray	nocurrentpoint	errordict
/ostackarray	rangecheck	errordict
/ostackarray	stackoverflow	errordict
/ostackarray	stackunderflow	errordict
/ostackarray	syntaxerror	errordict
/ostackarray	timeout	errordict
/ostackarray	typecheck	errordict
/ostackarray	undefined	errordict
/ostackarray	undefinedfilename	errordict
/ostackarray	undefinedresult	errordict
/ostackarray	unmatchedmark	errordict
/ostackarray	unregistered	errordict
/ostackarray	VMerror	errordict
/ostackarray	.error	systemdict
/p	StandardEncoding	systemdict
/P	StandardEncoding	systemdict
/paragraph	StandardEncoding	systemdict
/parenleft	StandardEncoding	systemdict
/parenright	StandardEncoding	systemdict
/parity25	setsccstreams	serverdict
/parity9	setsccstreams	serverdict
/percent	StandardEncoding	systemdict
/period	StandardEncoding	systemdict
/periodcentered	StandardEncoding	systemdict
/perthousand	StandardEncoding	systemdict
/plus	StandardEncoding	systemdict
/print	1	specialswitch
/print	intidleproc	serverdict
/printererror	start	userdict
/printpageflag	0	specialswitch
/proc	0	$printerdict
/proc	18	$printerdict

name	*where found*	*dict found in*	*name*	*where found*	*dict found in*
/proc	2	$printerdict	/three	StandardEncoding	systemdict
/proc	24	$printerdict	/tilde	StandardEncoding	systemdict
/proc	8	$printerdict	/timeout	execjob	serverdict
/proc	a4	userdict	/timeout	timeout	errordict
/proc	b5	userdict	/Times-Bold	idleArry	$idleTimeDict
/proc	legal	userdict	/Times-Bold	ROMnames	$idleTimeDict
/proc	letter	userdict	/Times-BoldItalic	ROMnames	$idleTimeDict
/proc	setpage	$printerdict	/Times-Italic	ROMnames	$idleTimeDict
/prompt	executive	userdict	/Times-Roman	idleArry	$idleTimeDict
/Q	StandardEncoding	systemdict	/Times-Roman	ROMnames	$idleTimeDict
/q	StandardEncoding	systemdict	/transparent	setsccstreams	serverdict
/question	StandardEncoding	systemdict	/two	StandardEncoding	systemdict
/questiondown	StandardEncoding	systemdict	/typecheck	typecheck	errordict
/quitflag	execjob	serverdict	/U	StandardEncoding	systemdict
/quitflag	executive	userdict	/u	StandardEncoding	systemdict
/quitflag	quit	userdict	/undefined	undefined	errordict
/quotedbl	StandardEncoding	systemdict	/undefinedfilename	executive	userdict
/quotedblbase	StandardEncoding	systemdict	/undefinedfilename	undefinedfilename	errordict
/quotedblleft	StandardEncoding	systemdict	/undefinedresult	undefinedresult	errordict
/quotedblright	StandardEncoding	systemdict	/underscore	StandardEncoding	systemdict
/quoteleft	StandardEncoding	systemdict	/unmatchedmark	unmatchedmark	errordict
/quoteright	StandardEncoding	systemdict	/unregistered	unregistered	errordict
/quotesinglbase	StandardEncoding	systemdict	/V	StandardEncoding	systemdict
/quotesingle	StandardEncoding	systemdict	/v	StandardEncoding	systemdict
/R	StandardEncoding	systemdict	/VMerror	dictfull	errordict
/r	StandardEncoding	systemdict	/VMerror	dictstackoverflow	errordict
/rangecheck	rangecheck	errordict	/VMerror	dictstackunderflow	errordict
/report	printererror	statusdict	/VMerror	execstackoverflow	errordict
/ring	StandardEncoding	systemdict	/VMerror	invalidaccess	errordict
/s	StandardEncoding	systemdict	/VMerror	invalidexit	errordict
/S	StandardEncoding	systemdict	/VMerror	invalidfileaccess	errordict
/saveswitch	checkquit	execdict	/VMerror	invalidfont	errordict
/saveswitch	execjob	serverdict	/VMerror	invalidrestore	errordict
/saveswitch	setstreams	serverdict	/VMerror	ioerror	errordict
/saveswitch	start	userdict	/VMerror	limitcheck	errordict
/sccok	setsccstreams	serverdict	/VMerror	nocurrentpoint	errordict
/section	StandardEncoding	systemdict	/VMerror	rangecheck	errordict
/semicolon	StandardEncoding	systemdict	/VMerror	stackoverflow	errordict
/sendctrld	appletalkopen	serverdict	/VMerror	stackunderflow	errordict
/sendctrld	setsccstreams	serverdict	/VMerror	syntaxerror	errordict
/server	start	userdict	/VMerror	timeout	errordict
/setidlefonts	ReadIdleFonts	userdict	/VMerror	typecheck	errordict
/seven	StandardEncoding	systemdict	/VMerror	undefined	errordict
/six	StandardEncoding	systemdict	/VMerror	undefinedfilename	errordict
/slash	StandardEncoding	systemdict	/VMerror	undefinedresult	errordict
/space	StandardEncoding	systemdict	/VMerror	unmatchedmark	errordict
/stackoverflow	stackoverflow	errordict	/VMerror	unregistered	errordict
/stackunderflow	stackunderflow	errordict	/VMerror	VMerror	errordict
/stat	printererror	statusdict	/VMerror	.error	systemdict
/stdin	appletalkopen	serverdict	/W	StandardEncoding	systemdict
/stdin	setsccstreams	serverdict	/w	StandardEncoding	systemdict
/stdin	stopPred	$idleTimeDict	/waittimeout	settimeouts	serverdict
/stdin	watchstreams	serverdict	/watchstreams	0	serverdict
/stdname	appletalkopen	serverdict	/watchstreams	0	specialswitch
/stdname	setsccstreams	serverdict	/watchstreams	1	serverdict
/stdname	stopPred	$idleTimeDict	/watchstreams	1	specialswitch
/stdname	watchstreams	serverdict	/watchstreams	3	serverdict
/stdout	appletalkopen	serverdict	/watchstreams	intidleproc	serverdict
/stdout	setsccstreams	serverdict	/width	0	$printerdict
/stdout	stopPred	$idleTimeDict	/width	18	$printerdict
/stdout	watchstreams	serverdict	/width	2	$printerdict
/sterling	StandardEncoding	systemdict	/width	24	$printerdict
/stmtfile	executive	userdict	/width	8	$printerdict
/stop	start	userdict	/width	a4	userdict
/stopPred	UseIdleTime	userdict	/width	b5	userdict
/stringtype	findfont	systemdict	/width	legal	userdict
/stringtype	handleerror	errordict	/width	letter	userdict
/stringtype	stack	systemdict	/width	setpage	$printerdict
/stringtype	=	systemdict	/wtimeout	start	userdict
/stringtype	=print	systemdict	/X	StandardEncoding	systemdict
/svlv	start	userdict	/x	StandardEncoding	systemdict
/Symbol	ROMnames	$idleTimeDict	/xoffset	0	$printerdict
/syntaxerror	syntaxerror	errordict	/xoffset	18	$printerdict
/T	StandardEncoding	systemdict	/xoffset	2	$printerdict
/t	StandardEncoding	systemdict	/xoffset	24	$printerdict

