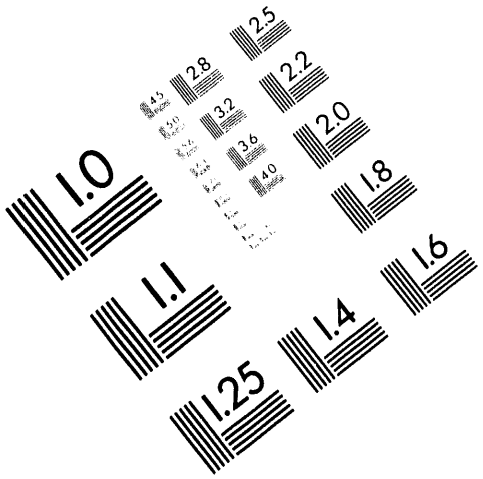
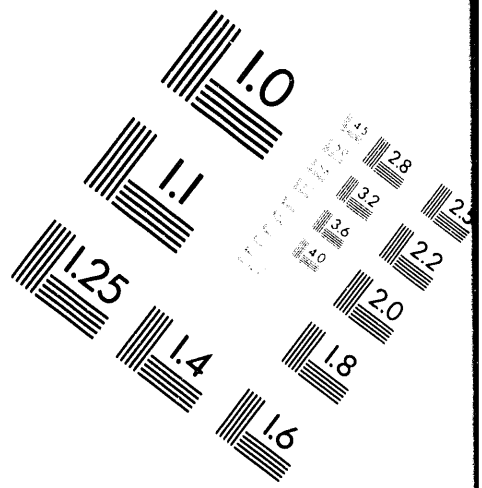
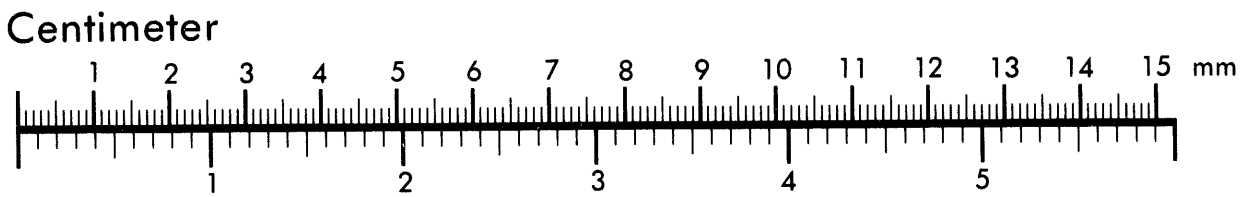


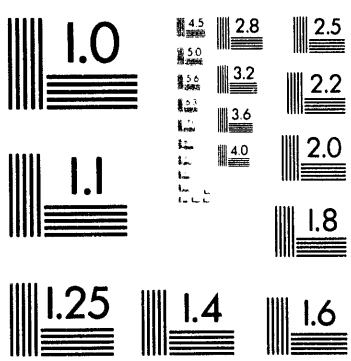
Association for Information and Image Management

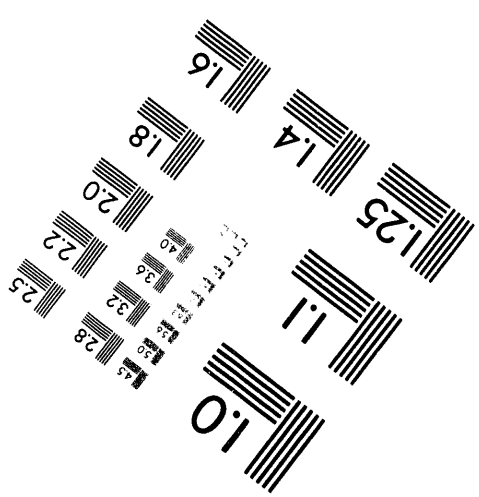
1100 Wayne Avenue, Suite 1100
Silver Spring, Maryland 20910

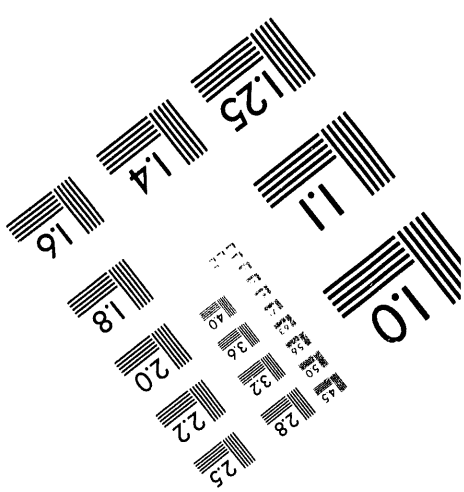
301/587-8202











MANUFACTURED TO AIIM STANDARDS
BY APPLIED IMAGE, INC.

1 of 1

ANL/IPNS/PP--79234

MPRINT: VAX Printing Made Simple

by Tom Worlton

Introduction

Users with stand-alone personal computers and personal printers usually find printing simple, but on a VAX computer or VAX cluster there may be many printers of different types located in different areas. The print queues set up for these printers may require different form qualifiers and may not all be able to print all documents. This article describes the basic steps a VAX system manager should take in setting up and managing print queues on a VAX and tells how to access these queues from a VAX, Unix, Macintosh, or DOS Computer. It gives a basic overview, but includes several helpful items that are obscure or completely undocumented. Following this overview, there is a description of a Fortran program, MPRINT, written to simplify printing for users.

The MPRINT program simplifies the choice of printers and print forms for users of VAX print queues by allowing them to select from a one line per queue menu. The menu includes queue descriptions and only lists printers which can correctly print the specified file. MPRINT selects the correct form to use based on file type and maximum record length of the file. MPRINT may be useful at your site, and provides examples of a number of system services. MPRINT includes routines to do a user open of a file and get information from the File and Record Access Blocks (FAB and RAB), and a routine to obtain information about print queues. There are also routines to parse a file name, detab a character buffer, and trim trailing nulls and blanks from a character string.

Types of Printers

Most VAX sites have Postscript laser printers, non-Postscript laser printers such as the LN03, and line printers. Many sites also have various types of plotters which are connected in the same way as printers. Postscript files contain formatting instructions, font selection and scaling, graphics, etc. in addition to text so will not print correctly on non-Postscript printers. Conversely, text files will not print on Postscript printers unless they are converted to Postscript format. Postscript printers such as the Apple Laserwriter were first used by Macintosh computers, but are now widely used on VAX computers because of the powerful font scaling and graphics they provide. Some of the newer Postscript laser printers have automatic emulation sensing and data conversion which allows sending Postscript, non-Postscript, and perhaps other types of files to the printer without any special setup modules. There are now also software products to do the conversion automatically before the file gets to the printer.

Connecting Printers

Printer connection includes the physical connection of the printer, any needed logical definition of the I/O port, and establishment of device characteristics. The latter two steps must be done each time the VAX node using the printer is rebooted, so must be initiated through SYS$MANAGER:SYSTARTUP_V5.COM.

One of the oldest and simplest methods of connecting a printer to a VAX is the direct serial connection. If your VAX has serial terminal lines available, you can connect to the printer serial port using a null modem cable with connectors which match the computer port and the printer.

Terminal server ports can also be used to set up connections to the serial ports of printers. This is especially useful if you have more than one VAX and need to share printers. You connect printers to terminal server ports in the same way that you connect serial printers to VAX serial I/O ports, but you must also issue the commands to define the port characteristics on the terminal server and issue the commands to create the port on the VAX using the LAT Control Program (LATCP).

Set characteristics of terminal server ports used for printers from a privileged port on the terminal server. For example, to connect a 9600 baud printer to port 3 of a terminal server which has DECnet node name dsv2, set the port characteristics with the commands:

```
Server> define port 3 access remote speed 9600 autobaud disabled
Server> set port 3 access remote speed 9600 autobaud disabled
```

The `define` command changes the permanent database, and the `set` command changes the volatile database. Parameters modified with the `set` command are lost when the port is logged out. Some terminal servers allow using the `change` command to modify both databases at once.

To use a printer attached to a LAT terminal server port, first create a logical LAT port on each VMS node which is to be able to send print jobs to the printer. In a cluster it is helpful to have two or more different nodes which can access the port in case one of the nodes fails. On VAX nodes, include commands to create the logical devices to correspond to terminal server ports in `SYS$MANAGER:LAT$SYSTARTUP.COM`:

```
$ lcp :==$latcp
$ lcp set node/state=on
.
.
.
$ lcp create port lta800:/queued/application/node=dsv2/port=port_3
```

The `create port` commands should be issued after setting the node state to on. Also note that you can choose any number you like for the LTA device number, but you should avoid very low numbers. Users connecting to a VAX through terminal servers will be assigned LTA numbers starting at 1 and increasing with each login.

