To: Distribution

From: Michael R. Jordan

Date: 07/27/78

Subject: PPS Support on Multics

This MTB outlines a plan for total support of the Page Printer System (PPS) as an offline printer on Multics. The PPS is presently supported in this fashion by all systems on which support exists, but plans are underway for online support for level 66 hardware. Support for the PPS exists presently on IBM OS and DOS systems, level 66 (GCOS), and various other systems.

Questions and comments should be directed to MJordan.Multics on system M or MIT or mailed to:

M. R. Jordan
Honeywell Information Systems
575 Technology Square
Cambridge, MA 02139

All comments must be received before August 25.

INTRODUCTION

This MTB outlines a plan for support of the PPS on Multics. First, a brief description of the PPS is given. Following this are sections stating why the PPS should be supported on Multics and some problems forseen in this support. This is all followed by the important stuff - how could we do it? And finally, a suggested schedule and some comments concerning the release of this software to customers.

THE PPS HARDWARE

The Page Printing System (PPS) is an off-line, non-impact printing system capable of printing up to 210 pages per minute (equivalent to 18,000 lpm). The PPS consists of one or more read-only tape drives, a system controller, a print unit, and one or more stacker units.

The read-only tape units are capable of reading tapes written at 556, 800, or 1600 bpi.

The system controller is a Honeywell 716 or Level-6 processor.

The print unit employs a fixed electrographic print mechanism that moves the paper at a constant rate of 30 inches per second (20 ips on slower models) past an operator mounted format drum containing the image of a "preprinted form" for fixed data such as horizontal lines or company logo. From there the paper passes a print station capable of printing up to 132 character lines read from the input tape. This unit is capable of printing with 4, 6, 8, or 10 lines per inch vertically and 10 or 12 characters per inch horizontally. The print unit is also capable of printing up to 255 copies of a report while only reading the input tape once. Also included in the print mechanism is a paper cutter and hole punch which allow a variety of page sizes and punched hole configurations. Each stacker unit provides 8 trays, each capable of holding up to 500 sheets. Stacker algorithms supported include stacking one report per tray, one copy per tray (for multiple copy reports), and simple overflow from one tray to the next with or without separator sheets (a separator sheet is a blank page slightly longer than a printed page).

WHY SUPPORT THE PPS?

There are a number of reasons that can be given at this time for supporting the PPS on Multics. A few are listed below:

1) The PPS is a very fast, efficient printing system.

2) Some Multics sites have PPS and the support they get is from other systems the customer has in-house (GCOS at AFDSC and Burroughs at DCC).

3) A number of prospective customers have expressed interest in this support. These are both prospective customers for the PPS or Multics or both (GM and Ford).

4) Internally, system M users have access to a PPS, but no support exists on Multics.

In summary, the PPS seems to be a "hot" item supported on a many systems but not on Multics.

PROBLEMS SUPPORTING THE PPS ON MULTICS

There are a few problems that come up fairly quickly when one looks at the PPS from a Multics viewpoint and talks to some of the prospective users of this package.

One of the problems is the inability of the PPS to overstrike print lines. This is due to hardware restrictions including the fact that the paper is fed at a constant velocity (this means that the 18,000 lines per minute is really 18,000 lines each minute). Since Multics allows (and users use) any overstruck print image possible, this presents a real problem. This problem can be solved by having the support hardware translate overstruck character sequences into printable characters on the PPS. This leads to the restriction of only a total of 256 different characters (a PPS limit due to 8-bit bytes). In fact the approach that will be taken in the support of the PPS on Multics is to keep a PPS print image for each line and any time an ASCII character is overstruck with a PPS character, look the resulting character up in a table. This table will be generated by a tool, supplied as part of the PPS support package, so that any custom character set can be supported.

