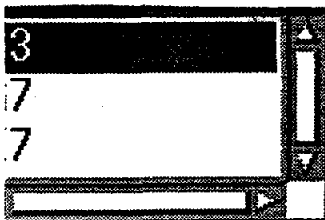


PRTRPT: A General Report Printing Method

Phil Busby, Live Data Systems, Inc. P. O. Box 13144, RTP, NC 27709

This SAS/AF® Frame method prints a report from data displayed in a dynamic extended table. It was written using SAS® 6.09 under Unix. It formats the report for the page size of the printer you choose from a list of available printers on your system.

The following frame display has a PRINT pushbutton and an extended table of real-time data from a manufacturing application. When you click on the PRINT pushbutton, the SCL code for the pushbutton writes the extended table data to a sequential file and calls the PRTRPT method.



EXIT

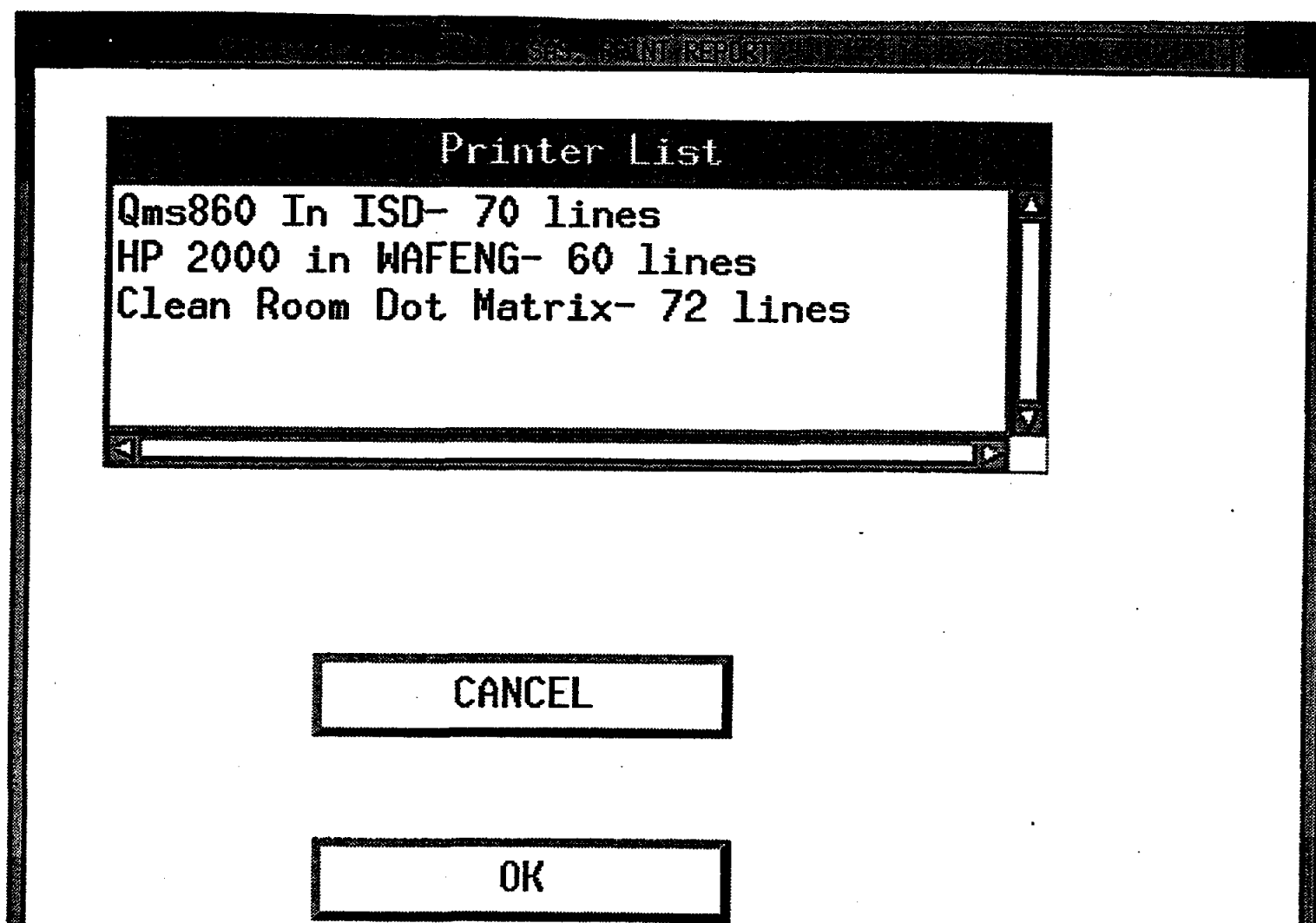
PRINT

CHIP / MASK		PU01	PU02	PU03	PU05	PU06	PU07
C5M41000BV	52E	XXXX	XXXX		XXXX	XXXX	XXXX
MEM41000B	70ASA			D003			
C5M41000BV	52E						XXXX
MEM41000B	74AXA	RRRR					
C5M41000BV	52E						XXXX
MEM41000B	71AXA						
C5M41000BV	52E	XXXX		XXXX	P1Q1	XXXX	
MEM41000B	64ASA				D006		

