

"Building a Simple Time Tracking System Using Screen Control Language"

**O.V. Hanger
Nielsen Media Research**

INTRODUCTION

With Version 6 of the SAS® System, Screen Control Language (SCL) was made available for use with SAS/AF® and SAS/FSP®. This new facility provides the developer tools for creating a wide variety of powerful interactive applications. This purpose of this paper is to demonstrate the basics of building a Personal Time Tracking System using SAS/AF software and SCL. The primary function of this application is to provide the user with a means for recording, categorizing and summarizing time spent performing tasks (such as work activities). Included in this application description is the summarization of the primary SAS System files that make up the application, invocation and initialization, the use of PMENUs, custom commands, extended tables, validating user input and the generation of summary reports via SQL SUBMIT blocks. The necessary SCL code for performing these various functions is also given in this document. The Personal Time Tracking System was developed with Version 6.06 of the SAS System of Software and, therefore, does not demonstrate any new features of subsequent Version 6 releases. It provides a useful, simple and interesting way to explore the various new features of SAS/AF, SCL and the SAS System of software.

PERSONAL TIME TRACKING SYSTEM

Computers and related software are great tools for storing and manipulating data. In this many times hurried late 20th century information age that we live in, the making of 'lists of things' is evolving to new heights through the application of computer technology. Just as it is useful to make lists of 'things to do', it can also be useful to make lists of 'things done'. For work related accountability, it is basically a requirement to periodically be able to review and report on those tasks, events and/or activities that have been accomplished. The Personal Time Tracking System discussed in this paper is the foundation for creating such a tool. It is

intended for someone who wants to record, save, change and review daily accomplishments via a daily hours log.

Each logged activity includes the date, hours taken, description of activity and activity type code (which can be created, changed and deleted by the user). It is assumed that there will be no more than 15 entries for any specific date and that the user will want to see all entries for the current date when the system is invoked. In addition to this 'daily view', the user may sometimes want to view all log entries or those for a specified period. In the case of a view for a specified period, the default start and end dates are for the current work week (Saturday-Friday). When the user is uncertain what activity type code should be used for a particular entry, they will be able to enter a '?' in the code field and be provided a list of currently defined codes from which a selection can be made. Any entry where the value for number of hours is missing is deleted. When summarizing 'things done' for a specific period, the current work week is the default period and the user may chose between a summary by date or a summary by code. For ease of use, the application is driven via pull down menus and fill in the blanks. For portability, no operating system commands or features unavailable to a user with only base SAS (Version 6.06 or above) are used. Operating system specifics are documented separately.

As a SAS/AF application, the Personal Time Tracking System (PTTS) can be invoked via PROC DISPLAY or the AF display manager command. In each case the proper syntax is:

```
PROC DISPLAY CATALOG=  
MYTIME.MAINCAT.MYTINIT.PROGRAM;  
RUN;
```

```
AF MYTIME.MAINCAT.MYTINIT.PROGRAM
```

The libref for the SAS library that contains the PTTS components must be 'MYTIME'. This reference can be created during a SAS session via a LIBNAME statement as follows:

```
LIBNAME MYTIME 'operating system specific';
```

For regular use of PTTS, the user may want to create a file of SAS programming statements or a SAS macro to do the above steps so that the system can be invoked with ease. The means by which this is done is highly dependent on the operating system and the preferred SAS invocation mode.

In addition to whatever program that the user may create to invoke the application, PTTS is comprised of the following SAS files that are maintained in the MYTIME library:

-----Directory-----

```
Libref:    MYTIME
Engine:    V606
Physical Name: E:\SASLIB\MYTIME
# Name      Memtype
-----
1 CONTROL   DATA
2 DALYLOG   DATA
3 MAINCAT   CATALOG
```

In this case, the CONTENTS procedure is showing the MYTIME library that is kept on an OS/2® based PS/2® in the subdirectory named \SASLIB\MYTIME on the systems E: drive. Of the 3 SAS files in the library, the MAINCAT catalog is the only file that is delivered to the user. It contains all the PF key definitions, PMENU's and SCL programs necessary to properly execute the PTTS application. It also contains a SOURCE entry that has all the SAS source needed to build the PMENU entries (though this is normally not necessary). This application has been successfully ported to VM®, MVS® and OS/2 by creating/importing transportable SAS files via the CPORT and CIMPORT procedures (Note: PF key entries had to be recreated and the program entries had to be recompiled for the OS/2 ported versions due to transporting the SAS catalog down a release level: 6.07-6.06). When the Personal Time Tracking System is invoked for the first time (or when it is invoked and none of the required SAS data files exist), the two SAS data files used by the application are created automatically.

THE PTTS DATA FILES

The primary SAS data file in the MYTIME library is dataset DALYLOG. This is where the PTTS log entries are stored. Each observation represents an individual log entry for a specific day (up to 15 entries per day). A modified CONTENTS procedure output for the DALYLOG variables follows:

Data Set Name: MYTIME.DALYLOG

-----List of Variables and Attributes-----

Variable	Type	Len	Format	Informat
DATE	Num	5	mmddyy8.	mmddyy8.
HOURS	Num	5	5.2	5.2
CODE	Char	3	\$3.	\$3.
DLYDESC	Char	50	\$50.	\$50.

Each PTTS log entry contains the date, number of hours (up to 2 decimal places), activity type code (from 1 to 3 characters) and, optionally, up to 50 characters of descriptive text. The total length of an observation is 63 characters which is ideal for the display of an individual PTTS log entry on a single line of the typical 3270 type terminal. Judicious use of the DLYDESC variable (field) in conjunction with an appropriate CODE value system should suffice for the tracking/reporting of activities.

The second SAS data file in the MYTIME library is the CONTROL dataset. This is where activity type codes are created and managed. A modified CONTENTS procedure output for the CONTROL variables follows:

Data Set Name: MYTIME.CONTROL

-----List of Variables and Attributes-----

Variable	Type	Len	Format	Informat
CODE	Char	3	\$3.	\$3.
CSDESC	Char	25	\$25.	\$25.
DEPT6	Char	6	\$6.	\$6.
PREFIX	Char	4	\$4.	\$4.
SUFFIX	Char	5	\$5.	\$5.
CLDESC1	Char	50	\$50.	\$50.
CLDESC2	Char	50	\$50.	\$50.
CLDESC3	Char	50	\$50.	\$50.
CLDESC4	Char	50	\$50.	\$50.
CLDESC5	Char	50	\$50.	\$50.

The CODE variable is the primary key (so to speak) and there will be one observation in the CONTROL dataset for each unique value of the CODE variable. The CDESC variable provides for a short description of the code (which is displayed with the code variable when a code selection list is called), whereas, the five CLDESC variables (CLDESC1 - CLDESC5) provide for up to five lines of additional descriptive text that can be associated with each unique value of CODE. The DEPT6, PREFIX and SUFFIX variables are intended for additional types of summarization and reporting which is not included in this base system.

