JANUARY/JUNE 1982
VOL. 5, NO. 1,2

JOURNAL

OF THE HP 3000 INTERNATIONAL USERS GROUP, INCORPORATED

JOURNAL
OF THE HP 3000 INTERNATIONAL USERS GROUP, INCORPORATED
JOURNAL
OF THE HP 3000 INTERNATIONAL USERS GROUP, INCORPORATED
JOURNAL
OF THE HP 3000 INTERNATIONAL USERS GROUP, INCORPORATED
JOURNAL
OF THE HP 3000 INTERNATIONAL USERS GROUP, INCORPORATED
JOURNAL
OF THE HP 3000 INTERNATIONAL USERS GROUP, INCORPORATED
JOURNAL
OF THE HP 3000 INTERNATIONAL USERS GROUP, INCORPORATED
JOURNAL
OF THE HP 3000 INTERNATIONAL USERS GROUP, INCORPORATED

CONTENTS

JOURNAL
OF THE HP 3000 INTERNATIONAL
USERS GROUP, INCORPORATED

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

JOURNAL
OF THE HP 3000 INTERNATIONAL USERS GROUP, INCORPORATED

The Change from MPE III to MPE IV on a Series III from a System Manager's Point of View

Dr. Bjorn Dreher
Institute for Nuclear Physics
Mainz University
D-6500 Mainz,
West Germany

Introduction

Our institute is a medium sized research institute in the field of basic research in nuclear physics. We have used an HP 3000 system since 1976, at that time an HP 3000 Series II, today an HP 3000 Series III.

Our main applications are scientific calculations (number crunching!) and the development of such programs. Besides that, several HP1000 systems and other minicomputers are attached to the HP 3000 as front-end processors to perform various real-time tasks for experiment control and data acquisition. The HP 3000 supplies its resources to the front-ends via our own communications software (we started when DS3000/DS1000 did not yet exist). Therefore we have much experience with the HP 3000 and with MPE.

Since August (1981) we have run MPE IV (C.00.01) and are experiencing - as a previous MPE III user - various, sometimes unexpected, differences in the system's behaviour as compared to MPE III. In general, MPE IV certainly is a major breakthrough in the HP 3000's operating systems, in particular regarding efficient performance and reliability. Most readers probably know the new key features of MPE IV. Let me give just two examples from our experience:

- Access to disc files in default mode (buffered) needs only 50%-60% of the CPU time that was needed under MPE III. This makes most disc I/O operations much more effective.
- It is not unusual that the system runs for six weeks and longer without any problems. That is to say, MPE IV is today already a very stable system.

My remarks that follow in this paper, mainly about the new scheduling algorithm, should be taken as a positive criticism, hopefully as input for an even better MPE in the future.

MPE III Scheduling

Let us go back somewhat and recall how the MPE III scheduler used to work. In MPE III we had a fixed time slice for all processes, whether they were SESSIONs (normally in the CS queue) or JOBs (normally in the DS or ES queue). The differences in the three queues were that the processes could drop differently deep in priority (ES deeper than DS, and DS deeper than CS) and ES processes fell faster than DS, and DS processes faster than CS processes. All this could be fine-tuned with the help of the QUANTUM command. It was pretty easy to tune the system, so that processes in the DS queue were punished much harder for using excessive CPU time than those running as SESSIONs in the CS queue.

But there was a provision that after some time - this could be several minutes in a very busy system - it was guaranteed that each DS process (and even an ES process) could again acquire the processor for the short time of a time slice.

MPE IV Scheduling Queues

In MPE IV we still have the three circular scheduling queues CS, DS, and ES. The QUANTUM command no longer exists; instead we now have the TUNE command. With the TUNE command, you can not only select the worst priority a process can ever get in a particular scheduling queue (as with the QUANTUM command in MPE III). but also the highest priority. When the system comes from HP, there is no overlap at all between the three scheduling queues, as is shown below:

```
      - 152
      |
CS    |
      |
      - 200
      - 202
      |
DS    |
      |
      - 238
      - 240
      |
ES    |
      |
      - 253
```

In this respect we now have much more flexibility in distributing the priorities than in MPE III. You now can let batch jobs run only when *no* process from a SESSION needs the CPU. But, in MPE IV there is now only *one* mechanism that gives a process a higher priority than it had before: the termination of a terminal read operation, i.e. when you press RETURN or ENTER. Then the process gets the highest priority it can get in its scheduling queue (in the above example 152 for CS and 202 for DS processes).

If now - in the above configuration - a process in CS

with pretty high priority even at daytime. CPU bound SESSION users are not always quite as happy. They now have always (as long as two JOBs are active) to compete against two JOBs. But in general this new configuration has proved to be acceptable to most users. Interactive applications, like editing, still get good response times.

Recently we made a little change in JOB priorities. Several of our users spent nights and weekends at the computer to get their work done faster than at daytime. Therefore we switch during nights and at weekends to the original priority distribution. There are seldom so many terminal users busy at night that the JOBs are not completed until the next morning. And the users, who spend part of the night or the weekend at the computer, get a high priority service.

Different Time Slices for SESSIONs and JOBs

There is another new tuning feature in MPE IV. You have different time slices for the three scheduling queues. The ones for DS and ES processes are fixed, the CS time slice is being recalculated by the system to represent the current mix of CS processes. The result of the recalculation is restricted to a certain range that can be selected with the TUNE command. It is preset to 0 - 300 msec and we did not change these values. The time slices for DS and ES processes are preset with 1000 msec (= 1 second).

