Two purposes of monitoring and reporting system utilization are to determine when the limits of hardware capacity will be met and to determine what the computer resources are being used for. These goals of capacity planning and cost allocation can be addressed by reporting the utilization statistics collected by CP in the accounting records. Other data are essential in fully understanding system utilization but the accounting data is an excellent overall first look. I have developed a set of SAS macros that make the reporting of this accounting data a relatively easy task. These macros are used to produce a set of monthly performance reports which are reviewed by systems people and high level management as well as to produce ad hoc reports as needed.

I. The Sources of Data of System Utilization

There are three data sources which report system utilization: 1.) the real time monitor or SMART 2.) VMMAP reports of CP monitor data and 3.) the accounting data. Any of these taken by itself is not a complete picture of utilization.

The real time monitor reports system utilization as it is happening in three minute time frames. This is useful in benchmarking a new application and in determining the source of a system degradation as it is happening. It cannot however give a very good report of system utilization over long time intervals

The CP monitor data reports on system utilization for longer periods than SMART and can be useful in looking at utilization over longer time frames. Monitor data can be collected each month, or whatever time frame is appropriate to the growth in utilization. These reports over time can be compared to determine growth in utilization. However, the collection of the monitor data is enabled by the CP monitor command or DMKSYS which collects data for specified time(s) of the day and for specified durations. All analysis depends on the correct choice of time(s) and duration that are typical of peak or average utilization. If the wrong times are selected because of inconsistent utilization trends, good analysis will be misleading because the data is atypical. However, VMMAP does give a great deal of detailed utilization statistics which is not available anywhere else and is essential in getting a complete picture of system utilization.

The accounting data reports overall system utilization all the time. Thus reporting this data routinely can alert you to significant changes in system utilization and can report utilization over very long or short time frames. I report utilization at least every month from the accounting data which shows: 1.) utilization by month for the twelve month period ending the current month, 2.) day by day utilization for the current month, 3.) utilization by working unit (cost center or line of business) and 4.) utilization by each project or application for the current month. These reports can help estimate how far in the future hardware capacity will be reached, who is using the system and when it is being used. These reports are generated by a set of general use SAS macros which report accounting data collected by SAS monthly.

The reports on the accounting data can also alert you to changes which can be looked at in more detail with monitor data and SMART. The accounting data is an early warning signal which is supplemented by other data when indicated.

II. The Accounting Data

The accounting data base is a hierarchical file with eight standard record types and several additional record types definable by the site. Please refer to table one for a definition of the standard record types. Each of these records has a field called account number whose value is the one assigned in the user's directory entry. This field is not very useful, since system utilization has to be reported by more specific BY groups than this. The field does not offer much more specificity than the userid field which is also present on the records. To remedy this problem a CO type record is produced which contains a four position project id. The user produces a CO record by executing an EXEC every time he begins work on a different project.

When producing reports, this project id is merged against a SAS data set which adds a great number of variables to the accounting data which allow the data to
be reported by any BY group imaginable. Reports can be done on one user for one project at any given time - or reports can be done for a cost center, line of business, work group, etc. etc.

The challenge with using this CO record is to assign the project id to all the other record types. If the world were perfect, one could simply sort the file by userid and datetime and RETAIN the project id in the CO record down through all other record types until the next CO record. The world is not perfect though and sometimes a CO record, or another record type can be missing from the file. If this happens, an incorrect project id would be assigned to utilization statistics.

It is possible to determine if a record is missing though. The 01 record should always immediately precede a CO record and also be the last record for a logon session. This 01 record contains a field indicating the total time connected to VM. So if this total connect time is subtracted from the 01 datetime stamp, the CO datetime stamp should result. That is, if the datetime stamp of the 01 record is 2:30 and the total connect time is 30 minutes, then the CO time should be 2:00. In this way a from- and to-datetime can be determined for every CO record. In the above example, the project id in the CO record should apply to all records for the user from 2:00 until 2:30. The world is even a little imperfect here too though - the 01 datetime minus the total connect time may be one or two seconds different from the CO datetime, so a fuzz value has to be implemented. But the principle remains that a datetime range can be determined for each CO record and its project id for each user.