When talking to users two points come up quickly: "control blocks" and "Can I use dprint?". The first item, control blocks, refers to the control block mechanism supported by the PPS. These control blocks allow a user to specify everything that the PPS system could want to know about the user's data. Such things

as paper size, lines per inch, characters per inch, the distribution list, etc. can be specified in the control block that preceeds a user's data on a PPS tape. The first subject will be handled by providing interfaces to allow the I/O module to put the control blocks on the output tape and by providing subroutines callable from PL/I to allow a user to personally handle these control blocks. A compiler for these control blocks will also be provided. And finally there will be a driver provided that will allow users to send output to the PPS using the dprint command.

THE PLAN

The basic support will be an I/O module. This I/O module, pps_, will provide its user with full control over the generation of a tape to be printed on the PPS. The I/O module will allow stream_output (put_chars) and the control and modes operations required to allow full control of all functions of the PPS. It will use prt_conv_ to handle all page formatting which will make it very compatible with Multics printer software.

The question of the character set of the PPS (which can be unknown in the case of custom fonts) is covered by the cv_ppsct tool. This command allows a Multics user to define Multics ASCII equivalents for all PPS characters including overstruck ASCII sequences.

Control blocks will be compiled from an ASCII source segment and will be accessible through the I/O module attach description, a control order, and a subroutine interface. The user can, with the control order or subroutine interface, get the Multics representation for the control block and modify it in any way the user wishes. A subroutine interface will also be provided to produce a PPS formatted control block from the Multics representation. This "extra step" is considered necessary due to the very strange representation of the control block on the PPS tape.

And finally to complete the package a daemon driver capable of using the pps_ I/O module will be provided. There are unanswered questions in this area. For example, how does a user specify a control block to the daemon? If this is answered, the next question is how much of the information in the users control block should override the information in the request type control block (surely not page size, but maybe font number, lines per inch, etc.). It is felt that these questions can be answered, but need not be at this time.

SCHEDULE

In MR7.0 we should offer the I/O module with a few capabilities lacking (such as control blocks), the cv_ppsct command, a user command to make a PPS tape using pps_, and a modified spooler (as an interim daemon). The attached document is suggested as the documentation for this package. Note that sections IV and VII are missing. This is because they address the issues of control blocks, some of which are not yet resolved.

In MR8.0 we could make the daemon support better and release the control block software.

COMMENTS

It is the author's opinion that this software should be released as unbundled software in MR7.0 and that the documentation should remain a separate manual. The rationale for the documentation remaining this way is that once we get this software done only those users directly invloved with the PPS will care about this information.

It should be pointed out that all of the software proposed for MR7.0 is very close to being complete with the exception of the modifications to the spooler. These changes have, in the past, been trivial (<50 lines of code).

CONTENTS

SECTION I

INTRODUCTION

This document describes the software provided in the Honeywell Multics operating system to support the Honeywell Page Printing System.

THE PAGE PRINTING SYSTEM

The Page Printing System (PPS) is an off-line, non-impact printing system capable of printing up to 210 pages per minute (equivalent to 18,000 lpm). The PPS consists of one or more read-only tape drives, a system controller, a print unit, and one or more stacker units.

The read-only tape units are capable of reading tapes written at 556, 800, or 1600 bpi.

The system controller is a Honeywell 716 or Level-6 processor.

The print unit employs a fixed electrographic print mechanism that moves the paper at a constant rate of 30 inches per second (20 ips on slower models) past an operator mounted format drum containing the image of a "preprinted form" for fixed data such as horizontal lines or company logo. From there the paper passes a print station capable of printing up to 132 character lines read from the input tape. This unit is capable of printing with 4, 6, 8, or 10 lines per inch vertically and 10 or 12 characters per inch horizontally. The print unit is also capable of printing up to 255 copies of a report while only reading the input tape once. Also included in the print mechanism is a paper cutter and hole punch which allow a variety of page sizes and punched hole configurations. Each stacker unit provides 8 trays, each capable of holding up to 500 sheets. Stacker algorithms supported include stacking one report per tray, one copy per tray (for multiple copy reports), and simple overflow from one tray to the next with or without seperator sheets (a seperator sheet is a blank page slightly longer than a printed page).