The MYTIME.MAINCAT Catalog

All programs, KEY and PMENU entries necessary to execute the PTTS application are stored in the MAINCAT catalog. None of these entries should need modification when porting to a new user (although this can be otherwise as noted above). A modified BUILD directory of the MYTIME.MAINCAT catalog follows:

+BUILD: DIRECTORY MYTIME.MAINCAT+

Name	Type	[Description]
MYTCNTL	KEYS	[pf1=help/pf7=mytback/pf8=mytforw]
MYTNONE	KEYS	[All PF Keys undefined]
MYTVALL	KEYS	[pf7=backward/pf8=forward]
MYTCNTL	PMENU	[help/add/deleted/end]
MYTRPT1	PMENU	[help/cancel/run]
MYTRPT2	PMENU	[help/end]
MYTVIEW	PMENU	[help/view/report/control/end]
MYTCDEL	PROGRAM	[Verify control delete]
MYTCNTL	PROGRAM	[Edit control dataset]
MYTCREA	PROGRAM	[Datasets create]
MYTINIT	PROGRAM	[Invocation pgm]
MYTRCOD	PROGRAM	[Display code report]
MYTRDAT	PROGRAM	[Display date report]
MYTRMAN	PROGRAM	[Report program]
MYTVALL	PROGRAM	[View dalylog data]
MYTVDLY	PROGRAM	[Daily view of data]
MYTVPER	PROGRAM	[Period view of data]
MYTPMNU	SOURCE	[Pmenus create code]

In the MYTIME.MAINCAT catalog, there are four basic entry types: source, pmenu, key and program.

The program entries contain all the SCL code for running the system and will be reviewed in detail to explain the inner working of the system later on in this application development description.

HELP, PF KEYS and PMENUs

Ideally, a fully developed interactive application should have online help available to the user. The SAS/AF software provides both general and Computer Based Training help development tools to the programmer. The PTTS system provides the potential for help functionality via a menu selection or (on occasion) a PF key. But the associated help or CBT entries have not yet been created at this time (and will, therefore, not be described).

There are 3 PF key entries used by some program entries in the MAINCAT catalog: MYTCNTL, MYTNONE and MYTVALL. The 'mytcntl' pf key definitions include two custom commands: 'mytback' and 'mytforw'. These are made available when viewing the CONTROL dataset for moving to the prior/next observation the the dataset (the CONTROL dataset is viewing one observation at a time). The 'mytmone' is used by programs that require no PF keys to be defined (actions only available via pmenu selections). The 'mytvall' pf key definitions include the SAS commands 'backward' and 'forwards'. These are needed for scrolling through the DALYLOG dataset (which is viewed in tabular form). All of the KEY entries were created by opening, editing and saving (to the MYTIME.MAINCAT catalog) a SAS display manager KEY window (available via the KEY display manager command).

The user interacts with the PTTS application via a windowed environment. Each window has an associated set of PMENU selections that are displayed at the top of the window. To make a selection the user moves the cursor to the desired selection and presses enter to execute that selection. Some selections will, in turn, display another set of selections to fully qualify the desired action. PMENU entries are created in the SAS System by using the PMENU procedure. The SAS source code used to created the PMENU entries used by the PTTS application is stored in the MYTPMNU source entry and is as follows:

```
proc pmenu cat=mytime.maincat;
/* Define MYTCNTL pmenu */
```

```

menu mytcntl;
  item HELP; item ADD ;
  item DELETE; item END ;
run;
/* Define MYTVIEW pmenu */
menu mytview;
  item HELP; item 'VIEW' menu=view;
  item 'REPORT' menu=report;
  item CONTROL; item END;
menu view;
  item 'DAILY' selection=daily;
  item 'PERIOD' selection=period;
  item 'ALL' selection=all;
  selection daily 'vdaily; end';
  selection period 'vperiod; end';
  selection all 'vall; end';
menu report;
  item 'BY-DATE' selection=bydate;
  item 'BY-CODE' selection=bycode;
  selection bydate 'bydate';
  selection bycode 'bycode';
run;
/* Define MYTRPT1 pmenu */
menu mytrpt1;
  item HELP; item CANCEL; item RUN;
run;
/* Define MYTRPT2 pmenu */
menu mytrpt2;
  item HELP; item END;
run; quit;

```

A full description of PROC PMENU can be found in the 'SAS Procedures Guide, Version 6, Third Edition'. The above code will create four PMENU entries in the MYTIME.MAINCAT catalog. Each selection from any pmenu will ultimately return a text string to the SAS command interpreter. This text string can be either a command that is known to the SAS system or a custom command that is known to the associated SCL program ('vdaily', 'vperiod', 'vall', 'bydate' and 'bycode' are examples of custom commands used by the PTTs application). The SAS System does not save the source used to generate PMENU entries. This must be done by the programmer. Here it is saved in the MAINCAT catalog.

THE SAS/AF PROGRAM ENTRIES

The ten program entries in the MAINCAT catalog make up the bulk of the PTTs application. These entries were created using the SAS/AF BUILD procedure. Of the program entries, 2 relate to the

invocation (MYTINIT, MYTCREA), 3 are used for viewing the daily log entry dataset (MYTVDLY, MYTVPER, MYTVALL), 2 relate to the control dataset maintenance (MYTCNTL, MYTCDEL) and 3 are used for summary reporting (MYTRMAN, MYTRCOD, MYTRDAT). All program entries have associated display windows with the exception of the 2 invocation related program entries. In the following paragraphs, each of the four groups of program entries in this application are discussed in more detail. Though the SCL source code for all these program entries is included in this paper, it is not reviewed in its entirety. Instead, it will be discussed in general with references to the complete source. If a program entry has an associated display window, a rough example of how it may appear from within the BUILD procedure is given. Any important general and field attribute parameter values are indicated and can be assigned via the BUILD procedure general and field attribute windows. Display window sizes are not given and are left to your own preference. An in-depth discussion on how to use PROC BUILD can be found in 'SAS/AF Software: Usage and Reference, Version 6, First Edition'. The manual, 'SAS Screen Control Language: Reference, Version 6, First Edition' provides a description of all SCL statements.

Invocation Related Programs

The MYTINIT program entry is always the first to be executed when the system is invoked as described earlier. It has two primary purposes: to check for the existence of the 'dailylog' and 'control' datasets in the MYTIME library and to invoke the MYTCREA program if either dataset does not exist. It has no display window but parent general attribute value must be given so that control will be given to the 'daily' view program when the initiation of the application is complete.

```

ENTRY: MYTINIT.PROGRAM
[Initial program to invoke MYTIME appl]
No Display

```

***** GENERAL ATTRIBUTES *****

Parent: MYTVDLY.PROGRAM

***** SOURCE *****

INIT:

```

if ^(exist('mytime.dalylog')) then
  call display('mytcrea.program');
if ^(exist('mytime.control')) then
  call display('mytcrea.program');
return;

```

```

MAIN:
return;

```

```

TERM:
return;

```

The MYTCREA program creates the 'dalylog' and 'control' SAS data files in the MYTIME library if they do not exist. When the 'dalylog' dataset is created an observation is added for the first day of the current year (code=HOL for holiday). When the 'control' dataset is created, 3 observations are added (GEN=general, VAC=vacation, HOL=holiday). The SCL code for all of this seems rather large but follows a basic pattern: the dataset is opened as new, the variable and attributes are defined and, then, the dataset is closed. The dataset is opened again, a 'set' is done, values are assigned to variables and then appended for the number observations being added and then the dataset is closed again. There is no display window or general attributes specified for this program entry and control returns to the MYTINIT program entry when MYTCREA completes execution.