User created SAS formats offer an excellent method of assigning a project id. The value statement can specify a list of datetime ranges and their associated project id. The syntax would look like:

\[
\text{(from datetime)} - \{\text{to datetime}\} = \{\text{project id}\}
\]

A separate format has to be created for every user in the data base, since two users could have different project ids at the same time and result in overlapping datetime ranges. Once these formats are created, the formatted datetime of any record will return the project id. Thus, separate formats are created for each user which lists the datetime ranges for each project id. Then when the accounting data is read the formatted value of any record's datetime stamp returns the project id. SAS was somewhat cumbersome in implementing this logic however.

II. Problems in coding SAS to Report Accounting Data

Problems arose when matching this CO project id to the other record types. Certain limitations in SAS made this a difficult task. The SAS user-created formats make an efficient table look-up technique. As applied to this problem the datetime stamp on the CO record and the datetime stamp on the 01 record (which occurs immediately before each CO record) make a range of time to which the previous CO project id can be applied. For example, consider the data in table two. The first CO project id should apply to all record types from 12:30:21 until 13:55:21 and the second project id should apply from 13:55:22 until 14:50:22. The two lines in the value statement should look like this:

'04APR89:12:30:21'DT- '04APR89:13:55:21'dt=x
'04APR89:13:55:22'DT- '04APR89:14:50:22'dt=y

(Where the DT literal works like the date literal - the real value statement would contain something like 908345185 - 908435813 = X). Both the creation of this kind of format and the invocation of it proved to be cumbersome in SAS - even with macros.

First, to create the formats (one per userid, to avoid overlapping times) I wanted to use "CALL SYMPUT" to create macro variables containing the start time, end time and project id and then use the macro variables to write out a PROC FORMAT step. But "CALL SYMPUT" will not allow the macro variable to be used until after the current data step has completed. Therefore, it is not possible to assign values to macro variables from values in a data set and then use those macro variables while in the same data step. So you cannot do the following within a single data step:

\[
\text{call symput(fromtime,datetim1);}
\text{call symput(totime,datetim2);}
\text{call symput(project,proj id);}
\text{&fromtime = &totime = &project}
\]

I tried letting the data step which executed the calls to symput complete and then write the code with the resultant macro variables. But three macro variables were needed for each time range that existed in the data set and this quickly used up huge amounts of memory. Macro variables like &fromtime&x, &totime&x and &project&x (where x was a simple counter
which incremented by 1 each time a
new range had to be written out) had to
be created and a two megabyte virtual
machine quickly ran out of memory.
What I finally had to resort to is to
use the PUT statement to write out the
PROC FORMAT step to an external disk
file and then INCLUDE this file back
into the original program—a very ancient
SAS technique that macros should
take care of.

The second coding technique problem
occurred during the use of this format.
A separate format had to be created for
each userid in order to avoid overlaps
time ranges. That is, two users
could have the same time range refer to
different CO project ids, but one user
never could. Therefore, the format to
be used to look up the project id
depends on the userid. SAS does not
allow the put function to accept a
character variable in the format name
position, so I could not code:

```
proj_id = put(datetime,userid.);
```

Where "USERID" is a SAS character
variable. Again "CALL SYMPUT" will not
allow use of the macro variable until
the data step completes, so I could not
assign a macro variable the value of a
character variable and use it in the
same data step as follows:

```
call symput(fmtnameJuserid);
proj_id = put(datetime,&fmtname.);
```

Again I had to use the older technique
of using the PUT statement to write out
a long series of IF.. THEN.. ELSE as
follows:

```
IF USERID = 'A' THEN
  proj_id = put(datetime,a.);
ELSE IF 'userid = 'B' THEN
  proj_id = put(datetime,b.);
ELSE IF userid = 'C' THEN
  proj_id = put(datetime,c.);
  
else proj_id = 'none';
```

This code was then INCLUDE'd in a
subsequent data step.