Screen 1: An extended table of real-time data with a PRINT pushbutton.

PRTRPT reads the Unix system file of available printers and the pagesize of each printer. When you select a printer, PRTRPT writes an output file from the data file passed to it, using the first lines in the input data file as header lines printed at the top of every page. The method can be called with any data file to be printed, not just data from an extended table. PRTRPT supplies page numbers in the upper right corner of every output page printed.

The parameters passed to PRTRPT are the file name,
the number of header lines in the file, and
the linesize of the file.



Screen 2: PRTRPT requestor window of available printers.

Project: SPACE/MMP
Module: prtrpt.scl
Version: 06dec94

Function: Print a file on user-selected printer.
Caller specifies file name and number of hdr lines.
Object reads \$SETC/printtcp and parses each available printer into cmd, desc, and pgsz.
Listbox displays description and gets user choice.
Input file is formatted for lines on specified printer's page. Page breaks can occur only where there are blank lines in the input file.
Output file is written to user's \$HOME directory and system is called to print it with the system command corresponding to the selected printer.

```
length tt      $ 5;
length dd      8;
length z       $ 20;
length strval  $ 9;
length fileref $ 8;
length cmdstr  $ 88;
length ffname  $ 64;
length line    $ 200;
length line1   $ 200;
length line2   $ 200;
length saveline $ 200;
length prtcmd  $ 80;
length prtdesc $ 40;
length pagesz  8;
length linesz  8;
length headsz  8;
length cc      $ 1;
length wherestr $ 80;

length msg      $ 40;

length head1-head20 $ 200;
array head{20} head1-head20;

entry filenam $ headln $ linesize $;
init:
    head1 = ' ';
    selected = 0;
    headsiz = input(headln, best.);
    if dd > 0 then
        put 'headsiz=' headsz;
    linesz = input(linesize, best.);
    if dd > 0 then
        put 'linesz =' linesz;
    rc = 0;
    pageno = 0;
    indent = 3;
    ffname = '$SETC/printtcp';

    prtrlist = makelist();

    rc = fillist('FILE', ffname, prtrlist);

    prtds = 0;
    prtdsn = 'sasuser.prtds';
    prtds = open(prtdsn, 'n');

    rc = newvar(prtds, 'prtcmd', 'c', 80);
    rc = newvar(prtds, 'prtdesc', 'c', 40);
    rc = newvar(prtds, 'pagesz', 'n', 8);

    rc = close(prtds);
    if (rc) then put 'prtds close rc=' rc;
    chipds = 0;

    if dd > 0 then
        put 'open prtds for update';
    /* Open prtds data set for output. */
    prtds = open(prtdsn, 'u');

    call set(prtds);
    nprtrvs = listlen(prtrlist);

    do j = 1 to nprtrvs;
        line = getitemc(prtrlist, j);
        link parselin;

        rc = append(prtds);
        end;
    rc = close(prtds);

    prtds = open(prtdsn, 'i');
    call notify('prtlist', '_repopulate_');
    return;

cancel:
    _status_ = 'H';
    link closum;
    return;
```

```

prtlist:
  if dd > 0 then
    put 'prtlist: ';
  msg = ' ';
  call notify('msgline', '_SET_TEXT_', msg);
  /* get last selected row and text value */
  call notify('prtlist', '_get_last_sel_',
    selrow, issel, line);
  if issel then selected = 1;
  if dd > 0 then
    _msg_ = 'line-' || line;
  if dd > 0 then
    _msg_ = 'issel-' || issel;

  if issel > 0 then do;
    line = getitemc(prtrlist, selrow);
    link parselin;
  end; /* get prtcmd, prtdesc. pagesz */

  subx = index(prtcmd, '%s');
  if subx < 0 then
    do;
      _msg_ = 'ERROR: no %s in prt-spec';
      put _msg_;
      put line;
    end;
  else /* write filename to print.out */
    do;
      link dofile;
      cmdstr = substr(prtcmd,1,subx-1) ||
        ' print.out ' ||
        substr(prtcmd,subx+2);
    end;

  if fidout > 0 then
    do;
      rc = fclose(fidout);

      if rc > 0 then
        do;
          put 'WARNING: output file close rc=' rc;
        end;
    end;

  return;

ok:
  if dd > 0 then
    put 'OK:' cmdstr;
  if selected = 0 then do;
    _msg_ = 'Please select a printer.';
    call notify('msgline', '_SET_TEXT_', _msg_);
  end;
  else do;
    rc = system(cmdstr);
    _msg_ = 'NOTE: system rc=' || put(rc,5.);
    _status_ = 'H';
    link closum;
  end;
  return;

prthead:
  pageno = pageno + 1;
  if dd > 0 then
    put 'prthead page ' pageno;
  substr(head[1], linesz-8, 4) = 'Page';
  substr(head[1], linesz-3, 4) =
    put(pageno, 24.);
  cc = '1';
  do j = 1 to headsz;
    rc = fput(fidout, head[j]);
    rc = fwrite(fidout, cc);
    cc = ' ';
  end;
  linecnt = 0;
  if saveline > ' ' then
    do;
      rc = fput(fidout, saveline);
      rc = fwrite(fidout, cc);
      saveline = ' ';
    end;
  return;

outline:
  if dd > 0 then
    put 'outline: ' line;

  if linecnt + headsz = pagesz-1
    and line > ' ' then
    do;
      saveline = line;
      line = ' ';
    end;
  if linecnt + headsz >= pagesz
    then link prthead;
  rc = fput(fidout, line);
  rc = fwrite(fidout);

  linecnt = linecnt+1;
  return;

exitbtn:
  _msg_ = 'EXIT =' || exitbtn;
  link closum;
  _status_ = 'H';
  return;

term:
  if dd > 0 then
    put 'term prtrpt.';
  link closum;
  if _status_ = 'C' then return;
  return;