Set terminal characteristics for all printer and terminal devices used for print queues with `SET TERMINAL` commands. These should be executed during startup (after port creation in `LAT$SYSTARTUP.COM`). For instance, for one of our LN03 print queues we set up the terminal as follows:

```
$ SET TERMINAL LTA800:    /DEVICE_TYPE=LN03/PERM/EIGHTBIT/PASTHRU-
                          /WIDTH=255/PAGE=63/SPEED=9600/HOSTSYNC -
                          /NOTYPEAHEAD/NOBROADCAST/NOECHO
```

The /PASTHRU qualifier is needed to allow printing of Tektronix graphics files. The page length has been set to 63 instead of 66 because we prefer to use 6 lines per inch since we sometimes print labels, and 0.25" at the top and bottom of the page are not printable.

Set your LAT port devices spooled to prevent other users from writing directly to them and causing problems with the queues. When a process directs its output to a spooled printer, the output is put in a temporary file on the storage device. When the output completes, the file is closed and submitted for printing on the associated output queue. The example terminal can be defined as a spooled device with the command:

```
$ SET DEVICE LTA800:/SPOOLED=(NPB,SYS$SYSDEVICE)
```

The arguments to the SPOOLED qualifier are the queue name and the disk device to contain the intermediate file. The SET TERMINAL and SET DEVICE commands should be put in a file such as DEVICE_SETUP.COM and executed from SYSTARTUP_V5.COM as follows:

```
$@SYS$MANAGER:DEVICE_SETUP.COM
```

Printers with an AppleTalk connection can be connected just like any other AppleTalk node and set up for VAX user access through Pathworks for Macintosh. Pathworks software also allows Mac users to access VAX printers. The AppleTalk name of the printer is initially the default name for that type of printer. You will probably want to use the LaserWriter Utility program on a Macintosh computer to rename the printer to a name specific to your site and the printer's location. The LaserWriter Utility should be run after selecting the desired printer using the Chooser. When the desired printer is displayed in the "Current name:" field, type the new name of the printer in the "New name:" field. Anyone who has chosen the renamed printer as their default printer will have to reselect the printer from the Chooser.

The Queue Manager

On a VAX the batch and print (output) queues are controlled by a queue manager which maintains a database with queue information. User processes, symbionts, and job controllers on each node communicate with the centralized queue manager through a shared interprocess communication (IPC) link[1] . Before you can enter an queue command, you must start the queue manager. Prior to VMS 5.5 the queue file was JBCSYSQUE.DAT and the queue manager had to be restarted in SYS$MANAGER:SYSTARTUP_V5.COM. The VMS 5.5 queue manager restarts automatically and uses three database files:

```
QMAN$MASTER.DAT
```

---

[1] *See Guide to Maintaining a VMS system, Chapter 5, Batch and Print Operations.*

```
SYS$QUEUE_MANAGER.QMAN$QUEUES
SYS$QUEUE_MANAGER.QMAN$JOURNAL
```

VMS 5.5 assumes that the queue database files are in SYS$COMMON:[SYSEXE] unless the logical name QMAN$MASTER is defined in SYLOGICALS.COM to point to a different location[2] . On a cluster, QMAN$MASTER must be identically defined on all nodes. This means that if you have multiple system disks, you should not specify SYS$SYSTEM as the value of QMAN$MASTER. This new queue manager has a fail-over capability which allows queue management to be transferred to another node if the controlling node fails and leaves the cluster. The VMS 5.5 queue manager is started with the command:

```
$ START/QUEUE/MANAGER[/ON=(node-list)] [dirspec]
```

The /ON=(node-list) option allows you to specify a list of nodes for the queue manager to run on. It will run on the first active node in the list and fail over to the next node if that node goes down. The node list can be terminated with an asterisk (*) so the queue manager will always have at least one node in the cluster to run on. The job controller will automatically start the queue manager during reboot unless you have stopped it manually, so the command to start the queue manager does not need to be included in SYSTARTUP_V5.COM.

When you upgrade to VMS 5.5 you have the option of keeping the old queue manager. Since Alpha AXP computers do not run VMS 5.5 software, you should stick with the old queue manager if you will be using Alpha machines.

There are two types of batch and print queues: generic queues and execution queues. A print execution queue is a queue associated with a specific printer. A print queue going to a terminal (serial) line is called a terminal queue. To simplify maintenance of execution queues, they may be designated as autostart queues so you will not have to issue a separate START/QUEUE command for each queue when you reboot. Instead you include the single command:

```
$ ENABLE AUTOSTART/QUEUES
```

in SYSTARTUP_V5.COM. This will start all autostart queues, unless they have been stopped manually.

A generic queue is an intermediate queue that holds a job until an appropriate execution queue becomes available to initiate the job. One generic queue can supply jobs to several similar print execution queues. Execution queues must be created before creating associated generic queues. A logical queue is a special type of generic queue which transfers jobs into the execution queue specified with the ASSIGN/MERGE command. SYS$PRINT is a special generic queue which will receive all jobs started by a print command without a /QUEUE qualifier. It can target one or more physical queues. Our

---

[2] *Guide to Maintaining a VMS System, 5.3.1.5 Moving Queue Database Files.*

physical line printer queue is called CI600 and we use the following command to start SYS$PRINT:

```
$ init/queue/start/generic=(CI600) SYS$PRINT
```

Restarting The Queue Manager

There are sometimes problems with queue access from one or more nodes in a cluster. To restart the queue manager in this case, log into the SYSTEM account and do the following:

1. Find the JOB_CONTROL process id by using the SHOW SYSTEM command.
2. Use the STOP/ID= command to stop the JOB_CONTROL process.
3. Issue the command:

   ```
   $ @SYS$SYSTEM:STARTUP JOBCTL
   ```

   to start the job controller and queue manager.

Print Symbionts/Supervisors

A print symbiont or printer supervisor manages printers, formats files for printing, and handles communication with printers. The default print symbiont process for VMS is named PRTSMB. Printers attached to terminal server LAT ports must use the print symbiont LATSYM. There are some layered products and third party SYMBIONTS which not only manage printers, but also increase their functionality.

DEC has just announced an impressive new product called DECprint Supervisor for OpenVMS (DCPS).[3] . DCPS plus does automatic data type detection and translation for PCL, DECPPL3, ANSI, Proprinter, DDIF, ReGIS, and TEK files. It also allows printing more than one page per sheet.

Pathworks for Macintosh includes conversion utilities for converting text files to Postscript, but it is up to the user to choose the correct form to use, while DCPS does automatic file type detection by examining the file, and converting it to Postscript if needed. The disadvantage of DCPS is that it supports only the serial line connection for third party printers.

Device Control Libraries

Printer characteristics can be changed through control sequences sent to the printer. Groups of these control sequences are generally stored in a device control text library in SYS$LIBRARY:. If you do not specify a library

---

[3] *DECprint Supervisor for OpenVMS (DCPS) replaces DECprint Printing Services for VMS which previously replaced Scriptprinter Software. DCPS is available in base, open, and plus versions. The base license is included with OpenVMS and supports only DEC Postscript printers and only includes ASCII file conversion. The Open and Plus versions support Apple and HP printers and add more file conversion capabilities.*

when you create the queue, the system assumes the library SYSDEVCTL. We define separate device control libraries for each type of printer. For instance, for LN03 type printers we have created LN03DEVCTL.TLB. Some of the setup modules in our LN03 device control library are: letter, listing, label, tekon. The setup modules in the library and the form names used may be the same for different printers, but the control sequences contained in the setup modules may be quite different in the different libraries. Printer queues created through Pathworks for Macintosh use device control library MSAP$DEVCTL.TLB. It contains setup modules such as MSAP$APPLEDICT71, PS_PLAIN_PROLOG, LTR_10_PROLOG, etc. Setup modules can be extracted from the library and examined with a text editor. They also can be modified and replaced in the library, but this requires stopping all queues using that library.