ENTRY: MYTCREA.PROGRAM
 Create MYTIME datasets (if don't exist)
 No display
 No general attributes specified

***** SOURCE *****

```

INIT:
if ^(exist('mytime.dalylog')) then do;
  dlogid=open('mytime.dalylog','n');
  rc=newvar(dlogid,'date','n',4,
    'Date','mmddyy8.','mmddyy8.');
```

```

  rc=newvar(dlogid,'hours','n',4,
    'Hours','5.2','5.2');
```

```

  rc=newvar(dlogid,'code','c',3,'Code','$3','$3.');
```

```

  rc=newvar(dlogid,'dlydesc','c',50,
    'Entry Description','$50','$50.');
```

```

  rc=close(dlogid);
  dlogid=open('mytime.dalylog','u');
  call set(dlogid);
  date=mdy(1,1,year(today())); code='HOL';
  hours=8; dlydesc='New Year's Holiday';

```

```

  rc=append(dlogid); rc=close(dlogid);
end;
if ^(exist('mytime.control')) then do;
  cntlid=open('mytime.control','n');
  rc=newvar(cntlid,'code','c',3,'Code','$3','$3.');
```

```

  rc=newvar(cntlid,'csdesc','c',25,
    'Code Description: Short','$25','$25.');
```

```

  rc=newvar(cntlid,'dept6','c',6,
    'Associated Dept Code','$6','$6.');
```

```

  rc=newvar(cntlid,'prefix','c',4,
    'Associated Prefix Code','$4','$4.');
```

```

  rc=newvar(cntlid,'suffix','c',5,
    'Associated Suffix Code','$5','$5.');
```

```

  rc=newvar(cntlid,'cldesc1','c',50,
    'Code Description: Long - Line 1',
    '$50','$50.');
```

```

  rc=newvar(cntlid,'cldesc2','c',50,
    'Code Description: Long - Line 2',
    '$50','$50.');
```

```

  rc=newvar(cntlid,'cldesc3','c',50,
    'Code Description: Long - Line 3',
    '$50','$50.');
```

```

  rc=newvar(cntlid,'cldesc4','c',50,
    'Code Description: Long - Line 4',
    '$50','$50.');
```

```

  rc=newvar(cntlid,'cldesc5','c',50,
    'Code Description: Long - Line 5',
    '$50','$50.');
```

```

  rc=close(cntlid);
  cntlid=open('mytime.control','u'); call set(cntlid);
  code='GEN'; csdesc='General activities';
  cldesc1='This work category (code=GEN)''
    'is created when the ';
```

```

  cldesc2='MYTIME system is invoked for ''
    'the first time. It ';
```

```

  cldesc3='is a default work category. You ''
    'may delete or ';
```

```

  cldesc4='edit this CODE control record as''
    'you see fit. ';
```

```

  cldesc5=' ';
```

```

  rc=append(cntlid);
  code='VAC'; csdesc='Vacation Time ';
```

```

  cldesc1='This work category (code=VAC)''
    'is created when the ';
```

```

  cldesc2='MYTIME system is invoked for ''
    'the first time. It ';
```

```

  cldesc3='is a default work category. You ''
    'may delete or ';
```

```

  cldesc4='edit this CODE control record as''
    'you see fit. ';
```

```

  cldesc5=' ';
```

```

  rc=append(cntlid);
  code='HOL'; csdesc='Business Holiday ';
```

```

  cldesc1='This work category (code=HOL)''
    'is created when the ';
```

```

cldesc2='MYTIME system is invoked for 'll
        'the first time. It ';
cldesc3='is a default work category. You 'll
        'may delete or ';
cldesc4='edit this CODE control record as'll
        'you see fit. ';
cldesc5=' '; rc=append(cntlid);
rc=close(cntlid);
end;
return;

MAIN:
return;

TERM:
return;

```

Together the MYTINIT and MYTCREA programs provide the basic functional foundations for invoking and initializing the PTTS application.

PROGRAMS TO VIEW LOG ENTRIES

There are 3 SCL programs in the PTTS application that provide for the viewing and editing of the daily log data: MYTVDLY, MYTVPER and MYTVALL. Each of these provide a slightly different view of the data: one day at a time, a select period of dates and all dates in the dataset. All three views present the data as an extended table with one row representing one observation in the DALYLOG dataset. But because of the difference in 'view', each has a slightly differently designed window (though with some commonalities) and some different underlying assumptions. This causes some of the SCL source code to be different (particularly in the INIT section) but the associated PMENU definition and, therefore, the code in the MAIN section of the SCL programs are basically the same.

This discussion will focus primarily on the SAS/AF entry for the initial default 'daily' view. For each of the 3 program entries, the a crude example of the entry's display window is given that shows the basic text of the display and the display fields with their associated SCL variable names (when displayed these fields will display as blank unless provided with a value by the program or the user). General attributes for the AF entry are given and the notable attributes for each display field (SCL variables) are provided. Since much of the code for the MAIN section of all these program entries is the same, the discussion of it for the MYTVDLY will suffice for

all three. The discussion of the MYTVPER and MYTVALL program entries will be more general and focus their differences from MYTVDLY.

The MYTVDLY program entry is the user's first view of the data when the application is invoked. If another view is desired during any particular use of the application, then the user can select the VIEW button from the menu bar and then select the desired view from the pull down menu that appears. By default, this program will always initially display any existing data for the current date. If no data exists then the current date is displayed and all the daily log entry fields are blank (have a null value). Up to 15 entries for any particular date are allowed. To add (or edit) one or more observations for the displayed date to the DALYLOG dataset, the user tabs to the desired log entry line and enters the desired values (hours, code and description). To add/edit for another date, the user tabs to the date field and enters the new date value (in mm/dd/yy format). For any date that is indicated, the program also displays the current total number of hours summed across all observations for that day. The menu bar also allows the user to edit the CONTROL dataset, to request a summary report of hours across a period of days by code or date and to select to end the application.

The display window has 7 fields (SCL variables) defined. Of these, four have the same name or alias as a corresponding variable in the DALYLOG dataset (date/dt, hours/hrs, code/cd and dlydesc). Three of the fields are protected (totl, dt and obnum/o) and two of the fields are non-display (dt and o). The MYTNONE set of PF keys are defined for the entry which is given the system option of 'extended table'. No prompt character is given and the command menu associated with the display is MYTVIEW (the same for the other 2 'view' entries).

ENTRY: MYTVDLY.PROGRAM
Daily view of DALYLOG observations

***** DISPLAY *****

Date: &xdate_ (mm/dd/yy) Total Hrs: &totl

Hours	Code	Description of Task
&dt &hrs_	&cd	&dlydesc_____ &o

******* GENERAL ATTRIBUTES *******

Function keys: MYTNONE
 Command menu: MYTVIEW
 Prompt char:
 System Options: EXTENDED TABLE

******* FIELD ATTRIBUTES *******

Field name: XDATE Length: 17
 Alias: XDATE Type: NUM
 Format: WEEKDATE17.
 Informat: MMDDYY8.
 Options: CAPS CURSOR AUTOSKIP

Field name: TOTL Length: 5
 Alias: TOTL Type: NUM
 Protect: YES Format: 5.2
 Options: CAPS AUTOSKIP

Field name: DT Length: 3
 Alias: DATE Type: NUM Protect: YES
 Options: CAPS AUTOSKIP NON-DISPLAY

Field name: HRS Length: 5
 Alias: HOURS Type: NUM
 Format: 5.2 Just: RIGHT Informat: 5.
 Options: CAPS AUTOSKIP

Field name: CD Length: 3
 Alias: CODE Type: CHAR
 Format: \$3. Informat: \$3.
 Options: CAPS AUTOSKIP

Field name: DLYDESC Length: 50
 Alias: DLYDESC Type: CHAR
 Format: \$50. Informat: \$50.
 Options: AUTOSKIP

Field name: O Length: 2
 Alias: OBNUM Type: NUM Protect: YES
 Format: Z2. Informat: 2.
 Options: CAPS AUTOSKIP NON-DISPLAY

The source component of this and the other two 'view' programs has 5 labeled sections: the usual INIT, MAIN and TERM sections and, because they use the 'extended table' system option, they have a GETROW section (for assigning data to fields) and a PUTROW section (for when fields in the table are modified). This application uses 'custom commands' (commands not known to the SAS System but have special meaning to these program entries). So the

'control' statement is used in the INIT section to ensure that the MAIN section will always be executed when the user presses enter after the window is displayed and an SCL variable (dlogword) is defined that will be used in MAIN to evaluate the 'command' (action requested by the user from the menu). Since the application is designed to use PMENUs (and MYTVLDY is the first program invoked that has a display window), the 'pmenu' SAS command is executed to ensure that the pmenu facility is turned on when the program is invoked.