```

dofile:

```
/* given filename, read it, storing first headcnt lines in header array. */
/* Stuff 'Page nnn' in first header line. Read the rest of the data */
/* lines, writing pagesz - headsz data lines to print.rpt. */
```

```
if dd > 0 then
    put 'dofile for ' filename;

fileref = 'printin ';
rc = filename(fileref, '');
rc = filename(fileref, filename);
fidin = fopen(fileref, 'i');
if fidin < 1 then
    do;
        put 'ERROR: cannot open input print file ' filename;
        _msg_ = sysmsg();
        put _msg_;
        return;
    end;
else if dd > 0 then
    put 'input file open fidin=' fidin;

fileref = 'printout';
rc = filename(fileref, '');
rc = filename(fileref, 'print.out');
fidout= fopen(fileref, 'o');
if fidout < 1 then
    do;
        put 'ERROR: cannot open output prt file print.out';
        _msg_ = sysmsg();
        put _msg_;
        return;
    end;
else if dd > 0 then
    put 'output file open fidout=' fidout;
```

```
closum:
    if prtds > 0 then
        do;
            rc = close(prtds);
            rc = delete(prtdsn);
        end;
    return;
```

inlines = 0;

```
/* priming read */
readrc = fread(fidin);
if dd > 0 then
    put 'first readrc =' readrc;
do while(readrc ne -1);
    rc = fget(fidin, line, 200);
    inlines = inlines + 1;
    if inlines <= headsz then
        do;
            head[inlines] = line;
            if dd > 0 then
                do;
                    put 'head ' inlines;
                    put line;
                end;
            end;
        if inlines = headsz then
            do;

                if dd > 0 then
                    put 'First link to head';
                    link prthead;
                end;
            if inlines > headsz then
                do;
                    /* data line to be written now. */
                    link outline;
                end;
            readrc = fread(fidin);
        end;
    rc = fclose(fidin);
    return;
```

parselin:

```
if dd > 0 then
    put line;
    ll = index(line, ':');
    if ll < 0 then
        do;
            put 'PRTRPT:: oops :';
            put line;
            ll = 1;
        end;
    prtcmd = substr(line,1,ll-1);
    prtdesc = substr(line, ll+1);
    ll = index(line, 'lines');
    if ll < 0 then
        do;
            put 'PRTRPT: oops lines';
            ll = 1;
        end;
    else
        do;
            pagesz = input(substr(line,ll-3,3), best.);
            if dd > 0 then
                put 'pagesz=' pagesz;
            if pagesz < 1 or pagesz > 999 then
                do;
                    put 'PRTRPT:invalid pagesz ' pagesz;
                    pagesz = 50;
                end;
            end;
        end;
    return;
```