MULTICS INTERFACES

Two user interfaces to the PPS are available on Multics. They are both described in this document. They are: the pps_ I/O module and the PPS I/O daemon.

The first, the pps_ I/O module, requires the user to take an active part in the preparation of the PPS tape. The user (or user's program) must attach, open, write to, close and detach the I/O switch being used. This method also requires access to use a tape drive on the Multics system.

The second interface requires some setup by the system administrator and operator of the Multics system. However, once this setup has been accomplished, the user need only use the dprint command to submit requests to be processed by the PPS I/O daemon.

The I/O module approach is usually better for producing large reports and offers the user maximum control over the PPS functions employed in the production of the output. For sites with many Multics users with a need to print on the PPS the I/O daemon is an attractive alternative.

SECTION II

USE OF THE pps_ I/O MODULE

The pps_ I/O module is designed to afford the user maximum control over the PPS reports produced. It provides for the generation of multiple report tapes while allowing full page format control (such as page labels, indentation, channel stops, etc.).

USING THE I/O MODULE

The pps_ I/O module should be attached and then opened for each output report being produced. The only opening mode supported is stream_output. Each time the I/O switch is opened a new report is started on the output tape. Once the report has been written the I/O switch is closed and is then ready for another report. Subsequent reports simply require the user repeat this open, write, close sequence as often as necessary. When all requests for the PPS have been processed the I/O switch is detached.

PROGRAMMING EXAMPLES

The examples of this section show how a user can produce PPS output reports using the supported languages of Multics. See section VIII for a complete description of the pps_ I/O module.

PL/I

The PL/I example in figure 2-1 queries the user for two arguments. They are the name of the file to be printed and the tape on which the output is to be placed. The program opens the input and output files, reads and writes the text, and finally closes the two files.

The example of figure 2-2 shows the steps that must be taken to produce multiple reports on the same tape. The extra steps invloved are necessary due to the operation of PL/I I/O. The same example is repeated in figure 2-3 with the explicit attach and detach as part of the PL/I code.

The following is a sample terminal session for the execution of the PL/I program listed in figure 2-2. All input lines typed by the user are preceeded by the exclamation (!) character. This is done to distinguish input and output lines and should not be considered as part of the input typed by the user.

```
! io attach ... to be supplied
```

FORTRAN

To be supplied.

COBOL

The COBOL example program in figure 2-4 reads the file attached to the insw I/O switch and displays the content of this file on the PPS. Note that the COBOL program thinks that the PPS attachment is a printer and that the resultant report will be double spaced.

The following is a sample terminal session for the execution of the COBOL program listed in figure 2-4. All input lines typed by the user are preceeded by the exclamation (!) character. This is done to distinguish input and output lines and should not be considered as part of the input typed by the user.

```
! io attach insw vfile_ userfile
  r 1306 0.049 5.546 79
! io attach outsw pps_ -volume ppsvol
  r 1307 0.277 23.628 415
! example
  example: Run-unit example terminated (line 32).
  r 1308 1.209 46.856 279
```

BASIC

To be supplied.

Figure 2-1

To be Supplied.

Figure 2-2

To Be Supplied

Figure 2-3
To Be Supplied

```
COMPILATION LISTING OF SEGMENT example
Compiled by: Multics COBOL, Version 3.0 of October 8, 1977
Compiled on: 07/06/78  1413.8 edt Thu
    Options: sc;

    1               identification division.
    2               program-id. pps-test.
    3               environment division.
    4               configuration section.
    5               source-computer. Multics.
    6               object-computer. Multics.
    7               input-output section.
    8               file-control.
    9                   select external pps assign to outsw-printer.
    10                  select external qaz assign to insw;
    11                  organization is stream.
    12              data division.
    13              file section.
    14              fd pps data record is pps-rec,
    15                  label records are omitted.
    16              01 pps-rec picture x(120).
    17              fd qaz data record is qaz-rec,
    18                  label records are omitted.
    19              01 qaz-rec picture x(120).
    20              working-storage section.
    21              procedure division.
    22              open-files.
    23                  open input qaz.
    24                  open output pps.
    25              loop.
    26                  read qaz record; at end go to close-files.
    27                  move qaz-rec to pps-rec.
    28                  write pps-rec after advancing 2 lines.
    29                  go to loop.
    30              close-files.
    31                  close qaz, pps.
    32                  stop run.
```

Figure 2-4. A COBOL program to list a file on the PPS.