Defining Queue Forms

A form is used to specify how the text is to be printed on the page and what setup module(s) are needed to prepare the printer. The form definition can include margins, length, width, type of paper (stock type), setup modules to be sent to the printer, etc. Our form definitions are made in the file SYS$MANAGER:QFORM.COM. Our form for printing 132 column text in landscape mode is called LAND8 and is defined as follows:

```
$DEFINE/FORM LAND8 5/STOCK=DEFAULT/WIDTH=132/LENGTH=66-
                   /NOWRAP/NOTRUNCATE/MARGIN=(BOTTOM=0)-
                   /DESCRIPTION="66/132 Landscape mode"-
                   /SETUP=LNLAND8
```

You must be careful to standardize the names of the stock types or jobs may be held up unexpectedly. When the stock specified in a print command or print form does not match the current stock type, the job will be put into a pending state due to a form/stock mismatch.

Pathworks for Macintosh print forms are defined in the command procedure:

```
msa$root:[msa.msap$utility]msap$build_form.com
```

Because we had previously used DEFAULT as the stock type for 8.5x11 white paper, when we installed Pathworks for Macintosh, we edited msap$build_form.com to change /stock="Plain_Paper" to /stock="DEFAULT".

Creating Print Queues

Print and batch queues are created through the INITIALIZE/QUEUE command, except for queues to AppleTalk printers which are set up using Pathworks. For example, to create a print queue on an LN03+ printer on terminal server port LTA800, you could issue the command:

```
$ INITIALIZE /QUEUE /FORM=DEFAULT /LIBRARY=LN03DEVCTL -
    /PROCESSOR=LATSYM /RETAIN=ERROR -
    /DEFAULT=(NOBURST,NOFLAG,NOTRAILER) -
```

```
/DESCRI="Bldg. 360 E101--LN03+ printer in main control room." -
/AUTOSTART_ON=(ANLPNS::LTA800:,ANPNS8::LTA800)  NPB
```

The /AUTOSTART_ON qualifier is only used on systems running the VMS 5.5 queue manager. Older systems should use /ON instead. Under the VMS 5.5 queue manager, you can specify a list of nodes and devices to which an autostart queue can fail over. In this example, the LTA800 port must have been created previously on nodes ANLPNS and ANPNS8 and the terminal characteristics set on both nodes.

Generic queues are started with the command:

```
$ INIT/QUE/GENERIC=(execution-queue-list) queue-name
```

Queues for Postscript printers which are connected to the AppleTalk network are created using Pathworks for Macintosh as follows:

```
$ ADMIN/MSA
MSA$MANAGER>ADD PRINTER DP960_ANLPNS/QUEUE=DP960-
_$/DESTINATION="IPNS 360 DP LZR 960@IPNS 360"-
_$/SETUP=(MSAP$APPLEDICT71) /FONTS=MSAP$FONTLIST_APPLE35.TXT-
_$/DEFAULT=(FORM=(PS_PLAIN))/NORECEIVER
MSA$MANAGER> START PRINTER DP960_ANLPNS
MSA$MANAGER> EXIT
```

The Pathworks printer name (here DP960_ANLPNS) must be different from the VAX queue name (here DP960) and each VAX node must use a different Pathworks printer name, so we have appended the node name in our definitions. The destination is the Apple printer name as set using the Apple LaserWriter Utility on the Macintosh[4] , followed by "@" and the name of the Macintosh zone containing the printer (in our example, the zone is "IPNS 360"). The values for /FONTS and /SETUP assume that you are running Macintosh system 7.x and have a Postscript printer with the 35 fonts included in the Apple Laserwriter plus. If you are using a Postscript printer with a different font set, there are other choices of fonts listed in the Pathworks documentation. The /NORECEIVER is an undocumented qualifier which prevents Pathworks from creating a chooser entry with the Pathworks printer name. If you do not specify this qualifier, each VAX that creates a queue for the AppleTalk printer will create another LaserWriter entry in the Mac Chooser.[5] DEC provides a file, MSA$ROOT:[MSAP.MSAP$UTILITY]MSAP$SET_PARAMS.PS, with Pathworks for Macintosh which can be copied to an Apple Laserwriter to

---

[4] *See the last paragraph in the "Connecting Printers" section.*

[5] *In the current example, without the /RECEIVER qualifier, Mac users would see the following two entries:*

IPNS 360 DP LZR 960@IPNS 360
DP960_ANLPNS

With the /RECEIVER qualifier, they would only see the first entry.

initialize the baud rate, etc. Make sure your printers actually match the settings in this file before using it (9600 baud, etc.) The file MSAP$SHOW_PARAMS.PS can be printed or copied to Postscript printers to cause them to print a sheet listing the Baud rate, partity, etc.

VAX print queues can also be made available to Mac users by using Pathworks to create a Chooser entry for the VAX queue as follows:

```
$ ADMIN/MSA
MSA$MANAGER> ADD PRINTER "LW on VAX"-
_MSA$MANAGER> /QUEUE=LW_VAX/DEST=LTA820:
MSA$MANAGER> START PRINTER "LW on VAX"
MSA$MANAGER> EXIT
```

This example creates a Chooser entry "LW on VAX" to allow printing on the VAX queue "LW_VAX" which is connected to terminal server port LTA820:. The destination for a serially connected printer would be a physical device name such as "TXA5:". The quotation marks are necessary in the example whenever you have spaces and/or mixed case characters in the Pathworks printer name.

Prior to VMS 5.5 all queues had to be restarted individually when the system was rebooted. With the VMS 5.5 queue manager, a single `ENABLE AUTOSTART/QUEUES` command will start all autostart queues that have not been stopped manually.

Using Print Queues

Mac users select a printer by going into the Chooser, selecting either the LaserWriter icon or the LW 7.1 N-up icon, then selecting an AppleTalk Zone and a LaserWriter[6] . This only has to be done once unless the name of the printer changes, or you want to change your default print device.

PC users use the "use" command to choose printers to be associated with LPT1, LPT2, and/or LPT3. For instance the command

```
USE LPT1:\\MYVAX\MYPRINTER
```

will cause applications which send output to LPT1 to print on queue MYPRINTER on VAX node MYVAX.

VAX users print files using the print command with the `/QUEUE=` qualifier specifying the queue and the `/FORM=` qualifier specifying the form to use. Postscript printers connected to AppleTalk and using MSAP$SYMBIONT and the MSAP$DEVCTL library can print Postscript files or text files. The user signifies the need for the symbiont to translate text into Postscript through the form specified in the print command. For instance a Fortran source file and listing file could be printed on queue DP960 with the commands:

---

[6] *This could be any Postscript printer with an Appletalk connection or any VAX printer defined through Pathworks for Macintosh*

```
$ print/que=dp960/form=ltr_10 program.for
$ print/que=dp960/form=lpt_plain program.lis
```

A Postscript file would be printed on the same queue by using /form=ps_plain. While these features are nice, it can be confusing to users because of the requirement of different forms for different kinds of files.

Users on Unix systems can print to OpenVMS print queues if the OpenVMS system has TCP/IP networking and has enabled LPD access from the Unix system. Our VAX computers use Multinet from TGV, Inc. to provide TCP/IP networking. In this case LPD print queue access is enabled by the following procedure:

```
$ MULTINET CONFIGURE /SERVERS
SERVER-CONFIG>SELECT LPD
SERVER-CONFIG>SET ACCEPT-HOSTS
. . .
Add Address: 192.41.228.1
Add Address:
SERVER-CONFIG>RESTART
SERVER-CONFIG>EXIT
$
```

(Substitute the correct Unix system Internet address in place of "192.41.228.1".)

The Unix system which needs to access VAX print queues must include entries for the VAX queues in the file /etc/printcap. The printcap entries must include the :rm (remote machine) parameter and the :rp (remote printer) parameters in additional to the normal printcap parameters. The remote machine value is the Internet address of the VAX where the print queue is located. The remote printer value is the normal print queue name used on the VAX. A sample entry in /etc/printcap is:

```
lp_960|lp|0|DP960 Laser printer as postscript\
        :lf=/usr/adm/lpd-errs:\
        lp=:\
        :mx#2000:\
        :rm=anlpns.pns.anl.gov:\
        :rp=dp960:\
        :sd=/usr/spool/lpd1:
```

Users print to these queues using the lpr command.

```
>lpr -Plp1 file.c
```

The MPRINT Program

MPRINT is a menu-driven print program which will determine the file type of the file to be printed and display a one line per queue menu[7] of print queues (with descriptions) which can be used to print the specified file. For text files, MPRINT will also determine the longest record length and select the proper form for that size of record. The source code for MPRINT.FOR is shown in Fig. 1.

