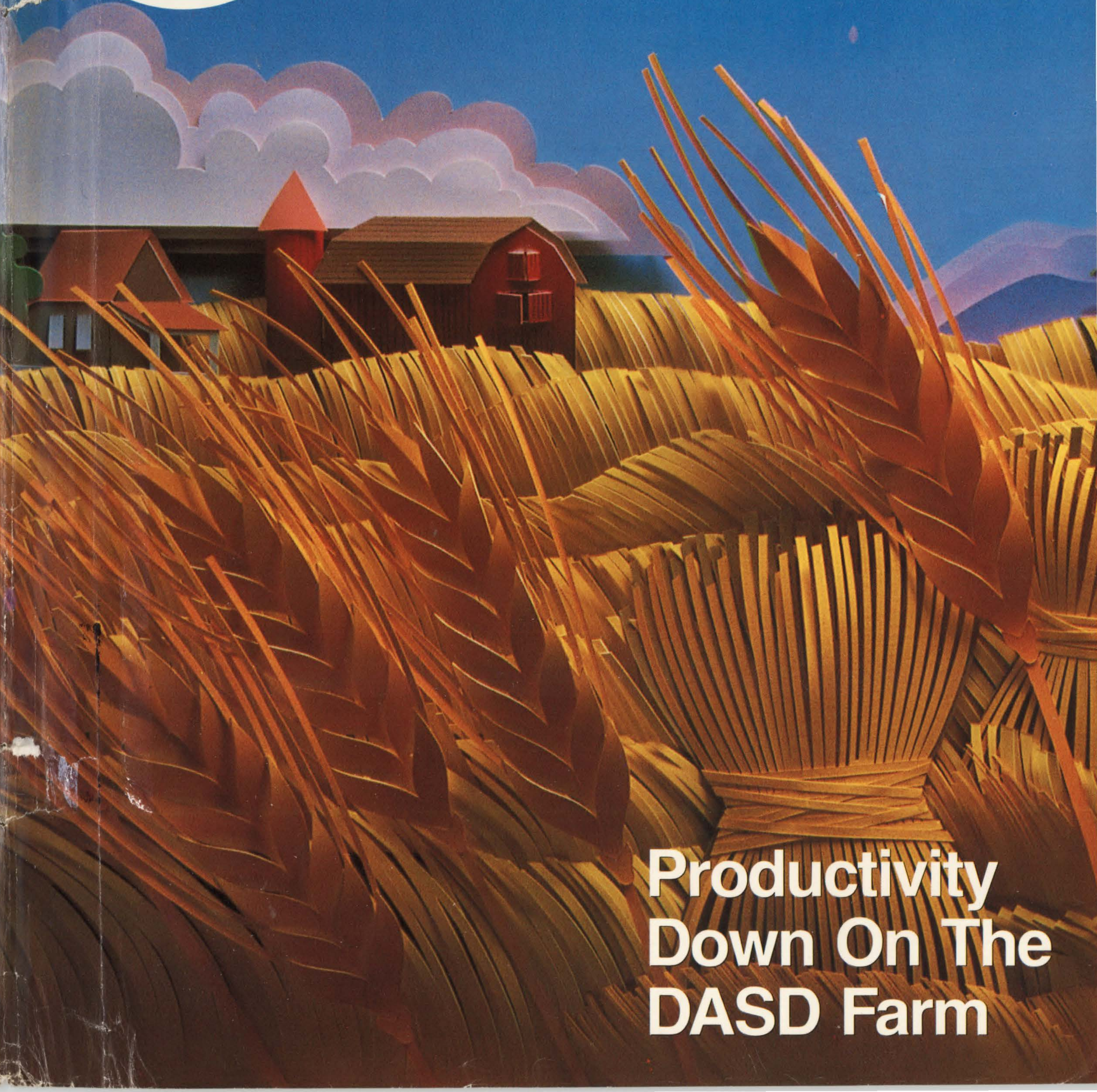
MAINFRAME JOURNAL

For Users of IBM System/370 Architecture & Compatible Systems

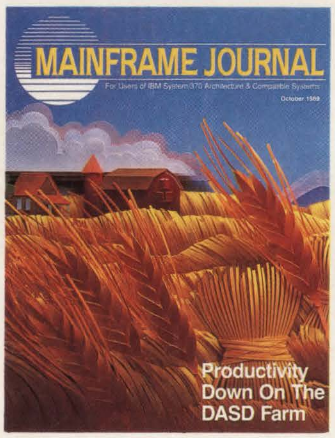
October 1989



C O N T E N T S

October 1989 • Volume IV • Number 10

DEPARTMENTS



COVER:

Down on the DASD farm you will look at DASD utilization from the productivity perspective with a tongue-in-cheek comparison to a crop-production survey. *Turn to page 104 for details.* *Cover illustration by Leo Monahan.*

MAINFRAME JOURNAL© (ISSN 0895-5751) is published monthly by Thomas Publications, Inc., 10935 Estate Lane, Suite 375, Dallas, TX 75238, (214) 343-3717. Second class postage paid at Dallas, TX. SUBSCRIPTION RATE: Subscriptions are free within the USA and Canada. One-year foreign subscriptions are $96. POSTMASTER: Send address changes to: *MAINFRAME JOURNAL*, P.O. Box 551628, Dallas, TX 75355-1628.

MANAGEMENT

SYSTEMS

APPLICATIONS

OPERATIONS

COMMUNICATIONS

Systems software for MVS data centers:

Enter the world of total administration, total support.

Presenting **CA-UNIPACK™/DCA**—an advanced software system from Computer Associates that automates the complex tasks of MVS data center management.

CA-UNIPACK/DCA—DATA CENTER ADMINISTRATION
Consisting of: CA-NETMAN®/FINANCIAL-INVENTORY, CA-NETMAN®/PROBLEM-CHANGE, CA-NETMAN®/OLCF and CA-NETMAN®/MRM.

CA-UNIPACK/DCA automates and integrates: hardware and software inventory management, help desk and problem tracking, configuration changes, invoice reconciliations, cost allocations, budgets, vendor contracts, user chargebacks, order tracking and more. It has the unique ability to interrelate these diverse activities so that a change in one area is immediately and automatically reflected in another. Additionally, CA-UNIPACK/DCA provides specialized functionality in managing and analyzing activities related to PCs and workstations.

CA-UNIPACK/DCA provides online, realtime control over critical managerial functions while it reduces costs, increases staff productivity and ensures sound decision making. Total administrative control. Only from Computer Associates.

And only Computer Associates offers **CA-UNISERVICE®/II**, a secure link between your mainframe and CA's Customer Service System 24 hours a day. You get online access to software fixes, interactive problem resolution, plus product tutorials and more!

Call Dana Williams today:
800-645-3003

COMPUTER ASSOCIATES®
Software superior by design.

- World's leading independent software company.
- Broad range of integrated business and data processing software for mainframe, mid-range and micro computers.
- Worldwide service and support network of more than 100 offices.

Resource & Operations Management • Financial • Banking • Graphics • Spreadsheets • Project Management

CIRCLE #90 on Reader Service Card ▲

CONTINGENCY JOURNAL: The Magazine For Business Continuity Planning

Bob Thomas

When we began the process of selecting the editorial content for the September issue of *MAINFRAME JOURNAL,* one topic we had planned to feature with several articles was contingency planning. The area of contingency planning, as well as disaster recovery and security, has proven to be extremely popular with our readers. In fact, we have had more requests for article reprints on these topics than any other. After evaluating the need for this type of information, we decided to forego periodic coverage in *MAINFRAME JOURNAL* and instead, to launch a completely new magazine titled *CONTINGENCY JOURNAL: The Magazine For Business Continuity Planning*.

"Contingency Planning is the ability of an organization to fulfill its mission, *no matter what,"* according to Philip Jan Rothstein, President of Rothstein Associates, Inc., a management consulting firm in Ossining, NY. "Those *'no matter whats'* are tough. Sure, fires, floods and 'dust and rubble' belong here but what about the disruption of critical information?" Some of the *'no matter whats'* to be covered in upcoming issues of *CONTINGENCY JOURNAL* are:

- Contingency Planning
- Disaster Avoidance
- Disaster Recovery
- Security
- Data Recovery
- Software Viruses
- Power Outages
- Fault-Tolerant Systems
- Electronic Vaulting
- Legal Liability
- Business Continuity
- Hostile Takeovers
- Loss Of Key Employees
- Strikes

The premier issue of *CONTINGENCY JOURNAL* will be launched in January 1990. If you think *CONTINGENCY JOURNAL* might be of benefit to you or someone else in your organization, reserve your copy now by sending in the Free Subscription Form in the ad on page 11.

FOCUS Newsletters To Be A Reality

Your response to *MAINFRAME JOURNAL's* six FOCUS Newsletters has exceeded our expectations.

Quite frankly, back in June when we first announced the FOCUS Newsletters in that issue of *MAINFRAME JOURNAL* and in our Action Card Deck, they were more of a speculative idea than a concrete actuality. At that time, we did not know if there would be sufficient interest to support not only the costs incurred with a series of newsletters, but also more importantly, the time and effort that development and production would take. Since June, the faith and confidence demonstrated by all of you who subscribed "sight unseen" is very much appreciated and has dictated that we cast the concrete and make the FOCUS Newsletters a reality.

Just as producing a solid software package that lives up to the customer's expectations takes time, so does developing solid FOCUS Newsletters on MVS, VSE, VM, CICS, VSAM and DB2. The objective of each FOCUS Newsletter is to provide more specific and timely information than is possible in *MAINFRAME JOURNAL*. Our target date for all six FOCUS Newsletters is January 1990. For subscription information see page 45.

Bob Thomas

MAINFRAME JOURNAL
For Users of IBM System 370 Architecture & Compatible Systems

Publisher and Editor-in-Chief
Bob Thomas

Associate Publisher
Martha Thomas

Editor
Carol M. Hoag

Copy Editors
Judy Beller
Cecilia Perry
Pat Warner

Art Director
David Kramer

Assistant Art Director
Ken Buerer

Magazine Production
Diane Dishman

Marketing Services
Sally Webb

Circulation Manager
Janice Porter

Assistant Circulation Manager
Nancy Crawford

Administrative Manager
Marian Davenport

Advertising Sales
Denise Thomas — Mark Cauto

BPA

List Rental Information
Contact AllMedia (214) 985-4060

SUBSCRIPTION RATES & INQUIRIES: Subscriptions are free within the USA and Canada. One-year foreign subscriptions are $96.

All subscriptions, remittances, requests and changes of address should be sent to *MAINFRAME JOURNAL* at 10935 Estate Lane, Suite 375, Dallas, TX 75238, (214) 343-3717.

The Power Package!

Take the shortcut to fast response times. DATAPACKER/II, the CICS transmission optimizer from H&M, will reduce outbound and inbound data streams by 50-85%. The advantages speak for themselves: considerably improved response times, no screen blinking, lower line costs and free capacity for additional terminals and new applications.

DATAPACKER/II
CICS · TRANSMISSION · OPTIMIZER

CIRCLE #118 on Reader Service Card ▲

H&M.
SOFTWARE'S FUTURE

H&M Systems Software, Inc., 25 E. Spring Valley Ave., Maywood, N.J. 07607-2150, 1-201-845-3357, 1-800-FOR-DEMO

Data Processing — Career Or Obsession?

"Why am I doing what I am doing?" is the common question asked by many a brave data processing professional while scratching his way through another day "in the business." Why is this question asked? Let's get back to the basics, the ground floor, the place where it all started — *the beginning*.

In the beginning, computer technology was the new frontier that lured young men (few women ventured into the forbidden zone) into searching for the perfect career. Visions of white lab coats, blonde girlfriends and new Corvettes made the life of a data processing professional the only logical choice. You did not start the adventure as an executive — you were allowed to start at the beginning. Dues had to be paid, hours had to be worked before you could be worthy of receiving the ground-floor opportunity. Ground-floor opportunities were abundant and, unfortunately, were rewarded with ground-floor wages and could be equated to working until one was ground into the floor. Take a look at the industry from the '60s to the present.

The '60s brought the excitement of the new frontier. Mainframes were growing in size and capacity. Data processing was a brave new world to be explored. Creativity was the key word in programming, limited only by the amount of bytes of storage available. The cloak of mystery surrounded the data processing field. The programming wizard was free to weave his magic spells — card input (what a wonderful innovation). COBOL was a godsend, RPGII was a language to be reckoned with, PL/1 was the language of the future.

The '70s brought larger mainframes and more storage. Brave new concepts for program development appeared: structured programming, documentation and code walk-throughs. Restriction of programmer creativity became the trend. Standardization and management restrictions now replaced the limitations that mainframe storage once commanded. Programming languages were developed that even the novice could code and understand. Yes, it was a dark time for the creative wizard. But all was not lost, a method for producing on-line, user friendly, interactive programs was being refined. Yes, CICS was becoming the state-of-the-art tool.

The '80s, the years of budget restraints, staff reductions and new wave management proved to accelerate the demise of the coding wizard. Databases, fourth-generation languages and computer industry publications proved to be a formidable foe for the wizards. Assembler language became the Holy Grail of the '80s, lost forever. The refinement of the personal computer further eroded the programmer mystique. Laymen understanding the inner workings of programming tools and computers stole the soul from the magic spells; the wizard was now perceived as being just a mere mortal.

Look at what it took to become a wizard. You were allowed to work your way to Valhalla in progressive stages: tape librarian to production control clerk to tape hanger to operator to programmer to programmer/analyst and, if fortunate enough to survive, the first marriage, the kids, the long working hours, the divorce, the cure for alcoholism, the second marriage, the high blood pressure and medical problems that accompany years in data processing. You will finally excel to the position of systems analyst.

But life at the top is not all it is cracked up to be. You find that through the years of trials and tribulations, you neglected to receive the one thing that eluded you over the years — a college degree in data processing or a related field. It seems that 20 years of experience is now replaced by two-to-four years at an accredited institution of learning. Yes, you have reached the bottom line. Real-world experience and treachery does not replace the rookie with a degree from fantasy land. It is true, "Data is just a four letter word."

Thomas E. Williams, Jr.
Sprouse-Reitz Stores Inc.
Portland, OR

VM Bigotry Revisited

I would like to add to the letter of Rich Greenberg in his August 1989 response to Michael Seadle's June 1989 article, "VM in the Development Center."

