What is the best way of controlling the output generated by a program? The answer to this question is simply - by controlling the input.

Input in its simplest form consists of data and program parameters. Assuming the data is valid (which is a very great assumption indeed), we then need to concentrate on insuring the validity of the program parameters. If we can insure that only valid parameters are passed to the executable code by means of a parameter filter, we can then be relatively sure that the output generated will also be valid.

Error Routine Objectives:

The following objectives were defined and adhered to during the development of the new Consistency Error Checking Routine:

1) Check the validity of input for each user variable.
2) Check combinations of the above input values.
3) Assign default values.
4) Automatic continuation when all criteria has been met.
5) Allow one consistent method of execution (ENTER key).

The result obtained by keeping the above objectives in mind was an error routine which would assign default values to the variables, check for errors, and if no errors were found, would issue an END command. The program portion of the module would then be executed. However, if an error was detected, an alarm would sound, an error comment would be displayed, and the cursor returned to the point of the error.

Error Routine Structure:

The basic structure of the new error routine consists of three parts. They are 1) the variable initialization loop, 2) the user variable validity check, and 3) the error comment check.

The variable initialization loop will be executed only the first time into the module. Within the do loop are %LET statements which assign default values or a null value to each of the user variable fields and to any other macro variables used within the routine. The assigned value can either be a global macro variable or a constant value, or null. Once these default values have been set, they can be changed by simply typing a new value into the user field. If using a global macro variable, it must have been defined in a module prior to entering the current module. By assigning a null value to a user variable, any values previously assigned to the user variable will be flushed.

Once the default values have been assigned, the screen has been displayed, and the ENTER key pressed, the user variable validity check is processed. This is the major portion of the error routine and may contain as many checks as are necessary to insure the validity of the program parameters. When checking for the user variable values, use the %QUOTE and %SUPERQ functions to prevent undesired system effects from occurring when the value of the user variable is blank. The first check should be for the value of the macro variable CONTINUE. If its value is blank, the validity check will be processed. If the value of CONTINUE is not YES, it means that either it is the first time through the routine, or that an error had been found in the last pass through the
routine. However, if the value of CONTINUE is YES, it
means the error routine has been executed once
and no errors were found. This flag is set in
order to prevent the routine from being executed
again. Without the flag, the routine would not
only be less efficient, but it would get caught in
an infinite loop due to the re-issuance of the END
command.

Following the check for CONTINUE, check for
the presence of all the required variables.
Because of the automatic continuation portion of
the routine, the LIST and REQUIRED variable
attributes on the attribute screen cannot be used.
The next check should be for valid values for each
user variable field, and finally for the
combinations of the values of the user fields. The
use of %ELSE %IF causes the system to drop past
following %ELSE %IF statements to the next %IF
statement once a condition has been found to be
true. If an error is detected, two variables are
defined to flag the error condition. The first
variable called ERRCOM is defined with an error
comment as the name of the user
variable where the error occurred. This variable
determines the placement of the cursor. Both of
these values are reinitialized at the beginning to
the variable validity check portion of the routine.

The final portion of the error routine is the
error comment check. It consists of two 'do'
loops. The first loop executes if ERRCOM does not
contain a null value. This will be true only if
the ERRCOM variable had previously been set when
checking for errors. If executed, this loop sets
certain automatic system variables; DALAR= equal to
X to sound the alarm, _DCURSOR equal to APLACCUR
to place the cursor at the point of error, and
_DENSG equal to &ERRCOM to display the error
comment. The second do loop will execute if the
value of ERRCOM is null. It sets the variable
CONTINUE equal to YES, and _DCMD equal to END. By
setting _DCMD equal to END, the user can
consistently press one key (the ENTER key) and let
the error routine control the execution of the SAS
code. The Cancel command can still be used to exit
any module.

Variable Attributes:

The definition of each user variable is very
important for this routine to function as intended.
The variable type must be defined correctly (as
ACTION, CHAR, or NUM), the Caps option should be
flagged for easier error checking, justification as
necessary, field protection and non-display
attributes flagged as appropriate, and the SAS
macro variable defined. Keep in mind that it is
the SAS Macro Variable name which is used in the
Error Routine, and not the user variable name. To
avoid confusion on this issue, it is best to define
the SAS Macro Variable name the same as the user
variable name. SAS automatically checks for the
variable type and issues an error comment if the
user input does not match the variable data type.

As previously mentioned, the REQUIRED and LIST
fields should not be flagged when using the new
Consistency Error Checking Routine. Use 'if'
statements to check for valid user variable values.

