PROJECT TRACKING SYSTEM USING SAS/AF®SOFTWARE

David M. Hartman, SmithKline Beecham Pharmaceuticals

Introduction

Monitoring the status of data processing for nearly 100 clinical protocols for each of a dozen drug projects is a monumental task. In an effort to make this job easier, a menu driven project tracking system (PTS) using SAS/AF®software has been developed.

PTS is a user-friendly interactive system which provides clinical project team members with an up-to-date status of critical stages in the processing of clinical research data. The following are some of the functional capabilities of PTS:

1. All 26 windows use PROGRAM entries which provide flexibility thereby giving the system developer virtually unlimited creative possibilities when designing the screens.

2. TAB and RETURN keys are all that are needed to move around within PTS. The exception is at the lower level data entry and retrieval screens which require the use of the PF3 function key (found on Digital VT terminals) to return to the previous window.

3. Two permament SAS datasets can be created at the drug project level. One dataset stores dates and the other stores comments.

4. New data can be entered or existing data can be updated.

5. Data can be retrieved for either viewing on the screen or output to a printer.

6. Pull down screens give the user the option of either typing appropriate information or selecting information (e.g. drug identification number) from a list.

Environmental Description

Developed in a VAX/VMS environment, PTS is built in an upper level directory which contains all of the 'generic' source code. The stored SAS®datasets created by this system are located in drug specific data directories. With the aid of logicals, a user can access the system from anywhere within the secured data processing environment. With the use of the AUTOEXEC command located in a DCL procedure, all that is needed to invoke PTS is to type PROJECT at the '$' prompt.

Main Menu

The first window to appear in PTS is called main.program which has the following choices: (1) Select Drug, (2) Create Dataset, (3) Enter/Update, (4) View Data, (5) Print Data, and (6) Exit. This window makes use of the BLOCK function providing a good visual effect by displaying '3-dimensional' boxes as the user begins his/her journey through PTS (Appendix 1). Within the BLOCK function is a series of CALL DISPLAY functions which makes the selected window 'active'. This combination of BLOCK and CALL DISPLAY makes the PROGRAM entry function just like the MENU entry available in SAS/AF (Fig. 1).

```
init:
 choice=1;
return;

main:

 LOOP:

  do while(choice ne 0);

   choice=block('Main Menu',
   'SB Project Tracking System',6,
   'Select Drug','Create Dataset',
   'Enter/Update','','View Data',
   'Print Data','','','Exit','','');

 select(choice);
  when (1) call display ('adl.program');
  when (2) call display ('new_ssd.program');
  when (3) call display ('ent_upd.program');
  when (5) call display ('view.program');
  when (6) call display ('print.program');
  when (10) call display ('ext.program');
  end;

 end;

call endblock();

return;

term:
return;
```

Figure 1. Source Code for Main Menu.

Select Drug

In the 'init' section of main.program, choice=1 (Fig. 1). This causes the cursor to automatically point to the 'Select Drug' option. The 'Select Drug' option is

mandatory because it performs two critical functions. One function is the creation of a macro variable called DRUGID which contains the drug identification. DRUGID is created by using the statement

```
call symput ('drugid',drug);
```

This macro variable is called in the title of every window within PTS. The second function is to point you to the drug specific directory where the data resides. This is performed as follows:

```
rc=system('set default dln$disk99:[drugid.data]');
rc=libname('data','[]');
```

The SYSTEM command acts much the same way as the X command in Base SAS® software in that SYSTEM invokes the DCL command 'set default'. Libname then assigns a library reference for a SAS data library.

Create Dataset

If no datasets exist for your selected drug, then your next selection would be 'Create Dataset' (Appendix 2). It is in this window that you are given a choice of setting up two datasets. One will contain mostly date information and the other dataset will contain comment information. With the use of 'choice group: view', and 'initial' in ATTR (Appendix 3), and the use of CALL DISPLAY, this window (like main.program) works much the same way as MENU entries while providing the ability to be more creative. All that is required by the users is to tab to the desired option and hit RETURN.

The creation of stored SAS datasets for protocol (App. 4) and comment information involved building the datasets without entering any data. This was basically a two step process. The following statements open a dataset and add a new variable:

```
dsid=open(new_dsn,'n');
rc=newvar(dsid,'prot','c',5,'prot');
```

In this example, 'prot' was added to the dataset as a 5 character variable. At this point, no data have been entered.

Enter/Update

Once you have created your datasets, your next selection would be 'Enter/Update'. From this menu you can select whether you want to enter data into the dataset for the first time (dataset contains 0 observations) or update the dataset. The ATTR screen and source code which drive this window are essentially the same as 'Create Dataset'.