I agree that coding REXX programs are for the most part a "question of style." However, in the example given, you will experience a performance degradation, especially as the program grows in size.

The REXX interpreter will attempt to interpret all non-literals before passing the expression to the CMS environment. In the example given:

```
EXEC NOTE MICHAEL '(' NOTEBOOK PROJECT
```

REXX must translate the first four variables to their respective values and then pass them to CMS. If on the other hand, the expression were coded as Rich suggested:

```
'EXEC NOTE MICHAEL ( NOTEBOOK' PROJECT
```

there is only one variable to be translated.

Again, the program in question is a simple one and the particular style does not make any difference. However, as programs become large with more and more variables, you can see that *style* can make a *big* difference.

Robert L. Hatcher
Southern Company Services, Inc.
Atlanta, GA

4341 To AS/400

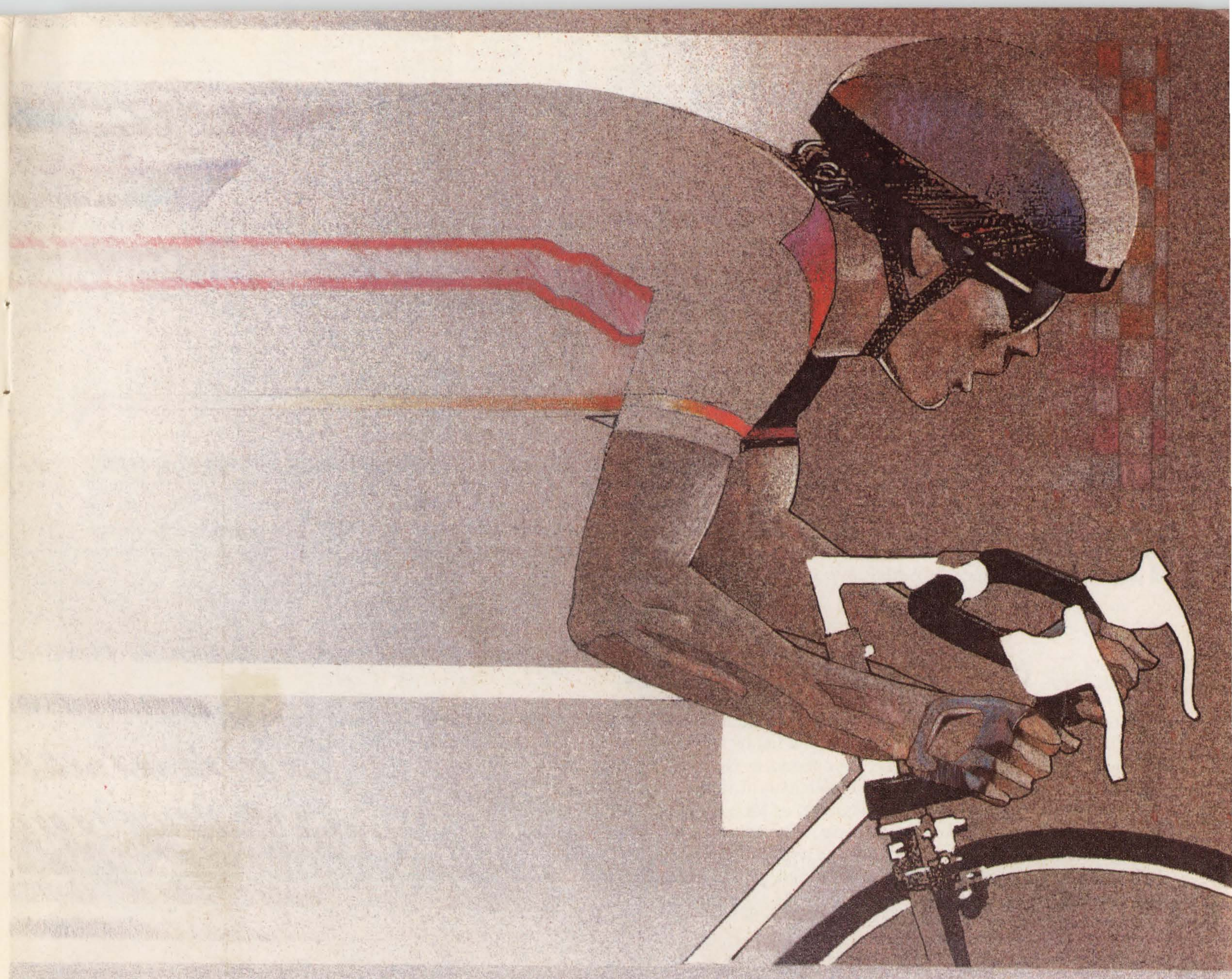
In the May 1989 issue the article, "Carolina Steel Makes the Most of VSE," by Mary Lou Roberts could have been written about my company, Keyes Fibre Company. While there are many similarities, there are some significant differences:

- We rely 100 percent on our database from Software AG called ADABAS and its Natural language
- We rely heavily on third-party software
- We do not have a systems programmer
- Our centralized staff of 12 programmers/analysts supports a 4341, several System/36s and PC platforms in eight states throughout the United States
- Our System/36 computers are linked together coast-to-coast
- Our department's budget has decreased over each of the last four years with a corresponding increase of service, quality and reliability
- Our operations is partly automated; we now have only one shift and three people which includes our operations manager.

Even though Keyes Fibre Company seems to be successful, the parent company is pushing a conversion of my company's 4341 to an AS/400. The conversion will probably take two years, cost $750,000, resulting in less capability.

Will you consider an article on converting from a 370 machine to an AS/400 or highlight a company that has made such a conversion?

John M. Murray
Keyes Fibre Company
Waterville, MA



3390 Disk Drive Delays

What exactly is causing the 3390 holdups is uncertain, but likely sources of delay may be the production of new platters or the head-to-disk assembly. Another strong possibility is that IBM is sitting on the disks until it can get the 3990 Model 3 controller to market; the advanced 3990 is not crucial to the operation of the new DASD in the short term, but it offers some important facilities that would greatly enhance this launch. In particular, the 3990's Dual Copy facility will prove to be a considerable boon to the new drives. *Source: Insight IBM publihsed by Xephon.*

Pre-printed Computer Lease Enforced

The Missouri Court of Appeals has overturned a trial-court ruling and held that a pre-printed standard from lease agreement for computer equipment was not an unenforceable contract of adhesion. Because the lease did not leave the lessee without a remedy, the court was not able to make a finding of "oppressive unconscionability." The decision is important to both software and equipment companies that use pre-printed standard agreements. It underscores the need to draft form contracts that contain reasonable terms and conditions, provide a remedy and take into account statutory consumer protections. *Source: Computer Law & Tax Report, published by Roditli Reports Corp.*

Researchers Make Guinness Book Of World Records

A team of six researchers at Amdahl Corporation (John Brown, Landon Curt Noll, B. K. Parady, Gene Ward Smith, Joel F. Smith and Sergio E. Zarantonello) has found the largest known prime number, a discovery worthy of the Guinness Book of World Records. The 65,087-digit number was found in the early-morning hours of August 6, 1989 as a result of the development of highly optimized algorithms and a combination of research, logistics and computer power.

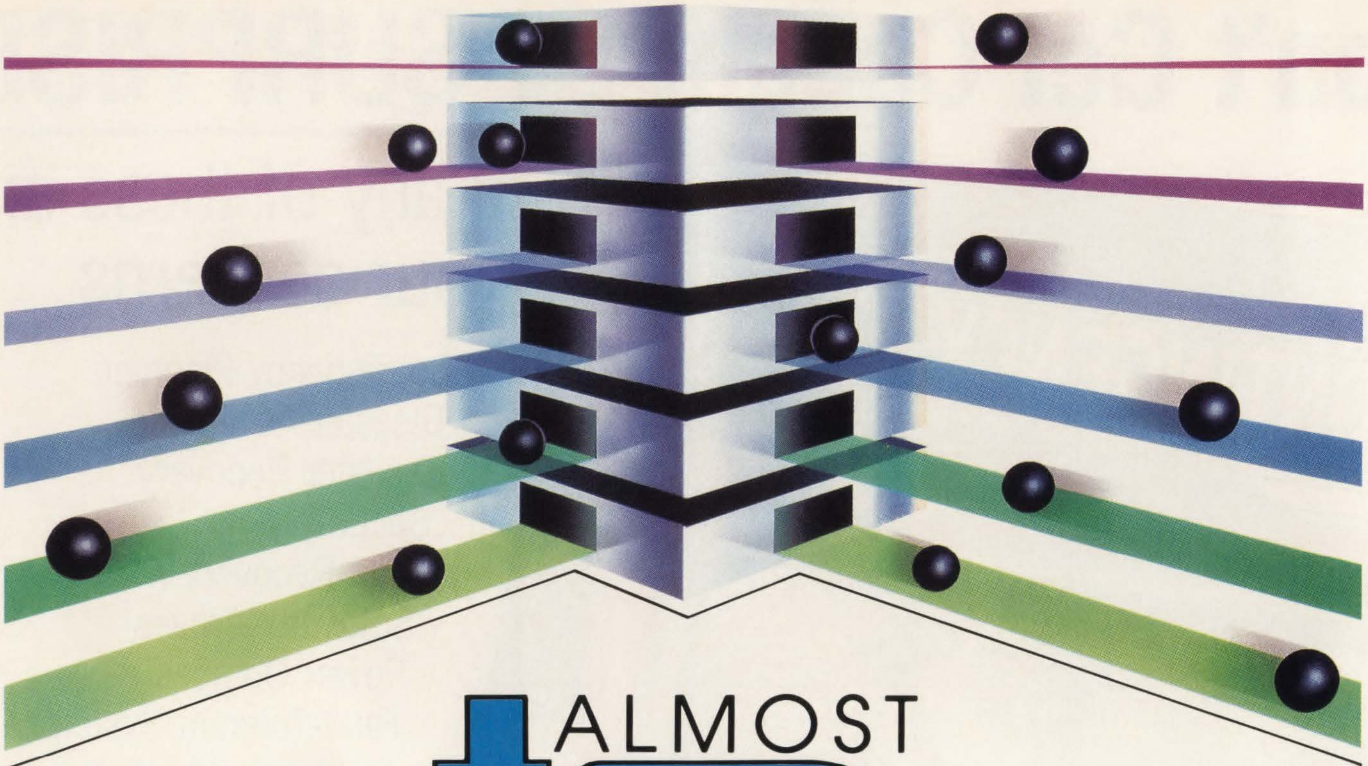
Finding the largest known prime number is the computing world's equivalent of climbing Mt. Everest. The number, at more than 65,000 digits, is unimaginably large. By comparison, the number of atoms in the known universe can be expressed in less than 100 digits. Its usefulness lies in the application of computer techniques used to find the prime number, some of which can be applied to speed computer programs for geophysicists in oil exploration, aeronautical engineering, weather forecasters' models and automotive design engineers and so on.

U.S. Department Of Commerce Honors BMC Software

The United States Department of Commerce has selected BMC Software, Inc. as recipient of the President's "E" Award for Excellence in Exporting. To earn the "E" Award, an organization's percentage of exports to total sales must exceed the industry average, both annually and over a four-year period. "The tremendous growth in our international organization is the result of BMC's strategy to expand aggressively outside North America," explains Richard A. Hosley II, President of BMC. "Last year we opened an office in Tokyo and a European Support Center in England. This past August we opened an Australian office in Melbourne and there are plans for expansion in Spain and Denmark before our fiscal year is over."

Help Desk Institute International Conference

The Help Desk Institute will hold its International Help Desk Conference February 11-14, 1990 at the Buena Vista Place, Walt Disney World. The Institute members are from large MIS organizations that have support staffs to help resolve hardware, software and communications problems and to perform analysis of problem history in order to reduce future problem occurrences. The conference will include industry leaders speaking on problem management, expert systems, problem resolution techniques, call management and the future outlook for support software. Highlighting the conference will be general sessions, multi-track special interest break-out sessions, vendor exhibitors, Birds Of A Feather (BOFs) and social activities. This conference is the single annual event that brings together leaders in Help Desk/Problem Management from industry and vendor organizations. For more information or to register, call (800) 248-5667.



ALMOST tso™

Almost anything TSO can do, *Almost* TSO™ can do better.

***Almost* TSO** supports dozens of users per address space instead of TSO's one. Edit files, submit jobs and preview sysout for a fraction of TSO's cost. Postpone expensive hardware upgrades. Prolong the useful life of your mainframe. No re-training is needed for ISPF PDF users. Save TSO for users who really need it.

Under TSO, *Almost* TSO™ makes the best better yet!

With ***Almost* TSO** and TSO together, you can edit files of **any size** and LRECL. Edit and browse multiple files concurrently. Cut and paste from edit, browse and even sysout files. The combination is unbeatable.

***Almost* TSO is a multi-user VTAM and/or a TSO application. Stand-alone or put together it's a winner. Call or write for all the details and ask about our 30 day FREE trial.**

Applied Software, Inc.

Quality Software Since 1973

840 U.S. Highway 1, Suite 250 • North Palm Beach, FL 33408 • (407) 626-4818

CIRCLE #71 on Reader Service Card ▲

Raising Your IQ (ISPF Quotient)

ISPF Techniques For Advanced Users

By David Shein

You have been using ISPF for a while now. For the most part, you like it. It provides a handy set of tools that are reasonably easy to learn and use. Also you have been using it long enough to find it reasonably comfortable. Even so, you sometimes find yourself performing tasks that are awkward or repetitive enough to make you wish for some productivity-increasing shortcuts. Maybe you have even heard passing references to things like "edit macros" and "command tables" and you wonder whether these mysterious mechanisms are things you should know about.

Does the above description apply to you? Read on!

The focus of this article is on ways to increase your productivity when using ISPF. I will assume you already know your way around ISPF panels pretty well and that you are interested in knowing about some advanced techniques that can help to leverage your productivity even further.