SECTION III

THE PPS I/O DAEMON

The PPS I/O Daemon is an I/O daemon driver that uses parts of the pps_ I/O module to format its output tapes. Once it has been set up by the system administrator, PPS printing is available to anyone who has access to the queues being used by the daemon.

USING THE PPS I/O DAEMON

The use of the PPS I/O Daemon is the simplest form of user interface to the PPS system. Once the system administrator has modified the I/O Daemon tables to include the PPS I/O Daemon (see appendix C for details), the user need only use the dprint command and specify the proper request type. For example, if the system administrator has designated the request type pps_8_11 as a PPS request type the user simply types

```
dprint -rqt pps_8_11 my_data
```

to have the file my_data printed by the PPS.

SECTION V

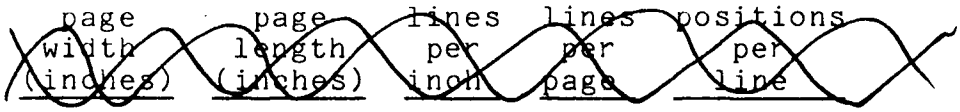
LIMITATIONS OF PPS SUPPORT ON MULTICS

There are a few limitations placed on the Multics user by the PPS interface described in this document. Some are avoidable by extra user coding, while others are due to the PPS system itself.

PAGE FORMAT

The page format limitations are due to physical limitations of the PPS hardware. The PPS line length is limited to 132 print positions. That is, no print line can represent more than 132 columns of printed output. The page length is limited to a maximum of 93 lines per printed page. These values may be restricted further by the physical page dimensions. Table 5-1 shows the line length and page length limits for all of the allowable page sizes.

CHARACTER SETS



In all cases the Multics PPS support software attempts to produce PPS output that is visually similar to output produced by Multics on a terminal or line printer. With the PPF6025 hardware option on the destination PPS system almost all underscored text and many other overstruck character combinations are possible. If the user specifies the "-ct ppf6025" control argument in the pps_ attach description any overstruck character combination that can be reproduced on the PPS will be accomadated. Any combination not representable on the PPS will be displayed as a special "black box" character. This "black box" character is just as its name suggests - a one character black box. With the standard PPS font (NIP optimized) any underscored text will have the underscores removed and any other overstruck character combinations that cannot be reproduced on the PPS will be

displayed as a "black box".

It is possible, using the cv_ppsct command, to define Multics ASCII strings (either single characters or sequences of overstruck characters) that represent any of the 8 bit characters of the PPS.

SECTION VI

PPS COMMAND DESCRIPTIONS

The commadns described in this section allow the user of the I/O module pps_ to create tables to control character conversions done by pps_ and to create listings tapes for printing on the PPS.

The format of these command descriptions is the same as that used in MPM Commands (order no. AG92).

Name: cv_ppscf

The cv_ppscf command converts a data file known as a PPS character file into a source segment that is then assembled to create a PPS character table. This character table can then be used to control the translation of Multics ASCII characters and overstruck character sequences to PPS characters (a modified EBCDIC character set).

Usage

cv_ppscf path {-control_args}

where:

1. path
   is the pathname of the PPS character file. If the suffix ppscf is not supplied it is assumed.

2. control_args
   can be chosen from the following:

   -list, -ls
   causes a listing file to be produced. The name of this file is the source path with the ppscf suffix replaced by the ppsctl suffix.

   -long, -lg
   causes a message reporting the usage of the available positions in the character matrix to be printed.

