Design of the DECprint Common Printer Supervisor for VMS Systems

Richard Landau and Alan Guenther

1 Abstract

DECprint Printing Services software controls a variety of printer features for a wide range of printers. It supports several different page description languages, handles multiple media simultaneously, and uses different I/O interconnections and communication protocols. Operating within the VMS printing environment, it implements a large number of user-specified options to the PRINT command. DECprint Printing Services functions as the supervisor in the VMS printing system for all PostScript printers supplied by Digital. The common printer supervisor has an especially flexible internal structure and processing method to serve complex printing environments. [The DECprint paper starts here.]

The increasing variety and complexity of printing devices in the last decade have strained the abilities of operating systems to support them. Users demand access to, and control over, the increasingly sophisticated features of their printers. At the same time, application programming resources are stretched by the requirement to support various devices and features. Modern operating systems include printing systems that support printers and insulate applications from many details of printing.

DECprint Printing Services software was designed to handle a wide variety of printers, with a range of I/O connections, media handling capabilities, finishing equipment, data syntaxes, and so forth. It provides the controlling software that supports the full range of Digital printers capable of printing PostScript documents.

DECprint Printing Services functions as a component of the VMS printing system at the level of printer supervisor, called symbiont in VMS terminology. The supervisor is known within Digital as the DECprint common printer supervisor or common print symbiont (CPS). It is called common because it replaces a number of different symbionts and is common to a range of printers. CPS is a completely new program developed by the Video, Image and Print Systems Group.

This paper explores the environment in which printing systems now reside. It describes the structure and functions of DECprint Printing Services and the design of CPS, focusing on its capabilities within the VMS system. The paper then discusses the operation of the VMS printing system and the enhanced printing environment made possible by CPS.

2 Printing System Dimensions

A printing system is the set of software and hardware components through which print requests pass from the time the user decides to print a document until the appropriate hard copy arrives.

The variety of printing devices in use is a challenge for the printing system and for application programmers. We use the word "printer" in this article to imply the full range of output devices that are attached to systems and networks. A system today must support a wide number of dimensions: marking technologies, media, medium sizes, speeds, transmission rates, and interconnects.

3 The DECprint Model of Printing

The DECprint model of printing is composed of several layers. Each layer has defined functions and I/O interfaces. The layers of the DECprint model and their relationships to VMS and CPS are shown in Figure 1. This model of printing describes a useful structure with consistent functions and responsibilities.

- Application. An application program creates information that the user may want to print. All types of applications fit into the model at this level, from data processing programs and simple text editors to high-quality document formatting and publishing applications. The application may present a printing interface directly to the user, or may create a final form document from which the user can access other printing interfaces.

- User printing interface. A user expresses the desire to print through a user interface to the printing system. The interface may be oriented to written commands, to user selection of menu choices, or to a point-and-select graphical interface.

- Job submission interface. User interface programs communicate with the lower levels of the printing system through an application programming interface (API) to the print client. The API contains full capabilities for creating, destroying, and managing print jobs of all types. The job submission interface may be operating system-specific or may be based on emerging standards for network printing.

- Print client. The client accepts requests through its API, performs defaulting for the user, assists in selecting the correct print service, gathers the print instructions and document files, and submits the job to the print service. The protocol used to submit the job may be operating system-specific or may be based on emerging standards for network printing. The print service may be local to the print client (and the user), or it may be located elsewhere in the network.

- Print service. The print service is a convenient abstraction that includes the print spooler and all subsequent layers in the execution of the print job, for some set of physical printers. Printers are often grouped together based on their static characteristics, such as type of printer, printer data syntax, and default media.

- Print spooler. The print spooler accepts the print job from the client, spools the files and queues the job for later execution if necessary, and then schedules the job for execution. If the job requires resources that are not immediately available, human intervention may be necessary. For example, if a job requires a special print medium, then an operator or other printer attendant must provide the medium for the printer. If the job requires a special font, the spooler may be able to obtain the font from a library without human intervention.

- Printer supervisor. The supervisor directly controls the printer. It interprets the print instructions for the job, manages the printer and its finishing equipment, and writes the document data to the page description language (PDL) interpreter. It also monitors the status of the printer, supplies some resources on demand, and responds to error conditions. On the VMS operating system, the printer supervisor is called a symbiont; on ULTRIX and UNIX systems, a daemon.