name	*where found*	*dict found in*
/xoffset	8	$printerdict
/xoffset	a4	userdict
/xoffset	b5	userdict
/xoffset	legal	userdict
/xoffset	letter	userdict
/xoffset	setpage	$printerdict
/y	StandardEncoding	systemdict
/Y	StandardEncoding	systemdict
/yen	StandardEncoding	systemdict
/yoffset	0	$printerdict
/yoffset	18	$printerdict
/yoffset	2	$printerdict
/yoffset	24	$printerdict
/yoffset	8	$printerdict
/yoffset	a4	userdict
/yoffset	b5	userdict
/yoffset	legal	userdict
/yoffset	letter	userdict
/yoffset	setpage	$printerdict
/Z	StandardEncoding	systemdict
/z	StandardEncoding	systemdict
/zero	StandardEncoding	systemdict
/.notdef	StandardEncoding	systemdict
-	execjob	serverdict
-	initprinter	serverdict
-	printerstatus	serverdict
-	setrealdevice	serverdict
-	warmedup	serverdict
-	***dict entry***	systemdict
-print	***dict entry***	systemdict
-string	cvsprint	--dict
-string	operatortype	--dict
-string	***dict entry***	systemdict
--	pstack	userdict
--	Run	systemdict
--	***dict entry***	systemdict
--dict	***dict entry***	--dict
--dict	--	systemdict
[	***dict entry***	systemdict
]	***dict entry***	systemdict

Index

Learn the secrets of the PostScript masters!

What does the server loop really do and how does it control job execution? What is the ==dict; does it have a special structure; where does it reside? How is the stopped operator used to control your job execution? Idle time font caching; what is it and how does it work? How are binding and immediately evaluated names used to improve efficiency?

Lots of questions? *Inside PostScript* has the answers.

Inside PostScript documents the internal procedures and structure of Adobe's PostScript interpreter showing programmers the techniques used by Adobe to control PostScript printers. Detailed discussions of error handling, printer control, job execution and file I/O are a few of the topics presented. With this insight, you are able to write better drivers for your applications, and create better debugging tools for code development and field support.

For the beginning PostScript programmer, the many code examples presented allow complex applications to be developed with less learning time. For the experienced programmer, new insight is gained into the operation of the PostScript interpreter.

"*Inside PostScript* will measurably improve the gereral level of skills and understanding in the PostScript progrmming community."

David Holzgang, author *Understanding PostScript Programming and PostScript Programmer's Reference Guide*

"Good reading for advanced PostScript programmers."

Patrick Wood, editor *The PostScript Language Journal*