Paper 215-26

A SAS® V8 AIX Web Application Utilizing ODS, FTP Files from MVS, and SAS® Proc Greplay in Batch Mode

Jon A. Patrick, IBM Microelectronics Division, Essex Junction, VT 05452

ABSTRACT

IBM's semiconductor fabricator in Burlington, VT creates mega data each day that is stored in numerous db2 tables on our MVS host system. The data is in the form of Etest, process (date, time, tool), probe, final test, etc. Our manufacturing engineering characterization team makes extensive use of our Intranet. This approach allows process engineers, management, and product engineers to view critical information at the click of a mouse. Most jobs are scheduled by crontab in the early hours of the morning to run data retrievals, SAS data analysis, and data transmissions to our OEM customers. My usual day begins by checking our project's web page to determine what has been tested in the past 24-hours for the product set for which I am responsible.

I will share one of the daily charts that covers ODS (Output Delivery System), FTP (file transfer protocol), SAS macro variables, and proc Greplay run in batch mode. The program generates four plots on one page that is ready for the web via ODS. The plots are trend charts of wafer probe defects at various points in the process line. Each process step is recorded in db2 with the date, time, and tool number for the lot that was processed.

INTRODUCTION

This 4-up program is broken down into three distinct parts. Part one sets up ODS commands and retrieves the data from the MVS host via FTP. Proc cport (at the MVS side) and cimport inside this job are used. Part two merges the probe data with the process date, time, tool data and generates the plots. Part three uses proc Greplay commands in batch to generate the four plots per page. A brief description of the code will be given after each part. As a program note, all SAS programs are debugged using display manager. Once complete, they go into a batch mode file that is called by crontab for execution. The SAS program manager is a useful tool for editing AIX files if one is not familiar with the VE editor in AIX. I would like to see SAS Institute add the "copy" "on", or cc cc oo oo, where blocks can be copied over other lines. That would be a useful addition to the program editor.

```
ods listing close;
goptions   reset=all      border      nocells
nocharacters   gunit=pct   cback=white   hby=2
nodisplay device=gif ;
```

This program creates 4 graphs/1 page ;
Remove old gif files from gif directory;

```
x"rm/afs/btv/u/jpatrick/scorpion/gif40/*.gif"
;
x"rm/afs/btv/u/jpatrick/scorpion/gif40/*.htm"
;
x"rm/afs/btv/u/jpatrick/sasgraphv7/fourup.*";
run;

libname  sasdb  '~jpatrick/sastempv7/cmos8s';
* date,time,tool data;
libname  sasdb1  '~jpatrick/sasdatav7/';  *wft
or probe data;
libname  grafcat  '~jpatrick/sasgraphv7/';  *
graphs;
libname tempcat '~jpatrick/scorpion/tempcat';
*template;

options  nocenter;  *  macrogen  symbolgen
nomprint mlogic ;

Filename
odsout'/afs/btv/u/jpatrick/scorpion/gif40/';

ods html
      contents='gcmos-contents.htm'
        frame='gcmos-frame.htm'
         body='gcmos-body.htm'
         path=odsout
         (url=none)
      newfile=page;
run;


filename input   ftp
'BARNEY.FWAFDATA'               user='D123456'
pass="&tsopw" rcmd='binary' host='btvmanb';
run;
proc cimport library=work.map file=input;
run;


 data sasdb1.barneywft;
    set work.map;
run;
```

*********************End Part 1************************

Part One covers the ODS and FTP commands. When starting with ODS, it is important to know that this feature starts in SAS v7. It is easier when building web pages based on SAS output, because it automatically builds the HTML code, which is then ready for web use without any inconvenience.

The first line of the program identifies prep for ODS; "ods listing close" closes all existing ODS commands that may be present when running SAS in interactive mode.

The Goptions "reset=all" is a recommended option. The "cback=white" is useful when directing the output to the web, as anyone who has seen the black back round can attest.