- PDL interpreter. Generally, final form document data is written in a data syntax intended for printing, but it is not in the native form required by the marking engine. A PDL interpreter transforms the printer language into the lower-level form for the marking engine. For example, in a typical laser printer, a PostScript interpreter transforms the PostScript language into a device-level bit map and media control instructions for the print engine. In a simpler impact printer, the controller turns characters and control sequences into pin timing and paper movement instructions.

- Marking engine. The marking engine consists of the media transport and printing mechanisms, generally controlled at a low level. Marking may be done by a wide spectrum of technologies, and the media used may also vary widely. For the most part, descriptions in this paper use raster devices such as laser printers as examples.

- Finishing equipment. The overall printing system includes finishing options that are not often considered part of the (largely electronic) printing system. Currently affordable components of the printing system are typically automated. For example, several years ago duplex (two-sided) printing was not economical for most office applications; today it is, and many office printers include this finishing feature.

Stapling, on the other hand, is still not economical for most office applications, though it is implemented in many high-end production printers.

Implementations of the model in various operating systems and printers may express the layers differently, sometimes skipping certain layers. The VMS printing system contains components at most levels of the DECprint model. The DECprint common printer supervisor (CPS) operates within the VMS system, as indicated in Figure 1. We designed CPS to satisfy the requirements and projected needs of users, system managers, and programmers. In the next section we discuss the design of CPS.

Sharing Devices

Printers are often shared, especially high-end or specialized, expensive devices. Since shared printers are not always immediately available to the user or application program, the printing system is required to hold jobs for printing later. The system must be able to store the user's instructions for printing, along with the contents of the document, until they are needed.

Insulating the Application from Details

A printing system insulates applications from the details of printing devices. For example, DECprint Printing Services provides communications mechanisms and protocols, determines whether a shared device is currently busy, and sometimes translates printer data syntax.

Application programmers generally prefer to deal with as few external interfaces as needed to perform the task. Thus it is desirable to minimize the number of different classes of printing devices while maximizing the variety and flexibility of printing devices. The DECprint architecture specifies that the printing system take responsibility for matching the needs of the application to the capabilities of the output device, whenever possible. For example, a printing system might need the ability to transform the printer data stream from a data syntax used by the application to a data syntax used by the printer. Hidden transformation makes the system easier for applications to use. DECprint Printing Services provides a certain number of printer data syntax transformations of this type, from languages such as DEC PPL3 (which is commonly referred to as "ANSI" within Digital) and ReGIS to PostScript, and from PostScript to printer bit maps.

4 Internal Structure of CPS

In designing CPS, our primary goal was to create a flexible system that would handle all the printer features we could foresee and many that we could not foresee, a system that could be modified as needed to handle not just new printers but new classes of printers. CPS is capable of managing a wide variety of character, line, page, and document printers.

To create a flexible printing system, we needed to design a highly modular internal structure. This internal structure combines modules into sequences at several levels to provide a general framework for controlling and manipulating I/O devices.

At the bottom level of the structure are filter modules, which are lightweight, independently schedulable subprocesses within a VMS process. Filter modules communicate with each other by means of I/O routines and a shared data structure containing job information. Pointers to the I/O routines and shared data are supplied in the invocation of the filter module. The effect of the stream I/O routines is much like that of pipes in the UNIX operating systems.

At the next higher level is a set of communicating filter modules; each stream of filter modules is called a job step. Finally, a module called the print job analyzer combines a sequence of job steps to handle a complete print job.

Filter Modules and Job Steps

Filter modules can read input from a preceding filter module and write data to a succeeding filter module. Filter modules may perform functions such as reading a file, converting carriage control, translating data syntax, or writing data to the printer. A filter module receives as arguments an input stream and an output stream, like a UNIX process, and a shared data structure, unlike a UNIX process. A simple filter module reads data from the input stream, processes data, and writes data to the output stream.

A filter module may condition its operation based on information from the shared data structure or the contents of the data stream. For example, a translator filter module might format data based on the page size, margins, and aspect ratio specified in the shared data structure, or based on control sequences in the data stream, or both.

Not all filter modules use the input or output streams. The file reader filter module reads from the file instead of the input stream. Similarly, the device output module writes to the printer instead of the output stream.

A job step is a set of filter modules piped together to perform one complete subtask. A subtask may be as simple as "create a separator page" or as complex as the sequence "read a file, perform carriage control conversion, add /HEADER, translate from ANSI data syntax to PostScript, and write the result to the printer." A print job is a set of job steps that performs all functions the user requests explicitly or implicitly. The CPS facility that translates selected printer data syntaxes into the PostScript language is discussed in the section Data Syntax Translation.

