

VMS® Environment and SAS® Language Applications: Getting the Best of Both Worlds

M. Rita Thissen, Research Triangle Institute, Research Triangle Park, NC

This paper reviews various features of the VMS operating system and its companion command language with respect to developing and implementing batch SAS programs. Specific examples include moving information between VMS symbols and SAS macro variables, referencing libraries or files through logical equivalence names, and accessing data in indexed ASCII files. Program examples are drawn from survey data processing applications.

INTRODUCTION

With efficiency goals in mind, programmers need to take advantage of all available tools to make data processing as reliable and manageable as possible. When routine and repetitive tasks are implemented as batch processes, they make better use of time and resources, raising productivity. The VMS and OpenVMS operating systems offer a variety of features to SAS language programmers for developing and controlling batch jobs. Applications programmers can adapt these features to their specific needs.

One area which relies heavily on batch processing is survey data management, where large files and cyclic operations of updating, editing and reporting are the norm. After the database structure and normal operations have been defined, data collection may proceed for years. National longitudinal surveys, which are designed to measure societal changes, accumulate large databases that must be preserved through generations of hardware and software, while the tasks to be performed periodically by the applications may remain almost unchanged. Batch processing techniques, SAS software and VMS capabilities can be combined to ensure consistency of operations and

accessibility of data over the life of the survey.

Both the SAS system of data management and the VMS operating system offer stability and provide a high-level programming language. The stability of the SAS system, with backwards compatibility in new releases and availability for a wide variety of platforms, provides programmers with a way to span changes in the computing environment smoothly. The VMS operating system's strengths also include stability and a versatile command language. Combining the multiplatform capabilities of the SAS system with the individual strengths of the operating system makes the best of both environments and produces code that remains usable for long periods of time.

SURVEY DATA PROCESSING

Survey data management includes a variety of computational activities relating to various data types. Responses to questionnaires form one component of data produced or required by a major survey. Other supporting data may include census or other population profiles, sampling information, respondent demographics, interviewer expense data and event tracking. Each

type of data contributes to the survey processes in some way.

The data management operations which generate or use this information can be grouped into a few major stages. Table 1 condenses survey activities into preliminary, intermediate and final operations. Initial efforts consist of collecting information about the target population and choosing the sample. Intermediate stages of administering the survey constitute the bulk of the activity, including the accumulation of questionnaire responses and management of the progress of the project. Ultimately, the collected information must be converted to its final form for presentation. These stages may occur concurrently or sequentially, depending on the design of the project.

Table 1. Survey Data Processing Stages

- Collection of information about prospective respondents
- Selection of sample population
- Processing responses to survey questions
- Tracking data flow
- Monitoring expenses
- Evaluating and providing feedback on field staff performance
- Compiling, analyzing and delivering survey results

A variety of hardware and software tools may be needed to support these operations. Data from the field may be collected through World Wide Web sites, computer-based telephony, computer-assisted personal interviewing or traditional pencil and paper methods. In any case, the data are assembled, checked and stored at a central site, often through automated or partially automated batch

processes on a powerful midrange computer.

Due to the longevity of some surveys, especially federally mandated national efforts, previously developed applications from various software vendors must be maintained as part of the data processing arrangement. While SAS language applications may be capable of performing most of the computerized data processing activities needed by a survey, in practice, most survey centers use a combination of data management tools.

BATCH PROCESSING

In what ways can an operating system contribute to satisfying data management needs? An operating system controls the execution of jobs, stores information on a transient or permanent basis and provides a command language by which a programmer can influence its operations. Whether an application is interactive or run as a batch job, the operating system manages the movement of data and scheduling of resources for it. In batch processing, the actions of the operating system are more visible than in interactive sessions, and the programmer can tailor the sequence of operations to suit the requirements of the task. When writing a batch application, the programmer has full access to the command language, scheduling capabilities and other features of the operating system.

The choice between interactive or batch mode may depend on many aspects of the task at hand. Interactive programs are preferable in some circumstances, and batch jobs are advantageous in others. The paragraphs below discuss some of the considerations which might lead to

the selection of batch processes over interactive ones.