Notes

The PPS character file consists of lines of character definitions. The first line of the file defines the default character and space character as two hexidecimal values seperated by white space. The default character is used to represent any character or sequence that is not defined in the remainder of the character file. The space character defines the PPS equivalent for the space character. The remaining lines define Multics ASCII equivalents for the PPS characters (one definition per line).

A PPS character is defined by specifying the hexidecimal value for the character followed by the ASCII equivalent. This ASCII equivalent can be a series of ASCII characters, in which case the overstruck sequence defined by these characters is

defined. The sequence of ASCII characters can contain any printable character (no spaces or control characters other than backspace are allowed) with optional backspaces which are ignored.

A PL/I like comment can be specified at the end of any line in the PPS character file.

The delimiters used in parsing these definition lines are white space. This includes the characters space and horizontal tab.

The one restriction on any overstruck sequence that is being defined is that all subsets of the overstruck sequence must also be defined in the character file. This means that if a user is defining "A" overstruck with "_" (i.e. "A") both "A" and "_" must also be defined in the character file.

Name: cr_ppslt

The cr_ppslt command is a convenient means by which a user can create a PPS listings tape containing several source segments to be printed.

Usage

cr_ppslt target {-control_args} path1 {... {-control_args} pathn}

where:

1. target
   can be the name of a tape volume to be used with the default attach description (see Notes below) or one of the following control arguments:

   -volume XX, -vol XX
   specifies that tape volume XX is to be used in the default attach description (see Notes below).

   -target_description XX, -tds XX
   specifies that the attach description XX is to be used rather than the default attach description.

   -target_switch XX, -tsw XX
   specifies that cr_ppslt should use the I/O switch XX rather than attaching it own. The switch may be open or closed.

2. control_arg
   can be chosen from the control arguments listed below. These control arguments affect the printing of all pathnames following them unless overriden by another control argument later in the commad line.

   -bottom_label XX, -blbl XX
   specifies that the string XX is to be used as a label at the bottom of every page of output.

   -label XX, -lbl XX
   specifies the string XX as a label at the top and bottom of every page.

   -modes XX, -mds XX
   sets the modes specified in the mode string XX.

-top_label XX, -tlbl XX
specifies the string XX as a label to be used at the top of every page of output.

3. pathi
is the pathname of a source file to be printed.

Notes

If a volume name is specified for the target, the cr_ppslt command attaches a uniquely named target switch using an attach description of the form "pps_ -vol volid".

Examples

In the following example all of the segments in the user's working directory with the suffix list will be printed on tape x12763:

```
! cr_ppslt x12763 [segs **.list]
```

In the second example, all of the segmetns in the user's working directory with the suffix list will be printed as well as all of the segments with a pl1 suffix. All of these later segments will also have each page labeled with the string "SOURCE".

```
! cr_ppslt x12763 [segs **.list] -label SOURCE [segs **.pl1]
```

display_ppsct_entry

Name: display_ppsct_entry

The display_ppsct_entry command allows the developer of a PPS character table to selectively print the hexidecimal value for overstruck ASCII sequences.

Usage

display_ppsct_entry table_name strings

where:

1. table_name
   is the name of the PPS character table to be used.

2. string
   is a string of ASCII characters with optional backspaces.

Notes

This command uses the specified PPS character table to look up the PPS equivalent character value for the overstruck sequence specified by string and print it on the user's terminal.

SECTION VIII

PPS I/O MODULE

This section describes the PPS I/O module. The description is similar to the I/O module descriptions found in MPM Subroutines (order no. AG93) and MPM Peripheral Input/Output (order no. AX49).

For a general description of the I/O system see "Multics Input/Output System" in section V of MPM Reference Guide (order no. AG91).

Name: pps_

This I/O module attaches an I/O switch to a uniquely named target I/O switch such that a tape suitable for the Page Printing System (PPS) will be produced.

Entry points in this module are not called directly by users; rather the module is accessed through the I/O system.

Attach Description

The attach description has the following form:

```
pps_ volids {-control_args}
```

where:

1. volid
   is the name of a tape volume to be used for output.