Enter Data

The 'Enter New Data' menu is set up much the same way as the other selection windows and allows the user to select entry of general protocol information or comments. There is also very little difference in the way each data entry window is set up, including the source code (Fig. 2, Appendix 5).

```
init:
 dsid=open('data.tracdate','u');
 drug=symget('drugid');
return;

main:
 call set(dsid);
 call setcr('new!','return','modify');
 if parm ne ' '
 and protg ne ' '
 and parmg ne ' ' then do;
   rc=append(dsid);
   parm=' ';
   protg=' ';
   parmg=' ';
 end;
return;

term:
 rc=sort(dsid,prot parm');
 rc=close(dsid);
return;
```

Figure 2. Source code for entering new protocol data.

The dsid=open('data.tracdate','u') command allows an existing SAS dataset to be opened for update. Two other functions, SET and SETCR were used in the window. The SET function links the SAS dataset variables to SCL variables of the same name and data type while SETCR provides a means of controlling the movement of the cursor when data is entered. Once all of the fields on the screen are entered, the APPEND function is invoked which takes the data from the Data Set Data Vector and puts it into the stored SAS dataset. When the user exits the window, the dataset is sorted and closed.

Update Data

Once the information is entered then your next choice would be 'Update'. It is here that the bulk of your data in the form of dates or comments are entered for the first time or updated. In the 'Update Data' window, the user has the choice of selecting Protocol/Parameter, Mapping, Source Code, or Comments. The windows for these choices are very similar to each other in the way they are 'set up'.

Some new tools used in these windows are FETCH, PUTVARC, VARNUM, and UPDATE (Fig 3.). The FETCH function was used in a loop which allowed

movement down rows of a dataset until the row of data to be updated was reached. The VARNUM function provided the number of the desired SAS dataset variable. This number was then used in the PUTVARC function which wrote the character value from the screen to the Data Set Data Vector for a SAS dataset. All of this simply takes a value entered by the user and writes it to a permanent stored SAS dataset. The UPDATE function takes the data from the Data Set Data Vector and overwrites data already existing in the stored SAS dataset.

```
init:
 dsid=open('data.tracdate','u');
 vnum=0;
 prot=' ';
 parm=' ';
 drug=symget('drugid');
return;

main:
 call set(dsid);
 call setcr('newl','return','modify');
 if parmv ne ' '
    and (basev ne ' ' or blindv ne ' ')
 then do;
  do while(fetch(dsid) ne -1);
   if prov eq prot
   and parmv eq parm then leave;
  end;

  vnum=varnum(dsid,'base');
  call putvarc(dsid,vnum,basev);

  vnum=varnum(dsid,'blind');
  call putvarc(dsid,vnum,blindv);

  vnum=varnum(dsid,'froz');
  call putvarc(dsid,vnum,frozv);

  rc=update(dsid);

  parmv=' ';
  basev=' ';
  blindv=' ';
  frozv=' ';
 end;
return;

term:
 rc=close(dsid);
return;
```

Figure 3. Source code for updating Protocol Information.

View Data

Once data has been stored, you can select 'View Data' which allows you to view the data on the screen. This window is another menu type PROGRAM entry screen and is set up much the same way as the Enter/Update menu. The user's choice of selections are (1) Protocol, (2) Mapping, (3) Source Code, (4) Comments, and (5) All of the Above. One of the SAS/AF functions used in these selections is REWIND (Fig. 4). The REWIND function provides the means by which the pointer is moved back to the top of the dataset. This gives the user the ability to select data from multiple protocols and/or parameters in the same session.

```
init:
 dsid=open('data.tracdate','i');
 prot=";
 prov=";
 drug=symget('drugid');
return;

main:
 call set(dsid);
 do while(fetch(dsid) ne -1);
   if prov=prot then leave;
 end;
 rc=rewind(dsid);
return;

term:
 rc=close(dsid);
return;
```

Figure 4. Source code for viewing data.

The OPEN function has been used in previous windows except instead of using 'u' for update, 'i' is used for input. Unlike the data entry screens, the FETCH function used here provides the means by which data from the stored SAS dataset can be viewed on the screen. Just as with the other windows that access the stored SAS dataset, this dataset is closed when the user exits the screen. In the 'View All' window, the 'Extended Table' attribute was used (Fig. 5, Appendix 6). The SETROW function was used to generate seven rows of data while the GETROW function was used to read the data from the stored SAS dataset and put it into the seven rows.

```
init:
 dsid=open('data.tracdate','i');
 drug=symget('drugid');
 prot='    ';
 prov='    ';
return;

main:
 call set(dsid);
 call setrow(7);
 rc=rewind(dsid);
return;

term:
 rc=close(dsid);
return;

getrow:
 do while(fetch(dsid) ne -1);
   if prov eq prot then leave;
 end;

 parm=getvarc(dsid,varnum(dsid,'parm'));
 base=getvarc(dsid,varnum(dsid,'base'));
 blind=getvarc(dsid,varnum(dsid,blind'));
 mstrt=getvarc(dsid,varnum(dsid,mstrt'));
 mdone=getvarc(dsid,varnum(dsid,mdone'));
 mod=getvarc(dsid,varnum(dsid,'mod'));
```

```
val=getvarc(dsid,varnum(dsid,'val'));
return;

putrow;
return;
```