MPRINT first uses SMG to create a pasteboard, a virtual keyboard, and a virtual display to contain the full file specification and the menu of appropriate queues for that file. It then gets the file name from the command line or prompts for a file name. LIB$FIND_FILE is then used to find the full file specification, and the file name is parsed into a name and type. Unless the file type is TK4014, the file is opened (READONLY) and the first record is examined to determine the type of file and the longest record length.[8] Since MPRINT will only list Postscript printers when the file is a Postscript file, it prevents users from trying to print a Postscript file on a printer which does not handle Postscript.

For text files, MPRINT also determines the longest record in the file, so that it can choose the correct form qualifier. The longest record length can be obtained from the XAB block obtained through a user open. It is contained in XABFHC.FHC.XAB$W_LRL. MPRINT_OPEN opens the file READONLY and references the Fortran user open routine OPEN_CONTIG. OPEN_CONTIG obtains the longest record length, lrl, and passes it to MPRINT_OPEN through common. MPRINT_OPEN then reads the first record of the file for file type determination and returns the longest record length and the first record from the file to MPRINT. MPRINT uses the longest record length to select the correct form for printing text files. We don't have any printer queues set up to print records wider than 132 columns, but MPRINT could be modified easily to handle that case. The printout of a text file with records longer than 132 characters will have truncated or wrapped lines on our queues.

The subroutine GET_QUE_INFO is then called to find all the queue names and the descriptions and library names for each print queue using the SYS$GETQUIW system service with the QUI$_DISPLAY_QUEUE function with QUI$_SEARCH_NAME. The search flags used include QUI$M_SEARCH_WILDCARD, QUI$V_SEARCH_TERMINAL, and QUI$V_SEARCH_PRINTER. If the queue has no description and MPRINT is being executed from the SYSTEM account, you are prompted for a description, and a command is spawned to add the description to the queue.

SMG routines are then used to create a menu of queues with descriptions (one line per queue) for the user to select from. The default menu selection is the queue matching the name of the logical MYQUE. The last menu entry is "Cancel". Unless the user selects "Cancel" a print

---

[7] *MPRINT uses SMG menu routines.*

[8] *The first two characters of a Postscript file should be "%!". See The Postscript Language Reference Manual, Appendix C, P. 281. Not all people follow this convention, so MPRINT also assumes files of type "PS" are Postscript and files that start with " initgraphics" are Postscript files.*

command is composed with a form qualifier based on the file type and the device control library for the queue which was chosen.

Installing MPRINT

To install MPRINT on your system:

1. Choose a disk device, dev, and create an [MPRINT] directory
   ```
   $ create/dir dev:[MPRINT]
   ```
2. Copy MPRINT.FOR and MPRINT.INC to the directory you created.
3. Use a text editor to add the following symbol definition to SYS$MANAGER:SYLOGIN.COM:
   ```
   $ MPRINT:==$dev:[MPRINT]MPRINT.EXE
   ```
4. Edit MPRINT.INC to put the proper form names for each type of printer on your system (i.e. for each device control library).
5. Compile and link MPRINT with the commands:

   ```
   $ FORTRAN MPRINT
   $ LINK MPRINT
   ```

Make sure the files in dev:[MPRINT] have a protection of WORLD:RE.

Customizing MPRINT

MPRINT is dependent on a knowledge of which forms are appropriate for different files to be printed. These forms should be the same for all printers using a given device control library. Data statements are used to define the library name and corresponding form name for each type of printer. To simplify customization, the data statements are in a separate include file, MPRINT.INC (see Fig. 2). Four types of forms are defined:

1. psform(i) for Postscript files
2. tkform(i) for Tektronix files
3. txt80form(i) for 80 column or less text
4. txt132form(i) for text wider than 80 column

A system manager can customize MPRINT for the forms in use at a given site by editing MPRINT.INC and changing the form names and/or device control library names as needed. If additional types of printers are needed, add the definitions in the same format as the previous entries, and increase the value of NUMLIB to match the number of printer types defined.

If the user has defined the logical name MYQUE to point to a valid queue, that queue will be the default. Users may want to include a definition of MYQUE in their LOGIN.COM file. For instance,

```
$ DEFINE MYQUE DP1260
```

will cause the DP1260 queue to be the default queue selection for MPRINT. If MPRINT has been used by the current process, the previously chosen queue becomes the default queue.

Using MPRINT

Once you have installed MPRINT, you can use the verb MPRINT instead of the PRINT command. If you include a filename, MPRINT will examine that file and then display a menu of appropriate printer queues. If you do not specify a file name, MPRINT will prompt you for a file name before displaying the menu. You should not specify any qualifiers. MPRINT will furnish the `/QUEUE`, `/FORM`, and `/NOTIFY` qualifiers.

The menu presented to the user will only show queue names and descriptions. Either the arrow keys or the mouse can be used to make a queue selection. To go to the bottom or top of the queue list, enter the PF1 key followed by the down-arrow key or up-arrow key. When the user selects an output queue, MPRINT will compose a print command and write it to SYS$OUTPUT then execute the command.

Some examples of MPRINT commands are:

```
$ mprint draw.ps

$ mprint program.for

$ mprint program
```

In these examples, if "draw.ps" is a Postscript file, only Postscript print queues will appear in the menu (see Fig. 3), but the menu for printing "program.for" will include all printers capable of printing a text file (see Fig. 4). In the third example no file type is specified, so MPRINT will assume ".LIS". If `program.for` has records less than or equal to 80 bytes and `program.lis` has longer records, MPRINT will furnish different forms qualifiers for printing the two files. The user will only know this through the echoing of the print command generated by MPRINT.