The INIT section also does the following:

- obtains the current date
- opens the proper datasets
- applies the where statement
- 'sets' SCL variables to dataset variables
- obtains value for total hours
- read data into the table from dalylog

INIT is only executed once (at program invocation).

After the initial window is displayed (and the INIT section has been executed), the program will wait for the user to press enter before continuing. The user can modify fields and/or make a selection from the menu. When enter is pressed, the program first checks to see if any fields in the extended table have been changed. If so, the PUTROW section of the program is executed for each modified table row. When PUTROW is complete or if no extended table fields have been modified, the statements in the MAIN section are executed and then the GETROW labeled section (to redisplay the extended table). This is the natural flow of logical control in SCL extended table processing which ensures that any actions due to extended table field modification will occur prior to any other activity.

The GETROW section is first executed by the INIT section via the SETROW statement and then after PUTROW and MAIN (if any table fields have been modified by the user). The SETROW used in this program indicates 15 rows in the table. So when GETROW is 'called' it executes 15 times. With the where clause in effect for the indicated date, the fetchobs will load the next observation that meets the where condition into the next row of the table. If the return code from the fetch is not zero (no more observations that meet the condition), then the value of the 'obnum' SCL variable is set to '0'. Otherwise, obnum is given the observation number. This way a modified extended table row can be determined to be either a modified or new observation.

When executed for each modified table row, the PUTROW section will delete the corresponding observation from the dataset if the 'hours' field is blank or contains a missing value. If the value of the 'code' field is a question mark, then a selection list of valid codes (in the 'control' dataset) is displayed with a short description of each code. This is done using the 'datalistc' SCL function. Any entered/modified code value that cannot be found in the 'control' dataset (using 'locatec') will cause the code value to be reset to _blank_, an 'invalid code' message to be displayed and PUTROW processing discontinued for that row. If the value of the obnum variable is zero then the data in the extended table row will be appended to the 'dalylog' dataset. Otherwise, the data is 'updated' in the dataset.

The GETROW and PUTROW sections of the SCL program constitute the bulk of extended table processing which allows the viewing of data in rows.

***** SOURCE *****

INIT:

```
control always; call execcmdi('pmenu on');
length dlogword $ 8 ; xdate=today(); totl=0;
_msg='Enter data for date or enter new date.';
dlogid=open('mytime.dalylog','u');
cntlid=open('mytime.control','i');
rc=where(dlogid,'date'='||putn(xdate,7.)');
call set(dlogid);
do while (fetch(dlogid)^=-1);
    totl=totl+hours;
end;
rc=rewind(dlogid); call setrow(15);
return;
```

MAIN:

```
if _status_ ^= 'E' then dlogword=word(1,'u');
if dlogword='CONTROL' then do;
    rc=close(cntlid); dlogword='';
    call display('mytcntl');
    cntlid=open('mytime.control','i');
end;
if dlogword='BYDATE' then do;
    rc=close(cntlid); rc=close(dlogid);
    if exist('work.rdates') then
        rc=delete('work.rdates');
    if exist('work.dlyrpt') then
        rc=delete('work.dlyrpt');
    call display('mytrman',dlogword); dlogword='';
    if exist('work.dlyrpt') then
        call display('mytrdat');
```

```
dlogid=open('mytime.dalylog','u');
cntlid=open('mytime.control','i');
call set(dlogid);
end;
if dlogword='BYCODE' then do;
    rc=close(cntlid); rc=close(dlogid);
    if exist('work.rdates') then
        rc=delete('work.rdates');
    if exist('work.codrpt') then
        rc=delete('work.codrpt');
    call display('mytrman',dlogword); dlogword='';
    if exist('work.codrpt') then
        call display('mytrcod');
    dlogid=open('mytime.dalylog','u');
    cntlid=open('mytime.control','i');
    call set(dlogid);
end;
rc=where(dlogid,'date'='||putn(xdate,7.)'); totl=0;
do while (fetch(dlogid)^=-1); totl=totl+hours; end;
rc=rewind(dlogid);
return;
```

TERM:

```
rc=close(dlogid);
rc=close(cntlid);
dlogid=open('mytime.dalylog','u');
rc=sort(dlogid,'date code descending hours');
rc=close(dlogid);
if dlogword='VDAILY' then
    call goto('mytvldly.program','r');
if dlogword='VPERIOD' then
    call goto('mytvper.program','r');
if dlogword='VALL' then
    call goto('mytvall.program','r');
return;
```

GETROW:

```
rc=fetch(dlogid);
if rc=0 then obnum=curobs(dlogid);
else do;
    date=.; hours=_blank_; code='';
    dlydesc=''; obnum=0;
end;
return;
```

PUTROW:

```
date=xdate;
if hours=_blank_ or hours=. then do;
    rc=fetchobs(dlogid,_currow_,_noset);
    rc=delobs(dlogid); return;
end;
if code='?' then code=
    datalistc(cntlid,'code csdesc');
```

```

if locatec(cntlid,varnum(cntlid,'code'),code)=0
then do;
    code=_blank_; _msg_='Code '||
        code||' not valid';
    return;
end;
if obnum=0 then do;
    rc=append(dlogid); return;
end;
rc=fetchobs(dlogid,_currow_,_noset');
rc=update(dlogid);
return;

```

The MAIN section in this program is used to perform activities based upon selections that the user has made from the command menu. It will always get a chance to inspect the 'command line' after the SAS system because CONTROL ALWAYS is in effect. This not only tells the SAS system to always execute MAIN when the user presses enter (even if no fields have been modified) but it also tells the SAS system to allow commands that it does not recognize to pass to the SCL program. Thus, the first line of the MAIN section checks the '_status_' SCL system variable to see if the END command has been processed. If not, then the first word in the command line is retrieved (this also removes the word from the command line).

The 'viewing' programs in the PTTS application have six custom commands (generated by related pmenu selections) that cause application activity in three basic areas: view selection for the dalylog dataset, control dataset maintenance and summary reporting. The custom commands generated by the VIEW pull-down menu selection ('vdaily', 'vperiod' and 'vall') are passed to the SAS command interpreter with the END command stacked behind them. Not initially recognized by the SAS system, the command line is then passed to the MAIN section of the program where the custom command is retrieved from the command line and assigned to the dlogword variable. The 'view' custom commands are also not recognized by the MAIN section but they have been removed from the command line. When control returns to the SAS System, the subsequent END statement is then interpreted by SAS and control is ultimately directed to the TERM section of the program.

When the CONTROL pmenu selection is made, a 'control' custom command is generated that is recognized by the MAIN section of the program.

Since the program that will be invoked requires update access to the CONTROL dataset, this program closes its current 'opening for input' of that dataset. The value of the dlogword is set to missing and the MYTCNTL program is called via the SCL display routine. When control is passed back to the view program, execution of MAIN continues. The CONTROL dataset is reopened again. The next executable code is at the end of the section where the 'where' clause is reevaluated and the 'totl' value is recalculated to be in line with data changes.