Print Job Analyzers

To simplify the addition of new printers and new classes of printers, CPS contains a software structure that corresponds to the hardware mechanisms of a printer.

A print job analyzer (PJA) determines which job steps are required to process a job. CPS includes a separate print job analyzer for each major class of printer that it supports: serial PostScript, PrintServer, and LN03 Image printer devices. When the symbiont begins execution, a PJA is chosen based on the type of device associated with the queue. This PJA is used until the symbiont is stopped. If a terminal device, such as a TT or TX or LT device, is associated with the queue, then the PJA for a serial device is invoked. If an LD device is used, then the PJA for an LN03Q printer is chosen. Otherwise, the PJA associated with PrintServer devices is used.

Each PJA contains a list of all job steps required to execute a job on the class of printers it supports. The PJA selects the job steps it needs from this list, depending upon the instructions received from the queue manager.

Job steps are linked together. The first job step chosen by the PJA is linked to the termination of the PJA itself; when the PJA finishes compiling the job, it terminates, thus starting the execution of the job. At the beginning of each job step, each filter module is assigned stack space and a stack frame. Its initial program counter address and arguments are stored in its saved registers for process activation.

CPS uses a piped stream I/O mechanism similar in function to a UNIX stream; a filter module's input comes from the output of the previous module, and its output becomes input to the following module. By convention, the first filter module of the job step is activated first in the job step; when a filter blocks for output, the next filter module is activated. That filter module then runs until it blocks for input or output, at which point the previous or following filter module is activated.

Table 1 shows a simplified listing of the job steps compiled by the serial PJA to process a simple job: one file to be printed in ANSI mode. Each of the job steps shown contains one or more filter modules piped together. For example the job-burst job step has two modules piped together: the job-burst module and the write-to-printer module. Figure 2 shows several job steps with several filter modules each.

Table 1: Simplified Job-step Sequence

Job Step	Function
init_ps_ device	Ensure device is "fixed up."
check_ prologues	Ensure that persistent prologues are loaded.
sheet_count	Get the beginning page count.
job_burst	Print job burst page.
sheet_size	Get the current sheet_size.
wait_sheet_ size	Wait for the sheet_size before continuing.
file_setup	Send any file /SETUP modules.
get_vmbytes	Get the amount of local printer memory available on the printer.
wait_vmbytes	Wait for the local printer memory message from the printer.

Table 1 (Cont.): Simplified Job-step Sequence

file_out	Read the file to print and send it to the DECansi translator.
sync	Wait for the printer to finish all pages.
init_ps_device	Ensure the device is "fixed up."
sheet_count	Get the ending page count.
wait_sheet_count	Wait for the page count to come back.
job_trailer	Print the job trailer page.
sync	Wait for the printer to finish the job-trailer page.
disconnect	Release the printer.

If an error occurs at any point in the processing of a job, CPS skips job steps until it reaches the identified error job step set by the PJA. In Table 1, the error job step points to the sync job step that precedes the job-trailer job step. In this case, CPS resynchronizes with the printer and prints the job-trailer page, including the error message, to tell the user what went wrong.

Event Handling

In addition to the output side of processing a job, there is a corresponding input side. The input side reads messages from the printer, parses them, and notifies the appropriate handler of the event. The handler is chosen based on the type of message sent.

- CPS internal messages are dispatched to the appropriate symbiont routines. For instance, printer resource messages contain information that affect CPS internal operations: paper size is stored for later use by layup (the general mapping of page images to sheets) and translators; virtual memory size is stored for translators; and page count is stored for later use in accounting.

- Printer status messages are dispatched to the operator and, in some cases, to the current user. CPS uses the normal VMS OPCOM notification mechanism to send messages to the system operator. If the user specified /NOTIFY in the print instructions, then CPS uses the VMS $BRKTHRU system service to send the message to the user also.

  In some cases, printer status messages require additional processing. For example, paper jams require special handling on some printers: since CPS cannot determine how many pages were lost in the jam, it invokes human intervention by placing the job on hold. The operator or user can determine what parts of the job, if any, to reprint.

- Program status messages and user data messages are dispatched to the job log. If the user specified /NOTIFY, then they are also displayed with the $BRKTHRU system service. These messages may be printed or logged.