Getting The Most Out Of PF Keys

The default PF key actions which ISPF provides are mostly useful ones. However, the ISPF designers recognized that different users would be using the product in different ways. They, therefore, made it easy to change PF key settings. You have two ways of changing PF keys. Method one is to select option 0.3 from the main menu. Method two, the preferred method, is to enter KEYS in the command field of any ISPF panel. Doing either of these will present you with a panel containing the definitions for 12 PF keys; to change a PF key setting, just type over the existing setting. If your terminal is equipped with 24 PF keys (and you have so indicated on panel 0.1), pressing ENTER will show you the definitions for the other 12. You can continue to toggle back and forth between the two groups of PF keys by pressing ENTER. When you enter END (PF3), the current PF key settings will be saved and (assuming you use the KEYS command to reach this point) you will return to the task you were performing when you entered the KEYS command.

A PF key can invoke almost any ISPF command. One very useful one is FIND IEF376I. When you are examining the output from a batch job using option 3.8, SDSF or other such utility, this FIND command will scroll past the JCL and system messages and go straight to the beginning of the SYSOUT for your job. This can be a great time saver.

You can also set a PF key to invoke line commands in ISPF edit. You do this by prefixing the PF key setting with a colon (:). For instance, if you find yourself using the "insert" (I) line command frequently, you can set a PF key to the value :I and then invoke the command for whatever line the cursor is on by just pressing that PF key.

Before you begin rearranging all your PF key settings, a cautionary warning: do not change the settings for PF3, PF7 or PF8. The defaults for these keys are among the most heavily used commands in ISPF and experienced ISPF users rely on them with a trust that borders on instinct. If you change them, anyone who uses your account (possibly including you) may get confused.

One final note. PF key settings are stored in your ISPF profile dataset so that ISPF can remember them between sessions. Some of the optional products that run under ISPF, however, have their own individual profiles with their own individual PF key settings. SDSF is one example. Usually (but not always!) the default PF key settings for such products will simply mirror the ISPF defaults. This presents a *caveat* and an opportunity. The caveat is, "always be aware of which function you are using because the PF keys may not do what they do elsewhere." The opportunity lies in the ability to have PF key settings that are uniquely tailored to the active function. For instance, DA OJOB is a command that SDSF users tend to find themselves using frequently. So if you get into SDSF, enter the KEYS command and set one of the PF keys to DA OJOB. You can then invoke that command using a PF key but *only* in SDSF, which is the way you want it.

Edit Macros

Edit macros are pretty much what their name implies: prebuilt sequences of ISPF edit commands. Using the edit macro capability, you can define "canned" edit command streams to perform repetitive or complex functions. You can then invoke the macros during an edit session to perform the specified functions on demand, without having to key in the commands each time. As a general rule, an edit macro can issue nearly any edit command. There are also some additional functions to help you control the movement of the cursor while an edit macro is executing; this is important when, for instance, a macro needs to issue line commands.

F I G U R E 1

JC Macro

```
ISREDIT MACRO
ISREDIT LINE_AFTER 0 = '//TS0007   JOB  +
                         (accounting info),'
ISREDIT CHANGE " ' " .ZF .ZF ALL
ISREDIT LINE_AFTER 1 = '//                    CLASS=A,MSGCLASS=R'
ISREDIT LINE_AFTER 2 = '/*JOBPARM    R=room#'
ISREDIT LINE_AFTER 3 = '//*'
```

For MVS, an edit macro is just a CLIST beginning with the statement ISREDIT MACRO and containing edit macro commands. You invoke the macro by entering its name (that is the CLIST name) in the command field of the edit panel. Your macro may or may not require operands, depending on what it does. For more detailed documentation of the rules for coding edit macros, see the IBM manual, *ISPF Edit and Edit Macros (SC34-4121).*

Examine a few useful edit macros to find out what they can do. The first macro is called Job Card (JC). This macro automatically inserts a // JOB card at the beginning of the member being edited. The macro is shown in Figure 1. With this macro, you do not have to store JOB cards in your ISPF libraries and retrieve them with the COPY command. If you need a job card for the member you are editing, just type JC in the command field and the macro creates one for you.

The next two macros complement one another. They are called Truncate Forward (TRF) and Truncate Backward (TRB). Assume you are editing a large member and you want to delete all the data from a certain line onward. The "longhand" way of doing this, of course, is to place the cursor in the line command area of the first line to be deleted and type D99999. With the TRF macro, you can simply type TRF on the command line, place the cursor on the first line to be deleted and press ENTER. Everything from the cursor down will be deleted. TRB works in a similar way but in the opposite direction; it deletes everything from the cursor *up*, that is, toward the beginning of the member. These macros are shown in Figures 2 and 3.

The Magic Of Command Tables

When you want to interrupt a task temporarily while you do something else, split screen is the obvious answer. But what if both screens are already active and you do not want to terminate either one? Or, what if, for whatever the reason, it just is not convenient to use split screen to interrupt the current task?

Remember the KEYS command we used earlier? It allowed us to interrupt a task and return to it later *without* invoking split screen. The command table mechanism allows you to define your own commands which can be used in this way. Even better, your commands can invoke standard ISPF services. Consider the following examples.

Assume you need to save a member you are editing but in a different dataset. It would be ideal if you could split the screen, invoke option 3.2 and allocate a new dataset. Then you could use the CREATE command to save your data in the new dataset. But suppose you already have the other (split) screen active with some unrelated function. You are stuck unless you have a command that will suspend your current edit session, take you directly to panel 3.2 and let you return to EDIT afterward.

Figure 4A shows a command table entry that will do exactly that. Our command is called U32. If your command table contains this entry, then entering U32 in the command field of *any* ISPF panel will interrupt the current task and display panel 3.2. When you finish with panel 3.2, END (PF3) will return you to the previous task and you can resume where you left off.

Another example involves SDSF, a product used by many installations for viewing completed jobs to decide whether

FIGURE 2

TRF Macro

```
ISREDIT MACRO
ISREDIT (LINE,COL) = CURSOR
IF &COL = 000 THEN DO
   SET LINE = 000000
   ISPEXEC VPUT (LINE) SHARED
   SET &ZEDSMSG = &STR(POSITION CURSOR FIRST)
   ISPEXEC SETMSG MSG (ISRZ001)
   EXIT
   END
ISREDIT (LLINE) = LINENUM .ZLAST
IF &LINE NE &LLINE THEN —
   ISREDIT DELETE .ZCSR .ZLAST
ELSE —
   ISREDIT DELETE .ZCSR .ZCSR
IF &LINE > 1 THEN DO
   SET &LINE = &LINE - 1
   ISREDIT CURSOR = &LINE &COL
   END
SET &ZEDSMSG = &STR(OK)
ISPEXEC SETMSG MSG (ISRZ000)
```

FIGURE 3

TRB Macro

```
ISREDIT MACRO
ISREDIT (LINE,COL) = CURSOR
IF &COL = 000 THEN DO
   SET &ZEDSMSG = &STR(POSITION CURSOR FIRST)
   ISPEXEC SETMSG MSG (ISRZ001)
   EXIT
   END
ISREDIT (LASTLN) = LINENUM .ZLAST
IF &LINE NE 1 THEN —
   ISREDIT DELETE 1 .ZCSR
ELSE —
   ISREDIT DELETE 1 1
IF &LINE < LASTLN THEN DO
   SET &LINE = &LINE + 1
   ISREDIT CURSOR = &LINE &COL
   END
SET &ZEDSMSG = &STR(OK)
ISPEXEC SETMSG MSG (ISRZ000)
```

FIGURE 4a

U32 Command

```
VERB  T  ACTION
U32   0  SELECT PGM(ISRUDA) PARM(ISRUDA2)
```

FIGURE 4b

SDSF Command

```
VERB  T  ACTION
SDSF  2  SELECT PGM(SDSF) PARM(/&ZPARM)
```

hard-copy output should be printed.

The usual way to access SDSF is via a menu option. But SDSF is the kind of tool you often need to use while interrupting something else. For instance, you have just created some JCL and issued the SUBMIT command to run it. Since this is the first time you have run this particular job, you suspect it may contain errors. So, without leaving EDIT, you would like to invoke SDSF, inspect the output, leave SDSF and either save your JCL or else correct and resubmit it, as appropriate. The command in Figure 4B will let you invoke SDSF from any ISPF screen by keying SDSF in the command field. When you finish with SDSF, just press PF3 as many times as needed to return to the screen you invoked SDSF from.

There are a few mechanical details you need to know in order to set up your own command tables.

Command tables are stored in your ISPF Table library, which is a partitioned dataset that is allocated to DD name ISPTLIB in your TSO session JCL or logon CLIST. A command table is just a member in this library. ISPF imposes a strict naming convention for command tables. A command table name is of the form xCMDS where x is one to four alphanumeric characters. Since the command table's name is also its PDS member name, the name must adhere to standard PDS member name requirements; that is, it must begin with an alphabetic character.

The characters represented by "x" above are usually ISR; ISRCMDS is the command table for most of the built-in ISPF functions such as edit and browse. However, you will sometimes find that optional features such as SDSF have their own command tables. If this is the case in your installation, check with your local systems programmer to determine the names of the other command tables in your environment.

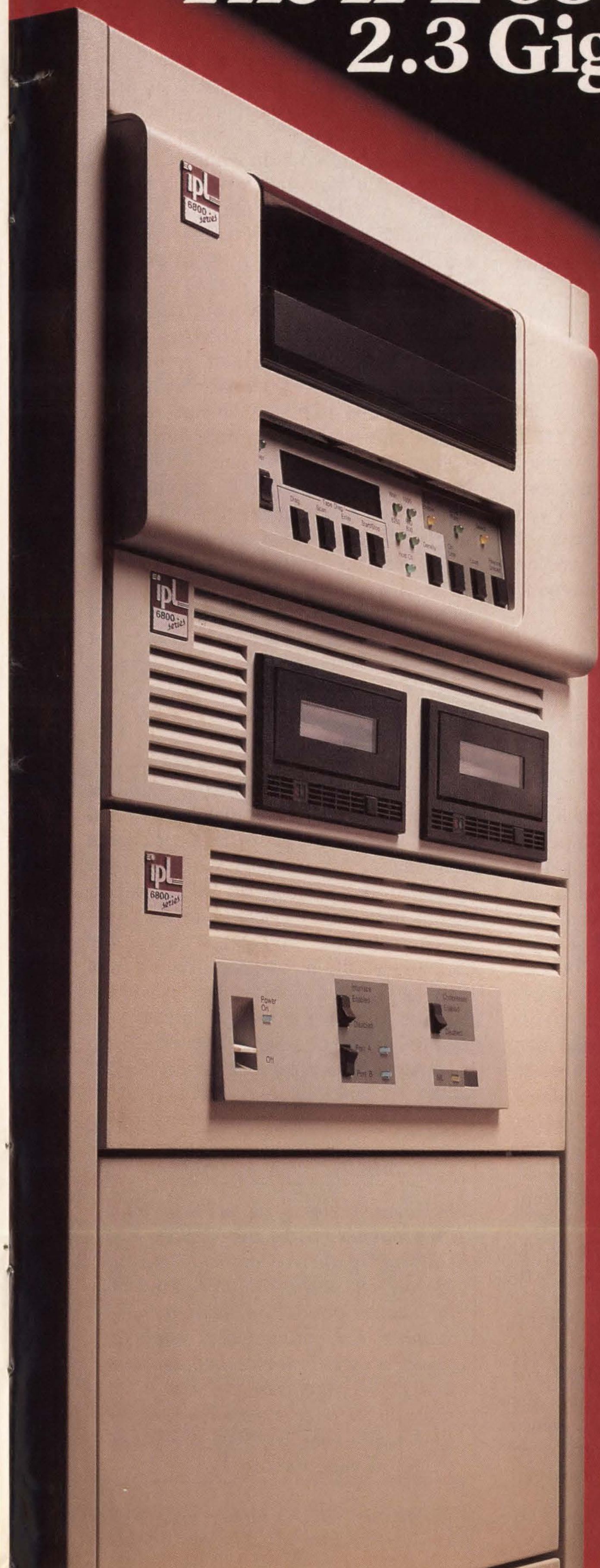
Once you have identified the command table to be modified, use ISPF option 3.9 to update it. This is a specialized edit function for command tables only. If you are accustomed to using ISPF Edit, you will find 3.9 pretty straightforward. Refer to the HELP tutorial if you get stuck. The examples in Figures 4A and 4B depict actual command table entries which can be keyed in using option 3.9.

The Double-Digit Utilities

When you choose option three on the ISPF main menu, you are offered a list of utility functions. Choices 12, 13 and 14 are recent additions to ISPF which provide some much needed functions. Actually, these three options all invoke the same program but with different parameters. The program is SuperC, a *very* handy utility which has been around for some time but which, until recently, was used internally within IBM and not made available to users.

The first option is option 3.14, the *search-for* function. When working with ISPF libraries, a common requirement is the need to search a library for a particular character string. For instance, you might need to scan a library of JCL to find all the places where a particular dataset name or DD name is referenced. Option 3.14 provides this capability, and *it is a colossal time saver.*

When first selected, option 3.14 displays a panel on which you can specify the library to be searched and the char-



acter string to search for. (An option on this panel lets you display a secondary panel on which up to nine additional strings can be specified.) You can search an ordinary sequential file, an entire library or just specific members. When you have keyed in the necessary parameters, press ENTER and SuperC will be invoked to perform the search. You can also specify where the listing containing the search results is to be stored. (This listing is displayed in "browse" mode on the terminal when the search is complete.)

This option can get a bit complex, so you may need to consult the manual the first few times you use it. It is well worth the effort!

Option 3.12 provides an interface to the *compare* feature of SuperC. This lets you compare two files or library members to determine where and how they differ. Option 3.12 is reasonably straightforward to use: you specify the library members and/or files to be compared, a listing file which is to contain the results and some processing options to control such things as the type of listing to be produced. You can also specify a *profile* dataset, which is a file containing detailed commands to control the comparison. This lets you override the defaults to specify some processing options, which are sometimes useful and which cannot be specified on the panel. For instance, the profile dataset can be used to control whether sequence number fields and comment lines are included in the comparison. (Profiles can be created with the Activate/Create function of option 3.13.) As with 3.14, the manual is an indispensable aid in learning to use this option.

Options 3.12 and 3.14 are limited in scope — they provide access to a useful subset of SuperC functions. Also available, however, is option 3.13, which gives full access to all of SuperC's features. Option 3.13 gives you more flexibility and options when performing search and compare functions and also offers some capabilities which are particularly useful to software developers. For instance, given two pieces of source code which we will call source A and source B, 3.13 can compare the two, determine where they differ and generate IEBUPDTE control statements that will transform source A into source B.