Of course, I realize the conceptual
issue with macros here. The macro inter-
preter must resolve macro language
before the SAS supervisor can begin to
execute the SAS code. But the lack of
data step interfaces between SAS data
variables and macro variables makes
life difficult. As does the problem
with the PUT function. The SUGI
conference in 1988 included several
papers on the good efficiency of SAS
format tables to not only decode

variables, but also to direct access
very large SAS data sets. At least one
of these papers used the technique of
creating a format that indicates
another format name to do the actual
lookup in because smaller format tables
are more efficient than larger tables.
If the SAS institute could support
these two syntaxes, these techniques
could be more frequently and easily
used.

III. A Short Description of the SAS
Macros

There are 9 macros that can be directly
used by a programmer to report accounting
statistics. Each macro conforms to
the look and feel of SAS syntax. The
macros are as follow:

```
%acntfmt Creates format library TXTLIB
%acntcnv Converts raw accounting
  records into a SAS data set
%acntupd Updates the accounting tape
  file and initializes the disk file
%acntabl Allows user-defined table to
  create additional variables
%acntabp Called by ACNTDAT to use
  table user created with ACNTABL
%acntdat Assigns project id to all
  accounting records
%acntrep Produces requested report
type on accounting data
%acntdte Called by ACNTDAT to assign
default start and end date
%acntprj Merges accounting data with
  project id data sets
```

ACNTFORMAT and ACNTCONVERT are
similar to the SYMPUT macro, and
ACNTPUT is similar to the PUT macro.
ACNTPROJECT is similar to the
PUT macro. The ACNTUPDATE macro
is similar to the PUT macro. ACNTDAT
is similar to the PUT macro.

ACNTTABLE is similar to the PUT macro.
ACNTDTE is similar to the PUT macro.
ACNTPRJ is similar to the PUT macro.

ACNTFORMAT and ACNTCONVERT are
similar to the SYMPUT macro, and
ACNTPUT is similar to the PUT macro.
ACNTPROJECT is similar to the
PUT macro. The ACNTUPDATE macro
is similar to the PUT macro. ACNTDAT
is similar to the PUT macro.

ACNTTABLE is similar to the PUT macro.
ACNTDTE is similar to the PUT macro.
ACNTPRJ is similar to the PUT macro.

ACNTFORMAT and ACNTCONVERT are
similar to the SYMPUT macro, and
ACNTPUT is similar to the PUT macro.
ACNTPROJECT is similar to the
PUT macro. The ACNTUPDATE macro
is similar to the PUT macro. ACNTDAT
is similar to the PUT macro.

ACNTTABLE is similar to the PUT macro.
ACNTDTE is similar to the PUT macro.
ACNTPRJ is similar to the PUT macro.

ACNTFORMAT and ACNTCONVERT are
similar to the SYMPUT macro, and
ACNTPUT is similar to the PUT macro.
ACNTPROJECT is similar to the
PUT macro. The ACNTUPDATE macro
is similar to the PUT macro. ACNTDAT
is similar to the PUT macro.

ACNTTABLE is similar to the PUT macro.
ACNTDTE is similar to the PUT macro.
ACNTPRJ is similar to the PUT macro.

ACNTFORMAT and ACNTCONVERT are
similar to the SYMPUT macro, and
ACNTPUT is similar to the PUT macro.
ACNTPROJECT is similar to the
PUT macro. The ACNTUPDATE macro
is similar to the PUT macro. ACNTDAT
is similar to the PUT macro.

ACNTTABLE is similar to the PUT macro.
ACNTDTE is similar to the PUT macro.
ACNTPRJ is similar to the PUT macro.

ACNTFORMAT and ACNTCONVERT are
similar to the SYMPUT macro, and
ACNTPUT is similar to the PUT macro.
ACNTPROJECT is similar to the
PUT macro. The ACNTUPDATE macro
is similar to the PUT macro. ACNTDAT
is similar to the PUT macro.

ACNTTABLE is similar to the PUT macro.
ACNTDTE is similar to the PUT macro.
ACNTPRJ is similar to the PUT macro.

ACNTFORMAT and ACNTCONVERT are
similar to the SYMPUT macro, and
ACNTPUT is similar to the PUT macro.
ACNTPROJECT is similar to the
PUT macro. The ACNTUPDATE macro
is similar to the PUT macro. ACNTDAT
is similar to the PUT macro.