"Device=gif" states that you want your plots in gif format. You also need to provide a filename called "odsout" that will be the target location for all your output, be it proc print, proc corr, or gplot. There is no magic in making the directory; in AIX a simple "mkdir" {name} is all that is needed. The files are automatically generated by SAS during execution.

The next lines are "x commands" that allow SAS to use AIX commands within the program. This is great when all that needs to be done is remove (RM) any existing files within the directory. This is very necessary when using ODS because, if you run 20 debug jobs, you'll have 20 versions of output within you directory. The proc Greplay noofs; delete _all_; has the same effect.

The ODS HTML commands are taken directly from the ODS handbook. I removed the page= option because it is unnecessary for my application. The option “url= none” is there because of the interaction of our department server’s web page with my directory which contains all my gif files. Since the department’s server (called crystal) points to the full path, having anything in the url option causes an invalid url address.

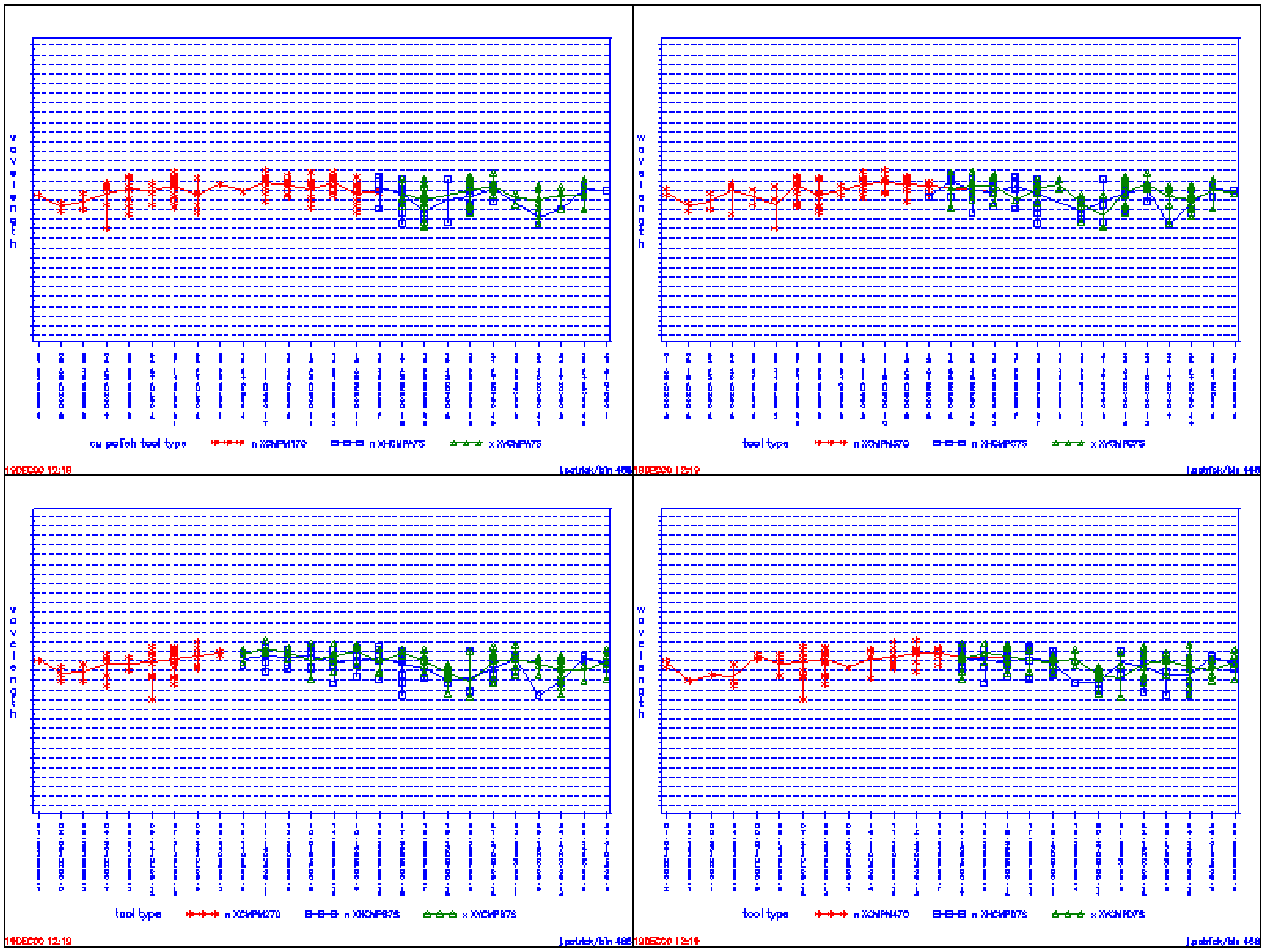
The FTP code transfers data between MVS (host) and AIX with little effort. The execution of jobs in the crontab stream allows a subset of data to be retrieved and stored using SAS cport. This is done on the host. The next job in the stream retrieves the data using cimport, as shown in this example. The MVS/TSO password is necessary; I usually put mine in my autoexec.sas file so it can be executed and put into the macro variable &tsopw upon execution of the batch job or while I start SAS v8 in the interactive mode. This is also a very easy way to transfer data among our customers. A few of our OEM customers have installed SAS because of the data analysis capabilities and the ease in transferring inline data using cport and cimport. The files are also easily pgp (pretty good privacy) protected prior to rftp transmission.