If we would choose those values in conjunction with our daytime priority limits, CPU bound DS processes would get 3 to 4 times more CPU time than CPU bound CS processes. This would certainly upset the terminal users. Therefore we changed the DS time slice to 200(!) msec. With this value we achieve on the average an almost equal distribution of CPU time among CPU bound SESSIONs and JOBs.

Preemption

Another change regarding the dispatching of processes comes with MPE IV. It has to do with "preemption." What is meant by "preemption"? Let us assume a process A with relatively high priority releases the CPU to wait for the completion of an I/O operation (I/O suspension). Another process, say B, with the next lower priority is therefore scheduled for execution and gets the processor. What happens when the I/O operation of process A completes while the process B is still in the middle of its time slice? Two possibilities exist:

- The process B is interrupted although it has not used up its total time slice and process A gets back the CPU. This is called "preemption."

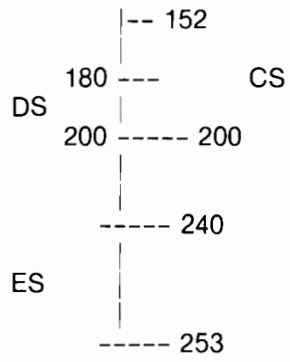
(SESSION) needs the processor for a long time (just a little DO loop in FORTRAN or doing serious number crunching), then no process in the DS or ES queue will even get a microsecond of CPU time. You cannot even abort the JOB, since this is being done with the same low priority.

Our Priority Limits in MPE IV

In our installation we have a mix of SESSIONs and JOBs, normally two active JOBs and several SESSIONs. People working in SESSIONs are explicitly allowed to do CPU bound work. That's the type of application we got the system for - five years ago. It has to be possible for the users to test and debug their number crunching programs. In other applications, users have programs where phases of number crunching alternate with interactive phases, e.g. graphic displays, data input, etc.

Therefore, after having switched to MPE IV, we had long phases where JOBs did not get any service from the CPU at all. The result was, that nobody wanted to submit batch jobs anymore. All the work was done from SESSIONs no free terminals were available.

We solved this problem very quickly by using the TUNE command. We adjusted the priority limits in the following way:



This means the lowest priority is the same for CS and DS queues, i.e. for SESSSIONs and JOBs. Only the best possible priority for JOBs is lower than that for SESSIONs but that does not make a big difference. JOBs do not climb up in priority anyway since there are no terminal reads completing in JOBs.

With these priority limits, we now have a system where CPU bound SESSIONs and CPU bound JOBs are treated almost the same. It can even happen - see below under "Time Slices" - that JOBs get more CPU time on the average than SESSIONs. But in this configuration a JOB can never be "killed" by a CPU bound SESSION.

The result in our institute was that generally the JOB users are now quite happy. Their work is being done

- The process B is allowed to continue until its time slice is entirely consumed, although another process (A) with higher priority is waiting for the CPU.

The second case has the advantage that less system overhead is generated and possibly a higher total system throughput is achieved. The disadvantage is that response time for I/0 bound processes becomes bad, and I/0 bound processes will get on the average less CPU time than CPU bound processes.

Let us look at a RESTORE operation. As always, the file to be restored is almost at the end of the tape volume. Many tape marks have to be skipped. This is a comparatively trivial operation in regards to CPU time. After issuing the skip-to-tape-mark request the RESTORE process is being I/O suspended. Another process gets the CPU until the tape mark has been found.

With preemption - if the priority is high enough - the RESTORE process would immediately get the CPU back and issue the next request: the tape would move pretty fast. *Without* preemption the RESTORE process has to wait until the currently active process has consumed its time slice. This can take one second or longer, depending on the current filter values.

In MPE IV preemption only takes place if the preempting process is "interactive", i.e. running in a SESSION. Processes running in JOBs do not preempt. Therefore, do not be surprised when your SYSDUMP or STORE *job* runs very slowly under MPE IV. No other JOB or SESSION should be active (e.g IDLE from the Contributed Library, not even in the ES queue).

But even if you run a RESTORE in a SESSION you may experience a slow down in the tape motion after the first few tape marks. The reason for this is the following:

Many requests to the file system to skip to the next tape mark eventually use up enough CPU time to let the RESTORE process drop gradually down in priority until it is at the lowest possible CS priority. There it lies together with all the other CPU bound and number crunching processes. As none of them will wait for a terminal read in the next time, none of them will ever get a higher priority. Now, processes with equal priority are scheduled in a round-robin fashion. And, since all those processes have the same priority, no preemption takes place. Therefore, even within one scheduling queue, I/O bound processes (non-terminal I/O) get worse service than CPU bound processes. The result is that our RESTORE operation becomes very slow after having skipped a couple of tape marks.

We know of no means to solve this problem. At the Berlin International Users Group Meeting in Octobr I talked about this problem to the HP people. Hopefully, in the near future HP will change their scheduling philosophy in this respect.

MPE- III Utilities

Many nice utilities from the Contributed Library or from elsewhere will not run correctly under MPE IV. Some of them will even crash the system. Examples are (★ means:crash the system):

SOO ★
SHOWPRI
IOSTAT2
TUNERn
OPERATOR ★
LISTFXX ★

IDLE should not be used to measure the free system time due to the problems mentioned above. RESP (measuring the system response time) now gives other results than under MPE III, essentially longer times.
On the Berlin Swap Tape I submitted MPE IV versions of SOO and of SHOWPRI. IOSTAT2 can be run under MPE IV, but gives no results about disc I/O, since disc I/O has now its separate I/O queue. I have a LISTFXX version running under MPE IV and not crashing it. It is currently being tested.