It is *strongly* recommended that you gain some practice with 3.12 and 3.14 before tackling 3.13. A study of the manual is an absolute must for using this extremely powerful tool.

One final point about these utilities. Both 3.12 and 3.13 give you the choice of performing the functions under ISPF, that is, while you wait, or submitting a batch job to run SuperC in background.

Miscellaneous Time And Work Savers

The LOCATE command is one you know well if you have used ISPF for any length of time. But this command has a handy option you might not be aware of.

Has this ever happened to you? You are editing data with the ISPF editor and you enter a C ("copy") or M ("move") line command on some data line. For one reason or another, however, you never type an A ("after") or B ("before") elsewhere in the data to complete the operation. Later, when you press PF3, the editor displays the message MOVE/COPY IS PENDING and refuses to terminate. Now you have to locate the line that contains the uncompleted command so you can resolve it. If your data consists of several hundred lines (or more), finding the offending line can be both annoying and time consuming.

LOCATE to the rescue! If you enter LOCATE CMD and press ENTER, the editor will find and display the first or next line that contains an unresolved command. You can then deal with the "orphan" command appropriately and continue on your way.

RETRIEVE is a command which has been added to ISPF recently and it is one of those "why-didn't-they-think-of-this-sooner" jewels. RETRIEVE, which can be entered in the command field of any ISPF screen, will redisplay the last thing entered in the command field. You can then press ENTER to re-execute the command or, if the command contains an error, you can correct it right there and *then* press ENTER. This is a real boon when you have just entered a long, complicated command which contains an error. Rather than key the whole thing in again (and maybe commit another error), you can use RETRIEVE to bring back the entire command so you can correct and retry it. RETRIEVE is so useful that it is now the built-in default setting for PF keys 12 and 24.

Option 3.4 is one that experienced ISPF users know and love. But this already-useful option has been enhanced to provide even more flexibility. When the screen contains a list of dataset names generated by 3.4, you can, of course, type action characters next to the displayed names to invoke functions such as edit, browse, delete and so on. But you are not limited to the built-in functions! Option 3.4 also lets you type the name of a CLIST next to the dataset name. The named CLIST will then be invoked and the dataset name will be passed to it as a parameter. This lets you extend the functionality of option 3.4 to a degree limited only by your creativity in writing CLISTs.

One installation has written a CLIST named "A" (short for "allocate"). When this CLIST is invoked by typing "A" next to a dataset name in a 3.4 list, it executes the LISTDSI command of TSO to acquire the attributes of the specified dataset. It then displays a screen, which is similar to the "allocate" screen of option 3.2, with the attributes of the specified dataset filled in. The user can change the displayed attributes, if desired, by simply typing over them. When ENTER is pressed, the ALLOCATE command of TSO is used to allocate a new dataset using the attributes shown on the screen.

The result of all this is to give the user the ability to easily allocate a new dataset using a dataset selected from a 3.4 list as the model. This is just an example of the flexibility available when writing your own CLISTs to be executed under option 3.4.

Concluding Thoughts

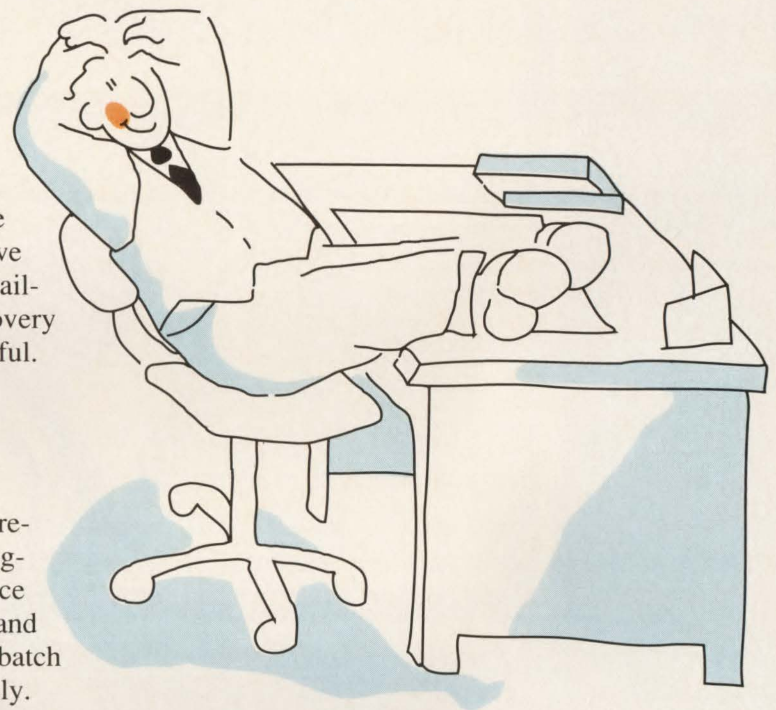
This article has attempted to highlight the enormous power and flexibility available to seasoned users of ISPF. It should be noted in passing that we have only scratched the surface of ISPF's labor-saving potential, especially in the areas of edit macros, command tables and CLISTs for option 3.4. Recent releases of ISPF have added features that let you extend the built-in functions to create your own unique labor-saving tools. ISPF has become such an open architecture that its ability to multiply your productivity is limited only by your own creativity.

ABOUT THE AUTHOR

David Shein is an MVS systems programmer with the Automobile Club of Southern California (Costa Mesa). The technical staff to which he belongs provides system software support and maintenance, capacity planning, performance management and DASD management. The environment supported includes multiple IBM mainframes, MVS/XA, CICS and a 3000-terminal network.

Now You Can Relax . . . CICS/VSAM Data Recovery Has Been Automated!

Data Recovery WAS Stressful



Traditional methods of data recovery are complex, and sometimes they don't work at all. Should data loss or corruption occur, you must start with such questions as . . . Do we have backups for this application, and where are they? What about the CICS and batch updates since the last backup? Will we have to re-key? Even if all recovery elements are available, the process of planning and coordinating a recovery is time consuming, prone to errors, and often stressful.

Integrity Solutions ELIMINATES Data Recovery Stress

Integrity Solutions, Inc. (ISI) has developed a comprehensive family of journal management and data integrity products to eliminate data recovery stress. Once you've installed the **Integrity Solution**, lean back and relax. Because you are assured that your CICS and batch VSAM data will be recovered quickly and accurately.

Automated Recovery: ISI's ICS/MANAGER automates your recovery by registering and managing all critical backup, journal and recovery information on a global Control Dataset. Customized JCL is **automatically** generated and submitted when needed for a recovery to save you valuable time and resources. ICS/MANAGER may be combined with any or all of ISI's other products to automate the recovery process.

Online Recovery: ISI offers two alternate online recovery methodologies. They provide fast and accurate FORWARD or BACKWARD recovery between backups for VSAM files updated by CICS/VS. One is unique in providing **24-hour/day availability of VSAM files to CICS** while the backup process takes place.

Batch Recovery: ISI provides comprehensive recovery for batch environments by transparently journaling VSAM updates. Job restart and re-run time is reduced, and redundant backups can be eliminated. This helps assure that your batch processing is completed on time.

Automatic Journal Archiving: You'll eliminate accidental overwrites of CICS journals and logs with automatic journal archiving. In addition, your entire CICS job submission process is managed and enhanced with numerous extended features.

All of Integrity Solutions' products are designed to work together to provide system-wide data integrity. However, each product may also be purchased separately to address your specific requirements.

And, if you need assistance at any time, our experienced technical support staff is at your service, 7 days per week, 24 hours per day.

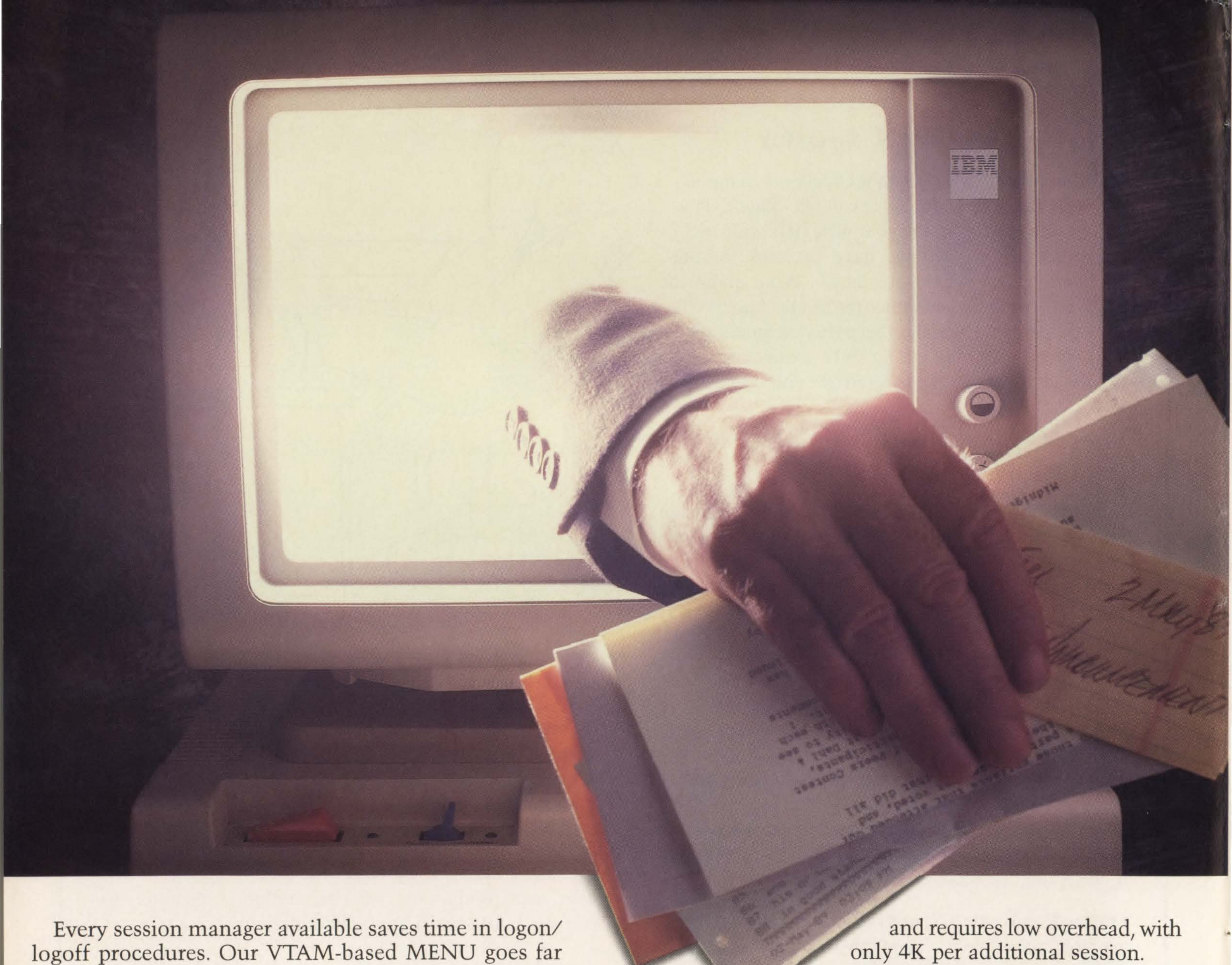
For more information, or to begin your free, 30-day trial, please call **1-800-289-9900** or **(303) 794-5505.**

Integrity Solutions, Inc.
7921 SouthPark Plaza
Littleton, Colorado 80120

Integrity Solutions . . . Setting the Data Integrity Standard

CIRCLE #125 on Reader Service Card ▲

If you're looking for a session manager that delivers more than multiple sessions, look into Menu. It also delivers the mail.



Every session manager available saves time in logon/logoff procedures. Our VTAM-based MENU goes far beyond this.

With its exclusive, built-in electronic mail feature, MENU from DTA lets you send messages to users or user mailboxes with incredible speed and ease. So memos, notes, requests, confirmations and the like are delivered with less paperwork, confusion and delay. And there's more.

MENU from DTA features automatic 3270 screen data compression, flexible security and authorization, auto logon/logoff, and complete on-line help screens. It also has single or multi session capabilities, is easy to install and requires low overhead, with only 4K per additional session.

To find out more about the many cost and time saving features built into MENU, or for a free 30-day trial, give us a ring soon. When it comes to effective session management, nobody delivers more.

DTA SOFTWARE... More efficient in every detail.

DTA®

Software Products Group
550 Waterford Park
505 North Highway 169
Minneapolis, MN 55441
800-521-6773/612-591-6100

CIRCLE #129 on Reader Service Card ▲

Automated Operations

Do Not Automate What You Can Eliminate

By Jeff Murrell

Much of the attention of computer operations managers over the past few years has been aimed at the subject of automating mainframe operations. You can read article after article, attend conference after conference and review product after product that is aimed at the automated operations market. Before committing yourself and your company to the high price of automation products and the resources it takes to implement them, there is much that can be done in the direction of *eliminating* functions, rather than automating them.

Many of the tasks that are performed by operations personnel are *non-value added* in terms of the product or service being provided to the customer. Why, then, is automating these functions considered before their need to be performed at all is examined? The automation of a task renders it *invisible* to the organization; nevertheless, it perpetuates the overhead (albeit reduced to machine speed) of performing that task. Elimination of tasks can and should be considered as the first phase of any automation strategy.

Nor is the concept of elimination of tasks limited solely to the mainframe console area. It can also successfully be applied in the areas of magnetic tape operations, output media handling, production (batch) processing and the Help Desk. This article will examine each of these areas in turn and will discuss ways to implement projects aimed at identifying those tasks which can be eliminated as a prelude to implementation of an automation program.

Involve The Supplier

The console operations area is perhaps the richest in terms of the availability of products to automate tasks and the number of different tasks that can be automated. Many IBM data centers have already utilized the Message Processing Facility of MVS to suppress some of the more obvious redundant messages. You can suppress a high percentage of messages which operators already ignore and automate what amounts to a "rubber stamp" response to many console requests (WTORs and the like).

As you determine which messages can be suppressed or automatically actioned, you should be feeding your suppression and automatic action lists back to hardware and software suppliers with a strong request that the base product be modified to eliminate these messages. Modification of the product will not be an overnight event and you will no doubt have to proceed with interim implementation of suppression or action rules through automation software. However, only when your suppliers begin getting this feedback from their customers can you begin to expect improvements.