Figure 5. Source code for viewing all data via the Extended Table.

Print Data

Another option to viewing the data is to obtain a printout. The selection 'Print Data' provides that option. The set up for this window is very much the same as 'View Data' with the exception of the selections. In this window, the user can select from the following: (1) Protocol, (2) Mapping/Source Code, (3) Comments, and (4) All of the Above. All of these selections are similar to other selection windows in terms of their screen design but their source code is different than anywhere else within PTS. Most of the code is written using Base SAS software which is embedded between 'submit continue' and 'endsubmit' (Fig. 6). The 'control always enter asis' command is used to execute the code within the submit function only when a specific protocol ID has been selected. The SCL function WOUTPUT is used to output the data to a printer. However, prior to output to printer, the data first appears on the output screen in Display Manager. Returning to the Print Data screen requires hitting the PF3 function key (found on Digital VT terminals).

```
init:
  prot  = ' ';
  parm  = ' ';
  base  = ' ';
  blind = ' ';
  froz  = ' ';
  drug=symget('drugid');
return;

main:
  if print='Y' then do;
    control always enter asis;
    submit continue;
      data date; set data.tracdate;
      proc sort data=date;
        by prot parm;

      data final; set date;
        by prot parm;
          dummy=1;
          if not first.prot
          or (base=' ' and blind=' ')
          then delete;
      run;
      proc sort data=final;
        by dummy prot;
      run;

      data _null_; set final;
        by dummy prot;
          length prot $6;
        retain newpage
            pagenum
            footnte1 0;
      file print header=head
            notitles
            pagesize=60
            linesleft=l;

    if first.dummy or newpage=1
    then footnte1=0;

    proto=prot;
    if protg='Y' then do;
      proto=prot || '*';
      footnte1=1;
    end;

    if (not first.dummy)
    and first.prot
    and newpage=0 then put @1 '';

    if first.prot then do;
      newpage=0;
      put @10 proto @;
    end;
    else
    if newpage =1
    and not first.prot then do;
      put @10 prot '(Continues)' @;
      newpage=0;
    end;

    put @47 base @58 blind;

    if last.dummy or l < 12 then do;
      pagenum=pagenum+1;
      line=repeat('-',56);
      put @10 line;

      if not last.prot
      then put @10 '(CONTINUED ON NEXT PAGE)';

      if footnte=1 then put @10
        ' * - Generic Protocol';

      n=l;
      if n>12 then do until (n=12);
        put @1 ' ';
        n=n-1;
      end;
      put @40 '- ' pagenum '-';

      if not last.dummy then do;
        newpage=1;
        put _page_;
      end;
    end;
return;

head:

proj_id=trim(left(&drugid));
f_title='Project Tracking for ' || proj_id;
pos1=(80-length(f_title))/2;
date=today();
current=put(date,mmddyy8.);
pos2=(70-length(current))/2;

line1=repeat('-',56);
line2=repeat('-',19);
```

```
  put #10 @pos1 f_title /
       @pos2 'Revised: ' current ////
       @10 line1 /
       @53 'Protocol' /
       @47 'Baseline'          @58 'Unblinded' /
       @47 line2;
  return;
  run;
  endsubmit;
  rc=woutput('print','report.form','','');
end;
return;

term:
return;
```

Figure 6. Source code for printing protocol information.

Exit

The final option available in PTS is 'Exit'. When this screen is invoked, the command

```
rc=system('set default libr_dir');
```

sends the user to the directory of drug specific SAS programs which the user selected prior to entering PTS.

Pull Down Screens

Those windows dealing with either selecting drug ID, entering new data, updating, viewing, or printing data request that a drug ID, protocol, or parameter (i.e. CLNEVENT, DEMOG, etc.) be entered by the user. As an aid to entering this information the user can type a '?' which pulls down a list of options to choose from. The selection window for each entry was created by editing a .LIST file. DRUG.LIST was created by using the following command at the command line in the BUILD procedure window:

```
edit drug.list
```

This file is stored in the catalog directory and is used primarily in the 'Select Drug' screen.