Part of the task of downsizing is to make it easy for various computing systems to work together. For most of us that means OpenVMS, Unix, DOS, and MacIntosh systems. This article has explained how those four computer systems can set up and share printers. The MPRINT program described here should be useful for any site with a lot of print queues and users who work from different locations. It does not support all the print command qualifiers, but is an easy way to select the correct form and queue for printing a given file.

```
      PROGRAM MPRINT
!+
! VERSION:      1-001
!
! ABSTRACT:     MPRINT is a program to simplify printing on a VAX by
!               allowing the user to select a queue from a menu.  MPRINT
!               examines the file to determine file type and maximum record
!               length and furnishes the appropriate print qualifiers.
!
! ENVIRONMENT:  User Mode, AST-reentrant
!
! AUTHOR:       Thomas G. Worlton               CREATION DATE:  1-Jan-1993
!               Argonne National Laboratory
!
! MODIFIED BY:
! 1-001 - Original. TGW 1-Jan-1993
! 1-002 - TGW 1-Mar-1993        Added MYQUE logical to set default queue
!-
      PARAMETER MAXQ = 100
      CHARACTER*76 header,fname,file,head1*20,head2*30
      CHARACTER type*10,name*30
      INCLUDE '($SMGDEF)'
      INCLUDE 'mprint.inc/list'
      INTEGER pbid, kbid, ldim, lmax, rows, columns,lhead1
      INTEGER dsid, ds2, ds3, ds4, NQ
      INTEGER SMG$CREATE_PASTEBOARD,SMG$CREATE_VIRTUAL_KEYBOARD
      INTEGER SMG$CREATE_VIRTUAL_DISPLAY,SMG$SET_KEYPAD_MODE
      INTEGER SMG$PASTE_VIRTUAL_DISPLAY,context,wide
      INTEGER SMG$CREATE_MENU,SMG$DELETE_MENU,SMG$READ_STRING
      INTEGER SMG$SELECT_FROM_MENU, SMG$FLUSH_BUFFER
      INTEGER SMG$DELETE_VIRTUAL_DISPLAY, SMG$DELETE_PASTEBOARD
      INTEGER SMG$REPAINT_SCREEN, SMG$READ_KEYSTROKE
      INTEGER*2 term_code
      INTEGER jlist(MAXQ), jlib(MAXQ), LMQ
      CHARACTER*16 q(MAXQ), MYQUE
      CHARACTER*39 LIBRARY(MAXQ)
      CHARACTER*56 description(MAXQ)
      CHARACTER*72 choices(MAXQ)
      CHARACTER*80 buffer
      CHARACTER form*9, rec1*13
      CHARACTER*132 command
      LOGICAL SHOW_QUEUE
!+
! Create the PASTEBOARD.
!-
      istatus = SMG$CREATE_PASTEBOARD (pbid)
!+
! Create a virtual keyboard.
!-
      istatus = SMG$CREATE_VIRTUAL_KEYBOARD (kbid)
!+
! Create the main virtual display
!-
      rows=22
      columns=78
      istatus = SMG$CREATE_VIRTUAL_DISPLAY(rows,columns,dsid,
     1                                     SMG$M_BORDER)
!+
! Translate MYQUE logical
!-
        call LIB$SYS_TRNLOG('MYQUE',LMQ,MYQUE)
!+
! Paste the virtual display
!-
      istatus = SMG$PASTE_VIRTUAL_DISPLAY(dsid,pbid,2,2)
```

```
      head1 = 'MPRINT--Choose Queue from a menu'
      ic1 = 0.5*(columns - jlen(head1))
      istatus = SMG$PUT_CHARS (dsid,head1,1,ic1,,SMG$M_BOLD)

!+
!  Get Print/Plot file name:
!-
      head2 = 'Print/Plot file name:'
      icp = jlen(head2) + 2
      wide = columns - icp
      main_choice = nchoices

      istatus = LIB$GET_FOREIGN(fname,,lfname)
      if(lfname .lt. 1) then
101     istatus = SMG$SET_CURSOR_ABS(dsid,2,1)
        call SMG$READ_STRING(kbid, fname, 'Print/Plot file name: ',
     1                       wide-2, , , ,lfname, term_code, dsid,
     2                        ,SMG$M_BOLD)
        if(lfname .lt. 1) goto 30       ! Exit on null file name
      end if
!+
! Find the file to be printed
!-

      istatus = LIB$FIND_FILE(fname,file,context,'*.lis;0')
      IF (.NOT. istatus) THEN
        CALL LIB$SIGNAL(%VAL(istatus))
        istatus = SMG$PUT_CHARS (dsid,'Press Enter to continue.',6,2,
     1                           SMG$M_ERASE_LINE,SMG$M_BOLD)
        istat = SMG$READ_KEYSTROKE(kbid, term_code)
        istat = SMG$REPAINT_SCREEN(pbid)
        GOTO 101
      end if
      istatus = LIB$FIND_FILE_END(context)

      lfname = jlen(file)
      istatus = SMG$PUT_CHARS (dsid,file(1:lfname),2,icp,,SMG$M_BOLD)

!+
!  Get the name and type from the full file specification
!-
      call fparse(file,name,type)
!+
! Find the file type from the file itself or file name/type
!-
      iftype = 0
      iformat = 0
      ls = 3    ! starting line for last menu
      lfname = jlen(fname)
      if (type .eq. 'GKSM' ) THEN
        istatus = SMG$DELETE_PASTEBOARD(pbid)
        istatus = LIB$DELETE_LOGICAL('METAFILE')
        istatus = LIB$SET_LOGICAL('METAFILE',fname(1:lfname))
        if(.not. istatus) CALL LIB$SIGNAL(%VAL(istatus))
        istatus = LIB$DO_COMMAND('$ RUN GPLOT_DIR:GKSPOST')
      else if (type .eq. 'DAT' .and. name .eq. 'PLT2'    .or.
     1                               name .eq. 'POPFIL'      ) THEN
        istatus = SMG$DELETE_PASTEBOARD(pbid)
        istatus = LIB$DELETE_LOGICAL('METAFILE')
        istatus = LIB$SET_LOGICAL('METAFILE',fname(1:lfname))
        if(.not. istatus) CALL LIB$SIGNAL(%VAL(istatus))
        istatus = LIB$DO_COMMAND(
     1          '$RUN UD5:[DISSPLA.CADIS110.DISMOD]POST11')
      else if (type .eq. 'TK4014' ) THEN
        iftype = 3      ! Tektronix file
      else if (type .eq. 'PS') THEN
```

```
      ' iftype = 1                ! Postscript file
        else
          istat = MPRINT_OPEN(file, lrl, rec1)
          istatus = SMG$SET_CURSOR_ABS(dsid,22,2)
          if(rec1(1:2) .eq. '%!') then    ! See Postscript Ref. Manual, App. C.,
            iftype = 1              ! Postscript file
          else if (rec1 .eq. ' initgraphics') then
            iftype = 1              ! Postscript file (TopDrawer)
          else
            if(lrl .le. 80) then
              iformat = 1
            else if (lrl .le. 132) then
              iformat = 2
            else
              iformat = 3
c              istatus = SMG$SET_CURSOR_ABS(dsid,20,2)
c              print *,'Warning: maximum record length =', lrl,
c     1                ' exceeds 132 characters'
            end if
            iftype = 2     ! Text file
          end if
        end if
!+
!  Get print queue names and descriptions
!-
          CALL GET_QUE_INFO(Q, DESCRIPTION, LIBRARY, NQ, MAXQ)
          istat = SMG$REPAINT_SCREEN(pbid)
        j = 0
        ndef = 1
        ltype = jlen(type)
!+
! Select useable queues for this file type as determined from the
! device control library used for each queue, and build list of queue
! names and descriptions.
!-
        do i=1,nq           ! Loop through all queues found
          jlib(i) = 0
          do il = 1,numlib          ! Find index to Device control library
            if(library(i) .eq. libname(il)) jlib(i) = il
          end do
          if(jlib(i) .ne. 0) then
            if(iftype .eq. 1 .and. psform(jlib(i)) .ne. '-') then
              show_queue = .TRUE.
            else if (iftype .eq. 2 .and. txt80form(jlib(i)) .ne. '-') then
              show_queue = .TRUE.
            else if (iftype .eq. 3 .and. tkform(jlib(i)) .ne. '-') then
              show_queue = .TRUE.
            else
              show_queue = .FALSE.
            end if
            if(show_queue) then
              j = j + 1
              jlist(j) = i
              if ( q(i) .eq. myque(1:lmq)) ndef = j
              choices(j) = q(i)//description(i)
            end if
          end if
        end do
        choices(j+1) = 'Cancel'
        nchoices = j+1
!+
! Create and paste new virtual display and create the menu.
!-
        rows = nchoices + 1
        if(rows .gt. 19) rows = 19
        columns = 74
```

```
      istatus = SMG$CREATE_VIRTUAL_DISPLAY(rows,columns,ds4,
     1                                     SMG$M_BORDER)
      ISTATUS = SMG$PASTE_VIRTUAL_DISPLAY(ds4,pbid,5,3)
      istat = SMG$PUT_CHARS(ds4,'Printer          Description',1,2,
     1                      SMG$M_ERASE_LINE,
     2                      SMG$M_BOLD.OR.SMG$M_UNDERLINE)
      irow = 2  ! start menu on 2nd row of virtual display
      istat = SMG$CREATE_MENU (ds4, choices,SMG$K_VERTICAL,,irow)
      IF (.NOT. istat) CALL LIB$STOP(%VAL(istat))
!+
! Select print queue from the menu
!-
      istat =SMG$SELECT_FROM_MENU(kbid,ds4,main_choice,ndef)
      IF (.NOT. istat) CALL LIB$STOP(%VAL(istat))
30    continue
      istatus = SMG$SET_CURSOR_ABS(dsid,20,2)
      istatus = SMG$DELETE_PASTEBOARD(pbid)
      if(main_choice .eq. nchoices) then
        write(6,*) 'Printing Cancelled.'
      else
        ip = jlist(main_choice)
        lq = jlen(q(ip))
        lfi = jlen(file)
!+