Conclusion

The purpose of this article is twofold. First, it should help all installations that switch from MPE III to MPE IV to smooth the transition and to understand what is going on in the computer due to the difference between the new and the old MPE. In addition, I hope that I could give some hints on how to overcome some initial problems in running MPE IV.

Secondly, this article should be a positive feedback to the HP people in charge of MPE IV, to show them some problems their old MPE III users have with MPE IV, and in which direction MPE IV can still be improved.

I personally would like to see essentially three areas changed in the future:

- Give an I/O suspended process a little kick in priority, but not as much as a process terminating a terminal read.
- Give the user the decision how much preemption he would like in his system, e.g. all CS processes should preempt whether SESSIONs or JOBs.
- Provide means again so that a JOB cannot be completely locked out by CPU bound SESSIONs.

JOURNAL
OF THE HP 3000 INTERNATIONAL USERS GROUP, INCORPORATED

Forms Families - I

Michael A. Casteel
Vice President
Computing Capabilities Corporation

This is a note on Forms Families, a VPLUS feature introduced some time back (MIT 2011 or Athena). The Forms Family concept is so useful that I am sure several articles would be needed to discuss the most significant implications.

In this note I will address a feature of Forms Families which is not discussed in great detail in the VPLUS manual: special initialization processing. (In fact, if you don't have the MPE IV version of the manual, it's not discussed at all). First, let me remind you of the most significant feature of the Forms Family: when going from one member of the family to another, VPLUS will not repaint the entire screen, as it would if the two forms were not members of the same family. The two forms of course have different names; they may also differ in all the specifications entered on the Field Menu for each field: Field Name, Display Enhancements, Field Type (Display Only, Optional, etc.), Data Type, Initial Value, and Processing Specifications. Basically, the picture stays the same, although different areas may be highlighted, protected, or unprotected when changing from one form to another.

Examples and explanations given in this note use a few conventions drawn from INSIGHT II, a transaction processor based on VPLUS. The conventions which appear in this note are:

- A form which is used to access a data set is given the name of the data set with a prefix code "DB__."
- A field in the form which is used to display or enter an item from the data set is given the name of the data item, again with the prefix code "DB__".
- A form used only to enter selection criteria for retrieving records from a data set is given the name of the data set with a prefix "DBE__". This is called an "Entry" form.
- A form with the special prefix code "DBP__" will be processed without pausing for user input, if all edits are satisfied.

An immediate application of the basic Forms Family feature is as advertised in the VPLUS documentation: for Change transactions we may use two forms, one to enter the key value, the other to view the selected record and make changes without changing the key value. With Forms Families, we can do this without repainting the screen.

For example, consider a "Change Employee Data" transaction which we wish to use to update name and address data in the PERSON data set. We wish the user to enter the Employee Number (EMP-NO), then view the current name and address. The user can change any part of the displayed data except EMP-NO, and press ENTER to update the database.

To accomplish this we design a screen layout depicting all the items of interest from the PERSON data set. We first create the Entry form, DBE__PERSON, to be used for input of the EMP-NO, and enter our screen layout for this form. This will be the original, or parent, form in our Forms Family. On the VPLUS Form Menu, we specify No Repeat and Clear before displaying the next form, DB__PERSON. Then, we go through the field. For this Entry form, we might make all fields Display Only except for the Employee Number, which we name DB__EMP__NO and make Required with Data Type DIG.

Next create the form DB__PERSON, being careful to enter DBE__PERSON in the Reproduced From box on the VPLUS Form Menu. This copies our screen layout and field definitions from the form we just created, making the new form the "child" in our Forms Family. Then we go through its Field Menus, making DB__EMP__NO Display Only (so the user can't change it), and assigning the desired attributes to other data fields. DB__LAST__NAME would be Required, while DB__INITIAL might be Optional, for example.

Now, when the user executes our Change transaction beginning with DBE__PERSON, he will be required to enter the Employee Number for the record to be changed. Once we retrieve that record from the database, we move on to the next form with VGETNEXTFORM to display the record contents and allow changes. As we display the record with VSHOWFORM, the user will see the Employee Number field "freeze", while the Last Name andlike data fields become available for entry of changes. The screen itself will not be repainted.

Then, after completing the desired change, the transaction returns to the beginning, displaying DBE__PERSON. The user will see all fields clear, and all "freeze" except the Employee Number which again becomes available for input.

This application of Forms Families adds a sophisticated look to the operation of your system.

From the user's point of view, why should the Employee Number field be unprotected when it can't be changed? Of course, as is the rule in data processing, there are tradeoffs to consider. In this case, we are adding another form to the Forms File, and there can't help but be additional overhead at run time in VGETNEXTFORM. But, had we used two unrelated forms to accomplish the task (as we had to in earlier releases of VPLUS), there would have been even more overhead involved in completely repainting the screen.

Another application of Forms Families is the entry (or display) of several records from several data sets on one screen, using multiple forms, without repainting or appending forms. Simply use a series of forms in one family: DB__SET1, DB__SET2, etc. On each form, assign DB__ names only to those fields which belong in the associated dataset, together with the right edits, protect/unprotect attributes, and so on. See below for an example.

Now, to the point of this note: consider VINITFORM operation on members of a Forms Family. As you will find when you use Forms Families, especially in applications like the last one above, VINITFORM works "just right", although not as it normally does. When initializing the fields in a form, VINITFORM normally starts by setting each field to the Initial Value entered on the Field Menu, often spaces ($EMPTY). However, if the form being initialized is a child form in a Forms Family, i.e. it was "Reproduced From" some other form, and the last form displayed by VSHOWFORM was a member of the same family, an empty Initial Value will not cause the field to be set to $EMPTY. Instead, the value left over from the previous form will be left! This means that, when using Forms Families, VPLUS automatically carries over field values from one form to the next.