The input and output sides of the symbiont run asynchronously most of the time, but occasionally it is necessary for the output side to wait for a message from the printer. This synchronization between the input side and output side of the symbiont is accomplished by an internal event-signaling facility. When synchronization is required, the output side waits for a specific named event and the input side signals that event when it is detected. For example, at the end of a job, CPS needs the final printer sheet_count in order to calculate the sheet_count for the job; this count is printed on the trailer page and stored in the VMS accounting records. When CPS needs the sheet_count, the output side waits for an event named sheet_count. The input side parses the incoming sheet_count message, stores the returned value in the shared data structure, and signals the sheet_count event. The processing of this event is asynchronous: at the time the message comes in, the output side may or may not have stalled while waiting for the sheet_count event. If the output side was waiting for that event, it is scheduled for further processing; if the output side was not waiting, the event is remembered, in case the output side attempts to wait for this condition in the near future.

In the next section we describe the ways CPS is controlled and managed in the VMS printing system and how it expands printing capabilities in the VMS environment.

5 The VMS Printing System Environment

CPS functions as a component of the VMS printing system at the level of printer supervisor. As such, it interacts with, and is shaped by, the other components of the VMS system. The term printer supervisor is used in this paper to be consistent with the terminology of the emerging International Standards Organization (ISO) Document Printing Application draft standard, ISO/IEC DIS 10175.

Components

The VMS Batch/Print system is a general queue management service, capable of queuing, scheduling, and executing jobs in response to a variety of user-specified instructions.[1] On the VMS system, the printing instructions are stored in a print job object, which is placed in a queue of jobs for a printer. Modern print jobs often resemble batch jobs, due to complex stored processing instructions and the heavy computing load placed on graphics printer controllers.

The VMS printing system contains components at most levels of the DECprint architectural model.

- User printing interface. The VMS system includes interactive Digital Command Language (DCL) interfaces for printing and managing print jobs, printers, and the printing system itself.[2] For DECwindows applications, the DECwindows Print Widget provides a graphical interface that permits users to specify all the options for printing, and the ALL-IN-1 application provides character-cell menus for choosing print options, including the enhanced options offered by CPS.

- Job submission interface. The VMS system includes program call interfaces that give the program all the capabilities of the DCL user interface.[3]

- Print client and service for remote printing. The distributed queuing services product currently provides transparent remote printing in networks using a proprietary network protocol.

- Print spooler. The VMS Job Controller, recently replaced by the VMS Queue Manager, functions as queue manager and scheduler. (The function of spooling printer data to temporary files is performed by the VMS file system and is transparent to most components of the printing system.)

- Printer supervisors. The VMS system provides two standard symbionts to support most line printers and serial printers. PRTSMB supports printers attached directly to communication ports on the CPU, e.g., the printer port on a VAX workstation. LATSYM provides support for printers attached to the serial or parallel ports of DECserver network communications servers. For PostScript printers, CPS is used instead of these standard symbionts.

The VMS printing system also contains components that affect CPS processing.

- Device control libraries are collections of small text sequences that can be inserted into the data stream from the symbiont to the printer. The sequences are ideally organized into text libraries containing named modules, with a separate library for each type of output device. Device control modules can be associated with a printer queue by the system manager as part of a FORM definition or a job reset function, or accessed directly by the user with the /SETUP qualifier.

  Device control libraries frequently contain device-specific control sequences that alter the format of the text and pages, for example, setting printer paper margins, setting character pitch, or enabling landscape printing. They may also contain downloadable font data or preprinted data for each page.

- VMS form definitions contain page size and margin specifications that guide the print formatting process for a print job. The user can also specify page setup strings and can prohibit the symbiont from wrapping lines during processing.

VMS Print Queues

VMS has several distinctly different types of queues. Execution queues process jobs through a symbiont, and generic queues transfer jobs to other queues. Often generic queues are used for load balancing: one generic queue may feed several printers of similar capability and location.

CPS also uses generic queues in an unusual way. Default attributes can be specified for generic queues that cause all jobs submitted through the queues to inherit certain default print instructions. For example, a queue can be established that, by default, assumes that jobs are PostScript documents, or assumes that jobs should be printed in landscape orientation. This ability to set default queue attributes is essential for supporting applications that can specify the queue name for a print job, but cannot specify certain other qualifiers such as DATA_TYPE. It can also permit users of old applications to access new features of the printing system.

VMS PRINT Commands and Interfaces

The VMS printing system is manipulated through DCL commands and qualifiers. Many of the qualifiers are handled by the queue manager and have no impact on the operation of print symbionts; others directly affect the operation of CPS.[2] The VMS system also supplies a call interface to these functions.[3]

VMS Interfaces to Symbionts

The VMS Job Controller/Queue Manager provides two interfaces for customizing print symbionts: the PSM module-replacement interface, and the SMB server symbiont interface. CPS is currently implemented as a single-stream symbiont through the SMB interface.