QUANTITY OF DATA. There may be no need to automate the operation of a small database, but those containing many megabytes of data require standardized processing routines in order to ensure consistency of treatment and timely completion. In some cases, the volume of processing corresponds to the size of the database. Since most tasks which can be reduced to a finite number of defined steps can be performed many times faster by a batch process than through an interactive session, the savings can be extensive for large databases. This is especially true if the duration or frequency of the task is proportional to the quantity of data.

DEGREE OF DATA

MANIPULATION. Batch processing may be superior to interactive work if lengthy or complex data manipulation is needed. The use of batch jobs may reduce vast amounts of data to more usable summaries, subsets or displays without making extensive demands on staff time.

REPETITIVE FUNCTIONS. Large dynamic databases, such as those found in survey research, undergo frequent updating with new information. The actions of adding or changing the data, generating reports, checking data consistency and other routine tasks can be performed in off-hours or at scheduled intervals by batch processes. Batch jobs can perform these repeated operations quickly and reliably.

SEQUENTIAL OPERATIONS. Batch jobs can be useful in controlling sequences of processes as well as individual actions. When applications which must be run in strict order, it may be desirable to handle their execution

through a single controlling program, rather than by manual release. Use of the operating system's scheduling capabilities may make management easier and more reliable.

TRIGGERED PROCESSES. Some periodic operations can be automated as triggered programs. For example, a program may be designed to run continuously as a batch job to check on a regular basis for the arrival of a transmitted file; when the file is found, the monitoring program can submit a subprocess which reads the transmitted file and updates a database. This and other intermittent tasks may be performed well under batch job control.

VMS (OpenVMS) FEATURES

This discussion focuses on SAS language applications under the VMS operating system, as they are used in survey processing. Some of the specific features of VMS and its command language are described, indicating how they may be employed to advantage by program developers.

The VAX and AXP family of computers are run under VMS or its newer incarnation, OpenVMS. Although the underlying nature of the two machines differs greatly, to the user they may appear to be identical because the interfaces are similar. In each, batch system support includes the Digital Command Language (DCL), symbol tables, logical equivalence names (logicals), indexed sequential files, job queues, job scheduling and a group of lexical functions for accessing or working with system-level information. The SAS system operates under both operating systems and offers host-dependent extensions for using the power of the special features. In this paper, the name

VMS will be used to refer to both VMS and OpenVMS, since the features discussed are consistent between the two.

Through DCL and job scheduling, VMS allows a programmer to automate a complex series of tasks, including applications developed under multiple software systems. Table 2 gives a generic outline for a batch job (the controlling program), and Table 3 provides a related outline for an application invoked by the batch job (the subprocess). The programming examples presented later in this paper, Programs 1 and 2, follow the outlines given in these two tables.

Table 2. DCL Program Outline

1. Check parameters for validity, if any were passed to the program.
2. Resubmit the job, if it is scheduled for periodic processing.
3. Define symbols and logicals to be used by subprocesses.
4. Release application programs (may be conditional, based on passed parameters, timing, etc.).
5. Transfer information between programs, as needed.
6. Distribute results, as needed.
7. Send notification of errors, completion status, etc.

Command files commonly include the items listed in Table 2. The first item is a safeguard. If the job has received any information in the form of parameters, it may be advisable to check their validity before proceeding. Item 2 refers to the ability of a command file to submit any other job, including another copy of itself, which is useful when a job runs on a regular basis. The third and fifth items relate to handling global information for use by the subprocesses, whether as

stored symbols, defined logicals or passed parameters. Following the release of applications, the job distributes results and notifies staff of completion. These last two tasks may be accomplished through release of another application or through the messaging process available through the operating system itself. Altogether, these steps compose the basic skeleton of a batch command file.

A similar outline of program steps is given in Table 3, which describes the parts of a SAS application program. Command files and applications may vary considerably in scope, from simple to elaborate, by using these basic components in more complex ways. In this way the applications programmer can use the language features to standardize and automate the survey management process.

Table 3. SAS Application Program Outline

1. Retrieve symbols or parameters.
2. Assign data library references through LIBNAME or FILENAME statements and VMS logicals.
3. Access data, optionally through use of VMS logicals in 2-level names.
4. Perform application-specific processing.
5. Store information in symbols or logicals, as needed.
6. Generate output.