For an example of this operation, let's consider an application like the last one above. Specifically, we wish a single screen to be used to add two records. Our "Add Employee" transaction using this screen will also allow us to put the employee's automobile on file. The screen is to be composed of two VPLUS forms, one for each record. We lay out the screen with PERSON data on the top half, and AUTOMOBILE data on the bottom half. We create one form, DB2__PERSON (we already have a form named DB__PERSON), with this layout. It's not a child form, since it's the first of its kind: it's going to be the parent in a new family. On the VPLUS Form Menu we specify No Repeat, Clear before displaying the next form, DB__AUTOMOBILE. Then we go through the Field Menus assigning edits, Data Types, Field Names, etc. for the fields in the PERSON records (DB__LAST__NAME ...). We might choose to make those fields from the AUTOMOBILE records Display Only (Field Type "D"). Now, we have a form suitable for adding PERSON records.

Next, we create the form DB__AUTOMOBILE, being careful to enter DB2__PERSON in the Reproduced From box on the VPLUS Form Menu. This copies our screen picture and field definitions from the form we just created. Then we go through its Field Menus, changing PERSON fields to Display Only and assigning the appropriate edits and attributes to the AUTOMOBILE fields. We also remove "DB__" from the PERSON field names and add it to the AUTOMOBILE fields.

If we define and execute an Add transaction which begins with the DB2__PERSON form, we will first see an empty form where we can enter PERSON data. The form is guaranteed to be empty, because DB2__PERSON is not a child form, so normal initialization took place. We enter valid PERSON data, the record is written to the data base, and we step forward to DB__AUTOMOBILE. VPLUS protects or "freezes" the PERSON fields and unprotects the AUTOMOBILE fields for us to enter, without repainting the screen. But look: the PERSON data is still shown on the screen! How convenient. This is because: 1) DB__AUTOMOBILE is a child form, and 2) in the same family as the previous form, DB2__PERSON.

So, we enter automobile data, place it on file and return to DB__PERSON to restart the transaction. Now, all the fields will clear once again because we are initializing a parent form.

Suppose we wish to allow the entry of several AUTOMOBILE records for each PERSON. We simply go to the Form Menu for DB__AUTOMOBILE and code it to Repeat in place, with Repeat Option "R". Now, after we enter the first AUTOMOBILE, those fields remain unprotected for entry of the next AUTOMOBILE. But, look now: the data we entered for the first AUTOMOBILE record is still on the screen! Here is one case where we might not wish to take advantage of this Forms Family feature. The solution is a simple one. We can override the special VINITFORM processing by coding explicit INIT processing specification on the Field Menus for the AUTOMOBILE fields, as follows:

JOURNAL
OF THE HP 3000 INTERNATIONAL USERS GROUP, INCORPORATED

INIT SET TO $EMPTY

A unique application of the special Forms Family feature is the possibility for the user to "pre-enter" information for use in later forms. Suppose, in our example, that the AUTOMOBILE fields were not Display Only on the DB2_PERSON form. Then, there's nothing to prevent the user from filing them in before pressing ENTER to add the PERSON record. When we step forward to the AUTOMOBILE form, that information stays in the form (unless we said SET TO $EMPTY), so we need only press ENTER. Of course, we could set up the AUTOMOBILE form to be processed automatically if the information has been "pre-entered", so the user could continue by just pressing ENTER once. For INSIGHT, just name the second form "DBP_AUTOMOBILE", where the "P" stands for Auto-Process.

I know that many VPLUS users have not yet made use of the Forms Family concept. It is my hope that this note will encourage you to consider its possible utility in your applications.

Before closing, I'd like to remark on the sequence of screen design steps narrated above. In the second example, the creation of the form DB_AUTOMOBILE copied the field definitions and edits from DB2_PERSON. This includes field names and processing specifications, many of which might not be correct for the DB_AUTOMOBILE form. In practice, you might consider making only those Field Menu entries which will be needed for both forms on the first form, then creating the second one. This could avoid the necessity of undoing many things on the second form which you just did on the first. In multi-form families, you can Reproduce From any member of the family. Pick the one whose field definitions will need the fewest changes to create your new form, and save yourself some work. ■

One-Pass Manual Master Deletion

Anthony P. Rizzo
Small Business Data Processing Corporation
4208 Airport Road
Cincinnati, Ohio 45226
(513) 871-7019

A problem arises when using IMAGE to serially read a record in a manual master data set, delete the record, and then go to read the next record. The problem, as many IMAGE users are painfully aware, is that of migrating secondaries. If the record that is serially read is a primary entry which has secondary entries on its synonym chain, after deleting the primary entry, its first secondary entry is physically moved to the address of the deleted primary entry. When the next record is serially read, the secondary record that migrated will be skipped. So much for the IMAGE tutorial.

I have seen all kinds of workarounds ranging from a two pass serial read of the manual master to pick up the migrating secondaries missed on the first pass; to writing out the key value of the records to be deleted to an MPE file, reading the file back in and getting the records from the data set by key value, and then deleting them. These solutions all require extra processing which is unnecessary.