To analyze console traffic, your system logs are an excellent source of information as are your operators themselves. A Pareto analysis is a tool used in quality improvement programs to identify the next logical problem to be addressed. Causes of activity are listed in descending order of frequency. It will usually be found that the top 20 percent of causes generate 80 percent of the occurrences. Thus, working on those more frequent causes will yield the more productive result. The technique is named after Vilfredo Pareto, who identified the principle of the "vital few" and the "trivial many." Figure 1 demonstrates the use of a Pareto chart in analyzing Help Desk activity.

Create a list of messages for each application, from the most frequent to the least. Begin at the top, examining which messages are unnecessary, which are automatically actionable and which are valid alerts requiring human intervention. When this exercise is completed, you will have an excellent document with which to feed your automation product, but more importantly, one which you can take to your supplier with a request for elimination.

The very number of consoles is another area for consideration. Where is it written that you must have a separate console for each system, each application and in some cases, a backup console "just in case?" Many automation products already address the capability to integrate the functions of several consoles into one operator workstation.

Professional workstations offer the opportunity to direct many *logical* consoles into one hardware device. Windows may be used to allow the operator to access many systems or many applications from one keyboard and one monitor. Further advancement is required to allow a window automatically to be activated when an alert is sent from the host system or application, interrupting whatever is currently executing. Without this advancement, the operator's role continues to be that of actively monitoring multiple host environments. However, the advent of multi-processing workstations should soon facilitate passive monitoring, whereby

FIGURE 1

Pareto Analysis Of Help Desk Calls

Call type	Number	Percent
Batch job status	43024	27.2
Workstation/terminal hardware problem	37570	23.8
Network access problem	20582	13.0
Security (e.g. lost password)	13881	8.8
Workstation software problem	4560	2.9
.....	....	...
.....	....	...
.....	....	...
Telephone problem	3020	1.9
Request for technical advice	2252	1.4
Other	7569	4.8
Total	157909	100.0

human intervention is only required when an alert is signaled. Add to this the ability to automatically telephone (or page) a support person, plus the ability of that person to emulate the operator workstation from a remote PC and you have the elements of a true lights out or unattended operation.

Console consolidation does not end with the mainframe and application monitoring functions. The operator workstation should also be designed to incorporate access to the external environment control system (air, water, heat, power), to the data center security (access control) system and to channel switches. Most such systems today run on separate, usually mini- or micro-based hosts but are capable of being integrated into a single workstation.

Migrate Small Tape Datasets to Disk

The tape library function is served by only a handful of automation products. Tape mount activity must certainly be the most mundane of operations tasks. As such, it is the easiest to consider automating and perhaps the least obvious to consider eliminating. However, if you analyze tape mount activity (via system logs), a likely discovery will be that a high percentage of mounts result in creation of or reference to a dataset that is less than 10MB in size (only five percent of the capacity of a single 3480 cartridge). With Direct Access Storage Device (DASD) technology continuing to reduce the cost per megabyte of on-line storage and with the cost of human resources for tape library services increasing, it now becomes economically feasible to store such datasets on disk.

If the dataset is accessed frequently (once per month or more), you will find that the cost of tape storage, mounts and processing often exceeds that of DASD. Even if the dataset is accessed less frequently, a facility such as IBM's DFHSM will ensure that it is returned to tape at an appropriate time. Now, however, you will be using the full capacity of a 3480 and making the maximum use of each tape mount, since many small datasets will be archived to the same physical cartridge.

Professional workstations offer the opportunity to direct many logical consoles into one hardware device.

To execute a project aimed at tape to DASD migration will require the involvement of your application systems groups and some special treatment of requests for sequential storage space. Tape has the characteristic that it is relatively *unlimited* in terms of how much data can be written once the dataset has been allocated. You can simulate this capacity by dedicating a pool of DASD to a new esoteric name (OLTAPE, for example). The customer's JCL can be changed to request OLTAPE with an indication of the logical record length, number of records to be written and retention period. Such a modification will also position the job for easy migration to Systems Managed Storage (DFSMS) when that IBM product is fully implemented.

Internal system exits can calculate the amount of primary and secondary disk space to allocate the dataset. With a pool of OLTAPE DASD large enough to accommodate all tape datasets below the 10MB limit, the customer should never experience an out-of-space abend. Further system modifications can be employed to print tailpage warnings when a job requesting OLTAPE has exceeded the 10MB limit or when a job requesting TAPE has written less than 10MB.

For jobs which read tape datasets, one can simply disable the UNIT=TAPE interpretation, leaving the system to determine from the dataset name whether or not that dataset now resides on real tape or OLTAPE.

The performance of sequential disk datasets is measured in milliseconds, rather than the minutes it takes to allocate a tape unit, obtain a mount and access the data. If performance and cycle time improvement is not sufficient incentive for the customer to migrate, pricing can be a further inducement. You should be able to price OLTAPE dataset space so as to demonstrate meaningful cost savings once tape storage and mount costs are compared.

From Less Paper To Paperless

The output media area is probably the most difficult to consider, both in terms of automation and elimination. Roll feed devices, burster-trimmer-stacker devices and advanced function printing all lend themselves to automating the manual tasks associated with managing the non-impact print function. However, in the end there are still people required to feed the "printing press" and to separate, bag and distribute the output. The paper culture is still alive and well in most corporations, so it will not be easy to promote a program of elimination. Many tactics may nevertheless be employed to reduce the load.

An on-line reporting system is a must. Most recipients of paper output have access to a terminal. If the output from their application systems is loaded to an on-line reporting system, they have immediate access to their results without the delays associated with print output and distribution. With appropriate indexing capability, keyed, for example, by a department number printed in the page

headings of the report, on-line reporting also allows direct access to the required portion of a report. Further, one copy of the report may be simultaneously read by many "recipients" from their terminal.

The implementation of an on-line reporting system may also involve some modification in the application systems. Reports which were designed for a 132 column by 66-line medium may require redesign for an 80-by-24 medium. The additional rows and columns do not need to be eliminated, since the on-line reporting system should allow scrolling for wide and long pages. However, the report might usefully be redesigned so that the most needed information is contained in the leftmost and uppermost portions of each page. Alternatively, the program could be altered to key off the medium and eject pages after 24, rather than 66 lines.

After on-line reporting, the next requisite is a good report distribution system. This system will receive the output of the application and will be capable of breaking a complete report into individual, smaller reports again keyed by data in the page headings. Each smaller report will contain only that information required by a particular recipient. The report distribution system should be driven by a database that knows each smaller report by name, who receives that report and whether that recipient requires on-line, printed or microfiche (one-day optical disk?) representation of that report. This database should be available to your customer via on-line inquiry, so that modification and deletion of requirements is a simple task. Again, the cooperation of the customer and the application systems groups will be required to successfully implement these concepts.

The third approach to print elimination will require sensitive management. Each recipient of a report needs to be challenged as to the ongoing need for the information. In many cases, unless the report distribution system is tied to the corporate human resource database, it will be discovered that the original recipient is no longer in the position held when the report was first implemented. Possibly that person no longer needs the report in his or her new capacity or even no longer works for the company. Alternatively, you may find that the person who now holds that position no longer requires the information. In many cases it has been easier for the new recipient to use the round file than to figure out how to cancel the report!

A survey periodically attached to each printed report is one way to motivate the recipient to reconsider the need. However, the survey must be accompanied by a simple, on-line means of acting on a decision to cancel. Another place to find unused reports is in the distribution area itself. Reports which are not picked up within a week of printing should be canceled via the distribution system. Again, if the recipient suddenly realizes something is missing, it should be a simple matter to reinstate the output via on-line access to the distribution system.

Correct Abends Before They Happen

Production (batch) processing is one area that has been well served by automation products for some time. Many data centers have already implemented library management and scheduling systems whereby application groups can register, validate and schedule job streams to execute at various frequencies without manual intervention. The area that has not been extensively addressed is what happens when something goes wrong. An abend usually requires that someone be alerted to take some action before the job can be rerun and further manual action to effect the restart. Such activity is usually required during second or third shift, which automatically extends the recovery cycle while the programmer is located (and awakened!). Automation of some abend recovery is possible. However, a project aimed at elimination of the cause of the abend will yield longer term benefits, not to mention more satisfied customers (and fewer bleary-eyed programmers!).

Start by generating a Pareto chart of the most common causes of production abends over some time period (several months). Begin at the top of the list and identify the application or technical support group (or supplier) whose product or service needs to be modified to eliminate the root cause. For example, many application systems will abend on finding that the expected input dataset is absent or empty. Electronic Data Interchange (EDI) applications are susceptible to this problem, since the external data source may or may not have data to transmit on the given day. The application could be modified to proceed without that particular input, to cancel itself (without abend) until the next run cycle or to reschedule itself later in the day, giving more time for input data to be accumulated. If the missing dataset is simply an error, the scheduling system could be modified to verify its existence at the time the job is placed on the schedule (a simple JCL scan is required). Advance warning could be provided to the application group, during prime time, enabling recovery of the dataset before it causes an abend and without that middle-of-the-night wake-up call.

With good monitoring tools, many other abends, such as time-outs, database full and out-of-space conditions and others can be foreseen, corrected and eliminated before they become a problem.

Help The Help Desk

The Help Desk is perhaps not the most obvious area to consider in terms of eliminating a task. Again, though, a Pareto analysis of the causes of Help Desk calls will yield a list of projects which can be undertaken to reduce the workload. Many calls may be addressed by placing the means to obtain the required product or service or to correct the problem in the hands of the caller. The banking industry has paved the way with its extensive use of automated teller machines and touch-tone account access.

Take, for example, the caller who wants to know the status of a batch job submitted some time ago. A simple inquiry can be developed that will enable that customer to interrogate the job entry subsystem directly and to receive a response indicating whether that job is running and, if not, where it is waiting.

Consider also the caller who simply needs information on how to execute an on-line inquiry. Why is that information not available at the touch of a function key? The developer of the inquiry should be encouraged to include context-sensitive help, such that if a given function key (F1 according to Systems Application Architecture's Common User Access standards) is depressed, a call is made to display the on-line documentation for that panel, without losing the contents of the original screen. To be even more critical, why is any kind of explanation even necessary? Could the inquiry developer alter the panel design so as to include more "user friendly" instructions and selection options? In such a way, the need for even help functions can be eliminated, not to mention the frustrated call to the Help Desk.

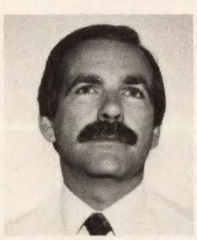
Conclusion

In all of the functional areas discussed, complete elimination of the operator's tasks is difficult, if not impossible to achieve, at least today. Some amount of console traffic, even if only alerts, will require automation and some will continue to require operator (human) attention. Tape is here to stay for some years to come and complete automation is a difficult economic proposition. Printed output cannot even be automated beyond the distribution area and the paperless society is for another generation. Some production abends will continue to require intervention and many Help Desk calls will require either automation via a voice response unit or the personal attention of a knowledgeable operator.

Before you spend too much time and money automating, however, try to minimize that which needs to be automated. At least then you will not be hiding the problem and perpetuating the overhead of dealing with it. You will also be providing a more cost-effective, available and higher-quality service to the data center customer.

The object of automation is quality. One way to improve service and product quality is to attack the number of non-value-added tasks that go into providing the product or service. This article has attempted to highlight some of the relatively simple projects that can be undertaken in any data center to minimize the number of tasks and to reduce the amount (and cost) of automation that must be deployed. It is hoped that such an approach will lead to a much more efficient operating environment, a more cost-effective utilization of automation technology and a more satisfied data center customer.

ABOUT THE AUTHOR



Jeff Murrell is Manager of Operations and Customer Support for the Information Systems and Services Division of Texas Instruments Inc., Plano, TX. Murrell has a 24-year career in systems, both in Europe and the U.S., the last four of which have been in systems development and now operations in an IBM mainframe environment.

Find Time For Your Expanding Role As A Database Administrator

By Robert Engle

Throughout the evolution of data processing, the role of the database administrator has been defined, refined and dramatically expanded. This area of responsibility has followed the trend from generalist to specialist like many other data processing careers. The days of the one-man band are gone and have been replaced by a symphony of specialists. Among these is the Database Administrator (DBA).

Evolution Of The Specialist

The industry has witnessed a shift of the responsibility for database administration from the lead programmer to the analyst to the database specialist. The function of database administration may appear under a number of titles throughout different companies, but each is basically the same. Titles such as database administrator, database analyst, database specialist, data analyst, data specialist and a whole array of others have surfaced throughout companies spanning the nation and the globe. For simplicity's sake, these titles will be referred to collectively throughout the remainder of this article as DBA.

The Database Expert

In the early days, a DBA in a large data center was most likely to be trained extensively in the hierarchical approach to database management, namely IBM's Information Management System (IMS) and Data Language 1 (DL/1). Even more, the DBA's concentration was centered around the logical design and physical implementation of the database structure.

The DBA was the primary contact and expert for all matters concerning the database. The programming staff was merely concerned with the DL/1 calls to access the database and was not concerned with the physical organization or structure of the database. It was the responsibility of the DBA and not the programmer to determine the most efficient method of retrieving the data needed by the program. This responsibility fit neatly into the structured programming and process-driven design methodologies promoted by Yourdon, DeMarco and others.

Technology and time-saving tips aid the DBA

These new methodologies included design and program *walkthroughs,* a method for individuals other than the author to review the program design and/or code. The DBA held a pro-active role in the design phase of a program or set of programs. It became a well-known fact that a good design and the proper amount of work upfront could prevent both logic and performance problems in the future.