Table 3 describes the components of an application running as a subprocess under a batch job. It first retrieves global information which was stored as symbols by the command file or passed to the application as parameters. Other information from the parent process may be used directly, such as logicals being used in LIBNAME statements to indicate

the location of data files or destination of output. After the computational body of the program, communication with the parent process can occur through storage of information in the symbol table or logical names. Output may be produced in various forms, including data files or listings. The steps shown in Table 3 give a very brief overview of the total application.

The command file and application programs have different purposes in the data management process. The command file manages global information for the application set, controls the sequencing of the applications, checks for error conditions and distributes information. DCL is powerful enough to perform many calculations or data manipulation functions, but from a program management viewpoint, it is better to contain all data processing within the applications programs. The purpose of the application programs is to retrieve and use the global information, perform computational steps and produce the information needed by other programs or by users.

The sample programs below provide typical usage for some of the host-specific operations available to SAS programmers under the VMS system. While the sample batch process shows only SAS applications and DCL, programs in other languages could be included easily. Together, the DCL and SAS examples, Programs 1 and 2, illustrate the usefulness of the VMS operating system to applications programmers.

VMS provides many useful features which may be employed in command files and through application programs. Some of these tools are listed in Table 4, with an indication of the SAS language

statements that access them. Many additional VMS commands and functions have a counterpart in the SAS language. For more information, refer to the *SAS Companion for the OpenVMS Environment, Version 6, Second Edition*, or to the classic DCL guide, *VMS User's Manual*.

Table 4. Selected VMS and SAS Language Features

<u>VMS Feature</u>	<u>SAS Language Access</u>
Symbol storage	GETSYM () PUTSYM ()
Logicals	2-Level naming LIBNAME FILENAME GETLOG ()
Indexed Files	FILE INFILE FILENAME
DCL	X VMS () SYSTEM ()
Lexical functions	FINDFILE ()
System data	&VMSSASIN &SYSDATE &SYSRC

FIELD PERFORMANCE GOALS: A SURVEY MANAGEMENT EXAMPLE

One way to encourage better overall performance in a survey group is to provide group members with frequent quantitative feedback and realistic

quantitative goals. The programming example presented here is drawn from a quarterly process of (1) evaluating regional trends in past performance, (2) determining the level of regional interviewing effort needed to achieve national survey goals, and (3) expressing those goals in terms of the number of completed cases expected for each week of the quarter for each group of interviewers. The goal-setting process is linked to periodic reports on performance.

The programs detailed here use prior performance data, current staffing data, and contractual obligations that define interview completion goals. They compute, report and distribute new weekly goals for each group of interviewers. The use of VMS features in DCL and SAS language statements will be explained in that context. *Italic letters have been placed beside lines of code in the command file (Program 1) and application program (Program 2) to indicate the topic being discussed.*

DCL BATCH PROGRAM

(a) QtrGoals.com, is designed to be submitted to a VMS batch queue with three parameters. Values for these might be 3, 96, 4600, to indicate the third quarter of 1996, with a completion goal of 4600 interviews. These parameters are referenced within the DCL code as p1, p2, and p3, by VMS naming convention.

(b) The opening lines of the command file are standard operations for almost any batch job. The first two executable lines turn on the VMS error-checking facility and cause the remainder of the DCL program to be bypassed at any point if an error condition occurs. This operation is similar to that of the SAS language statement `OPTION ERRORABEND`. `SET VERIFY` causes

the executed statements to be listed in the DCL log file. When parameters are expected by a program, it makes sense to verify their existence. More elaborate validation could be included, if desired, and additional lines of code could be added here to resubmit the job for its next scheduled run.

(c) This section of QtrGoals.com defines logicals for various directory locations. The logicals can be used in place of the full pathname in the program or in any subprocess spawned by the command file, such as within the application programs. Logicals are a convenient way to designate or change directories from one project to another or from test data to production data, especially when the directory structure is a many-layered hierarchy.