The SMB interface permits a user to replace the flow of control of the symbiont with a separate process. The process may be written in any style and structure suitable to the task at hand, and need follow only certain minor guidelines with respect to the operating system environment. To use the SMB interface, we replaced the entire symbiont process. The result was much greater flexibility, but we were required to write more program code.

The SMB interface provides services to the symbiont process through subroutine entry points and callbacks that pass messages between the symbiont and the VMS queue manager. Messages from the system to the symbiont specify functions such as start up, shut down, begin job, pause, resume, and interrupt. Messages from the symbiont to the system return information such as job status, job completed, device status and error information, and checkpoint and accounting data.

6 Range of Printers Supported

CPS currently supports the full range of PostScript printers supplied by Digital, from a low-speed color printer up to a 40-page-per-minute laser printer that can handle 11 different paper sizes.

Special I/O Processing

CPS supports several different means of communication with the printer: serial, Ethernet, and a special high-speed video connection.

The serial connection may be either a direct connection between the computer and the printer or a local area transport (LAT) connection by which printer is attached to a serial port of a DECserver terminal server. The two methods differ only in the way jobs are started and terminated. For LAT-connected printers, CPS must establish and dismiss the LAT connection at the start and end of each job.

Once the connection is established with the serial printer (via LAT or direct connect), CPS begins a dialogue with the printer using an asynchronous serial line protocol and PostScript programs. The asynchronous serial line protocol, defined by Adobe Systems Inc., consists of five control characters that alter or query the state of the printer.

The symbiont forces the printer into an idle state by a series of control/T, control/C, and control/D characters. When a control/T results in an IDLE message from the printer, the symbiont and printer are ready to process a job.

PrintServer printers on Ethernet networks are DECnet nodes. To write to a PrintServer printer, CPS establishes a DECnet task-to-task session at the beginning of the job. The dialogue required for synchronizing serial printers is not necessary for the Ethernet printers; the PrintServer protocols provide synchronization and device control operations through separate control channels.

Printers connected through Ethernet use several protocols, which are layered on DECnet task-to-task communications. The protocol used depends upon the version of the PrintServer code.

The local area print service (LAPS) protocol was developed for the PrintServer family and is still in use. The Common Printer Access Protocol (CPAP) will replace LAPS in all PrintServer printers.[4] PAP is based on the earlier Reid-Kent protocol, Internet Socket 170, and is being discussed as a possible new Internet standard.[5]

Special Processing for "Dumb" Printers

In some printer configurations, it is economical to use the workstation or CPU as the printer controller. In this case, the printer includes only the print engine and media handling and finishing equipment, and none of the electronics, computers, and interpreter programs that render the graphics language into the elements required by the print engine (usually an array of pixels). Such a "dumb" printer is physically connected to the computer by a very high speed link such as a direct video connection or data bus. For such a controller-less printer to be generally useful, the printing system must emulate an existing class of printer.

The LN03 Image printer (LN03Q) is a bit-map printer of this type. It uses a special high-speed DMA bit-map interface that plugs into a Q-bus and provides the speed required for printing scanned images. The protocol between this interface and the printer consists of bit maps and a small amount of status and synchronization information.

The engine itself includes only the laser imaging and paper handling equipment. CPS handles the rest of the controller functions in the host computer. Because of the level of support and emulation provided, the LN03Q printer appears to be an ordinary PostScript job printer with some special image capabilities.

For a given print job, CPS performs the normal processing up to the point at which the PostScript language data stream would normally be sent to the printer. At this point, CPS directs the data stream to a special PostScript interpreter subroutine that produces a bit-map image of the printed page in memory. The bit-map image is then sent to the printer through a special LNV21 direct memory access I/O interface on the Q-bus.

The software for the LN03Q printer also has one special processing path. The LN03Q printer is intended as an image printer for bit-map images. CPS supports image files containing page images that are scanned or precomputed at device resolution (300 dots per inch) and optionally compressed with Comité Consultatif Internationale de Télégraphique et Telephonique (CCITT) Group 3 (1D) or Group 4 (2D) compression methods. Image files can be transmitted directly to the printer without converting to PostScript. Image files can only be sent directly to the printer if they are printed one page per sheet; if the user requests printing multiple pages per sheet, or other layup functions, then the image is processed through the PostScript interpreter.