**********************Start Part 2************************

This macro calls plot programs;

```
    %macro screen;
    %do il=1 %to &n;
   * by lot;
      %include  '~jpatrick/sas/sascode/plot4up1';
   * by week;
    %include    '~jpatrick/sas/sascode/plot4upw';
   %end;
    %mend screen;

   *get the etest data;
   data xall;
    set sasdb1.barneywft;
   if defects=. then delete;
     keep lot defects wafer;
        run;

    proc sort data=xall; by lot;

     proc means noprint data=xall; by lot;
       var defects;
     output out=xall mean=defects ;

   *get the inline date/time/tool data;
     data dates ;
       set sasdb.datetool;
   rename otool_dt=outdate
          grp_oper=soper
          gatelvl =level
          otool_tm=outtime;
     operdesc=translate(operdesc,' ',"'");
     operdesc=translate(operdesc,' ',";");
     operdesc=translate(operdesc,' ',"/");
     operdesc=translate(operdesc,' ',":");
   title=substr(operdesc,4,16);
   *output the 4 opers necessary for the plot;
   * lm photo;
    if oper='XPH8S0LM' then output;
   * mc photo;
   if oper='EPHBPSB8' then output;
   * m1 photo;
   if oper='XPHMEM1H' then output;
   * pc photo;
   if oper='EPHPOL8S' then output;
 run;

 proc sort data=dates nodupkey out=dates; by
lot oper;

 proc sql;
 create table xall as
  select *
 from xall as a, dates as b
 where  a.lot=b.lot;
 run;

 proc sort data=xall; by outdate outtime;

* get list of unique operations;

 proc sort data=xall nodupkey out=datelist;
 by oper;

 proc sort data=datelist; by outdate;

 data datelist; set datelist; by outdate;
   order= _n_;
 desc=level|| ' ' ||left(put(title,$16.));
 keep desc oper order title level soper gate;
 run;

 proc sort data=datelist; by gate;

* create all the macro variables;
 data _null_;
  set datelist end=eof;

Call
Symput('title'||trim(left(_n_)),(trim(left(pu
t(title,$25.)))));

Call
symput('level'||trim(left(_n_)),(trim(left(pu
t(level,$2.)))));

Call
symput('gate'||trim(left(_n_)),(trim(left(put
(gate,4.)))));

Call
symput('operz'||trim(left(_n_)),(trim(left(op
er))));

Call
symput('soper'||trim(left(_n_)),(trim(left(so
per))));

Call
symput('n',(trim(left(_n_))));
run;

%put title1=&title1, lasttitle=&&title&n;
%put title1=&operz1, lasttitle=&&operz&n;

*run the macro, n , times ;
%screen;

run;
```