A sample of a better method is illustrated in the COBOL program that follows.

```
*ASSUME THE FOLLOWING DATA DEFINITIONS
01   GET-MODE                PIC 9999 COMP.
01   STATUSS.
     05  CONDTN-WORD         PIC 9999 COMP.
     05  FILLER              PIC X (6).
     05  SYNONYM-COUNT       PIC 9 (9) COMP.
     05  FILLER              PIC X (8).

*IF NECESSARY, REWIND DATA SET BEING ACCESSED.
  CALL "DBCLOSE" USING BASE-NAME, DATA-SET, MODE3, STATUSS.
     IF CONDTN-WORD NOT = 0 GO TO DISPLAY-STATUS.
     MOVE 2 to GET-MODE.
READ-DATA-SET.
     CALL "DBGET" USING BASE-NAME, DATA-SET, GET-MODE, STATUSS, LIST, RECORD-BUFFER, DUMMY.
     IF CONDTN-WORD= 11 GO TO CONTINUE-PROGRAM.
     IF CONDTN-WORD NOT= 0 GO TO DISPLAY-STATUS.
*CHECK TO SEE IF THIS RECORD IS TO BE DELETED.
     IF record NOT to be deleted
        MOVE 2 to GET-MODE
        GO TO READ-DATA-SET.
     IF SYNONYM-COUNT< 2
        MOVE 2 TO GET-MODE
     ELSE
        MOVE 1 TO GET-MODE.
     CALL "DBDELETE" USING BASE-NAME, DATA-SET, MODE1, STATUSS.
     IF CNDTN-WORD NOT = 0 GO TO DISPLAY-STATUS.
     GO TO READ-DATA-SET.
CONTINUE-PROGRAM.
.
.
.
/
```

After IMAGE performs a read on a manual master data set, doubleword 5-6 of the ten-word status array is equal to zero, unless the entry read is a primary entry, in which case, it is the number of entries in the synonym chain. If no secondary entries are present, the synonym count is 1. We use this fact to determine whether we need to reread a record address after deleting a record.

during the current segmenter session, you will still have to specify the ;RL = parameter if you wish to include routines from an RL file.

-EXIT

Ends interaction with segmenter program (SEGDVR.PUB.SYS)

Segmenter Strategies (RI's vs. SI's)

Assuming you want to have common routines for your shop, which file type should you use? RL's or SL's. The answer is "IT DEPENDS!"

Things to consider include:

1. Does the routine require global storage? (Access to what is known as DB-relative storage). If so, then the routine must not be in an SL file.

2. Is the routine likely to be changed or enhanced? If so, what would be the impact of having to re-PREP all the programs that use the routine.

3. If you put the routine in an SL and use the ";LIB = " parameter, will you need this routine in other groups or accounts? Would it create problems for you to maintain several copies of the routine?

There are many ways to allow for common subroutines at your shop. I will mention 5 possible alternative approaches that might be considered.

A. Use the COPYLIB facility of COBOL or the JOIN command of the EDITOR and include the common source routines as part of the source for your main program.

B. Compile the subroutine and keep the USL file. Using the SEGMENTER, reference the subroutine's USL file as the AUXUSL and copy the RBM's into another USL that contains the main program RBM's.

C. Place the routine in an RL. All other programs that need the routine must simply be PREPed with the ";RL = filename" parameter.

D. Place the routine in an SL in the group where the program will reside (or in the PUB group of that account). Programs that need the routine must be run with the ";LIB = G (or P)" parameter on the :RUN statement. (It could also be part of a UDC.)

E. Place the routine in SL.PUB.SYS. The program will not require the ";LIB = " parameter.

Given these alternatives, there are some pluses and minuses to each approach. They are:

A. Common source code.

Advantages: Easy to use; Requires no use of the SEGMENTER subsystem; Allows for non-dynamic subroutines.

Disadvantages: Changes to a common routine may require recompiling many programs; Compiler time is increased since the common routine is recompiled; Possible conflicts with data or paragraph names in main programs; Does not allow mixing of source languages.

B. Use an AUXiliary USL file.

Advantages: Allows complete flexibility in segmentation; Allows mixing of source languages for main program and subroutines; Allows non-dynamic subprograms.

Disadvantages: Requires extra commands in JCL jobstream; Requires more SEGMENTER expertise on part of programmers; Requires good documentation of shop standards and how to use them; Also requires more control; No easy way to determine if a program calls the routine since all references are internal to the object code.

C. Put the routine in an RL.

Advantages: Easy to use; :PREP uslfilename;RL = rlfilename Allows non-dynamic routines; The RL file can contain all the common routines for your shop; Only the referenced routines are brought into the object code at PREP time.

Disadvantages: All referenced routines are brought into one segment; This may have some performance implications and may also produce extremely large segments for some programs; A change to the RL routine requires SEGMENTER manipulation and may require that all programs which use the routine be re-PREPed; (If you no longer have the program USL, you will also have to recompile;) No way to clean or expand the RL file; If you run out of room, you must rebuild the RL from the USL's or possibly even the source code (and more recompiles.)

D. Put the routine in SL.group.account (or SL.PUB.account)

Advantages: No recompiling; All routines are current as of RUN-time; Allows for sharing of code segments; (i.e. Every program does not have its own copy of the routine;) The program file is smaller.

Disadvantages: Uses valuable CST entries; (System limit is 192) Requires ";LIB = G (or LIB = P)" parameter; (But could be in UDC); May

require many SL's to be generated with identical routines depending on your account structure.

E. Put the routines in SL.PUB.SYS

Advantages: Same advantages as SL.group.account but no ";LIB=" required; Minimizes duplicate code modules; Sharing on a system-wide rather than just an account-wide level; Somewhat easier to control than are many SL files; More sharing of common code could result in less swapping for certain environments.