ACNTTABLE is similar to the PUT macro.
ACNTDTE is similar to the PUT macro.
ACNTPRJ is similar to the PUT macro.

ACNTFORMAT and ACNTCONVERT are
similar to the SYMPUT macro, and
ACNTPUT is similar to the PUT macro.
ACNTPROJECT is similar to the
PUT macro. The ACNTUPDATE macro
is similar to the PUT macro. ACNTDAT
is similar to the PUT macro.

ACNTTABLE is similar to the PUT macro.
ACNTDTE is similar to the PUT macro.
ACNTPRJ is similar to the PUT macro.

ACNTFORMAT and ACNTCONVERT are
similar to the SYMPUT macro, and
ACNTPUT is similar to the PUT macro.
ACNTPROJECT is similar to the
PUT macro. The ACNTUPDATE macro
is similar to the PUT macro. ACNTDAT
is similar to the PUT macro.

ACNTTABLE is similar to the PUT macro.
ACNTDTE is similar to the PUT macro.
ACNTPRJ is similar to the PUT macro.

ACNTFORMAT and ACNTCONVERT are
similar to the SYMPUT macro, and
ACNTPUT is similar to the PUT macro.
ACNTPROJECT is similar to the
PUT macro. The ACNTUPDATE macro
is similar to the PUT macro. ACNTDAT
is similar to the PUT macro.

ACNTTABLE is similar to the PUT macro.
ACNTDTE is similar to the PUT macro.
ACNTPRJ is similar to the PUT macro.

ACNTFORMAT and ACNTCONVERT are
similar to the SYMPUT macro, and
ACNTPUT is similar to the PUT macro.
ACNTPROJECT is similar to the
PUT macro. The ACNTUPDATE macro
is similar to the PUT macro. ACNTDAT
is similar to the PUT macro.

ACNTTABLE is similar to the PUT macro.
ACNTDTE is similar to the PUT macro.
ACNTPRJ is similar to the PUT macro.
### Table 1.

<table>
<thead>
<tr>
<th>Record Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>User-defined record - assigns project id</td>
</tr>
<tr>
<td>01</td>
<td>General system utilization generated at logoff</td>
</tr>
<tr>
<td>02</td>
<td>Device detaches</td>
</tr>
<tr>
<td>03</td>
<td>Temporary disk detaches</td>
</tr>
<tr>
<td>04</td>
<td>Invalid logon passwords</td>
</tr>
<tr>
<td>05</td>
<td>Successful links</td>
</tr>
<tr>
<td>06</td>
<td>Unsuccessful links</td>
</tr>
<tr>
<td>07</td>
<td>VTAM information</td>
</tr>
<tr>
<td>08</td>
<td>Logoff/disconnect terminal address</td>
</tr>
</tbody>
</table>

### Table 2.

<table>
<thead>
<tr>
<th>Record Type</th>
<th>Description</th>
<th>Date time stamp</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>Project id X</td>
<td>04APR89:12:30:21</td>
<td>NA</td>
</tr>
<tr>
<td>03</td>
<td>Temp disk detach</td>
<td>04APR89:13:30:21</td>
<td>1 Hour 0 Mins</td>
</tr>
<tr>
<td>02</td>
<td>Device detach</td>
<td>04APR89:13:50:21</td>
<td>1 Hour 20 Mins</td>
</tr>
<tr>
<td>01</td>
<td>Logoff summary</td>
<td>04APR89:13:55:21</td>
<td>1 Hour 25 Mins</td>
</tr>
<tr>
<td>C0</td>
<td>Project id Y</td>
<td>04APR89:13:55:22</td>
<td>NA</td>
</tr>
<tr>
<td>03</td>
<td>Temp disk detach</td>
<td>04APR89:14:40:22</td>
<td>0 Hour 10 Mins</td>
</tr>
<tr>
<td>01</td>
<td>Logoff time</td>
<td>04APR89:14:50:22</td>
<td>NA</td>
</tr>
<tr>
<td>02</td>
<td>Logoff summary</td>
<td>04APR89:14:55:22</td>
<td>0 Hour 55 Mins</td>
</tr>
</tbody>
</table>