2. control_args
   can be chosen from the following:

   -bottom_label XX, -blbl XX
   specifies that the string XX is to be used as a label at the bottom of every page of output.

   -char_table XX, -ct XX
   specifies the pathname of the PPS character table to be used. This character table is created using the cv_ppscf command (see Section VI).

   -density n, -den n
   specifies the density of the tape to be produced, where n can be either 1600 or 800. If this control argument is not given, a density of 800 bpi is used.

   -label XX, -lbl XX
   specifies the string XX as a label at the top and bottom of every page.

   -modes XX, -mds XX
   specifies the initial mode string XX to be used. (See "Modes" below.)

-top_label XX, -tlbl XX
specifies the string XX as a label to be used at the top of every page of output.

-volume XX, -vol XX
specifies the tape volume named XX as the output tape volume for the target attachment. This control argument must be used when a volume name begins with a hyphen (-).

Open Operation

The only opening mode supported is stream_output. Opening the I/O switch results in the attachment and opening of the target switch.

Close Operation

Closing the I/O switch results in the termination of the current PPS output report. When the switch is in the attached and closed state, it may be re-opened to produce multiple PPS output reports.

Detach Operation

Detaching the I/O switch releases any resources assigned during the attachment of the I/O switch.

Control Operation

The I/O module supports two control operations.

get_label_info
set_label_info

In the descriptions below, info_ptr is the information pointer specified in the iox_$control call.

get_label_info
returns to the caller the page label information currently in use by pps_. The info_ptr must point to the following structure in which information is returned:

```
dcl 1 pps_label_info aligned based,
    2 bottom_label char(132) var,
    2 top_label char(132) var;
```

where:

bottom_label
is the label to be used at the bottom of each printed page.

top_label
is the label to be used at the top of each printed page.

set_label_info
sets the page label information to be used by pps_. The info_ptr must point to a structure declared as in the case of get_label_info. The label information is moved from the callers structure to the data structure used in formatting output pages.

Modes Operation

The modes operation is supported whenever the switch is attached. The recognized modes are listed below. These modes are also accepted following the -modes control argument in the pps_ attach description.

edited, ^edited
specifies that ASCII control characters that do not affect carriage or paper motion are to be escaped. Otherwise, these control characters are ignored. (Default is off.)

endpage, ^endpage
specifies that when the normal printed area of a page is overflowed, printing is continued on the next page. Otherwise, text is printed on every line of the physical page. (Default is on.)

esc, ^esc
specifies that the special processing of the ASCII ESC

character is to be enabled. (Default is off.)

fold, ^fold
specifies that lines that are longer than the line length are folded to the next line. Otherwise, such lines are truncated to n print positions (where n is the current line length). (Default is on.)

inn
specifies that each line of output is to preceeded by n spaces. (Default is 0.)

lln
specifies the length in print positions of the output line. When an attempt is made to use more than n print positions on a line the remaining text is moved to the next line or discarded depending on the setting of the fold mode. (Default is 132.)

pln
specifies the length in lines of the printed page. When an attempt is made to print the n+1th line on the page a form feed character is inserted causing the output to proceed on a new page unless the endpage mode is off. (Default is 60.)

vertsp, ^vertsp
performs the vertical tab and form feed functions. Otherwise, these characters are simply mapped into newline characters. (Default is on.)

Notes

There are two character tables supplied as part of the released PPS software. They are ppf6023 and ppf6025. The ppf6023 character table contains 92 printable ASCII characters with all of the underscored characters mapped to remove the underscores. The ppf6025 character table contains these same 92 printable characters as well as these same characters overstruck with the underscore character. For a full description of these character tables see Appendix A.

The I/O module may hold data in buffers between operations. For this reason no operations should be attempted on the target I/O switch while it is being used with the pps_ I/O module.