Disadvantages: It is hard to make changes to the system SL; This should only be done via the SYSDUMP procedure and may require you to shutdown your system to install any new routines or changes; A new cold load tape should be made after any change; You will have to reapply your changes every time there is an update to MPE; There may be a conflict with entry point names of your routine and those of MPE; Even if all routines are OK today there is no no guarantee that the next version of MPE will not have a name which conflicts with your routines; If you ever have to take your application programs to another site's machine, they may not want you to put things in their SL; Consider the implications of this in your disaster recovery plan.

Here is a chart which compares some of the differences between the RL and SL file types.

	RL	SL
1. Allows dynamic routines	Yes	Yes
2. Allows non-dynamic routines	Yes	No
3. Loaded at PREP-time	Yes	No
4. Loaded at RUN-time	No	Yes
5. Can contain more than 1 segment	No	Yes
6. File can be any name	Yes	No ★
7. Can reference common data	Yes	No
8. Permanent part of program file	Yes	No
9. Uses a CST entry	No	Yes

★ Although the SL file can be built with any name, it cannot be referenced at RUN-time with the ";LIB=" parameter unless the name of the file is "SL".

So what is the right answer? It still depends.

However you should probably not consider putting your routines in SL.PUB.SYS for the reasons just shown. Also non-dynamic (those that use global storage) routines cannot go into any SL. Other than that, it probably is a function of your account structure, control procedures and personal preference rather than a technical issue.

This should hopefully help you to answer whether you want to put the routine in an SL or RL. (Assuming you even asked the question!)

Summary

The SEGMENTER subsystem will typically be used for several reasons.

These would include:

1. Manipulating RBM's for effective segmentation - to create a more "efficient" program - to reduce the number of code segments - to use common code from another USL file.
2. Manipulating library files (RL's or SL's) - to list the contents of the files - to add or delete routines from the library file.
3. Utility functions such as cleaning or copying a USL or SL.
4. Miscellaneous functions - Activating or inactivating an older version of code in the USL file - Setting or resetting an internal flag so that code will be available only to other procedures within that segment.

The SEGMENTER subsystem operates on three different types of files. These include USL, RL, and SL files. All of the files store object code but are used for different purposes. Everything must first be compiled into a USL before a manipulation can take place. RL's and SL's are "grab bags" of commonly used routines that were first compiled into a USL.

This paper should hopefully have explained the SEGMENTER process in general terms. It does not have specific examples of the use of the various commands. We cannot effectively cover examples in this paper because of length limitations for Journal articles.

BUT there is GOOD NEWS! The SEGMENTER manual is in the process of being rewritten. It should be a much more useful and viable reference guide than the current edition. I don't know when it will actually be available but I certainly hope it will make it to distribution before the end of the FISCAL year. In the meantime -

HAPPY SEGMENTING! ■

JOURNAL
OF THE HP 3000 INTERNATIONAL USERS GROUP, INCORPORATED

Tips and Techniques For Technical Writing

Dr. John R. Ray, Editor
HP 3000 IUG Journal
Dept. of Curriculum and Instruction
University of Tennessee
312 Claxton Education Building
Knoxville, Tn. 37996-3400

Dr. Lloyd D. Davis, Director
Academic Computing Services and
Associate Editor - HP 3000 IUG Journal
University of Tennessee at Chattanooga
Hunter 209-C
Chattanooga, Tn. 37402

Why Write?

If you are not a writer by profession, you may be hesitant about writing for a professional publication such as the HP3000 IUG Journal. The fact of the matter is that your professional skills are more important than your writing skills. Like many professional publications, The Journal builds its reputation on being written by professionals in the field for other professionals.

If you have experience, then we encourage you to share your knowledge through a Journal article. To help you get your thoughts down on paper, we have put together some tips on writing for non-writers. The professional and personal benefits derived from writing an article are of major importance.

What are the benefits to be derived from writing an article? For one thing, having an article published in the Journal generates publicity for both the author and the author's firm. The author benefits by being recognized as having expertise on the topic. Your firm benefits by being recognized as having leading professionals on its staff and by having its name brought to the attention of professionals nationwide.

Another benefit of writing an article is the personal satisfaction that comes from having contributed to the betterment of the profession through sharing your knowledge. There is also the satisfaction of seeing your name and your ideas in print.

The Review Process

Reviewers evaluate all articles basically on content, not grammer or literary style. For each article, the reviewers fill out an evaluation form and recommend that the article be either: 1) published; 2) revised and published; 3) revised and reviewed again; or 4) not published. Articles that are original, timely and previously unpublished, devoid of sales or promotional material, and of national interest and value to a significant number of readers have the best chance of being published.

The review process usually takes four to six weeks. As soon as the reviewers' evaluations are received, the author is notified of their decision. If the reviewers recommend that an article be revised, the author will be provided with specific recommendations.

If the information in the article could easily become dated, the author should note this in a letter attached to the article when the article is submitted for publication. The staff will then make a special effort to publish the article before the information becomes out of date and will, if necessary, contact the author for updated information immediately before publication.

Most articles will require some degree of editing before publication. The staff may suggest refinements in the areas of literary style and organization. If there are corrections in the areas of spelling, punctuation, grammar, or word choice, these will be noted on the article. The article with annotated remarks will be returned to the author for approval prior to publication unless editorial changes are minor.

On occasion, the staff may telephone an author and ask a question about the information in an article. Although the editorial reviewers do review articles for accuracy of information, the author is still responsible for the accuracy of his or her article. Also, publication of an article does not mean that the ideas expressed in the article are endorsed by the HP 3000 International Users Group and/or its Journal editors.

Manuscript Requirements

Most articles submitted for publication in the Journal are four to eight double-spaced, typewritten pages, but articles longer and shorter than this have been published. Very long articles of twenty pages or more may be published in parts as a series.