Image files are structured in Digital document interchange format (DDIF), which expresses text, graphics, and images together. Files intended for the LN03Q printer must contain only image bit maps.

If the print job specifies DATA_TYPE=DDIF or the file is a DDIF file, then CPS examines the file in a special mode. If the file correctly contains only image bit maps, CPS decompresses the images in memory if necessary, using the DECimage Image Support Library routines, and then sends the uncompressed bit map directly to the LN03Q print engine. Thus the image goes directly to the printer without passing through the PostScript interpreter.

7 Special Processing in CPS

CPS includes a number of special features and functions to satisfy the requirements of the DECprint architecture and the VMS printing system. In this section, we discuss the features that extend the process of standard print symbionts or are completely new.

Reading Print Instructions

CPS reads the print instructions for a job from the VMS queue manager through the SMB$READ_MESSAGE and SMB$READ_MESSAGE_ITEM functions of the SMB interface. Print instructions are expressed as attributes with values. Each attribute has an associated numeric code and symbol, called an item code, and a value of a specific data type. The symbiont reads each item code and value, and stores the information in a static data structure. The information is used later to determine the processing sequence for the job, special information to be displayed on separator pages, and so forth.

Bidirectional Communication with PostScript Printers

CPS requires a full duplex communications path to PostScript printers since they report many conditions by sending messages to the host computer. These messages include device status messages, program status and error messages, user data messages, and replies to CPS inquiries.

CPS also requests information from the printer for synchronization, formatting, and accounting purposes. For instance, to determine how to format ANSI text, the symbiont needs to know what paper is loaded in the printer.

CPS receives the messages from the printer and parses them to determine what it should do with the message. If the message is device status, then CPS routes the message to the operator and/or the user whose job is being printed. If the message is an internal CPS communication, then CPS processes it. Otherwise, the message is either a program status message or a user data message. In either case it is logged for the user.

All messages are parsed except user data messages. Messages from the printer's interpreter are converted to a standard format that would, if desired, permit the message to be translated into the user's native language.

Data Syntax Translation

CPS provides a facility that translates selected printer data syntaxes into the PostScript language. The translating programs are subroutines, some quite large and complex, that accept a data stream in one format and produce a data stream in another format. The translators are responsible for all formatting, including sheet size, page orientation, aspect ratio, and type sizes; CPS is responsible for all I/O and coordination with the printer. The translation facility currently supports the following printer data syntaxes: DEC PPL3, ReGIS, Tektronix 4010 /4014, and PCL Level 4.

The translation facility has several restrictions. A file may consist of only one data syntax, and all files in a job must be of the same data syntax.

In general, CPS performs the translation from one data syntax to another on the host computer. In this way, simple printers that support only the PostScript language internally can be extended to support a number of printer languages. This reduces the requirement for a complex printer controller that supports multiple data syntaxes internally. Host translation can guarantee consistent use across jobs of the printer's internal fonts, page orientation, finishing equipment, and page layup The general mapping of page images to sheets supplied as part of CPS requires that the printer operate in PostScript mode. To ensure consistent use of fonts and consistent positioning of pages with respect to finishing such as duplexing and stapling, all language translation must be done by the symbiont.

Page Layup Multiple Pages per Sheet

Page layup is the process of printing more than one page image on a sheet of paper. When more than one page image is placed on a sheet of paper, the images are rotated and scaled to fit on the page, but are altered in no other way. The layup facility works with all data types, including PostScript and PCL data syntaxes. Layup also permits formatting for larger paper sizes and then printing on smaller sheets.

Layup is invoked explicitly with one or both of the extended qualifiers NUMBER_UP and LAYUP_DEFINITION. NUMBER_UP specifies the maximum number of page images that will be printed on a single side of a sheet; for example, two-up printing is specified by the "NUMBER_UP=2" option. Two or four page images per side may save significant quantities of paper for draft printing, handouts, and the like. Up to 100 page images may be placed on a single sheet of paper for thumbnail draft printing to review the overall layout of a document.

Layup may also be invoked through a combination of PAGE_SIZE and SHEET_SIZE with NUMBER_UP. For example, the combination of PAGE_SIZE=E,SHEET_SIZE=A,NUMBER_UP=1 permits printing draft copies of large-format documents on small paper. Conversely, the combination of PAGE_SIZE=A,SHEET_SIZE=B,NUMBER_UP=1 magnifies the smaller page to fit the larger sheet.