When the REPORT pmenu selection is made, a pull-down menu is displayed to select BY-DATE or BY-CODE summarization of the dalylog data. Depending upon the selection made, either the 'bydate' or 'bycode' custom command is passed to the MAIN section of the 'view' program. In either case, the dalylog and control datasets are closed, any temporary datasets related to reporting are checked for (and deleted if they exist) and the MYTRMAN program is called (passing to it the 'dlogword' variable value). When control returns, the report display program is then called. When control returns again, the dalylog and control datasets are reopened and the 'where' and 'totl' tasks are performed (end of MAIN).

The TERM section is executed (after MAIN) when the SAS System interprets an END command (the CANCEL command is not used in this application). This can either be when the END pmenu selection is made (to end the application) or when a different view of the dalylog dataset is requested. The first activity in this section is to close both the dalylog and control datasets. The dalylog dataset is reopened for update, sorted and closed. This sorting causes any dalylog observations that have been 'marked' for deletion (in PUTROW via the 'delobs' function) to actually be removed from the dataset. If termination of this 'view' program is via a VIEW pmenu selection, then the appropriate 'view' program is invoked with the GOTO routine using the 'r' value for the 'action' argument on the GOTO (removes top entry from the execution stack so that the next 'view' program will be able to end this application via the END selection).

The MYTVPER and MYTVALL Programs

These other two view programs are quite similar to the MYTVDLY program with the exception that they display data over a period of days (specified or

all) instead of displaying one day at a time and new observations cannot be added (appended) to the dalylog dataset (only updates/deletes can be done). Many of the SCL and data variables referenced in the display window and source code are the same in these view programs as in MYTVDLY. In both displays, the date field is displayed and not protected in the extended table portion of the display window. The MYTVPER program entry includes two date related SCL variables as fields in the display window and a protected field for displaying total hours across the selected view period. The MYTVALL program entry has no date or total hours fields in the display (only column headings and the extended table).

Both of these 'view' programs use a 'dynamic' extended table (so defined via the SETROW routine via the INIT section) that provides for unlimited rows in the table. Because of the high probability that the number of observations selected (rows in the table) will be greater than the number of rows available in the window, the MYTVALL function key settings provide two SAS commands (backwards & forwards) so the program can scroll across the extended table.

The MYTVPER program initially assumes that the desired period to be viewed is to be the current week which starts on Saturday and ends on Friday. The INIT section evaluates the last date in the period (xdate2) to be today's date and the first Saturday prior to today. The user may change the period once the program is invoked. The rest of the INIT section is pretty much the same as in MYTVDLY with variations on the WHERE function and the SETROW routine. The GETROW section is different (and simpler) than in the first view program because no observations are to be added in the dynamic extended table. The MAIN and TERM sections in the period view program are essentially identical to what is in the daily view program. For some code brevity in this document, commented statements are used in the code shown below to show the similarity (though not a substitute for the real thing). The PUTROW section of both period view programs are different only in that there is no provision for appending new data.

ENTRY: MYTVPER.PROGRAM
Period view of DALYLOG observations

***** DISPLAY *****

Period: &xdate1_ - &xdate2_ Total Hrs: &totl_

Date	Hours	Code	Description of Activity
&date_	&hrs_	&cd	&dlydesc_

***** GENERAL ATTRIBUTES *****

Function keys: MYTVALL
Command menu: MYTVIEW
System Options: EXTENDED TABLE

***** FIELD ATTRIBUTES *****

Field name: XDATE1 Length: 8
Alias: XDATE1 Type: NUM
Format: MMDDYY8. Informat: MMDDYY8.
Options: CAPS CURSOR REQUIRED AUTOSKIP

Field name: XDATE2 Length: 8
Alias: XDATE2 Type: NUM
Format: MMDDYY8. Informat: MMDDYY8.
Options: CAPS REQUIRED AUTOSKIP

Field name: TOTL Length: 6
Alias: TOTL Type: NUM Protect: YES
Format: 6.2 Just: RIGHT Informat: 6.2
Options: CAPS AUTOSKIP

Field name: DATE Length: 8
Alias: DATE Type: NUM
Format: MMDDYY8. Informat: MMDDYY8.
Options: CAPS AUTOSKIP

Field name: HRS Length: 5
Alias: HOURS Type: NUM
Format: 5.2 Informat: 5.
Options: CAPS AUTOSKIP

Field name: CD Length: 3
Alias: CODE Type: CHAR
Format: \$3. Informat: \$3.
Options: CAPS AUTOSKIP

Field name: DLYDESC Length: 50
Alias: DLYDESC Type: CHAR
Format: \$50. Informat: \$50.
Options: AUTOSKIP

***** SOURCE *****

INIT:

```
length dlogword $ 8 ; control always;
xdate2=today();
xdate1=xdate2-(weekday(xdate2)*
  (weekday(xdate2)^=7));
dlogid=open('mytime.dalylog','u');
cntlid=open('mytime.control','i');
rc=where(dlogid,'date>='||putn(xdate1,'7.')||
  ' and date<='||putn(xdate2,'7.));
call set(dlogid); totl=0;
do while (fetch(dlogid)^=-1);
  totl=totl+hours;
end;
rc=rewind(dlogid); call setrow(0,0,'','y');
return;
```

MAIN:

```
if _status_ ^= 'E' then dlogword=word(1,'u');
/* CONTROL do-group as in MYTVDLY */
/* BYDATE do-group as in MYTVDLY */
/* BYCODE do-group as in MYTVDLY */
rc=where(dlogid,'date>='||putn(xdate1,'7.')||
  ' and date<='||putn(xdate2,'7.)); totl=0;
do while (fetch(dlogid)^=-1);
  totl=totl+hours;
end;
rc=rewind(dlogid);
return;
```

```
TERM: /* Same as in MYTVDLY */
return;
```

GETROW:

```
if fetchobs(dlogid,_currow_) then call endtable();
return;
```

PUTROW:

```
if hours=_blank_ or hours=. then do;
  rc=fetchobs(dlogid,_currow_, 'noset');
  rc=delobs(dlogid); return;
end;
if code='?' then code=
  datalistc(cntlid,'code csdesc');
if locatec(cntlid,varnum(cntlid,'code'),code)=0
then do;
  _msg_='Code '||code||' not valid';
  code=_blank_; return;
end;
rc=fetchobs(dlogid,_currow_, 'noset');
rc=update(dlogid);
if rc=0 then _msg_='Record(s) updated';
return;
```

The MYTVALL program is the simplest of the three viewing programs in the PITTS application. There are no selection dates and, therefore, where function processing. There is no summing of hours across all rows in the table. Excepting for display screen modification and the elimination of source related to the above MYTVALL is identical to MYTVPER.