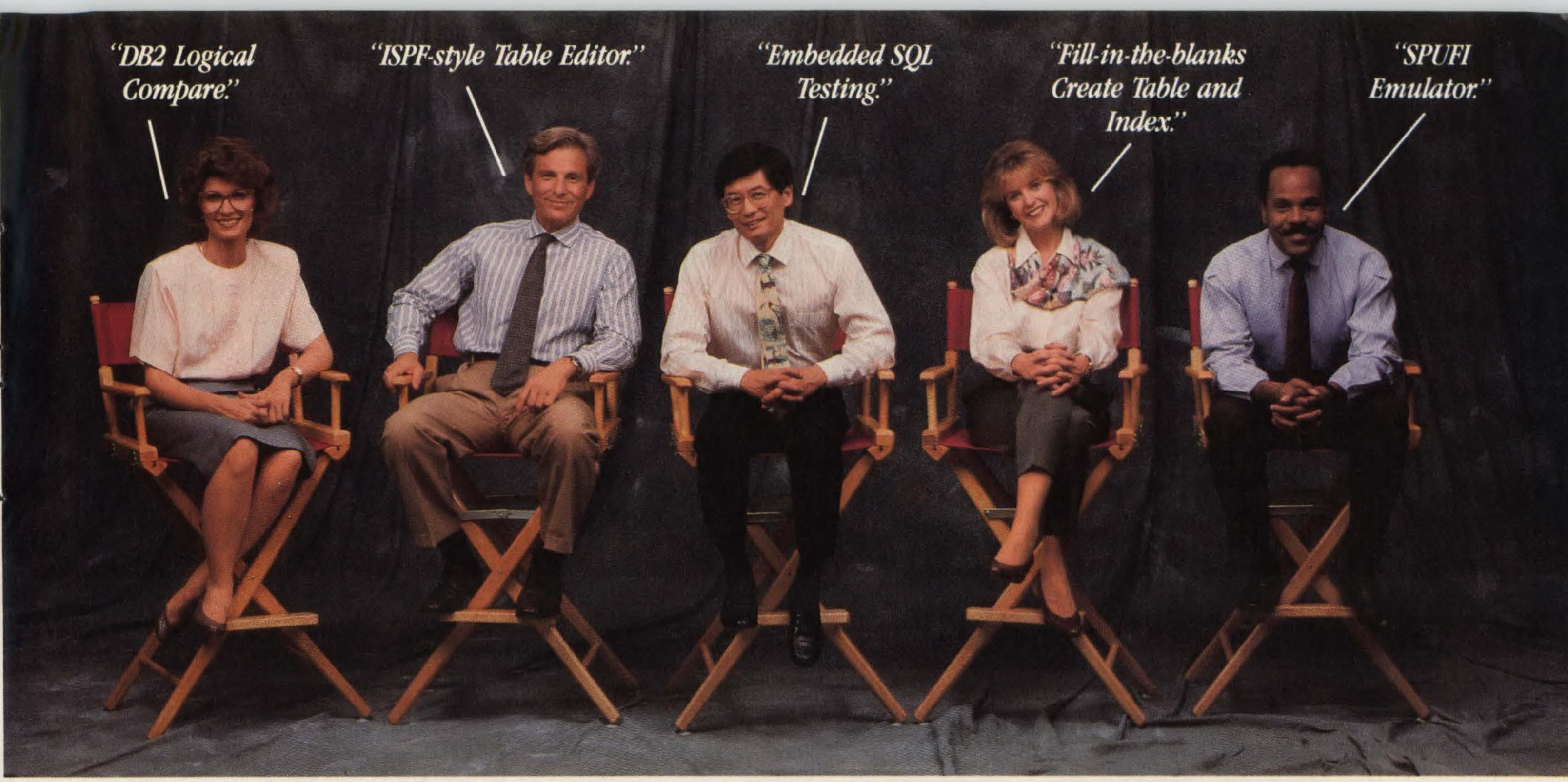
Performance And Tuning

In addition to maintaining the data used by these processes, it also became necessary for personnel to ensure that the level of performance of the applications accessing the data remained acceptable. As the use of interactive systems grew, performance of the applications became more critical than ever. Performance had to be measured and analyzed. Both good and bad performances had to be recognized. The good performance was used as a model and the bad one had to be improved to closer meet the standards set by good performance.

The IMS systems programmer was initially responsible for all aspects of performance of the IMS system and the applications accessing the IMS system. The trend has been to shift the responsibility and control of performance of the IMS system and its applications to the DBA. The IMS systems programmer still usually retains the responsibility for the MVS tuning considerations that affect IMS, but the DBA is now accountable for performance and tuning considerations within IMS.

System Maintenance

As the quantity of the data grew and the number of users accessing the data multiplied, the number of applications increased as did the interfaces between the applications. Suddenly, maintenance of the IMS system became a full-time job. In addition to performing the IMS control block changes, all changes to the system,



Why More Programmers Prefer ProEdit® For DB2 Application Testing.

The reasons include all of the above, and more. Just ask any user.

Programmers tell us ProEdit's ISPF-like interface speeds all their daily tasks, from building a DB2 test environment, to manipulating DB2 data, to testing SQL application code, and reporting on test results.

It gives them so much functionality, they've made it the industry leader for DB2 application testing. In fact, ProEdit is installed in more DB2 shops than any other DB2 product on the market.

And ProEdit continues to set the standards for DB2 application testing. For example, ProEdit now provides the industry's only DB2 Logical Compare Facility.

This new facility eliminates the tedious chore of manually comparing "before" and "after" images of DB2 tables following an application test. ProEdit does it automatically—and displays any differences for you.

VENDOR MEMBER
INTERNATIONAL DB2 USERS GROUP

A recent user survey revealed ProEdit increases DB2 application testing productivity by an average of 33%! That's because ProEdit was developed with the DB2 application programmer in mind—and product enhancements continue to be user-driven.

Plus, it's the only DB2 product that interfaces with the most comprehensive DB2 object management system around—ProAlter™/Plus.

Like all our products, ProEdit is offered with a lifetime trade-in guarantee so that the money you spend today is always available to meet your changing needs tomorrow. That makes us "The Safe Buy."

Call today toll-free **800-642-0177** for details on ProEdit and our full line of DB2 products and services. Or write us at Two Executive Drive, Fort Lee, NJ 07024.

On·Line Software
INTERNATIONAL

The Safe Buy.

CIRCLE #196 on Reader Service Card ▲

the applications and the databases needed to be coordinated and closely controlled. The task of modifying the IMS system control blocks became less technical and more administrative, thus came the switch of responsibility for the IMS SYSGEN process from the systems programming staff to the DBA staff.

Hardware technology progressed and system software development skyrocketed. Suddenly, many new features were being added to the IMS system software. IBM began to target its clients' industries and introduced functions such as Fast-Path, Multiple Systems Coupling and Intersystems Communication.

These new sophisticated software features required both an understanding of how the system software had functioned previously and an appreciation for how the applications already interfaced with the software. The DBA was expected to maintain the expertise for these new features in shops where they were used.

Availability And Recovery

Companies slowly became dependent on data processing for their business operations. Many employees could remain totally unproductive if the system or its applications were unavailable. The task of recovering from an outage, whether it be the database, the application or the system, became critical.

In addition to the client-oriented features of the software, IBM also introduced features within the IMS system software to aid the DBA staff in speeding the recovery process, including Database Recovery Control (DBRC), DASD logging, off-line dump formatting and Extended Recovery Facility (XRF). It became a requirement for a DBA to understand these features because they are crucial in expediting a recovery from an outage.

Together with concerns about the on-site recovery process came floods of concerns about how a company would survive if its computer operations became unavailable indefinitely, as would be the situation after a natural disaster. If a company were not prepared, a natural disaster could put it out of business. The ability to recover from a disaster became a high priority item on many corporate strategic plans. The DBA plays an integral part in every *good* disaster recovery plan.

CASE Technology And DB2, Front And Center

The applications development and production implementation cycle stabilized for some time. The bugs in the traditional approach had been pretty well worked out, but development of new applications was demanding on corporate budgets. Vendor-developed applications did not always foot the bill in terms of a company's operating needs and internal development of applications could not easily be cost justified. Thus came the birth of Computer Aided Software Engineering (CASE).

Although CASE technology remains in its infancy, there are a number of products on the market which are being used by many companies. The use of analysis and design tools, fourth-generation languages and application code generators have all dramatically changed the way applications development is performed and each is having a major impact on the responsibilities of the DBA.

At the risk of sounding trite, the last but definitely not the least contributor to the increased responsibility being placed on DBAs has been the maturation of the relational database technology, namely IBM's DB2. Many DBAs who were formerly the IMS experts are being used to support DB2.

Although IMS can be rated far more complex than DB2, DB2 may seem like an adventure through an unknown world, much like Dorothy and her friends in search of the Emerald City. Many of the concepts are familiar yet different, the way the Tin Man, Lion and Scarecrow were familiar yet strangers to Dorothy. The IMS DBA may search far and wide for difficult and highly technical explanations that may be brief and simply explainable within DB2.

Suddenly, the DBA has a multitude of responsibilities and roles which must be exercised on a daily basis. Some of these require more time than others, some require a large learning curve and some may only be exercised periodically. The DBA has become inundated with tasks, a to-do list that grows daily, multiple number one priority items and less time to spare.

Taking Advantage Of Technology

Fortunately, the rapid advances in technology are bringing a number of new Information System Staff products to the marketplace. CASE technology is a tremendous example. Beyond CASE technology, there are a number of other products on the market which will increase your productivity and give you greater flexibility in satisfying requests and solving problems. If you do not have any of these products available to you, then you should immediately investigate automation products.

Tools which aid you in the *design* process are a requirement for development of applications in this day and age. The old *flowchart* and *data flow diagram* templates should now reside in museums and not in your desk. There are too many good and cost effective products available for both mainframes and personal computers for you to be using the old manual drawing methods.

CLISTs To The Rescue

One product that is available in every shop having TSO is ISPF Dialog Management Services. You should become familiar with the product as soon as possible.

ISPF Dialogs are both easy and quick to write. They will be your first step to automating tasks and functions which you consider mundane. Even the most basic ISPF Dialog Services will allow you to gain great advantages over manual methods of completing tasks.

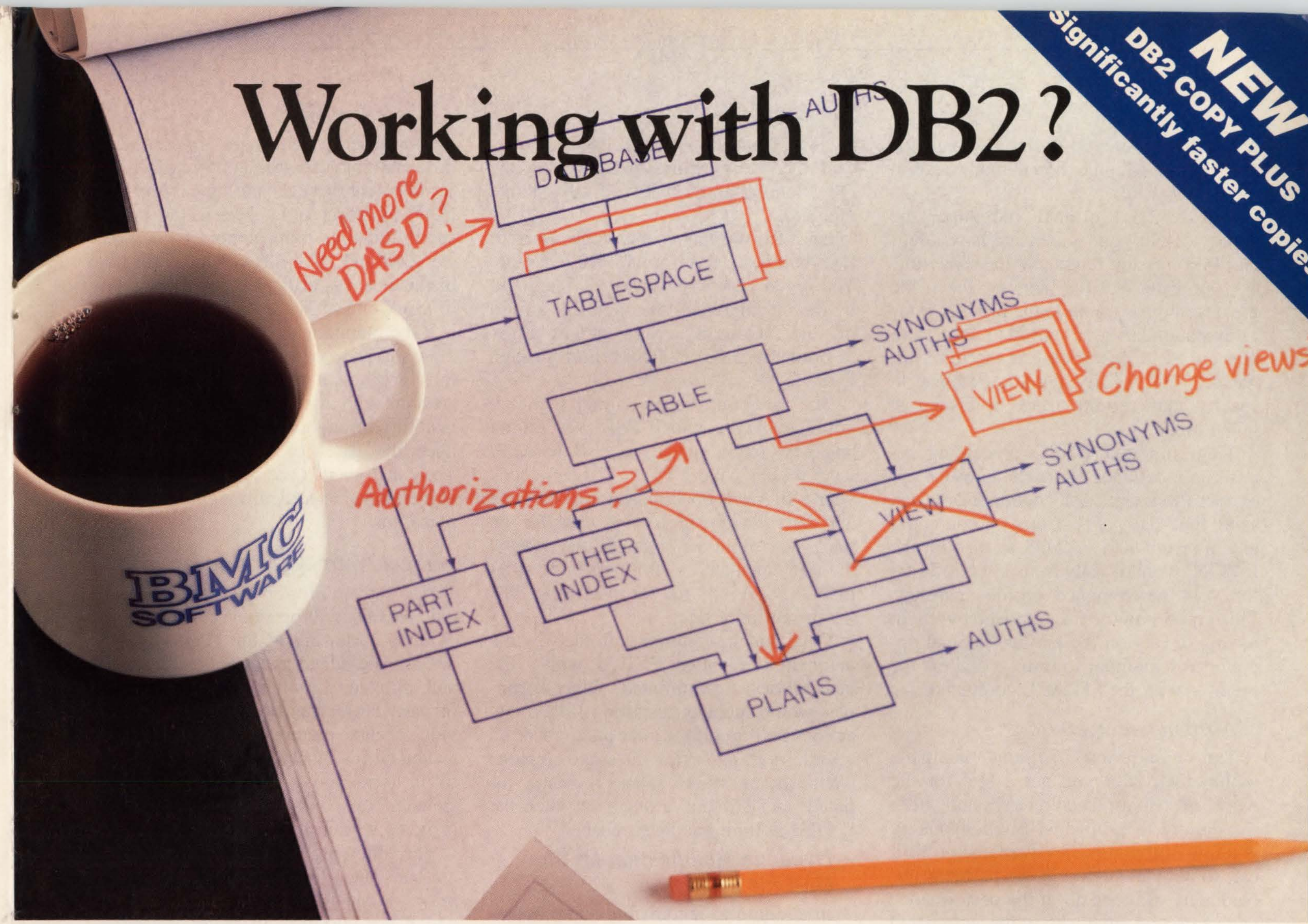
One basic example would be to create an ISPF Dialog that would allow you to generate the JCL and control cards necessary to print or copy some information from the IMS Secondary Log Dataset. Your JCL will be fairly static except for the tape volume serial number and control cards. The JCL and control cards may be included as skeletons in your ISPF dialog.

Your dialog could prompt you for input or a panel may be displayed to enter your input. This input may then be included in your JCL and control cards. The combined information may then be submitted. In the future when you need a log print, your dialog can be executed. It will be faster and more accurate than JCL generated manually.

Real-Time Performance Monitoring

The next type of product that will be most beneficial to you as a DBA will be a DBMS monitoring and automated operations product. One of the best and most widely-used products for IMS monitoring is Boole and Babbage's (Sunnyvale, CA) IMS Monitoring Facility (IMF).