Duplex Printing

Printing on both sides of the paper introduces a number of new options and interactions that require special processing in CPS. CPS begins each document on the first side of a new sheet, so that recto and verso (right-hand and left-hand) pages and alternating margins are aligned with the correct sides of sheets as they are stacked by the printer. This function also interacts with the direction in which the medium is physically loaded into the printer if the medium is not symmetric left-to-right, top-to-bottom, or front-to-back, such as pre-drilled paper.

The interactions of PDL coordinate systems, page layup, media selection, asymmetric media, duplex printing, and binding are the most elusive engineering problems in the printing application space. No general model of these interactions has been developed, despite considerable effort in standards committees. It appears that it is necessary to implement every possible option.

Separator Pages

CPS prints all the separator pages defined by the VMS queuing system as well as some generated by CPS. Flag, burst, and trailer pages for job and file levels are available as defined by VMS, and contain the same information presented in a highly legible format. In addition to the standard VMS information, the job trailer page also contains the first two PostScript language errors returned from the printer. This often makes it unnecessary to use MESSAGES=PRINT to see simple errors.

To ensure that the job separator pages can always be printed correctly, CPS resets the PDL interpreter in the printer before printing these pages. The CPS-generated separator pages do not alter the coordinate system of the interpreter; the user's document starts printing with the default PostScript state. File separator pages, in contrast, print in the current PostScript environment, including the altered page geometry, e.g., layup established by the print job.

CPS defines two new separator pages. The file error page is printed when a file cannot be opened or an error occurs while reading the file. The file error page informs the user of the error condition which caused it to be printed. The job log page contains up to 40 lines of the job log file. The job log file contains job events such as job start and job completion as well as program status messages and user data returned from the printer.

Managing Printer Resources

Once communication is established with the serial printer, the symbiont must establish what resources are available on the printer. These resources include prologues, which are commonly used PostScript routines, the amount of available virtual memory, and the medium in the default paper tray. For example, CPS persistently loads the PostScript prologue for the output of the ANSI text translator into the PostScript interpreter. This resource might be lost to the printer because of a power failure or might become obsolete due to a software upgrade. CPS interrogates the printer at the beginning of any job requiring the translator prologue and loads a new prologue, if necessary. CPS also performs similar processing for the PostScript prologue that is used to generate the separator pages.

For traditional resources such as paper, CPS relies on status messages from the printer to indicate that the printer is stopped because paper supply is empty or jammed. These conditions are relayed to the operator and to the current user by standard VMS mechanisms.

Library Search Lists

In the standard VMS print symbiont, only one device control library may be associated with a queue. This is not a problem since the standard VMS print symbiont deals with only one data syntax. (Recall that device control libraries are often written in device-dependent data syntax.) CPS, on the other hand, uses more than one data syntax when printing a non-PostScript job: the data stream to the printer is PostScript, but the data stream to the translator is in another data syntax.

Early versions of symbionts that supported PostScript suffered from the same restriction: only one device control library was available, and its modules were expressed in PostScript. This made it impossible for users to share device control libraries with their standard VMS print symbiont and their non-PostScript printers.

To solve the problem of multiple data syntaxes in a job, CPS introduced device control library search lists. The system manager, rather than specifying a single file specification in the INITIALIZE/QUEUE/LIBRARY command, creates a logical name instead. CPS translates that specific logical name and uses each element of the result as a device control library. Each library in the search list can have a data syntax associated with it by adding the qualifier, /DATA_TYPE=.

CPS supplies a device control library, CPS$DEVCTL, which must be included in the search list, usually as the first, or only, element in the search list.

8 Summary

The DECprint model of printing describes a useful structure with consistent functions and responsibilities. CPS is an advanced print symbiont that runs in the VMS printing system. It includes many specialized functions to support the features of a wide range of modern printing devices. It provides, we feel, an extraordinary level of support. It was designed with a highly modular and flexible internal structure to permit enhancements to be engineered with minimal interactions with current operations.

CPS is currently shipping its fourth version. This version fully supports the ten different PostScript printers supplied by Digital, which range from a low-speed color printer to a high-speed laser printer. It also supports five different data syntaxes in which applications can write documents. We expect that more printers and more capabilities will be added in future versions, and that CPS will require a minimum of additional engineering effort due to its very general internal structure.

9 Acknowledgements

We would like to thank Peter Conklin for actively initiating CPS and Gary L. Brown for even more actively expanding it. We would also like to thank past and current CPS developers: Ned Batchelder, Cathy Callahan, Mark DeVries, Rich Emmel, Dave Gabbé, David Larrick, Klara Levin, Mary Marotta, Doug Stefanelli, and Charlotte Timlege. We would like to thank Bill Fisher for his extensive comments on this article. Finally, we thank the many sites and people who have tested the DECprint Printing Services software.