! Find form used for this type of file by chosen queue.
!-
        il = jlib(ip)    ! Store library number in il
        if(iftype .eq. 1) then
          form = psform(il)
        else if (iftype .eq. 2) then
          if(iformat .eq. 1) then
            form = txt80form(il)
          else if (iformat .eq. 2) then
            form = txt132form(il)
          else if (iformat .gt. 2) then
            form = txt132form(il)
            print *,'Warning: file contains records up to',
     1                      lrl,' bytes long.'
            print *,'Some characters may be truncated or wrapped.'
          end if
        else if (iftype .eq. 3) then
          form = tkform(il)
        end if
        lf = jlen(form)
        command(1:) = '$ print/notify/que='//q(ip)(1:lq)//
     1                      '/form='//form(1:lf)//
     2                      ' '//
     3                      file(1:lfi)
        write(6,*) command
        istat = lib$spawn(command)
        istatus = LIB$SET_LOGICAL('MYQUE',q(ip)(1:lq))
      end if
      ndef = main_choice

      call exit
      end
```

```
!+
! Subroutine MPRINT_OPEN opens a file and specifies a Fortran
! user open routine (OPEN_CONTIG) which returns the largest
! record length (lrl) through common.  MPRINT_OPEN then reads
! and returns largerec=lrl and record1=1st record in the file.
!-
        SUBROUTINE MPRINT_OPEN(file_name, largerec, record1)
        IMPLICIT INTEGER*4 (A-Z)
        INTEGER*2 BLOCK_SIZE, RECORD_SIZE, LRL, GROUP, MEMBER
        BYTE RECORD_FORMAT, RSL
        CHARACTER*(*) file_name, record1
        CHARACTER*255 RSA

        COMMON /RMS/ BLOCK_SIZE, RECORD_SIZE, RECORD_FORMAT, LRL,
        1            GROUP, MEMBER, UIC, RSA, RSL

        EXTERNAL OPEN_CONTIG

        lname = jlen(file_name)
        OPEN (UNIT=10, FILE=FILE_NAME(1:LNAME), STATUS = 'OLD'
        1     , USEROPEN=OPEN_CONTIG, RECL=512, READONLY)

        READ(10, '(A)') record1
        CLOSE (unit=10)
        largerec = lrl

        RETURN
        END
!+
! Function OPEN_CONTIG
! This useropen routine accesses the FAB, the NAM, and two of the XAB
! blocks.  Please note that the pointers to the NAME & XAB blocks must
! be established before the RMS $OPEN (or $CREATE for new files).  This
! signals RMS to provide you with that information.
!
! Based on USER_OPEN/OPEN_CONTIG routines obtained from DEC, with some
! changes by Tom Worlton
!-
        INTEGER FUNCTION OPEN_CONTIG (FAB, RAB, LUN)
        IMPLICIT INTEGER*4 (A-Z)
        INTEGER*2 BLOCK_SIZE, RECORD_SIZE, LRL, GROUP, MEMBER
        BYTE RECORD_FORMAT, RSL
        CHARACTER*255 RSA

        COMMON /RMS/ BLOCK_SIZE, RECORD_SIZE, RECORD_FORMAT, LRL,
        1            GROUP, MEMBER, UIC, RSA, RSL

        INCLUDE '($FABDEF)'
        INCLUDE '($NAMDEF)'
        INCLUDE '($RABDEF)'
        INCLUDE '($XABDEF)'
        INCLUDE '($XABFHCDEF)'
        INCLUDE '($XABDATDEF)'
        INCLUDE '($XABPRODEF)'
        INCLUDE '($XABALLDEF)'

        RECORD /FABDEF/ FAB, /NAMDEF/ NAM, /RABDEF/ RAB

! The structure XXX sets up a single XAB block definition,
! with an optional sub-structure for each XAB type.  To
! see how VMS defines these structures you can produce a
! compiler list file with the following:
!     INCLUDE '($XXXDEF)/LIST'

        STRUCTURE /XXX/
                UNION
```

```
                MAP
                RECORD /XABDEF/ XAB
                END MAP
                MAP
                RECORD /XABFHCDEF/ FHC
                END MAP
                MAP
                RECORD /XABDATDEF/ DAT
                END MAP
                MAP
                RECORD /XABPRODEF/ PRO
                END MAP
                MAP
                RECORD /XABPRODEF1/ PRO1
                END MAP
                MAP
                RECORD /XABALLDEF/ ALL
                END MAP
                END UNION
        END STRUCTURE

        RECORD /XXX/ XABFHC, XABPRO

! The chain of XAB blocks, and the NAM block, must have their pointers
! established in the FAB block BEFORE the $OPEN or $CREATE.  When
! these pointers exist, RMS will check to see what type of XAB
! each one is, and supply the requested information in each block.
!
! Initialize FAB block and set up links to NAM block and first XAB.
        FAB.FAB$B_BID = FAB$C_BID
        FAB.FAB$B_BLN = FAB$C_BLN
        FAB.FAB$L_NAM = %LOC(NAM.NAM$B_BID)
        FAB.FAB$L_XAB = %LOC(XABFHC.XAB.XAB$B_COD)

! Initialize NAM block, in this case the ifnormation for the resultant
! file name.
        NAM.NAM$B_BID = NAM$C_BID
        NAM.NAM$B_BLN = NAM$C_BLN
        NAM.NAM$L_RSA = %LOC(RSA)
        NAM.NAM$B_RSS = 255

! Set up the first XAB block, a File Header Characteristics XAB, and the
! link to the next XAB block.
        XABFHC.XAB.XAB$B_COD = XAB$C_FHC
        XABFHC.XAB.XAB$B_BLN = XAB$C_FHCLEN
        XABFHC.XAB.XAB$L_NXT = %LOC(XABPRO.XAB.XAB$B_COD)

! Set up the Protection XAB, and link to the next block.  Note
! the Protection XAB has 2 Fortran structures, XABPRODEF %XABPRODEF1.
! There is no next XAB block, so XAB$L_NXT should be 0.
        XABPRO.XAB.XAB$B_COD = XAB$C_PRO
        XABPRO.XAB.XAB$B_BLN = XAB$C_PROLEN
        XABPRO.XAB.XAB$L_NXT = 0

        OPEN_CONTIG = SYS$OPEN (FAB)
        IF (.NOT. OPEN_CONTIG) RETURN
        OPEN_CONTIG = SYS$CONNECT (RAB)

!The FAB, NAM, & XAB blocks now contain information.  Move the desired
! data into variables in the COMMON list for access by the parent routine.
        BLOCK_SIZE = FAB.FAB$W_BLS
        RECORD_SIZE = FAB.FAB$W_MRS
        RECORD_FORMAT = FAB.FAB$B_RFM
        LRL = XABFHC.FHC.XAB$W_LRL
        GROUP = XABPRO.PRO1.XAB$W_GRP
        MEMBER = XABPRO.PRO1.XAB$W_MBM
```

```
    UIC = XABPRO.PRO1.XAB$L_UIC
    RSL = NAM.NAM$B_RSL                 ! RSA is already set

    RETURN
    END
```

```
!+
! Subroutine GET_QUE_INFO
! Subroutine to get print queue names, descriptions, and libraries.
!
! Programmer:   Tom Worlton, 3-Sep-1992
! (with help from a DECUS tape example.)
!-
        SUBROUTINE GET_QUE_INFO(Q_OUT, DESCRIPTION_OUT
        1                       , LIBRARY_OUT, NQ, MAXQ)
        IMPLICIT INTEGER*4 (A-Z)
        CHARACTER*(*) Q_OUT(MAXQ), DESCRIPTION_OUT(MAXQ), LIBRARY_OUT(MAXQ)
        INTEGER NQ, MAXQ, LUSER
        INCLUDE '($QUIDEF)'
        INCLUDE '($JBCMSGDEF)'
        INTEGER*4 SYS$GETQUIW, STATUS_Q

        ! Define item list structure
        STRUCTURE       /ITMLST/
          UNION
            MAP
              INTEGER*2 BUFLEN, ITMCOD
              INTEGER*4 BUFADR, RETADR
            END MAP
            MAP
              INTEGER*4 END_LIST
            END MAP
          END UNION
        END STRUCTURE

        ! Define I/O status block structure
        STRUCTURE /IOSBLK/
        INTEGER*4 STS, ZEROED
        END STRUCTURE

        ! Declare $GETQUIW item lists and I/O status block
        RECORD /ITMLST/ QUEUE_LIST(10)
        RECORD /IOSBLK/ IOSB

        ! Declare variables used in $GETQUIW item lists
        CHARACTER*31 SEARCH_NAME, DEVICE_NAME, QUEUE_NAME
        CHARACTER*32 GENERIC_TARGET
        CHARACTER*39 LIBRARY_NAME, PROCESSOR
        CHARACTER*255 DESCRIPTION, COMMAND, USER*30
        CHARACTER*6  PTYPE
        CHARACTER ANS
        INTEGER*4    SEARCH_NAME_LEN, SEARCH_FLAGS
        INTEGER*4    DEVICE_NAME_LEN, GENERIC_TARGET_LEN
        INTEGER*4    LIBRARY_NAME_LEN, DESCRIPTION_LEN, QUEUE_NAME_LEN
        INTEGER*4 QUEUE_STATUS, LQS, QUEUE_FLAGS, LQF, LNL
        INTEGER*4 MASK, MTEST

        MAXNAME = LEN(Q_OUT(1))
        MAXD = LEN(DESCRIPTION_OUT(1))
        LUSER = 0
        ! Define Queue name search string
        SEARCH_NAME = '*'
        SEARCH_NAME_LEN = 1

! Initialize item list for the display queue operation
        QUEUE_LIST(1).BUFLEN = SEARCH_NAME_LEN
        QUEUE_LIST(1).ITMCOD = QUI$_SEARCH_NAME         ! Input item code
        QUEUE_LIST(1).BUFADR = %LOC(SEARCH_NAME)
        QUEUE_LIST(1).RETADR = 0        ! RETADR must be 0 for input items
        QUEUE_LIST(2).BUFLEN = 4
        QUEUE_LIST(2).ITMCOD = QUI$_SEARCH_FLAGS        ! Input item code
        QUEUE_LIST(2).BUFADR = %LOC(SEARCH_FLAGS)       ! (the rest are output)
```

```
        QUEUE_LIST(2).RETADR = 0
        QUEUE_LIST(3).BUFLEN = 31
        QUEUE_LIST(3).ITMCOD = QUI$_DEVICE_NAME
        QUEUE_LIST(3).BUFADR = %LOC(DEVICE_NAME)
        QUEUE_LIST(3).RETADR = %LOC(DEVICE_NAME_LEN)
        QUEUE_LIST(4).BUFLEN = 31