**************************End Part 2**********************



Rick Guest, an IBM employee for the Dallas SUGI, wrote the first part of Part Two. This macro is great if you need to generate multiple plots. Its basic operation is to call a plot routine for each macro variable generated using the call symput commands. The actual plot routines are not included in this paper because they are typical. However, note that the plot routines require a "gout=" option when executing the gplot procedure. This is necessary so the four plots can be placed on one page via the greplay procedure, as discussed in Part Three. The heart of Part Two merges the probe data with the process (date, time, tool) data. The process data is organized in a vertical format, i.e., there is only one variable called "outdate" that changes for each process operation. This creates a situation where many observations of dates, all with the same lot number, are merging with a file that has a single observation per lot that contains the photo defects. The best way to merge these files is with proc sql. There are several translate commands with the date, time tool data step. They are necessary because the macro variables would cause execution to stop when a semicolon is embedded in the process title.
The last section of Part Two deals with generating a unique list of each process operation. This is done using proc sort with the "nodupkey" option. The list is fed into a _null_ datastep that generates a macro variable for each process gate, process level, and operation. This is the classic way to create numerous macro variables. By concatenating the system _n_ variable to the end of the macro variable name, a unique macro variable is generated for each process operation to plot. The system _n_ variable will max out with the total number of process operations; this is also the total number of times the macro "screen" actually runs.

**********************Start Part 3*************************

Create template;

```
   goptions  reset=all    border    vsize=6.5
   hsize=9.5 hpos=120 vpos=90
   nocells   nocharacters   cback=white   hby=2
   device=gif;

    goptions display;

     proc greplay nofs
      igout=grafcat.fourup
      tc=tempcat;

      tdef newfour des='fred 4 up chart, by lot'

      1/llx=0 lly=50 ulx=0 uly=100
        lrx=50 lry=50 urx=50 ury=100 color=black

      2/llx=0 lly=0 ulx=0 uly=50
        lrx=50 lry=0 urx=50 ury=50  color=black

      3/llx=50 lly=50 ulx=50 uly=100
        lrx=100    lry=50    urx=100    ury=100
   color=black

      4/llx=50 lly=0 ulx=50 uly=50
     lrx=100     lry=0     urx=100     ury=50
color=black;

  template newfour;

   treplay
     1:gplot
     2:gplot2
     3:gplot4
     4:gplot6 ;

  tdef  newfour  des='fred  4  up  chart,  by
week';

    treplay
     1:gplot1
     2:gplot3
     3:gplot5
     4:gplot7 ;

run;
quit;
ods html close;
ods listing;
run;
quit;
```

Part Three covers the proc Greplay code that generates a template and replays the graphs so 4-plots fit into 1 page. I duplicated the goptions statement except for the display option that needs to be "on" during this part of the execution. Proc Greplay is run to define where your graphs are stored, and to define the template catalog. It is necessary to use the "nofs" option at this point to be in "line mode". The "tdef" statement is used to define a template; in this example, the name of the template is "newfour". Most of this code can be found on the SAS Institute web page; I only had to modify the goptions for v8. For web applications, the use of the "des=" option is important because the description appears in the frame portion on the web. The macro calls a plot routine, by lot, and a plot routine, by week, for each process operation; therefore, the treplay calls gplot,2,4,6 for placement on one template and gplot1,3,5,7 for placement on the next template. The program ends with an html close so no subsequent program output is stored on your web page.

The html code for placing these graphs on your web page would look something like this.

```
   <a  href="gif40/gcmos-frame.htm">  <h3>  4up
 process chart </h3></a><br>
```

CONCLUSION:

SAS v8 makes it easy to store SAS output on the web via ODS. The fact that everyone is trying to reduce cpu costs has moved SAS applications down to different platforms. This application uses FTP to transfer the data between our MVS host and AIX. My hope is that the example is not too trivial for most users, and that it helps the productivity of others.