APPENDIX A

MULTICS PPS CHARACTER SETS

There are two PPS character sets supported by the Multics PPS support package. The character set in figure A-1 is the ASCII character set without underscored characters (ppf6023). The second character set allows a number of overstruck character sequences in addition to the characters in figure A-1. These additional characters are shown in figure A-2.

The figures that follow show the correspondence between the PPS character set and the Multics ASCII character set. The PPS characters are shown in hexidecimal and the ASCII characters or character sequences are shown without backspaces. Figure A-1 for example shows that the ASCII string A is the same as the PPS character represented by C1(16).

HEX	ASCII
40	
5a	!
5a	!_
7f	"_
7f	"_
7b	#_
7b	#_
5b	$_
5b	$_
6c	%_
6c	%_
50	&_
50	&_
7d	'_
7d	'_
4d	(_
4d	(_
5d	)_
5d	)_
5c	*_
5c	*_
4e	+_
11	+-
11	+-_
4e	+_
6b	,_
6b	,_
60	-_
32	-<
32	-<_
07	-=_
07	-=_
60	-_
8f	-¬
8f	-¦_
4b	.
4b	._
61	/_
61	/_
f0	0_
f0	0_
f1	1_
f1	1_
f2	2_
f2	2_
f3	3_
f3	3_
f4	4_
f4	4_
f5	5_
f5	5_

HEX	ASCII
f6	6
f6	6_
f7	7_
f7	7_
f8	8_
f8	8_
f9	9_
f9	9_
7a	:_
7a	:_
5e	;_
5e	;_
4c	<_
8c	<=
8c	<=_
4c	<_
7e	=_
ae	=>
ae	=>_
7e	=_
be	=¬
be	=¦_
6e	>
6e	>_
6f	?_
6f	?_
7c	@_
7c	@_
c1	A_
c1	A_
c2	B_
c2	B_
c3	C_
c3	C_
c4	D_
c4	D_
c5	E_
c5	E_
c6	F_
c6	F_
c7	G_
c7	G_
c8	H_
c8	H_
c9	I_
c9	I_
d1	J_
d1	J_
d2	K_
d2	K_
d3	L_
d3	L_

HEX	ASCII
d4	M_
d4	M_
d5	N_
d5	N_
d6	O_
d6	O_
d7	P_
d7	P_
d8	Q_
d8	Q_
d9	R_
d9	R_
e2	S_
e2	S_
e3	T_
e3	T_
e4	U_
e4	U_
e5	V_
e5	V_
e6	W_
e6	W_
e7	X_
e7	X_
e8	Y_
e8	Y_
e9	Z_
e9	Z_
ad	[_
ad	[_
e0	_
e0	_
bd	]_
bd	]_
5f	^_
5f	^_
3d	^¬
3d	^¦_
6d	_
81	a
81	a_
82	b_
82	b_
83	c_
83	c_
4a	c¬
4a	c¦_
84	d_
84	d_
85	e_

HEX	ASCII
85	e_
86	f_
86	f_
87	g_
87	g_
88	h_
88	h_
89	i_
89	i_
91	j_
91	j_
92	k_
92	k_
93	l_
93	l_
94	m_
94	m_
95	n_
95	n_
96	o_
96	o_
97	p_
97	p_
98	q_
98	q_
99	r_
99	r_
a2	s_
a2	s_
a3	t_
a3	t_
a4	u_
a4	u_
a5	v_
a5	v_
a6	w_
a6	w_
a7	x_
a7	x_
a8	y_
a8	y_
a9	z_
a9	z_
8b	{_
8b	{_
4f	¦_
4f	¦_
9b	}_
9b	}_

Figure A-1. The ppf6023 character table.