(d) The parameters passed to the command file are stored as symbols for easy retrieval by subprocesses and to provide more descriptive names. A constant value, the overall historical response rate, is stored as well, because it may change from one project to another but generally does not change by quarter as the other parameters do. The response rate is stored as a character string, because VMS numeric symbol storage only accepts integer values. The information stored through the command file will be accessed within the SAS application through reference to the symbol names.

(e) Having handled the global information, QtrGoals.com launches the SAS application, QtrGoals.SAS. This application forms the heart of the batch process, and so it is named consistently with the command file to indicate the relationship between the two. In short, it calculates quarterly yield values from the global information and other existing input data files. The results of this

program are written to external files, which are accessed by the next SAS application in this set, and also by programs written for another database management system.

(f) When the first SAS language application has completed execution, a second one is started. Note that these programs are run in the SAS batch mode, but are not sent to a VMS batch queue. By running them directly, their order is maintained—the second one does not start until the first one has finished. If the first had terminates with an error condition, the remaining parts of the command file are bypassed, except for sending the termination message.

The second SAS program, GoalsRpt.SAS, is maintained separately from QtrGoals.SAS, so that it can be re-run independently if the report needs to be regenerated later in the quarter. GoalsRpt retrieves data stored by QtrGoals.SAS and formats them into a report of weekly completion figures. Note that the quarter information is passed from the command file to the subprocess through the SAS system option SYSPARM, an alternative method of distributing global information.

(g) The weekly and quarterly figures are e-mailed to on-site staff for review, and a program is submitted for e-mailing to field supervisors around the country. The HOLD option on the SUBMIT statement prevents the subprocess from beginning execution, pending intervention by the programming staff. If the goals are approved, the final application is released through a simple interactive command and mailing proceeds automatically. If the goals are not approved, the job entry is canceled, adjustments are made to the input data, and the process is begun again.

(h) The DCL file finishes by sending a message to alert the appropriate programming staff member if a problem was encountered or to indicate that the job completed normally.

The steps described above rely only on DCL commands and the VMS system. The applications launched by a command file may be in any language, including other DCL command files. They may be conditionally released, monitored for triggering events, or released to VMS batch queues, depending on the needs of the data processing operation. The next discussion will show how the SAS application program, running as a subprocess, makes good use of the VMS features.

Program 1. QtrGoals.com, a DCL command file.

```

$! -----
$! Name:      QtrGoals.com
$!
$! Purpose:  Computation of
$! regional yields for the
$! quarter (SESUG example)
$!
$! Author:    Rita Thissen
$! Date:      July 1996
$!
$! Usage:     Update FS_YIELD.DBS
$! and FS_GOALS.DBS before
$! running this program. Submit
$! with the following parameters
$!   p1 = 1-digit quarter      (a)
$!   p2 = 2-digit project year
$!   p3 = total number of
$!         interviews needed for
$!         the quarter, with no
$!         embedded comma
$! For example
$! submit/noprint/keep -
$!   /param=(3,96,4600) qtrgoals
$!
$! Notes:     This program submits
$! SAS applications QtrGoals and
$! GoalsRpt.
$! -----
$!
$ set on      ! error checking (b)

```

```

$ on err then goto abort
$ set verify      ! write to log
$!
$! check parameters
$ if 'p1'.eqs."" .or. -
  'p2'.eqs."" .or. -
  'p3'.eqs."" then goto abort
$!
$! resubmission could go here
$!
$! define logicals for source
$! data and applications      (c)
$ define sas_lib -
  disk48:[rthissen.sasdata]
$ define pgm_lib -
  disk48:[rthissen.sesug]
$ define fics$sysl -
  disk48:[rthissen.fics]
$ define fsrs$dir -
  disk48:[rthissen.fics.q'p1']
$ set def pgm_lib
$!
$! store parameters as symbols(d)
$ qtr = 'p1'
$ yr = 'p2'
$ budget = 'p3'
$! from MCB calculations
$ resp = "0.8038"
$!
$! start the SAS program to
$! compute yields and goals      (e)
$ sas qtrgoals
$!
$! collect the new info into
$! standard report format      (f)
$ sas/sysparm="Q'p1'" goalsrpt
$!
$! mail the results for approval
$!                               (g)
$ mail -
/su="Qtr 'p1' goals for review"-
Q'p1'_goals_report.txt -
rthissen,rtnhsda
$!
$! submit a program on hold to
mail results to FSs, if approved
$ submit/hold/que=sys$batch -
/param=('p1') mail_goals      (h)
$!
$ goto done
$!
$! error conditions detected!
$ abort:
$ mail/su="QtrGoals
terminated abnormally" nl:
rthissen
$ set noverify
$ exit