        QUEUE_LIST(4).ITMCOD = QUI$_GENERIC_TARGET
        QUEUE_LIST(4).BUFADR = %LOC(GENERIC_TARGET)
        QUEUE_LIST(4).RETADR = %LOC(GENERIC_TARGET_LEN)
        QUEUE_LIST(5).BUFLEN = 31
        QUEUE_LIST(5).ITMCOD = QUI$_LIBRARY_SPECIFICATION
        QUEUE_LIST(5).BUFADR = %LOC(LIBRARY_NAME)
        QUEUE_LIST(5).RETADR = %LOC(LIBRARY_NAME_LEN)
        QUEUE_LIST(6).BUFLEN = 255
        QUEUE_LIST(6).ITMCOD = QUI$_QUEUE_DESCRIPTION
        QUEUE_LIST(6).BUFADR = %LOC(DESCRIPTION)
        QUEUE_LIST(6).RETADR = %LOC(DESCRIPTION_LEN)
        QUEUE_LIST(7).BUFLEN = 31
        QUEUE_LIST(7).ITMCOD = QUI$_QUEUE_NAME
        QUEUE_LIST(7).BUFADR = %LOC(QUEUE_NAME)
        QUEUE_LIST(7).RETADR = %LOC(QUEUE_NAME_LEN)
        QUEUE_LIST(8).BUFLEN = 4
        QUEUE_LIST(8).ITMCOD = QUI$_QUEUE_STATUS
        QUEUE_LIST(8).BUFADR = %LOC(QUEUE_STATUS)
        QUEUE_LIST(8).RETADR = %LOC(LQS)
        QUEUE_LIST(9).BUFLEN = 4
        QUEUE_LIST(9).ITMCOD = QUI$_QUEUE_FLAGS
        QUEUE_LIST(9).BUFADR = %LOC(QUEUE_FLAGS)
        QUEUE_LIST(9).RETADR = %LOC(LQF)
        QUEUE_LIST(10).END_LIST = 0

! Request search for terminal and printer queues.
        SEARCH_FLAGS = QUI$M_SEARCH_WILDCARD
        SEARCH_FLAGS = IBSET(SEARCH_FLAGS,QUI$V_SEARCH_TERMINAL)
        SEARCH_FLAGS = IBSET(SEARCH_FLAGS,QUI$V_SEARCH_PRINTER)

! Dissolve any internal search context block for the process
        STATUS_Q = SYS$GETQUIW (,%VAL(QUI$_CANCEL_OPERATION),,,,,)

! Locate next printer/terminal queue; loop until an error status is returned
        NQ = 0
        DO WHILE ( STATUS_Q)
          LGT = 0
          STATUS_Q = SYS$GETQUIW (,
        2                         %VAL(QUI$_DISPLAY_QUEUE),,
        2                         QUEUE_LIST,
        2                         IOSB,,)
          IF (STATUS_Q) STATUS_Q = IOSB.STS
          if     (generic_target_len .eq. 0
     1     .AND. STATUS_Q .NE. JBC$_NOMOREQUE) then
           LNL = LIBRARY_NAME_LEN
           IF(DESCRIPTION_LEN .LT. 1) THEN
             IF(LUSER .EQ. 0) THEN
               CALL GTUSER(USER,LUSER)
             END IF
!+
! If there is no description and user = SYSTEM prompt for description
!-
             IF(USER(1:LUSER) .EQ. 'SYSTEM') THEN
               WRITE(6,*) 'Que ',queue_name(1:queue_name_len) ,
     1                    ' has no description.'
               WRITE(6,FMT='(''  Please enter a Queue Description:'')')
               READ(5,FMT='(q,a)') description_len, description
               IF(description_len .GT. 0) THEN
                 COMMAND = '$SET QUEUE '//queue_name(1:queue_name_len)
     1                   //''/DESCRIPTION="'
```

```
     2                //description(1:description_len)
     3                //'"'
                    WRITE(6,*) command
                    CALL LIB$SPAWN(command)
                  END IF
                ELSE
                  DESCRIPTION_LEN = 14
                  DESCRIPTION(1:) = 'No Description'
                END IF
              END IF
              IF(NQ .LT. MAXQ) THEN
                NQ = NQ + 1
              ELSE
                WRITE(6,*) 'You have more than ', MAXQ, ' queues.'
                WRITE(6,*) 'Change MAXQ and rebuild MPRINT.'
              END IF
              Q_OUT(NQ) = QUEUE_NAME(1:QUEUE_NAME_LEN)
              DESCRIPTION_OUT(NQ) = DESCRIPTION(1:DESCRIPTION_LEN)
              IF(LNL .GT. 0) THEN
                LIBRARY_OUT(NQ) = LIBRARY_NAME(1:LNL)
              ELSE
                LIBRARY_OUT(NQ) = 'SYSDEVCTL'
              END IF
            end if
          END DO

          RETURN
          END
```

```
      SUBROUTINE DETAB(BUFFER, LBUFF)
!+
! Subroutine DETAB replaces tabs in a buffer with spaces.
! The buffer length may change.
!
! Programmer:   Tom Worlton, 21-Jan-1987
!-
      CHARACTER*(*) BUFFER
      CHARACTER TEMP*132, CTAB*1, BLANKS*8/'        '/
      INTEGER ITAB, NBLANKS, LBUFF

      CTAB = CHAR(9)
      TEMP = BUFFER

      ITAB = INDEX(TEMP,CTAB)
      DO WHILE (ITAB .NE. 0)
        NBLANKS = 8 - MOD(ITAB,8)
        TEMP(ITAB+NBLANKS+1:) = TEMP(ITAB+1:)
        TEMP(ITAB:ITAB+NBLANKS) = BLANKS
        ITAB = INDEX(TEMP,CTAB)
      END DO

      BUFFER = TEMP
      LBUFF = JLEN(BUFFER)

      RETURN
      END
!+
!     SUBROUTINE TO RETURN USERNAME OF CURRENTLY
!     EXECUTING PROCESS.
!
! AUTHOR:       Thomas G. Worlton              CREATION DATE:  30-Sep-1988
!               Argonne National Laboratory
!-

      SUBROUTINE GTUSER(USERNAME,NSIZE)
      IMPLICIT INTEGER*4 (A-Z)
      INCLUDE '($JPIDEF)/LIST'
      INTEGER*2 JPI_LIST(8)/128,JPI$_USERNAME,6*0/
      EQUIVALENCE (JPI_LIST(3),USER_NAME_ADDR)
      EQUIVALENCE (JPI_LIST(5),NAME_SIZE_ADDR)
      CHARACTER USERNAME*(*)
      EXTERNAL SYS$GETJPI

      JPI_LIST(1) = LEN(USERNAME)
      USER_NAME_ADDR = %LOC(USERNAME)
      NAME_SIZE_ADDR = %LOC(NSIZE)

      STATUS = SYS$GETJPI(%VAL(1),,,JPI_LIST,,,)
      IF (.NOT. STATUS) CALL LIB$STOP(%VAL(STATUS))
      STATUS = SYS$WAITFR (%VAL(1))
      IF (.NOT. STATUS) CALL LIB$STOP(%VAL(STATUS))
      DO WHILE(USERNAME(NSIZE:NSIZE) .EQ. ' '
     1  .OR.  USERNAME(NSIZE:NSIZE) .EQ. CHAR(0) )
          NSIZE = NSIZE - 1
      END DO

      RETURN
      END
```

```
      function jlen(string)
!+
! Function routine jlen returns the length of a character string
! without trailing blanks and/or nulls
!-
      character*(*) string
      integer jlen

      jlen = len(string)
      do while (string(jlen:jlen) .eq. ' '       .or.
     1          string(jlen:jlen) .eq. char(0)   .and.
     2          jlen .gt. 0)
        jlen = jlen - 1
      end do

      return
      end
```

```
      Subroutine fparse(file,name,type)
!+
! Subroutine fparse Gets the file name and type from a file specification
!-
      character*(*) file,name,type

      lfdim = len(file)
      lname = len(name)
      ltype = len(type)
      lfile = jlen(file)

      iket = index(file,']')
      icolon = index(file,':')
      if(icolon .gt. iket) then
        in1 = icolon + 1
      else
        in1 = iket + 1
      end if
      idot = index(file(in1:),'.') + in1 - 1
      isemi = index(file(in1:),';') + in1 - 1
      in2 = idot - 1
      it1 = in2 + 2

      if(isemi .gt. in1) then
        it2 = isemi - 1
      else
        it2 = lfile
      end if

      name = file(in1:in2)
      lname = in2-in1+1
      type = file(it1:it2)
      ltype = it2-it1+1

      return
      end
```

```
!
! Include file MPRINT.INC used by MPRINT.FOR
!
! To customize MPRINT for your site, edit this file to make form
! names match the names in use at your site.  If more libraries are
! specified, also increase the NUMLIB parameter.  If a printer cannot
! print Tektronix and/or Postscript files, set value(s) to '-'
!
      parameter maxlib=20
      character*39 libname(maxlib)
      character*31 psform(maxlib), tkform(maxlib)
      1           ,txt80form(maxlib), txt132form(maxlib)

! Default device control library, used where none is specified
      data libname(1)          /'SYSDEVCTL'/
      data psform(1)           /'-'/
      data tkform(1)           /'-'/
      data txt80form(1)        /'DEFAULT'/
      data txt132form(1)       /'DEFAULT'/

! Library for LN03-Plus laser printers
      data libname(2)          /'LN03DEVCTL'/
      data psform(2)           /'-'/
      data tkform(2)           /'TEK'/
      data txt80form(2)        /'LETTER'/
      data txt132form(2)       /'LAND8'/

! Library for Postscript printers on Appletalk
! If you want grey bars on 132 column text printed by Pathworks for
! Macintosh printers, change "LPT
      data libname(3)          /
      data psform(3)           /
      data tkform(3)           /
      data txt80form(3)        /
      data txt132form(3)       /

! Library for Printronix line pri
      data libname(4)          /
      data psform(4)           /
      data tkform(4)           /
      data txt80form(4)        /
      data txt132form(4)       /

! library for old Talaris (QMS) l
      data libname(5)          /
      data psform(5)           /
      data tkform(5)           /
      data txt80form(5)        /
      data txt132form(5)       /'LISTING'/

! Library for LJ250 color printer (not used for text files)
      data libname(6)          /'LJ250DEVCTL'/
      data psform(6)           /'-'/
      data tkform(6)           /'-'/
      data txt80form(6)        /'-'/
      data txt132form(6)       /'-'/

      PARAMETER NUMLIB=6
```