HEX	ASCII	HEX	ASCII	HEX	ASCII	HEX	ASCII
40		f6	6	59	L	8a	e
5a	!	3b	6	d4	M	86	f
25	!	f7	7	62	M	90	f
7f	"	3c	7	d5	N	87	g
26	"	f8	8	63	N	9a	g
7b	#	3e	8	d6	O	88	h
27	#	f9	9	64	O	aa	h
5b	$	3f	9	d7	P	89	i
28	$	7a	:	65	P	ba	i
6c	%	41	:	d8	Q	91	j
29	%	5e	;	66	Q	ca	j
50	&	42	;	d9	R	92	k
2a	&	4c	<	67	R	cc	k
7d	'	8c	<=	e2	S	93	l
2b	'	23	<=	68	S	cd	l
4d	(	43	<	e3	T	94	m
2c	(	7e	=	69	T	cf	m
5d	)	ae	=>	e4	U	95	n
2d	)	16	=>	6a	U	d0	n
5c	*	44	=	e5	V	96	o
2e	*	be	=¬	70	V	da	o
4e	+	18	=¦	e6	W	97	p
11	+-	6e	>	71	W	dc	p
04	+-	45	>	e7	X	98	q
24	+	6f	?	72	X	dd	q
6b	,	46	?	e8	Y	99	r
33	,	7c	@	73	Y	de	r
60	-	47	@	e9	Z	a2	s
32	-<	c1	A	74	Z	df	s
07	-=	48	A	ad	[	a3	t
31	-	c2	B	db	[	ea	t
8f	-¬	49	B	e0	\	a4	u
4b	.	c3	C	e0	\	eb	u
30	.	51	C	bd	]	a5	v
61	/	c4	D	75	]	ed	v
02	/0	52	D	5f	^	a6	w
be	/=	c5	E	22	^	ee	w
18	/=	53	E	3d	^¬	a7	x
34	/	c6	F	3d	^¦	ef	x
f0	0	54	F	6d	_	a8	y
35	0	c7	G	81	a	fa	y
f1	1	55	G	e1	a	a9	z
36	1	c8	H	82	b	fb	z
f2	2	56	H	78	b	8b	{
37	2	c9	I	83	c	fc	{
f3	3	57	I	79	c	4f	¦
38	3	d1	J	4a	c¬	fd	¦
f4	4	58	J	17	c¦	9b	}
39	4	d2	K	84	d	fe	}
f5	5	cb	K	80	d		
3a	5	d3	L	85	e		

Figure A-2. The ppf6025 character table.

APPENDIX B

MULTICS PPS TAPE FORMAT

The tapes produced by the Multics PPS support package are described in detail in "OS and DOS Support for the Page Printing System" (order no. AR86). This format in terms of a Multics tape_ibm_ attach description is:

```
tape_ibm_ -vol volid -format fb -rec 133 -block 1596
    -mode ascii ...
```

The tape is written using the ascii mode due to the fact that character translation to the PPS character set is accomplished using the PPS character table.

See MPM Peripheral Input/Output (order no. AX49) for a complete description of tape_ibm_.

APPENDIX C

PPS I/O DAEMON OPERATION

This appendix is intended to serve as a guide for system administrators who are installing the Multics PPS software. The material in this appendix should be sufficient to allow an administrator to set up the PPS I/O Daemon and generate new character tables as needed.

SOFTWARE PACKAGE DESCRIPTION

The Multics PPS support software consists of the bound segments bound_pps_support_ and bound_pps_driver_ and the character table ppf6025.

INSTALLATION

These segments should be installed in >unbundled in the same fashion as any other unbundled Multics software.

I/O DAEMON SETUP

CHARACTER TABLE GENERATION

Character table generation is performed using the cv_ppsct command described in section VI.

A simple example of this process if given below. The character table generated defines the characters o, ^, _, o̲, ô, ^̲, and ô̲.

```
! qx
! a
! 10 00 /* Default = 10 and space = 00 */
! 01 o /* This is the first character defined */
! 02 ^
! 03 _
! 04 o_
! 05 o^
! 06 _^
! 07 o_^ /* This is 3 characters overstruck. */
! w test.ppscf
! q
  r 2222 0.236 21.425 34
! cv_ppscf test -lg
  22 out of 24576 table entries used.
  r 2224 1.643 44.364 54
! alm test
  r 2227 3.637 63.126 78
```