ENTRY: MYTVALL.PROGRAM

View ALL DALYLOG dataset observations

***** DISPLAY *****

Date	Hours	Code	Description of Activity
^^^			
&date__	&hrs_	&cd	&dlydesc_____

***** GENERAL ATTRIBUTES *****

Function keys: MYTVALL
 Command menu: MYTVIEW
 Prompt char:
 System Options: EXTENDED TABLE

***** FIELD ATTRIBUTES *****

Field name: DATE	Length: 8
Alias: DATE	Type: NUM
Format: MMDDYY8.	Informat: MMDDYY8.
Options: CAPS AUTOSKIP	

Field name: HRS	Length: 5
Alias: HOURS	Type: NUM
Format: 5.2	Informat: 5.
Options: CAPS AUTOSKIP	

Field name: CD	Length: 3
Alias: CODE	Type: CHAR
Format: \$3.	Informat: \$3.
Options: CAPS AUTOSKIP	

Field name: DLYDESC	Length: 50
Alias: DLYDESC	Type: CHAR
Format: \$50.	Informat: \$50.
Options: AUTOSKIP	

***** SOURCE *****

INIT:

```
length dlogword $ 8 ; control always;
dlogid=open('mytime.dalylog','u');
```

```

cntlid=open('mytime.control','i');
call set(dlogid); call setrow(0,0,'','y');
return;

```

MAIN:

```

if _status_ ^= 'E' then dlogword=word(1,'u');
/* CONTROL do-group as in MYTVDLY */
/* BYDATE do-group as in MYTVDLY */
/* BYCODE do-group as in MYTVDLY */
return;
TERM: /* Same as in MYTVDLY */
return;

```

```

GETROW: /* Same as in MYTVPER */
return;

```

```

PUTROW: /* Same as in MYTVDLY */
return;

```

MANAGING THE CONTROL DATASET

There are two programs in the PTTS application that provide for the maintenance of the 'control' dataset: MYTCNTL and MYTCDEL. The first is the primary program for viewing and editing the control dataset one observation at a time and is invoked from one of the 'viewing' programs by way of the CONTROL pmenu selection. The second is invoked by the first control program to verify the delete of an observation from the control dataset. Both programs return control to the program from which they were invoked when they have completed execution.

The display window for the MYTCNTL program defines and displays fields (SCL variables) for all control dataset variables. The MYTCNTL PF keys and command menu are defined as general attributes and, between the both of them, define four custom commands that are used here: MYTBACK & MYTFORW (function keys 7 and 8 for moving from observation in the dataset to another) and ADD & DELETE (via pmenu selection for adding and deleting observations from the control dataset). In the display and attribute descriptions that follow, the field attributes for CLDESC2-CLDESC5 are not given. They are basically the same as CLDESC1.

ENTRY: MYTCNTL.PROGRAM

View and Edit MYTIME.CONTROL dataset

***** DISPLAY *****

Code: &cd Brief Description: &cldesc_____

Dept: &dept6 Prefix: &pfx Suffix: &sufx

Expanded Description of What Code Represents:

```

&cldesc1 _____
&cldesc2 _____
&cldesc3 _____
&cldesc4 _____
&cldesc5 _____

```

***** GENERAL ATTRIBUTES *****

Function keys: MYTCNTL
Command menu^6: MYTCNTL

***** FIELD ATTRIBUTES *****

Field name: CD Length: 3
Alias: CODE Type: CHAR
Format: \$3. Informat: \$3.
Options: CAPS CURSOR REQUIRED AUTOSKIP

Field name: CSDESC Length: 25
Alias: CSDESC Type: CHAR
Format: \$25. Informat: \$25.
Options: REQUIRED AUTOSKIP

Field name: DEPT6 Length: 6
Alias: DEPT6 Type: CHAR
Format: \$6. Informat: \$6.
Options: CAPS AUTOSKIP

Field name: PFX Length: 4
Alias: PREFIX Type: CHAR
Format: \$4. Informat: \$4.
Options: CAPS AUTOSKIP

Field name: SUFX Length: 5
Alias: SUFFIX Type: CHAR
Format: \$5. Informat: \$5.
Options: CAPS AUTOSKIP

Field name: CLDESC1 Length: 50
Alias: CLDESC1 Type: CHAR
Format: \$50. Informat: \$50.
Options: AUTOSKIP

Since there are custom commands used by the MYTCNTL program, the INIT section in the source sets control to ALWAYS. Three SCL variables are defined via a LENGTH statement that will be used for custom command interpretation and program

task control: cntlword, addflag and deltind. The addflag variable (used for identifying new observations to add) is initialized to a character zero, the control dataset is opened for update, the SET routine is invoked to connect SCL variables to dataset variables, the first observation in the dataset is fetched (which loads the variable values to their corresponding fields in the display window) and a text string is assigned to the _msg_ SCL system variable to remind the user how to scroll from one observation to the next. The program will then wait for the user to press enter (to select a pmenu displayed command or apply a modification to a displayed field for the current observation) or to press PF7/PF8 (which will cause the program to scroll to the prior or next observation in the dataset).

The first activity of the MAIN section is to obtain the value for the first word that may have been passed from the SAS command interpreter. MAIN checks to see if any of the fields in the display window have been modified. If so, and the value of addflag is a character zero (which indicates that this is data for an existing observation), then that observation in the control dataset is updated. The message returned to the user indicates whether or not the update operation was successful. If the value of the addflag variable is a character one (which means that this is a new observation to be added), then the display data is added as a new observation to the dataset. Again, a message will be displayed in regards to the success of the 'append' operation.

Of the four selections available from the pmenu associated with this program's display window, HELP/ADD/DELETE/END, 2 are custom commands (add & delete). When the ADD selection is made, the values of all fields are set to missing, the value of the addflag variable is set to a character one and control is returned from the MAIN section to the program display (awaiting user input). If the DELETE selection is made, then the value of the deltind SCL variable is set to 'NO' and is passed to the MYTCDEL program as it is called using the DISPLAY routine. A value for the deltind SCL variable is also returned from MYTCDEL. If it is still equal to 'NO' then program control is returned from MAIN with no action taken. Otherwise, the current observation is deleted, the dataset is sorted and the first observation in the dataset is displayed. A message indicating that the observation was deleted is displayed when program control returns.

The MYTBACK and MYTFORW custom commands used in this program are good examples of how to setup scrolling from one SAS dataset observation to another in an SCL program. In this application, these custom commands are generated via PF keys assigned to the program entry. When the MAIN section detects the MYTBACK command, a check is performed to determine if this is the first observation. If so, a message is generated and control is returned to the program display. Otherwise, the next dataset observation is fetched. With MYTFORW, an attempt is made to fetch the next observation. If a non-zero return code results, then the program generates a message and fetches the last observation.

The TERM section of MYTCNTL, closes the dataset, reopens the dataset, sorts it and then closes it again. Control is returned to the calling program.

***** SOURCE *****

INIT:

```
control always;
length cntlword $ 8 addflag $ 1 deltind $ 3;
addflag='0';
cntlid=open('mytime.control','u');
call set(cntlid); rc=fetch(cntlid);
if rc=0 then _msg_ =
'Use PF7/PF8 to page Backwards/Forwards';
return;
```

MAIN:

```
cntlword=word(1,'u');
if field('modified',
'code csdesc dept6 prefix suffix cldesc1
cldesc2 cldesc3 cldesc4 cldesc5')
then do;
if addflag='0' then do;
if update(cntlid)=0 then
_msg_='Record updated';
else _msg_='Problem with update ....';
return;
end;
if addflag='1' then do;
if append(cntlid)=0 then do;
addflag='0'; _msg_='Record added';
end;
else _msg_='Problem w/adding record ....';
return;
end;
end;
if cntlword='ADD' and addflag='0' then do;
```

```

code=_blank_; csdesc=_blank_;
dept6=_blank_; prefix=_blank_;
suffix=_blank_; cldesc1=_blank_;
cldesc2=_blank_; cldesc3=_blank_;
cldesc4=_blank_; cldesc5=_blank_;
addflag='1'; return;
end;
if cntlword='DELETE' then do;
  deltind='NO';
  call display('mytdel.program',deltind);
  if deltind='NO' then return;
  if delobs(cntlid)^=0 then return;
  rc=sort(cntlid,'code'); rc=rewind(cntlid);
  rc=fetch(cntlid);
  _msg_='Control record was deleted';
  return;
end;
if cntlword='MYTBACK' then do;
  if curobs(cntlid)<=1 then
    _msg_='Top of Data';
  else rc=fetchobs(cntlid,curobs(cntlid)-1);
end;
if cntlword='MYTFORW' then do;
  if fetch(cntlid)^=0 then do;
    rc=fetchobs(cntlid,attrn(cntlid,'nobs'));
    _msg_='End of Data';
  end;
end;
return;
TERM:
  rc=close(cntlid);
  cntlid=open('mytime.control','u');
  rc=sort(cntlid,'code'); rc=close(cntlid);
return;