Example:

The following example serves as an
illustration of the new Consistency Error Checking
Routine. The screen variables are &XMODE, &PRNTR,
&COPY, and &CONTINUE. The variable attributes for
each of the variables are:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Caps</th>
<th>Just</th>
<th>Pad</th>
<th>Protect</th>
<th>Non-display</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMODE</td>
<td>Char</td>
<td>X</td>
<td>L</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>PRNTR</td>
<td>Char</td>
<td>X</td>
<td>L</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>COPY</td>
<td>Num</td>
<td></td>
<td>L</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>CONTINUE</td>
<td>Char</td>
<td>X</td>
<td>L</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

All but the variable &CONTINUE are user
variables. It serves as a flag for continuation
and is not available for input from the user, nor
is its value displayed on the screen. The only
valid input values for the variable &XMODE are
'BATCH' and 'ONLINE'; the valid input values for
&PRNTR are 'SYSTEM' and 'LOCAL'; and the valid
range of input values for &COPY is 1 to 10. All
variables are required. Valid combinations of the
user variables are the following:

<table>
<thead>
<tr>
<th>XMODE</th>
<th>PRNTR</th>
<th>COPY</th>
</tr>
</thead>
<tbody>
<tr>
<td>BATCH</td>
<td>SYSTEM</td>
<td>1 - 10</td>
</tr>
<tr>
<td>ONLINE</td>
<td>SYSTEM</td>
<td>1 - 10</td>
</tr>
<tr>
<td>ONLINE</td>
<td>LOCAL</td>
<td>1</td>
</tr>
</tbody>
</table>

The first step in the routine is to assign
default values to the variables. The only default
value is for 1 copy of the reports. The next step
is to check for all required fields. Recall that
this is accomplished by using the %QUOTE, %SUPERQ,
and %STR macro functions. Be sure not to use the
supermark within the %SUPERQ function, as an error
will occur. If testing for the presence of a value
of a user variable, be sure to include a space
between the parentheses in the %STR function.

After the routine checks for values for each
of the required variables, it checks for valid
input for XMODE, PRNTR, and COPY respectively.
Next, the routine checks for combinations of user
variable values as defined above. If no errors are
found, the ERRCOM macro variable continues to be
null, the routine sets CONTINUE to YES, and the END
command will be issued, allowing the program
portion of the module to be executed. If an error
is found, ERRCOM is defined with an error comment,
PLACCUR is set to the name of the variable for the
cursor to be placed, and the system falls through
to the check for ERRCOM. If its value is not null,
the error comment is displayed on the screen, the
alarm sounded, and the cursor placed at the point
of error.
**NEW CONSISTENCY ERROR CHECKING ROUTINE EXAMPLE**

**SYSTEM PARAMETER INPUT SCREEN**

You are now executing the XXXX system.

In order to proceed, you must enter the information below.
The parameters entered will be used to generate reports throughout the current session.

- **Method of Execution:** Remote
  - (Batch or Online)
- **Destination Printer:** Appear
  - (System or Local)
- **Number of Copies:** Remote

**SAS Code or %INCLUDE**

```sas
%check;
SAS Code or %INCLUDE
Prot Display;
Run;
```

**Development Problems:**

As with all application development, several problems were encountered during the development phase. Below are five such obstacles with the solutions that were found to be appropriate.

1. **Spaces in the macro function %STR().**
   - **Solution:**
     - If the variable is a user variable, include a space between the parentheses.
     - If the variable is not a user variable, do not include a space between the parentheses.

2. **No log of the execution of the Error Routine for debugging.**
   - **Solution:** Place %PUT statements describing its point of execution throughout the Error Routine.

3. **Caught in an infinite do loop when the value of a user variable is:**
   - **a) Missing when the REQUIRED attribute field for the user variable is flagged.**
   - **b) Invalid according to the LIST of valid entries according to the user variable attributes.**
   - **Solution:**
     - Do not use these attributes, and include checks for missing or invalid input in the Error Routine.

4. **More efficient execution.**
   - **Solution:**
     - Check for the value of &CONTINUE before performing the error checking.
     - Make logical use of the %ELSE %IF statement when checking for errors.

5. **Incorrect reference to the user variable macro name.**
   - **Solution:** Use the same name for both the user variable and its associated macro.

**Conclusion:**

The SAS/AF software is a fine user interface tool, and when used with the Enhanced Consistency Error Checking Routine will insure valid user input, resulting in valid output from the application.