```

```

$!
$! normal termination
$ done:
$ mail -
/su="QtrGoals finished normally"-
nl: rthissen
$ set noverify
$ exit

```

BASE SAS APPLICATION

The points below detail access to the VMS features by the SAS application (Program 2). The program is started as a subprocess, with symbols and logicals pre-defined by the parent process, some data files already available in other directories, and a reporting program waiting for results. The host-specific SAS language extensions are annotated by letters in the margin.

(i) The program opens with a DATA _NULL_ step to retrieve the values which were stored as VMS symbols. Base SAS provides two functions, GETSYM, which reads values from symbols, and PUTSYM, which stores values as symbols. The values are moved into macro variables, for convenient usage within the body of the program.

(j) The filename statement makes use of the logical name, FICSS\$SYSL, instead of the full pathname of the designated file. The logical name can be used inside single or double quotes.

This filename statement exhibits two other host-specific language extensions, by referencing a VMS index on the data file through the KEY option, and by controlling access through the FAC and SHR options. FAC indicates the level of access which the program needs with respect to the data. In this statement, it has read (GET), write (PUT) and delete (DEL) access. Concurrent access is supported by the SHR option, which

must match the values of the FAC option, if they are used together.

There are several supported file types under VMS, including standard ASCII (flat) files and indexed sequential files. This type of indexed file is not the same as an indexed SAS data set; it can be viewed through editors or other ASCII text readers, and it may be indexed on multiple key fields simultaneously. The index structure is specified in a separate file with the extension .FDL. More details about File Definition Language (FDL) files are provided in the *VMS User's Guide* as well as through the VMS on-line help facility under CREATE, ANALYZE or CONVERT. Several data file options in the SAS language relate to indexed sequential files. The KEY= option on the FILENAME indicates which of the multiple keys to use and a companion KEYVALUE option can be used to subset the data according to the index values.

The options may be used on the FILENAME statement, as they are here, in which case they apply to all subsequent uses of the fileref, or they may be used on the FILE or INFILE statements, in which case they refer only to the relevant data step. If used in both statements, the FILE or INFILE options overrule those on the FILENAME statement.

(k) The SAS language for VMS also provides additional automatic macro variables, including &VMSSASIN which is shown in the TITLE2 statement. This variable provides a quick reference to the full pathname of the current application, a feature which is especially convenient when similar output can be produced by different programs, or when testing software in a separate application library.

(l) Within a SAS application, there are several ways to issue commands to the operating system. The X command acts the same in batch programs as it does in interactive usage, interrupting the program to spawn a subprocess. The command following the X will be acted upon by VMS. If special characters are included, the command must be enclosed in double quotes. The X command can be used for assigning or reassigning logicals, whose new definitions will be recognized by the parent process. However, symbol values assigned through the X command exist only for the duration of the spawned subprocess and will not be accessible to the remainder of the SAS application. Instead of the X command, the PUTSYM function should be used to write values to the VMS symbol table.

In this case, the X command is used to create an empty copy of an indexed sequential file, FS_YIELD.DBS, with the indexing characteristics specified in a related file, FS_YIELD.FDL. Both are stored in the directory specified by the logical name, FICSSSYSL. The indexed file is used here instead of a SAS dataset or flat file so that the information can be accessed by another database management system without requiring any conversion steps.

(m) Note that a two-level data set name is specified in this data step, but the first level does not appear in a LIBNAME statement. The first part is, instead, a VMS logical name. As long as the logical name conforms to SAS naming conventions of length and alphanumeric character usage, it can be used interchangeably with a libref defined in a LIBNAME statement.

(n) Another method of using the logicals is displayed in the file statement, in which the name of the output file is

specified by a logical pathname and literal filename. The file in use here is the one created above with an index, and the data written to it will be indexed automatically.