Initially, this product did not provide much in the way of problem determination, because it was executed under the IMS subsystem. If you needed to investigate a slow response problem on IMS, the old RTO product experienced the same slow response. Since the product execu-



Work with BMC.

DB2 can be a lot more work than you expected with quite a bit less help than you need. But when you've got BMC Software's DB2 Masterplan™—a comprehensive series of DB2 products—your work is complemented by a company that has worked extensively with DB2 and knows what you need to keep your system running efficiently.

DB2 ACTIVITY MONITOR—displays and collects real-time and historical data from MVS, IMS, CICS and DB2; provides more functionality than any other DB2 monitor available.

DB2 MASTERMIND™—complete DB2 administration in one product consisting of:

- DB2 ALTER—provides complete support for changing, copying and migrating DB2 data structures; includes data conversions, authorization-id switching and restart capabilities.
- DB2 CATALOG MANAGER—gives quick and easy catalog information, execution of SQL statements and DB2 utilities, audit logs and extended SQL function.
- DB2 DASD MANAGER—controls the life cycle of physical objects with comprehensive space analysis statistics; also includes space estimation, AMS command and utility jobstream generation, and triggers.

DB2 REORG PLUS—reorganizes DB2 tables 4-10 times faster than the supplied DB2 utility; reduces elapsed times, CPU cycles and EXCPs.

DB2 COPY PLUS—substantially reduces copy times and resource utilization; provides single or dual copies in one pass.

DATA PACKER™/DB2—reduces DASD requirements for DB2 tables an average of 50% to 70%; reduces EXCPs.

For more information, or to begin a 30-Day-Plus Free Trial of any or all of these products, complete the reader service number card. Or call BMC Software, Inc., **The Complete DB2 Company™** at **1-800-841-2031**.

BMC SOFTWARE

P.O. Box 2002
Sugar Land, Texas 77487-2002
713-240-8800
BMC also has offices in Australia, England, France, Italy, Japan and West Germany.

IBM Business Partner

CIRCLE #103 on Reader Service Card ▲

tion has been moved to the TSO subsystem, it has become much more helpful and easier to use.

Release 2.3.1 of IMF introduced the ability to split your screen and have a regular ISPF session running at the same time as your IMF session. This has been the greatest improvement to the product since its execution has been moved to TSO. Also introduced in this release is the time initiated EXEC facility that allows a user to specify that certain execs be issued at given times.

Time initiated EXEC processing removed the reliance on master terminal operators to execute IMS commands at the same time daily. For example, you could use time initiated EXECS to turn on the IMS DC monitor daily from 11:00 to 11:05 for your performance profile snapshot. This could now be accomplished without having to rely on the master terminal operator to remember to turn on and off the monitor with the /TRACE command.

Automated Operations

The monitoring facilities available within IMF help you get a feel for the workload being placed on IMS and aid in responding to problems. Simply looking at a few of the inquiries available on IMF can sometimes highlight the problem immediately. Beyond all of the performance and workload monitors is the automated operator portion of the product.

Automated operations will allow you to become aware of problems faster and can even begin to resolve the problems before you are notified. Any IMS or IMF message that is logged to the IMF journal may be intercepted and a specified set of actions may be taken. This is accomplished by including the IMF AUTO-OPERATOR exit in the link of the IMS nucleus. When a message is issued, the exit tries to schedule an IMF EXEC with the same name as the message ID.

IMF EXECs are similar to ISPF dialogs. In fact, most commands which may be executed from an ISPF dialog may be issued in your IMF EXEC. In addition, EXECs can be written to issue either IMF or IMS commands which may help you in resolving problems.

Suppose you are having a problem with a shortage of PSB WORK pool, but you are unsure of how much to increase the pool. You can set up a monitor using IMF to track the number of scheduling failures by type intent. Once the number of scheduling failures surpasses a specified maximum, a warning message will be issued by IMF. You can write an exec to intercept the warning message, turn on the DC monitor and /DISPLAY the active regions.

Once the warning condition has cleared, IMF will issue an informational message. You could also write an exec, identified by the informational message ID, to turn off the DC monitor and submit a job to print the EXCEPTION report through IMS ASAP.

The exception report would then tell you specifically which PSB was failing and how much space the PSB required from the PSB Work Pool. You can check your PSB Work Pool utilization at the time and determine the high water mark for the pool. You could then increase the pool by the size calculated as follows: high water mark + size required for the failing PSB = current pool size.

Once you are familiar with the way the automated operator EXECs work, the possibilities are unlimited. Other examples would include: restarting IMS transactions and programs after an application abend; starting extra message regions when the message queues become inflated; and DBRing a database when the EXTEND limit has been reached.

Dynamic Modification Of The On-Line System

In addition to automating some operation functions, it is important that you have some software allowing you flexible and dynamic modification of application-related IMS control blocks. There are a couple of products which will accomplish just that.

One such product is BMC Software's (Sugar Land, TX) DELTA IMS. DELTA allows the user to dynamically alter control blocks being used by IMS. Changes to the SYSGEN application information may be made dynamically and immediately. This will allow you to complete the work request faster and more effectively because it will no longer be required that you wait for an IMS SYSGEN to be scheduled and completed.

Beyond modification of the IMS control blocks, DELTA provides utilities to create IMS Stage 1 source from a list of DELTA commands which you generated using the product. This can save a great deal of time and errors when creating the source input to the IMS SYSGEN process.

Time-Saving Tips

Technology has helped us do our jobs more efficiently, but there are some fundamental time management tips which will complement these technological improvements. Getting organized is the key to successful time management for any occupation and, as a DBA, successful time management is critical if superior quality of work and above average performance is to be maintained.

Learning to save time and getting the most productive use of your time will create a greater variety of tasks, increase your visibility throughout the organization and open the gates of opportunity to you. There are several time-saving tips and each is mentioned specifically with a brief explanation.

Track Your Tasks And Activities

A complete *master to-do* list is essential. This list will be an ever-changing list of items requiring action. Items will be deleted and added as tasks are completed and initiated. This list can be a summary of past, current and future tasks. It provides a clear picture of progress and direction. It may include both long-term and fill-in type activities. This master list should correspond directly to a set of files or items within the files.

The following three sets of files should exist in your filing system in some form: *open* — consisting of projects and tasks which are currently in work; *closed* — consisting of projects or tasks which have been completed, including all follow-up activity; and *future* — consisting of backlogged projects or tasks which will be completed sometime in the future.

These files should be reviewed weekly to ensure that each file remains in the appropriate group and that each of them is still in progress. Completed open files should be moved to the closed files and open files put on hold should be moved to the future files. Future files with work that needs to begin should be moved to the open files and future files with work that has been reassigned or canceled should be removed from the future files.

Program And Status

These three sets of files will provide a good reference source when drafting a status report. Each closed file, which contains a completed project or task, may be listed as an accomplished item. Future files which have been put on hold may be listed as *waiting for input* or *supervisor action required*. Open files may be listed as items with work in progress.

Simply reviewing your files should al-


We help programmers to boldly go where no one has gone before.
A new exploration in productivity.
The XPERT Series of programmer productivity tools will help you explore data in places you couldn't imagine, and manipulate it in ways you never thought possible.
DATA-XPERT, IMS-XPERT and DB2-XPERT are compatible with all commonly used programming languages, file organizations and data base management systems in the IBM MVS environment.
They're also compatible with the way you work. Their ISPF-like design will have you learning skills quickly and getting results faster—it's productivity at warp speed.
Navigate through unexplored data.
Working with files such as VSAM, and IMS or DB2 data bases—regardless of size—you can selectively edit, browse, extract, load, convert, reformat and print any size file or record to get just the data you want.
Even in the demanding DB2 arena, the XPERT Series are proving themselves daily as a tool that lets programmers work faster, more efficiently and produce higher quality data, tables and applications.
DB2
Take command.
There's a good reason why the XPERT Series is rated #1 over the competition.* Call today for more information or a free 30-day trial of DATA-XPERT, IMS-XPERT or DB2-XPERT.
Call 1-800-344-9223; in Canada, 1-800-344-9224.
The XPERT Series. As far as productivity goes, it's out of this world.
XA
XA SYSTEMS CORPORATION
CIRCLE #134 on Reader Service Card ▲
*Hancock Survey, January 1989. DATA-XPERT and IMS-XPERT are registered trademarks of XA Systems Corporation. DB2-XPERT is a trademark of XA Systems Corporation. ISPF, IMS and DB2 are products of IBM.

low you to report status more completely and faster than trying to remember back over the last two weeks or month. Once the completed files have made their way on to a status report, the contents should be distributed into your regular filing system for future reference.

Even if these three sets of files exist only as stacks of paper on the corner of your desk, they should be easily accessible and you should always be familiar with their contents. This part of your filing system may seem a bit cumbersome at first, but once you are familiar with the way to use it, it should help you organize your work considerably.

The Telephone, Friend Or Foe?

Phone calls can mean the death of your time schedule if they are not handled properly. If you let it, the phone can be the greatest single contributor to time loss.

How many times have you started a long-term task, only to be interrupted by the phone. Then you have to restart the task from the beginning, possibly several times. Much of that time could have been saved if you will follow these few suggestions.

Every phone call should be logged immediately. A piece of scrap paper, such as the back page of an old SYSOUT, could serve as a log sheet.

Typical Phone Call Reaction

As the caller begins to speak, write the date, time and the caller's name and number at the top of the log sheet followed by a brief description of the problem or inquiry. Next, determine if the request can be satisfied quickly and determine the priority of the request.

If it can be satisfied quickly and the priority is high (that is, production is being affected), then a response should be given to the caller immediately and a brief description of the response should be written on the bottom of the log sheet. The log sheet may then be placed in the stack of closed files.

If the request will require little time but the priority is low and you are busy on another task, tell the caller you will return the call at a specified time. This way the caller knows that you are not putting him/her off indefinitely.

If you are going to get back to the caller, then note the date and time that you will call back on the log sheet. Then place the log sheet with the open files and schedule time for the task.

If the caller cannot wait, say you will need to research the answer before calling back. Write the word *now* on the log sheet and place it at the top of your open files. As soon as you come to a break point in your current task, complete the caller's request, write the brief description on the bottom of the log sheet, return the call and place the log sheet in your closed stack of files.

If the request cannot be satisfied quickly, the priority is low and you are busy on another task, explain that you are busy and that the request will be placed in your work request queue. Determine the date and time when you can satisfy the request and note this on the log sheet. Inform the caller of this date and time, place the log sheet with your open files and continue with your current tasks.

Control Your Telephone

The most important consideration is *do not* let your phone control your schedule. It is also important to remember not to let the sequence of calls or the tone of the caller's voice determine your task priorities.

You will not be able to avoid interruptions totally, but if you apply this method you will certainly be able to cut down on the number of interruptions.

If you are working on a critical task and you are being interrupted continuously, it may be wise to isolate yourself away from your regular work area. Make sure that your secretary or someone knows where you are and that you are only to be interrupted in case of an emergency. This will buy the time that you need to work on your critical task.

The Time Schedule

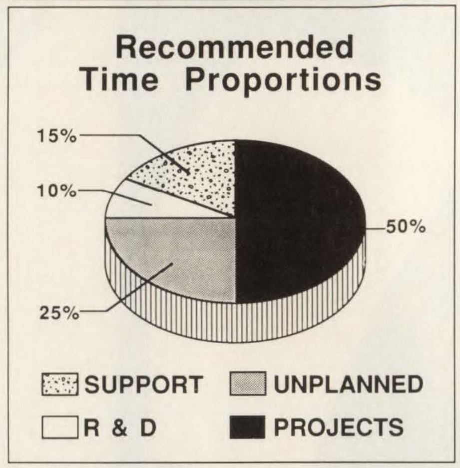
Scheduling your time is probably the most crucial of the time management techniques. If you do not have a road map to use as a guide for what you should be doing and when you should be doing it, you are likely to find yourself running in several directions, fighting multiple fires and going home at the end of the day wondering what you have accomplished.

Your time schedule should be completed on a weekly basis. Friday afternoon is usually a good time to plot next week's activities. Certain time blocks will remain consistent with the same task from week to week, like morning mail, previous night's follow-up and lunch. You may consider placing these fixed items permanently into your pattern schedule.

Draft Your Time Schedule

When scheduling your time, first determine the high priority items and items

FIGURE 1



which will require a significant amount of your time. Your priorities should be determined and agreed upon by you and your supervisor. Be sure that both of you agree which tasks should be completed ahead of others and when tasks are due to be completed. This will avoid miscommunications at a later date and give you a start on scheduling your time. Block out the times for meetings which have been scheduled for the next week. Next, determine the largest blocks of time that are most likely to be free of interruptions.

For example, Tuesday afternoon may be more likely to be free of interruptions than Monday morning. Finally, write in the blocks of time for the tasks with the highest priority and most significant amounts of time.

When scheduling your time, be sure to include free blocks of time for unexpected activities. Unexpected activities will always occur. They may vary in length of time required to complete, but they will occur. Planning for the unplanned can be challenging. However, after a few weeks of completing time schedules, you will have a much better feel for how much time you should allow for unplanned activities.

Polish Your Technique

If you are consistently spending too much time on unplanned activities, then perhaps better planning for projects or activities is needed by you or by others with whom you interact.

A chart of major activities and their suggested proportions of time as a percentage of the whole is shown in Figure 1. You will notice that about 25 percent of your time should be designated as free time for unplanned activities.

In addition, give yourself at least four hours a week for research- and development-type activities; this will give you a break from routine activities and will help ensure that you and your tasks are progressively moving forward with time.

Figure 2 shows an example of a completed time schedule for the week of 3/17/89. You will notice that Morning Mail and Follow-Up Activity are scheduled daily from 8:00-9:30. Lunch is scheduled daily from 12:00 to 12:30. These are fixed activities and appear on the schedule every week. When this schedule was completed, four meetings were scheduled for the week. The times for these meetings were blocked out first.

The tasks in progress, or open tasks, at the time were: Install Data Dictionary Release 6.0, Develop IMF 2.5 Operator Training and IMS Performance Monitoring. Blocks of time for these activities were scheduled after the meeting times were blocked out.

Free time for unplanned activities was scheduled along with the time for open activities. Fill-in activities for the week were listed under the Notes section. If at some point throughout the week you do not have tasks to complete during a free block, you can use this time for the fill-in activities or for continuing on your open task list.