Conclusion

SAS/AF is an extremely powerful tool which gives application programmers the ability to develop simple, 'user friendly' systems to perform a variety of different tasks. The application described in this paper covers some of the more likely uses of SAS/AF, mainly data entry and retrieval. Hopefully, this paper gave you a sense of how PTS works from both an end-user and application programmer point of view. When developing a system using SAS/AF, the programmer is limited only by experience and his/her imagination.

References

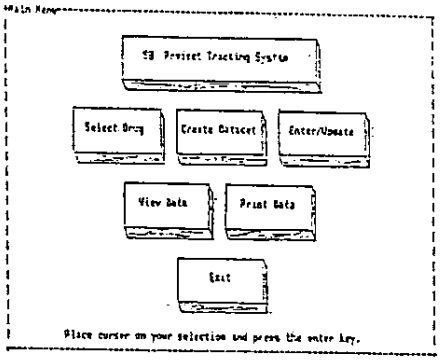
SAS®Language: Reference, Version 6, First Edition

SAS/AF®Software: Usage and Reference, Version 6, First Edition

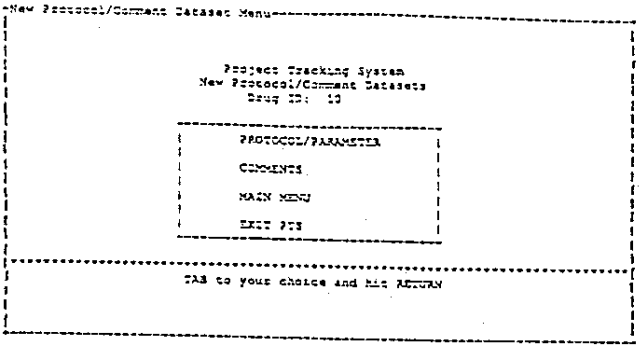
SAS®Screen Control Language: Reference, Version 6, First Edition

For further information contact:

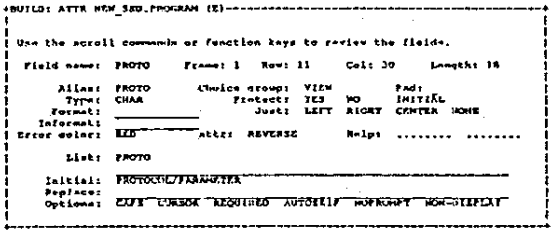
David M. Hartman
SmithKline Beecham Pharmaceuticals
Mail Code #FF0605
P.O. Box 1510
King of Prussia, PA 19406-0939



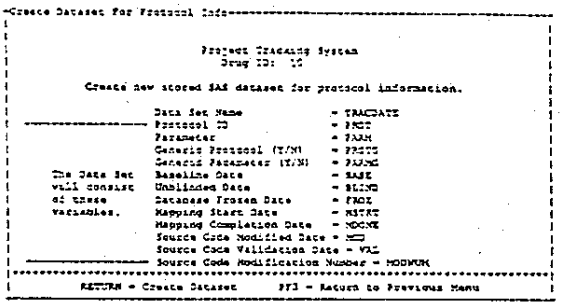
Appendix 1. Main Menu



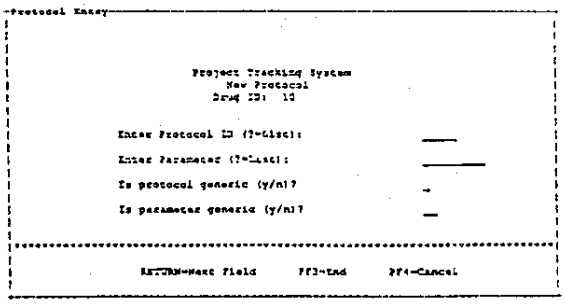
Appendix 2. Create Dataset Menu



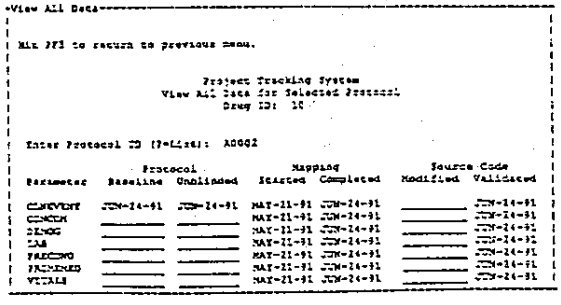
Appendix 3. ATTR screen, creating SAS dataset



Appendix 4. Screen creating protocol SAS dataset



Appendix 5. Screen for new protocol data



Appendix 6. Screen for viewing all data