```

The MYTCDEL program is called so the user can verify whether or not an observation in the control dataset should really be deleted. The verification in the program is done via a 'choice group'. The choice group in a SAS/AF entry allows for the selection of only one of two (or more) choices. In this case, the choice is either 'NO' or 'YES'. To make a selection, the user moves the cursor to the desired choice and presses enter. The value of the display field (which is given an initial value and protected) is passed to the SCL variable which has the same name as the choice group name. The ENTRY statement defines the SCL variable which will contain the value that is passed to and from this program in coordination with the calling program (MYTCNTL). There is no pmenu definition or PF keys (MYTNONE) for MYTCDEL and, by field attribute definition, the

cursor will be initially on the 'NO' selection (field name: I1). When the user presses the enter key execution of the program will be terminated (_status_='H') and the value assigned to the field that the cursor is on will be passed to the calling program (MYTCNTL in this case).

ENTRY: MYTCDEL.PROGRAM
Verify CONTROL observation deletion

***** DISPLAY *****

Are you sure you want to delete this control record?
 &i1 - Cancel Delete
 &i2 - Delete Record

***** GENERAL ATTRIBUTES *****

Function keys: MYTNONE

***** FIELD ATTRIBUTES *****

Field name: I1	Length: 3
Alias: I1	Choice group: DELTREC
Type: CHAR	Initial: NO
Options: CAPS CURSOR AUTOSKIP NOPROMPT	

Field name: I2	Length: 3
Alias: I2	Choice group: DELTREC
Type: CHAR	Initial: YES
Options: CAPS AUTOSKIP NOPROMPT	

***** SOURCE *****

entry deltre \$ 3 ;

INIT:
return;

MAIN:
 status='H';
return;

TERM:
return;

THE SUMMARIZATION PROGRAMS

The PTTS application provides for the summarization and display of summarized data using three program entries: MYTRMAN, MYTRDAT and MYTRCOD. The actual

summarization is done by submitting statements to SQL for processing. The advantage of this over doing a SUBMIT to the SAS System is that it does not require terminating the AF application and then re-invoking it to get the desired summary. In this application, summarization can be invoked from any one of the three viewing programs and is a four step process: select REPORT from the command menu, indicate the desired summarization class (BY-DATE or BY-CODE) from the resulting pull-down menu, edit the first and last date values in the window that is then displayed (if the defaults are not what is desired) and, then, select RUN from that window's pmenu selection. The summary is then displayed. Irregardless of the selection made from the REPORT pull-down menu, MYTRMAN is invoked from the calling viewing program. The value of the dlogword SCL variable in the calling program is passed to MYTRMAN and is used in the INIT section to determine the message that is displayed in the display window to remind the user of the summarization class that has been chosen. By default, the first and last date of the reporting period is determined to be for the current week (Saturday-Friday). These dates may be altered by the user who can also cancel the reporting function by selecting the CANCEL menu. Once the summarization period is set, the process is started by selecting RUN from the command menu. This is the only custom command created for this program entry. Whenever MAIN executes the value for the rtype variable is reexamined and the message to remind the user of the report type is displayed.

Some validation is done on the 'fdate' and 'ldate' display fields in MAIN. The values are checked for blank or missing values and a check is done to ensure that the value for the first date (fdate) is not greater than the value for the last date (ldate). If any of these conditions are detected, the field is highlighted, the cursor is positioned on that field and the appropriate message is generated when MAIN completes.

When the RUN custom command is selected and the display fields have been successfully validated, MAIN first creates a temporary dataset ('work.rdates') to which one observation is added that contains the values for the fdate and ldate variables. These values are also assigned to the SCL program variables 'rdate1' and 'rdate2'. These variables serve as macro variables whose values will be substituted when the SQL statements are

submitted to the SQL processor.

Depending on the 'rtype' value, a specific group of SQL statements are submitted to the SQL processor using the SCL SUBMIT statement. The value for the 'when' parameter is 'continue' and the value for the 'where' parameter is 'sql'. 'Continue' must be used when statements are submitted to the SQL processor (as opposed to submitting statements to SAS). This will cause all statements in the submit block to be executed by SQL and control to be returned to the SCL program (statements following the submit block).

If the summary is selected to be by date, the SQL statements submitted request a table (dataset) to be created using the date and hours variables from the 'mytime.dalylog' dataset. A column (variable) called 'tothours' is created whose value will represent the sum of the hours variable. The created table will be a summary that is grouped by date and the only data that is input from the dalylog dataset will be those observations whose value for date is within the rdate1 to rdate2 period. If summarization by code has been selected, then the submit block first creates a table called 'temp1' that is the code summarization equivalent to the table that is created for 'by-date' reporting. Then a 'codrpt' table (dataset) is created that includes the 'temp1' data and the short description of the code variable values from the 'control' dataset (via a match merge on 'code'). Irregardless of the report type, the _status_ SCL system variable is set to 'H' which invokes the termination of the MYTRMAN program.

ENTRY: MYTRMAN.PROGRAM
Primary MYTIME report program

***** DISPLAY *****

Enter report period dates (MM/DD/YY):
First > &fdate__
Last > &ldate__
Select CANCEL to exit, RUN to report.

***** GENERAL ATTRIBUTES *****

Function keys: NONE
Command menu: MYTRPT1

***** FIELD ATTRIBUTES *****

Field name: FDATE Length: 8
Alias: FDATE Type: NUM
Format: MMDDYY8. Informat: MMDDYY8.
Options: CAPS CURSOR REQUIRED AUTOSKIP

Field name: LDATE Length: 8
Alias: LDATE Type: NUM
Format: MMDDYY8. Informat: MMDDYY8.
Options: CAPS REQUIRED AUTOSKIP

***** SOURCE *****

entry rtype \$ 6 ;

INIT:

```
length dlogword $ 8 ; control always;
call nextcmd();
if rtype='BYDATE' then
  _msg_='Reporting BY-DATE for Period.';
if rtype='BYCODE' then
  _msg_='Reporting BY-CODE for Period.';
fdate=today()-(weekday(today())*
  (weekday(today())^=7));
ldate=fdate+6;
return;
```

MAIN:

```
dlogword=word(1,'u');
if rtype='BYDATE' then
  _msg_='Reporting BY-DATE for Period.';
if rtype='BYCODE' then
  _msg_='Reporting BY-CODE for Period.';
if fdate=. or fdate=_blank_ then do;
  erroron fdate; cursor fdate;
  _msg_='Enter Date Value or select CANCEL';
  return;
end;
if ldate=. or ldate=_blank_ then do;
  erroron ldate; cursor ldate;
  _msg_='Enter Date Value or select CANCEL';
  return;
end;
if fdate>ldate then do;
  erroron fdate; cursor fdate;
  _msg_='First date is greater than last date';
  return;
end;
if dlogword='RUN' then do; dlogword='';
  rdatid=open('work.rdates','n');
  rc=newvar(rdatid,'fdate','n',4);
  rc=newvar(rdatid,'ldate','n',4);
  rc=close(rdatid); if rc=0 then _msg_='';
  rdatid=open('work.rdates','u');
```

```
call set(rdatid); rc=append(rdatid);
rc=close(rdatid); rdate1=fdate; rdate2=ldate;
if rtype='BYDATE' then do;
  submit continue sql;
  create table dlyrpt as
  select date, sum(hours) as tothours
  from mytime.dalylog
  where date>=&rdate1 and
  date<=&rdate2
  group by date;
endsubmit;
end;
if rtype='BYCODE' then do;
  submit continue sql;
  create table temp1 as
  select code, sum(hours) as tothours
  from mytime.dalylog
  where date>=&rdate1 and
  date<=&rdate2
  group by code;
  create table codrpt as
  select a.code, a.tothours,
  b.csdesc
  from temp1 as a, mytime.control as b
  where a.code=b.code ;
endsubmit;
end;
_status_='H';
end;
return;
```