10 References

o VMS Utility Routines Manual (Maynard: Digital Equipment Corporation, Order No. AA-LA67B-TE, 1990).

o VMS DCL Dictionary, 2 vols. (Maynard: Digital Equipment Corporation, Order Nos. AA-PBK5A-TE and AA-PBK6A-TE, 1991).

o VMS System Services Reference (Maynard: Digital Equipment Corporation, Order No. AA-LA69A-TE, 1991).

o J. Jones, A. Kachrani, and T. Powers, "The Common Printer Access Protocol," Digital Technical Journal, vol. 3, no. 4 (Fall 1991, this issue):55-60.

o B. Reid and C. Kent, "TCP/IP PrintServer Print Server Protocol," Western Research Lab Technical Note TN-4 (Maynard: Digital Equipment Corporation, 1988).

11 General References

Guide to Maintaining a VMS System (Maynard: Digital Equipment Corporation, Order No. AA-LA34A-TE, 1990).

DECprint Printing Services User's Guide (Maynard: Digital Equipment Corporation, Order No. AA-PBZGA-TE, 1991).

DECprint Printing Services System Manager's Guide (Maynard: Digital Equipment Corporation, Order No. AA-PBZFA-TE, 1990).

Digital ANSI-Compliant Printing Protocol Level 3 Programming Reference Manual (Maynard: Digital Equipment Corporation, Order No. EK-PPLV3-PM, 1991).

Digital ANSI-Compliant Printing Protocol Level 3 Programming Supplement (Maynard: Digital Equipment Corporation, Order No. EK-PPLV3-PS, 1991).

PostScript Translator's Reference Manual for ReGIS and Tektronix 4010/4014 (Maynard: Digital Equipment Corporation, Order No. AA-PBWFA-TE, 1991).

PostScript Printers Programmer's Supplement (Maynard: Digital Equipment Corporation, Order No. EK-POSTP-PS, 1991).

PostScript Language Reference Manual, 2nd ed., Adobe Systems Incorporated, ISBN 0-201-18127-4 (Reading, MA: Addison-Wesley, 1990).

Information Technology-Text and Office Systems- Document Printing Application (DPA), ISO/IEC JTC1/SC18 N, Draft International Standards 10175-1 and 10175-2 (September 1991).

CDA Base Services Technical Overview (Maynard: Digital Equipment Corporation, Order No. AA-PHJYA-TE, 1991).

Creating Compound Documents Using CDA Base Services (Maynard: Digital Equipment Corporation, Order No. AA-PHK2A-TE, 1989).

Writing Converters Using CDA Base Services (Maynard: Digital Equipment Corporation, Order No. AA-PHK1A-TE, 1991).

CDA Base Services Reference Manual, 2 vols. (Maynard: Digital Equipment Corporation, Order Nos. AA-PHJZA-TE and AA-PHK0A-TE, 1991).

CDA: DDIF Technical Specification (Maynard: Digital Equipment Corporation, Order No. AA-PHK3A-TE, 1991).

CDA: DTIF Technical Specification (Maynard: Digital Equipment Corporation, Order No. AA-PHK4A-TE, 1991).

DDIS Syntax Specification (Maynard: Digital Equipment Corporation, Order No. EL-00081-00-1, 1987).

Richard B. Landau Richard Landau is the DECprint program manager for the Video, Image and Print Systems Group. Working to improve the interaction of printing software and hardware, he initiated the DECprint, Font, and PostScript programs. Prior to this, Rick was the program and development manager for the VAX DBMS, DATATRIEVE, CDD, and Rdb/VMS products and for the relational database architecture. Before joining Digital in 1974, Rick worked as an independent consultant and was also employed by Applied Data Research, Inc. He holds A.B. (cum laude, 1969) and M.A. (1973) degrees in statistics from Princeton University.

A. Alan Guenther As a member of the technical staff in the DECprint System Software Group, Alan Guenther is involved in the ongoing design and implementation of the DECprint common print symbiont. Prior to this, he was the primary designer and implementor of the distributed queuing services. Alan has worked at Digital since 1973, both as a full-time employee and as an independent consultant (from 1982 to 1990). After receiving a B.S. (honors, 1970) from the University of Montana, he worked at the university until he joined Digital.

===

==