To estimate the number of pages an article will be when published, have the article typed with 50 to 55 characters on each line. This will give you the approximate number of lines the article will be when published. The Journal uses a two column format with about 55 lines per column. Hence divide the total line count by 110 (two columns of 55 lines make one page). This quotient is your rough page count.

If space is required for exhibits such as formulas, tables, charts, diagrams, and figures, then the page count should be revised upward to reflect that space.

All articles submitted for publication in the Journal should be double-spaced, typewritten on one side of 8½ x 11 inch white paper. Ample margins of at least 1 to 1½ inches should be left on all sides. Articles typed with about 53 characters on each line would be appreciated, but this is not mandatory. Subheadings should be inserted where appropriate in the article, but again, this is not mandatory. Footnotes, tables, and figures should be on sheets of paper separate from the article. Indicate the placement of tables and figures within the article by giving each table, figure, etc., a number and using this number within the text.

Footnoting has as its goal the conveying of necessary information to enable the reader to accurately identify the location of the material to which reference is being made within the article. The most important traits of footnoting are accuracy, completeness, and style consistency. If you are not familiar with footnoting techniques, several good style guides are available to serve as references. These include:

- *Publication Manual of the American Psychological Association* Second Edition 1979
- *Form and Style - Theses, Reports, Term Papers* William Giles Campbell Stephen Vaughn Ballou Houghton Mifflin Company, Boston, 1974
- *A Manual for Writers of Term Papers, Theses, and Dissertations* Kate L. Turabian Fourth Edition University of Chicago Press, Chicago, 1973.

The above references give you the standards for footnoting. In practice when writing for our Journal or most other professional journals, read the journal in question for the style of footnoting used in that journal. By utilizing the examples found therein as a guide for your required footnotes (you may not need any), you can easily handle yours with a high probability of being correct and complete.

All pages containing copy, footnotes, tables and figures should be numbered sequentially, with tables and figures being the last of the pages. Numbering pages is important in case the pages do get out of sequence.

Black and white photographs to accompany the article are welcomed. An explanation of each photo (a caption) should be submitted with each photo. A caption can be written on the back of the photo or on a sheet of paper. If written on a sheet of paper, the sheet of paper should be numbered as the last page of the article, and if there is more than one photo, the caption should be numbered to indicate with which photo it is associated. Photos cannot be returned.

A short author's biography, including title, firm, membership in professional societies, special accomplishments and honors, should be submitted with the article.

If the article has been submitted to another publication or has been previously published, this should be noted in a letter accompanying the article. As mentioned before, if the information in the article can become dated soon, then this also should be noted.

Before mailing your article, read it over carefully. Recheck all figures and mathematical computations. Sometimes mistakes occur in typing. Remember, you are responsible for the accuracy of your article.

After you are certain that your article is accurate and to your liking, make a copy of it. Keep one copy for your records and mail the other to the Journal. All articles submitted to the Journal and all correspondence regarding publishing in the Journal should be addressed to:

John R. Ray
University of Tennessee
Dept. of Curriculum and Instruction
312 Claxton Education Building
Knoxville, Tn 37996-3400

Tips on Writing for Non-Writers

Often the task of writing seems too formidable to undertake. The ancient Chinese proverb states "Each journey of a thousand miles begins with the first step." This is also true in writing. You must eventually start placing words on paper or equivalently on other media. But how do you get started? We have listed several points that we believe will be helpful to those without previous, extensive writing experience.

1. Have something to say. When writing an article for the Journal, what you have to say is more important than how you say it. Ask yourself what subject you want to write about and what you want to say about it.
2. Make a list of the points you want to discuss in the article, then arrange these points in the tentative order you want to discuss them. This will give you an outline of your article. Use single words and phrases rather than complete sentences. If the list becomes too long or unwieldy, the subject may be too broad to be covered in the space of one article. In such a case, limit the subject and eliminate several lesser points.
3. Ask yourself who, what, when, where, why and how. This is another method of preparing an outline for your article. You could also compile a list of questions you will answer in the article.
4. Pretend you are writing a long business letter on the subject or preparing a report for your firm.
5. Write as you speak. If you have difficulty getting your thoughts down on paper, try

dictating them into a dictaphone or tape recorder.

6. Don't worry about saying it right the first time. Concentrate on your thoughts, not the words. Once your thoughts are written down on paper or transcribed from a dictaphone or tape recorder, you can go back and revise your wording.
7. Try "The purpose of this article is ..." if you have difficulty starting the article. You can change this first sentence later if you want, although this is an acceptable way to begin an article. "In summary" and "in conclusion" are acceptable and easy ways to end an article.
8. If all else fails, consider having a professional writer write your article for you. Your firm may have a public relations firm on retainer or a public relations writer on staff that you could use. Do make sure, though, that you provide the writer with in-depth information that is current and topical and that you review the article for accuracy and value upon completion. Otherwise, the article probably will not be of interest and value to the readers of the professional (HP 3000) journal and therefore runs the risk of not being suitable for publishing.

Format

There is no absolute format, standard, or arrangement that must be followed in preparing an article for a professional journal. Items that might be appropriate for one article might be totally inappropriate in another. After the author has selected those things about which to write, the format or physical arrangement (headings) can be determined. To aid the writer, sections with appropriate headings and subheadings might be selected from the following list (in about the same order):

Report Title

Introduction
- Background
- Problem Statement
- Information Sources

Procedures
- Design of experiment or solution
- Sample Selection
- Equipment
- Measures Used

Findings (Data)
- Presentation of facts and data
- Interpretation of findings
- Limitations of 'facts' meanings

Summary and Conclusions
- Short restatement of goals for article
- Brief statement of findings
- Any conclusions
- Suitable recommendations

References (Bibliography)

Appendix (if any)

Again we stress that the above list is a very formal list of of topics that might be found in some research papers. Rarely would all of these be found in the average journal article. However, some of these may provide an outline or skeleton upon which you may structure your writing and aid you in a readable, logical organization for your paper. Select from the topics on the list a topical outline that suits your article; utilize headings and structure to augment or replace these topics as your article requires.