TERM:

return;

The MYTRDAT and MYTRCOD report display programs are two variations on the same theme. Each is called by the viewing program after the termination of the MYTRMAN program and the successful creation of their corresponding dataset. They are both dynamic extended tables, use a similar display and follow the same program logic. The only real difference is the summary variable (and the csdesc variable that is in MYTRCOD). For brevity, the MYTRDAT program will be shown and described in detail. Only those components of the MYTRCOD program that are different from the MYTRDAT will be presented in this document.

The display window includes the period dates and the total hours for the period. The individual dates, total hours and percent of total hours for the period are shown in the extended table portion of the display window. In addition, an action field (the

single ampersand, '&', is automatically defined with the 'action' field attribute and named FIELD4). This field has not been made functional in the PTTS application yet, but is included for a future enhancement to these two programs that will allow for the display of detail data that makes up the summary data for each row of the displayed table. The FIELD4 display field is the only one that is not protected. The MYTVALL function keys and the MYTRPT2 command menu are specified as general attributes. The former provides PF7 and PF8 for scrolling the extended table and the latter attribute presents the user with only two menu selections: HELP and END.

ENTRY: MYTRDAT PROGRAM
Display report by DATE data

***** DISPLAY *****

```
-----
Report Period: &fdate__ - &ldate__
Total Hours : &TOTL_
Date                Hours    %Totl
^^
& &date_____ &hrs_ &pct_
-----
```

***** GENERAL ATTRIBUTES *****

Function keys: MYTVALL
Command menu: MYTRPT2
System Options: EXTENDED TABLE

***** FIELD ATTRIBUTES *****

```
-----
Field name: FDATE   Length: 8
Alias: FDATE        Type: NUM
Protect: YES        Format: MMDDYY8.
Options: CAPS AUTOSKIP
-----
```

```
-----
Field name: LDATE   Length: 8
Alias: LDATE        Type: NUM
Protect: YES        Format: MMDDYY8.
Options: CAPS AUTOSKIP
-----
```

```
-----
Field name: TOTL     Length: 6
Alias: TOTL          Type: NUM
Protect: YES        Just: RIGHT
Format: 6.2          Informat: 6.2
Options: CAPS AUTOSKIP
-----
```

```
-----
Field name: FIELD4   Length: 1
Alias: FIELD4        Type: ACTION
-----
```

Options: CAPS AUTOSKIP

```
-----
Field name: DATE     Length: 17
Alias: DATE          Type: NUM
Protect: YES        Format: WEEKDATE17.
Options: CAPS AUTOSKIP
-----
```

```
-----
Field name: HRS       Length: 5
Alias: TOTHOOURS     Type: NUM Protect: YES
Format: 5.2          Just: RIGHT
Options: CAPS AUTOSKIP
-----
```

```
-----
Field name: PCT       Length: 5
Alias: PCT           Type: NUM Protect: YES
Format: 5.1          Just: RIGHT
Options: CAPS AUTOSKIP
-----
```

The source code for both of the report data display programs is basically the same with the exception of a different dataset name from which to obtain the summary data. In the INIT section, the 'rdates' dataset is opened, read and closed in order to get the report period dates for the display window. The related report dataset is then opened, read and closed to get total hours across the report period. The GETROW section of the code is called (via the SETROW routine). The GETROW section fills the extended table with data and calculates the percent of total period hours for each row of the table. The MAIN and PUTROW sections perform no tasks and the TERM section simply closes the related report dataset before ending program execution.

***** SOURCE *****

```
INIT:
  rdatid=open('work.rdates','i');
  call set(rdatid); rc=fetch(rdatid); rc=close(rdatid);
  rdlyid=open('work.dlyrpt','i');
  call set(rdlyid); totl=0;
  do while (fetch(rdlyid)^=-1);
    totl=totl+tothours;
  end;
  rc=rewind(rdlyid); call setrow(0,0,'','y');
return;
```

```
MAIN:
return;
```

```
TERM:
  rc=close(rdlyid);
return;
```

GETROW:

```
if fetchobs(rdlyid,_currow_) then call endtable();
field4=_blank_;
pct=round((tothours/totl)*100,.1);
return;
```

PUTROW:

```
return;
```

As has been previously stated, the MYTRCOD program is essentially the same as MYTRDAT with the exception of a different summary report dataset and an only slightly different display window design (with almost identical display fields). All source code is the same with some minor modifications in the INIT section. Therefore, only the display window, field attributes for the CD (code) and CDESC fields, and the INIT section of the source code are shown.

ENTRY: MYTRCOD.PROGRAM

Display report by CODE data

***** DISPLAY *****

```
-----
Report Period: &fdate__ - &ldate__
Total Hours : &totl_
```

```
Code Description      Hours  %Totl
AAA
& &cd &csdesc_____ &hrs_ &pct_
-----
```

***** FIELD ATTRIBUTES *****

```
-----
Field name: CD          Length: 3
Alias: CODE             Type: CHAR
Protect: YES            Format: $3.
Options: CAPS AUTOSKIP
```

```
-----
Field name: CDESC       Length: 25
Alias: CDESC            Type: CHAR
Protect: YES            Format: $25.
Options: CAPS AUTOSKIP
-----
```

***** SOURCE *****

INIT:

```
rdatid=open('work.rdates','i');
call set(rdatid); rc=fetch(rdatid); rc=close(rdatid);
rcodid=open('work.codrpt','i');
call set(rcodid); totl=0;
do while (fetch(rcodid)^=-1);
    totl=totl+tothours;
end;
rc=rewind(rcodid); call setrow(0,0,'','y');
return;
```

CONCLUSION

This paper has presented a basic foundation for building a Personal Time Tracking System application using SAS/AF and SAS Screen Control Language. Since no special SAS System products are used, the PTTS application can be easily ported to other computer environments that only require the base SAS System software (release 6.06 or higher). The PTTS application provides examples of custom PF keys, pmenu commands, custom commands, viewing and editing data as a table, viewing and editing data one observation at a time and using the SQL processor to create summary data of the daily log of hours dataset. Having been developed using release 6.06 of the SAS System (the first version 6 release available for the IBM mainframe environment), this application does not utilize any enhancements that have been made available in subsequent releases. For anyone that may want to adopt and develop a similar application for their own use, there is a wide variety of functions that can be added using SAS/AF and SCL depending upon user needs for this type of facility (HELP entries, drilldown on summary reporting, the ability to print data, the elimination of redundant code, and so forth). Most importantly, this paper provides a fine example of the power and user-friendliness of the types of applications that can be developed using the SAS System of software.

SAS, SAS/AF and SAS/FSP are registered trademarks of SAS Institute Inc., Cary, NC, USA.

IBM, OS/2, MVS and VM are registered trademarks of International Business Machines Corporation.