Fig. 2

MPRINT--Choo

SD3

Printer	Descr.
399T4_LW	Bldg.
AST	Bldg. 3
[redacted]	Bldg. [redacted]
DP960	Bldg. 36
LAURA_223	Bldg. 223
LWIIG_223	Bldg. 223
LWP399	Bldg. 399
LZR1260_223	Bldg. 223 1
OSBORN	Bldg. 223 D.
Cancel	

Fig. 3

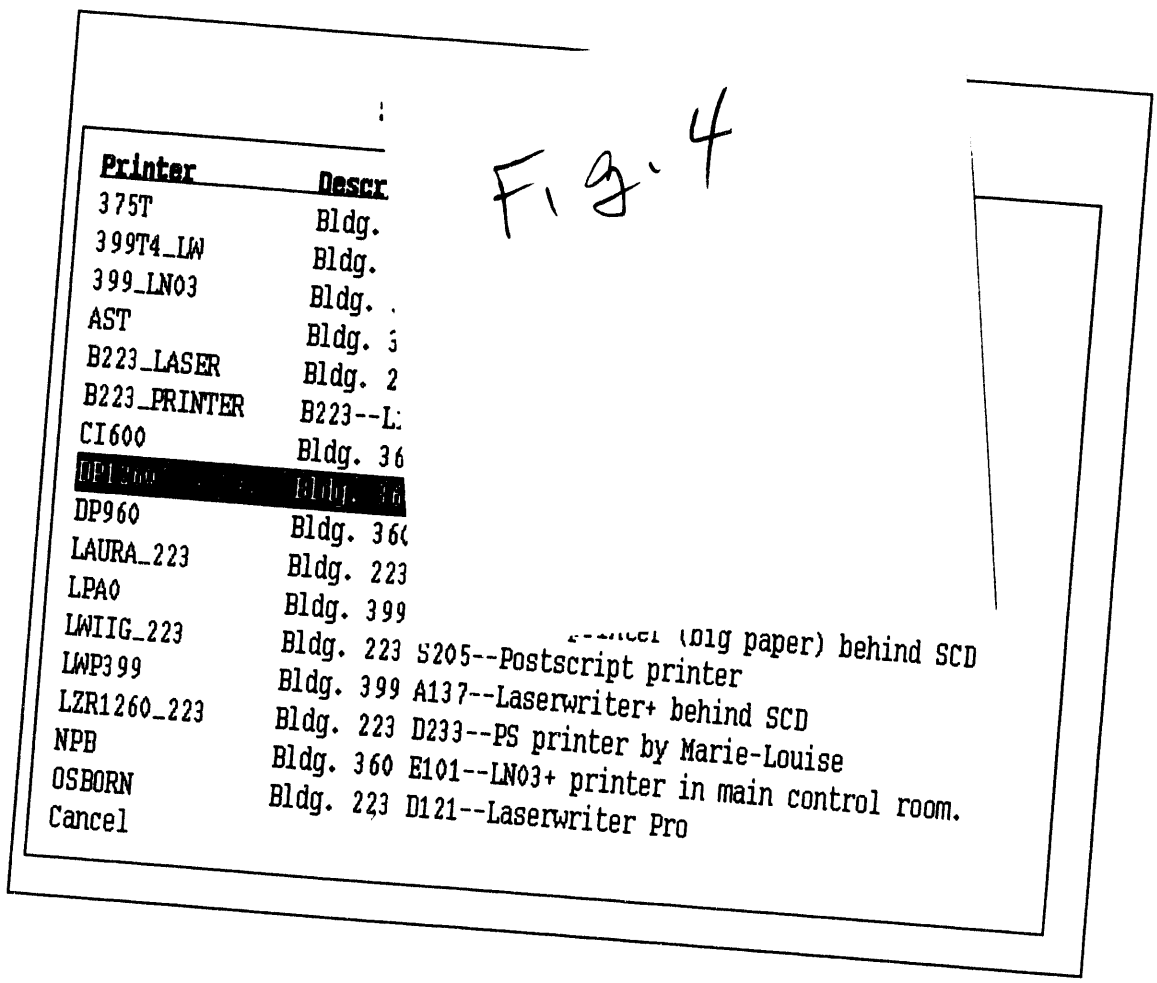

Printer
Descr
375T
Bldg.
399T4_LW
Bldg.
399_LN03
Bldg.
AST
Bldg.
B223_LASER
Bldg. 2
B223_PRINTER
B223--L
CI600
Bldg. 36
DP960
Bldg. 36
LAURA_223
Bldg. 223
LPA0
Bldg. 399
LWIIG_223
LWP399
LZR1260_223
NPB
OSBORN
Cancel
(big paper) behind SCD
Bldg. 223 S205--Postscript printer
Bldg. 399 A137--Laserwriter+ behind SCD
Bldg. 223 D233--PS printer by Marie-Louise
Bldg. 360 E101--LN03+ printer in main control room.
Bldg. 223 D121--Laserwriter Pro

Fig. 4

DATE

FILMED

9/9/94

END