Style and Readability

Articles for professional journals sometimes suffer from being too stiff and rigid and/or from being awkwardly worded. Authors should strive for a style that is clear, direct, and effective. Word choice should be appropriate for the populace that reasonably might be expected to read the article. Therefore word choice should be chosen so as to both convey the problem and its solution and as well not require the reader to use a dictionary for frequent translation. Articles should be written in a direct, straightforward manner without being overly elaborate and structurally complex. Although, as earlier mentioned, there are writing conventions common to writing for professional journals, these should not interfere to the point of making good writing bad. Rather each author should utilize his/her individual skills in communications to convey meaning to the reader. Several methods (3:41-3) for improving readabilty follow:

1. Appeal and interest increase readability.
2. Personalization means putting human interest into the report: through a review of previous investigations as a story of other persons' successes and failures, an account of how the author collected and treated the data, illustrative cases, and deviations from central tendencies.
3. Pattern or design should be made plain to the reader.
4. Through appropriate emphasis the reader should get the important points.
5. Too great density or concentration of ideas may make reading difficult, requiring some expansion or dilution.
6. Plain words are important in making a report readable.

Remember that style is to foster clear and effective communication, not to confuse it. Carter Good reports (2:409) "As long as young scientists and scholars write accurately, clearly, and attractively, their differences in expression may render science a happier way of life for them and for the reader."

Success

We have stressed those points that we believe important in writing a journal or other professional article. Many of these are somewhat mechanical and pro forma; others are good sense types of points. It all requires an idea, a suggestion, a fresh point of view, something important enough to justify your writing and others reading. You may say, "But no one has ever heard of me before. What chance have I to write something and see it published?" Not surprisingly, what you have to share and say is more important than who you are or where you are from. Thomas Frantz (1:384-386) surveyed the editorial boards of six professional journals and asked these editors to rank order criteria commonly used in evaluating manuscripts for journal publication. His findings follow in tabular form.

TABLE 1
Summary of 14 Criteria
For Evaluation of Manuscripts
Ranked in Importance by 55 Members
of the Editorial Boards of Six Journals

Criteria	Mean	S.D.
1. Contribution to knowledge	1.8	1.2
2. Design of Study	3.5	2.1
3. Objectivity in reporting results	4.7	2.3
4. Topic selection	5.5	2.9
5. Writing style and readability	5.7	2.7
6. Practical implications	6.4	3.3
7. Statistical Analyses	6.5	2.5
8. Theoretical Model	7.0	2.7
9. Review of the literature	7.2	2.3
10. Clarity of tabular material	8.1	2.3
11. Length	10.2	1.6
12. Punctuation	11.5	1.9
13. Reputation of author	12.6	1.9
14. Institutional affiliation	13.5	0.9

The above research reports that the contribution to knowledge the article makes is of primary importance. Also, as the article reports, among the top six criteria are objectivity, topic selection, writing style and readability, and practical applications. Of least importance are the author's reputation and institutional affiliation. The moral here is that who you are is not important; rather what you say, what it means, and how it reads are all nearly equally important.

Don't get discouraged if your article is not accepted for publication. Usually, the reason an article is not accepted for publication is that it is too general in scope and does not provide enough in-depth information to be valuable to other professionals. Keep in mind that many famous authors have had articles rejected for publication but did not quit trying. As the saying goes, if at first you don't succeed, try, try, again.

References

1. Frantz, T. F. *Criteria for Publishable Manuscripts,* Personnel and Guidance Journal, 47 (1968), 384-386.
2. Good, C. *Essentials of Educational Research. New York: Appleton-Century Crofts, 1966.*
3. *Strang, R. Principles of Readability Applied to Reporting Research,* Improving Educational Research. Washington: American Educational Research Association, 1948. p. 41-43.

The 1982 *JOURNAL* Publication Schedule, as given below, includes the deadlines for receipt in the Los Altos office of articles submitted for publication. Also given are the months of publication.

HP 3000 IUG JOURNAL PUBLICATION SCHEDULE

Article Submission Date	1982 Quarterly Issue	Month of Publication
July 15	Vol. V. #3	September
October 15	Vol. V. #4	December

This publication is for the express purpose of dissemination of information to members of the HP 3000 International Users Group. The information contained herein is the free expression of members. The HP 3000 International Users Group and Editorial Staff are not responsible for the accuracy of technical material. Contributions from Hewlett-Packard Co. personnel are welcome and are not to be construed as official policy or the position of the Hewlett-Packard Company.

Send your contributions c/o Editor, HP 3000 International Users Group, Inc., 289 South San Antonio Road, Los Altos, California, 94022, U.S.A

JOURNAL

OF THE HP 3000 INTERNATIONAL USERS GROUP, INCORPORATED

Journal
Administrative Offices
HP 3000 International Users Group, Inc.
289 South San Antonio Road
Los Altos, California 94022
U.S.A.

Ruth Ann Barrett, Association Manager
Tel. (415) 941-9960
ADDRESS CORRECTION REQUESTED
RETURN POSTAGE GUARANTEED

BULK RATE
U.S. POSTAGE
PAID
Permit No. 656
Los Altos, Calif. 94022