While this SAS program is not complex, it focuses on those operations which benefit most from the VMS system. Each of the annotated items might be handled in a different way under different circumstances, but the code shown here is easy to use, easy to maintain and takes advantage of the host-specific extensions where they are useful.

Program 2. QtrGoals.SAS, a SAS Language Program

```

/*****
* Name: QtrGoals.SAS
*
* Purpose: To generate yield
* values and expected numbers
* of interviews per week
* which add up to the
* budgeted contractual
* goal.
*
* Usage: This program is
* released by QtrGoals.com,
* which assigns values to VMS
* symbols resp (response rate),
* qtr (quarter), yr (year) and
* budget.
*
* Author: Rita Thissen
* Date: July 1996
* Note: SESUG example
*****/
option errorabend;

* retrieve parameters from symbol
table and store them as macro
vars; (i)
data _null_;
  length q $1 y $2 b $6 r $6;
  q = getsym('qtr');
  y = getsym('yr');
  b = getsym('budget');
  r = getsym('resp');
  call symput('quarter',q);
  call symput('year',y);
  call symput
    ('budget',b);

  call symput
    ('rr',r);
run;

filename yields (j)
  'fics$sys1:fs_yield.dbs' key=0
  fac=(get,put,del)
  shr=(get,put,del);

*calculate how many interviews
would be completed without
adjusting any values;
data qtryield;
  length dwunits yield inttot 8
  fsid $2 ;
  retain inttot 0;
  infile yields;
  input @1 fsid $2.
        @3 dwunits 4.
        @7 yield 4.2 ;
  * inttot is the cumulative
  expected number of interviews;
  inttot = (dwunits/yield) +
    inttot;
run;

* record the starting data as a
paper trail;
proc print data=qtryield;
title "Input data and calculated
total number of interviews for
&year Q&quarter";
title2 "Source program: (k)
&vmssasin";
run;

data _null_;
* compute a multiplier to bring
the total number of interviews
to what is needed for the
quarter;
length adjust 8;
set qtryield end=eof;
if eof then do;
  * the adjustment factor is
  the ratio of the desired
  number to the total expected
  number;
  adjust = &budget /
    (inttot*&rr);
  call symput
    ('adjust',put(adjust,best.)
    );
  put "Note from QtrGoals:
    adjustment factor = "
    adjust;
end; *if eof;
run;

```

```

* create an empty indexed file to
hold the new data;          (1)
x create
  /fdl=fics$sysl:fs_yield.fdl
  fics$sysl:fs_yield.dbs;

*write new data to the indexed
file, and create a permanent
SAS dataset with the same info,
for use by the reporting
program;
data sas_lib.yields;          (m)
set qtryield (drop=inttot);
file 'fics$sysl:fs_yield.dbs'
  mod;                          (n)
yield = yield / &adjust;
put @1 fsid $2.
   @3 dunits 4.
   @7 yield 4.2 ;
run;

proc print data=sas_lib.yields;
title "Adjusted yields for &year
Q&quarter";
run;

```

SUMMARY

In many areas of research and industry, applications programmers must develop systems which are influenced by available hardware, software and data management practices. To balance the workload with respect to staff and system availability while simultaneously producing reliable and flexible applications, programmers may take advantage of features of the operating system as well as the applications language. A batch processing environment can harness the strengths of the available systems, especially when the volume and rate of data flow is large. The discussions in this paper present some of the requirements of survey data management and show how the strengths of the VMS operating system and the SAS software system can be employed to meet them.

REFERENCES

SAS Institute Inc. (1996) *SAS Companion for the OpenVMS Environment, Version 6, Second Edition*, Cary, NC: SAS Institute Inc.

Digital Equipment Corporation (1989) *VMS User's Manual*. Digital Equipment Corporation. Maynard, MA

ACKNOWLEDGMENTS

The author wishes to thank Jane E. Mudano, Philip Cooley and G.G. Frick for their helpful comments during preparation of this paper.

SAS is a registered trademark or trademark of SAS Institute in the USA and other countries.

® Indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

The author can be reached at rthissen@rti.org.