FIGURE 2

Week Ending 3/17/89	0800	0900	1000	1100	1200	0100	0200	0300	0400	0500	Daily Comments
Monday 3/13	Mail and fllw up		new applct implmt meetng		L	IMS Performance Monitoring					increase PSB WORK POOL 3/17/89
Tuesday 3/14	mail and fllw up				U	install & test data dictionary (6.0)					
Wednesday 3/15	mail and fllw up	data dict testing (6.0)			N		dvlpmnt system sysgen				DD 6.0 ready for USER testing
Thursday 3/16	mail and fllw up			IMS 2.2 plan meet	C		Develop IMF 2.5 Operator Training				Schedule OPS Training Session w/o 4/10/89
Friday 3/17	mail and fllw up		DBA staff meet		H		wkend test meetng				Pseudo-Prod IMS support sat - 3/18/89

NOTES

Free Time Fill In Activities

- Verify Space Utilization
- Review Prod IMS scheduling
- Investigate RECON maintenance

Keep Meetings Productive

Much of your time schedule can be exhausted and many hours wasted by sitting idly in meetings. Avoid unproductive meetings. It is important to develop a skill for being able to determine if your presence at a meeting is required and necessary. Your invitation to a meeting could be for informational purposes or it could be directed to anyone with your expertise on a specific subject.

If you determine that your invitation is for informational purposes only, you can call the requestor of the meeting, inform him/her of your absence and request that a copy of the meeting notes be forwarded to you. If you determine that someone with your expertise is required, determine if you are the proper recipient of the invitation.

Even if your presence is required, be aware that at some point you will have to miss a meeting. If you have to miss a meeting and it is possible, send someone in your place. If it is not possible to send someone in your place, then contact the requestor, inform him/her of your absence and request that a copy of any notes or questions from the meeting be forwarded to you.

Standardize Your Note Taking

During meetings that you do attend, it is important to take adequate notes. Develop a systematic approach for taking meeting notes, including a personal dictionary of abbreviations, acronyms and symbols which have a specific meaning to you.

Symbols can be used as a standard method for highlighting your notes. The use of a standard set of symbols for highlighting your notes will make reviewing your meeting notes faster and more effective.

The same symbol should always be used for sections of notes that require action on your part. For example, '= => Copy T from P,all AC dbs', where = => designates action on your part. Other symbols can be used for sections of notes such as informational items and action items for others.

Items requiring your action should be clearly defined and understood by all attendees of the meeting. These items should also include a specific completion date. This is crucial for prioritizing your tasks and scheduling your time.

During the meeting, *do not* be afraid to voice new or different ideas. Opposing opinions may exist, but sometimes the best ideas are born out of dissent.

It is important to express your ideas and opinions during the meeting. This will keep the meeting productive and ensure that decisions reached during a meeting live to be enforced by the attendees of the meeting. Finaly, find out if and when a follow-up meeting is scheduled so you may add the requirement to your time schedule.

After the meeting, review your notes immediately and transfer your action items to your master to-do list and file your notes appropriately. Your notes may be filed with an existing open file or a new file may need to be opened. If necessary, prioritize and schedule time for your action items from the meeting and, if required, schedule your time for the follow-up meeting.

DBA-Specific Tips And Ideas

Although these ideas may be applied by professionals other than the DBA, they are instrumental in saving time for the DBA because of the multitude of different tasks which could be executed on any given day. There are a few other suggestions which may be applied specifically to your responsibilities as a DBA.

Floating atop the sea of knowledge are a few manuals which you will find yourself referencing repeatedly. If it is possible, obtain your own personal copy of these manuals. At the minimum, you should have your own copy of the messages and codes manuals for the products which you support.

Having your own copy of a manual means that you can add side margin notes of your own to the manual text. Many times the text contained in an IBM manual can be vague and nondescript. Having your own notes in addition to the manual text will lead to faster problem determination when a problem repeats itself. In addition to problem determination notes, it will be helpful to note other manual references directly next to the text.

Contacts — Invaluable Resources

As a DBA, it is imperative that you develop your contacts, both inside and outside of the company. This can be accomplished by attending user group meetings and conferences. The technical exchange of information that takes place at conferences such as GUIDE is invaluable.

Once you have started to develop your list of contacts, you will find yourself making that informal call to get the outside opinion when you may seem at a total loss. Never reinvent the wheel. Your time is much too valuable to repeat tasks which have already been completed by someone else. Having a list of contacts both inside and outside the company will provide you with a reference source if you have the suspicion that something has been done before.

Kill Two Birds

Similar tasks or tasks that would complement each other should be combined if possible. For example, if you have new software to be tested and you have someone screaming for an isolated development environment, you could satisfy the request and have someone test the new software without ever knowing it. The old adage of "killing two birds with one stone" should be applied as much and as often as possible.

Multitasking should be applied by yourself much the same way as it is applied by the machine. If you have some

See DBA *page 50*

MVS/ESA Architecture, Modeling And Performance

By J. William Mullen

The MVS/ESA operating system architecture has opened new horizons for systems designers and users through the ability to eliminate performance bottlenecks experienced in previous versions of MVS. Architecture providing dataspaces, hiperspaces, enhanced use of expanded storage and the potential for significant elimination of I/O for subsystems has provided a new set of problems in the performance and capacity planning arenas. This article addresses implementation and use of the new architecture and techniques for modeling the new architecture for both performance and capacity planning projections. Control over creation of dataspaces, hiperspaces and use of data windowing will be presented from both an architecture and performance view.

The MVS operating system continues to see expansion and implementation of new software technology. In many cases, new software technology is provided to optimize the use of existing hardware technology or alleviate constraints and processing deficiencies in the current software technology. In other cases, new software technology is available only because new hardware technology is available or new hardware technology cannot be utilized to its fullest potential with existing software.

With the availability of expanded storage on the 3090 processor complex, a new hardware technology was available for alleviating the existing performance problems encountered through use of auxiliary storage as a paging and swapping mechanism. The concept of ''processor storage swap'' was expanded beyond its original implementation through the logical swapping process. This second level mechanism for paging and swapping reduced the amount of time required to transfer pages to and from central storage from milliseconds to microseconds. While not operating at the speed with which logically swapped users can be reactivated, expanded storage provided a significantly faster alternative to the use of auxiliary storage while freeing a critical resource: central storage.

Use of virtual storage and Virtual Storage Constraint Relief (VSCR) is still a major area of concern for many installations. The size of operating system subsystems as well as application system address spaces continues to increase as additional function is provided. The requirement for storage and retrieval of vast amounts of data and the ability to consume large amounts of storage for buffering of data files and databases has increased with the transactions volumes and processing required in today's large installations. Processing delays due to I/O access requirements, ''I/O drag'' and the need to reduce or eliminate I/O operations where possible has only been available through the use of large virtual buffer areas within the user address space. Even with the availability of the MVS/XA operating system and 31-bit addressing, the VSCR and I/O elimination problems are still with us.

Availability of the MVS/ESA and the ESA/370 architecture is providing a direction for exploitation of the hardware architecture to resolve many of the existing problems with large storage requirements and I/O access delays. I will review some of the software implementations in the MVS/ESA operating system architecture and discuss both performance implications and approaches to using analytic modeling to predict the effects of conversion to the MVS/ESA operating system environment. Also, I will briefly review the Advanced Address Space Facility (AASF) and hardware architecture that support implementation of the new MVS/ESA facilities (see Figure 1).

Data-In-Virtual

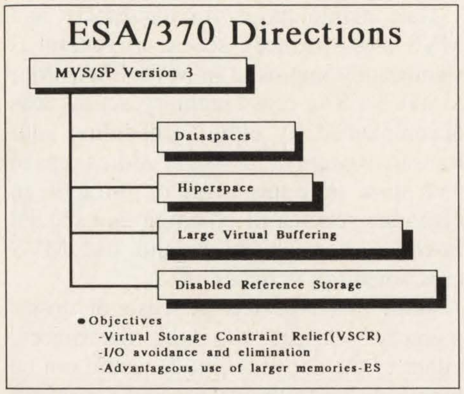
The Data-In-Virtual (DIV) facility was introduced in the MVS/XA operating system environment as the first implementation of what we shall term the ''processor storage data'' technology.

Based on the implementation of the Virtual Storage Access Method (VSAM), DIV was implemented as a 4K, block addressable architecture contained in a special type of VSAM dataset defined *linear.* Processing routines were available to access linear datasets through the definition of a virtual storage *window* and either Assembler language macros or subroutines to be called from high-level languages. The user of a DIV dataset could provide a virtual window large enough to hold only a few 4K blocks of data from the dataset or the entire dataset in storage.

The VSAM access method was the first to exploit MVS/XA 31-bit addressing and use of virtual storage above the 16MB line. The use of Local Shared Resource (LSR) buffer pools, which allows optimum use of VSAM shared buffers for multiple VSAM files, considerably reduced the requirements for virtual storage necessary for data buffering.

Virtual buffering requires real storage frames for backing and can create system paging problems. The amount of virtual

FIGURE 1



storage available below the 16MB line was also a limiting factor. Use of virtual storage above the 16MB line eliminated the latter and, though the amount of virtual storage for a single LSR pool is limited to 16MB, the user can define multiple 16MB LSR pools. You will see that VSAM linear datasets are the basis for much of future subsystems elimination of I/O and performance improvements.

ESA/370 Architecture

Three new facilities are available in the MVS/ESA operating system implementation and ESA/370 architecture. These facilities are termed dataspaces, hiperspaces and large virtual buffering. Each of these new facilities involve use of the AASF and the new cross-memory addressing capability of the ESA/370 architecture.

Dataspaces

A dataspace can be described as an MVS address space that did not quite make it. The dataspace is created in much the same manner as an address space without the system areas normally found in the address space (LPA, COMMON, private area). The dataspace thus provides the user two gigabytes of addressable *virtual storage*. The dataspace also lacks several of the other attributes of a normal address space (see Figure 2).

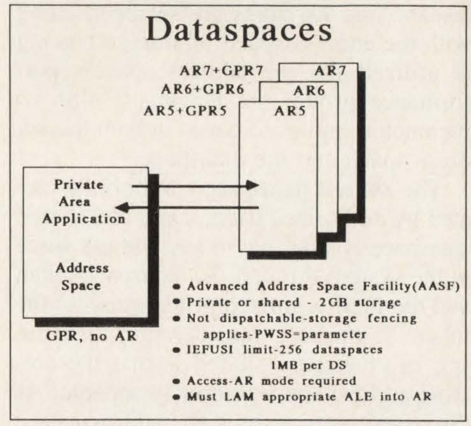
The first of these is dispatchability. The dataspace is owned by the creating address space but does not have an Address Space Identification (ASID). The dataspace is identified and referenced by both an eight character, unique name and an eight character unique *token*. Where ASIDs are reused as address spaces come and go during normal processing, a dataspace token is unique from system IPL to system IPL and not reused. The lack of dispatching capability disallows the execution of program modules in the dataspace. Information stored in the dataspace can be in many forms including executable load modules, but executable load modules must be transferred to an address space to execute.

Two limitations are placed upon dataspace creation. The first is the total number of dataspaces that can be active in the MVS/ESA system at any given time. This limit is set at 8192 and cannot be changed by the user. The second limitation is the number of concurrent dataspaces owned by an address space. The current limit is 256 of which zero, one and two are reserved for MVS/ESA use. A user can thus create 253 unique dataspaces. Exceptions to this limit and controls are discussed later in this article in the section titled, Dataspace And Hiperspace Controls.

Though dataspace virtual storage size must be defined in 4K blocks, the 4K blocks of storage in a dataspace are byte-addressable. The user can move data between the address space and a dataspace or between dataspaces with standard Move Character (MVC) instructions.

Dataspaces come in three flavors: single, shared and Disabled Reference

FIGURE 2



(DREF). The last is a special case reserved for authorized users. Dataspaces can reside in virtual storage with real storage backing, on expanded storage pages or on auxiliary storage. The most commonly used dataspace will be the single dataspace, created by a user address space for storage and retrieval of data. The dataspace will assume the characteristics of the creating address space and the real storage frames backing the virtual storage allocation in the dataspace will become part of the task working set. If the creating address space is swappable, the dataspace and its pages are swapped along with the address space. If storage fencing is utilized for the address space's performance group, the fencing (PWSS=) parameter applies to pages in both the address space and the dataspace.

The *shared* dataspace can only be created by authorized users. Data in a shared dataspace is available to any address space in the system through the dataspace token and the facility for acquiring access to the token. A primary consideration for dataspaces created as shared is that the creating address space be non-swappable. As discussed previously, if the address space is swapped out of storage, any dataspaces created by the address space go with it. Obviously, this would not work very well for shared dataspaces.

A significant trait of shared dataspaces is the lack of threads for access. Unless segments of storage in the dataspace are storage protected by the creating address space or serialized through the use of memory enqueue, an address space has immediate access to the data. This eliminates much of the queuing due to lack of threads seen in many data access and database implementations.

The Disabled Reference (DREF) dataspace is also limited to authorized users. DREF dataspaces can reside in virtual storage (with real storage backing) or expanded storage but not on auxiliary storage. The Expanded Storage Table Entry (ESTE) for expanded storage frames backing DREF dataspace pages are marked such that the Real Storage Manager (RSM) migration routines will not migrate them to auxiliary storage.

Why DREF dataspaces? Some MVS management routines execute in disabled states. This prevents them from taking a page fault where the page may reside on auxiliary storage. Prior to MVS/ESA, this required page fixing of the pages used by these routines, with much of the page fixing below the 16MB line due to the page involvement with I/O processing. Depletion of real storage frames below the 16MB line creates significant problems as the number of active address spaces increases and can result in swap-in failures for address spaces. Resolution of page faults from expanded storage eliminates much of the requirement for fixed pages and particularly fixed pages below the 16MB line. Again we see new architecture making more efficient use of existing technology.

Accessing Dataspaces

Data manipulation for dataspaces and MVS cross-memory access in general is significantly improved in MVS/ESA. Prior to MVS/ESA, cross-memory access was accomplished by either scheduling routines to execute in the target address space and store or return data or through an elaborate schema of chaining and special instructions to establish and use MVS cross-memory services.

With the expected increase of cross-memory services use with dataspaces, either of the above methods would not be practical due to the processing overhead. With MVS/ESA, a special set of hardware registers, termed Access Registers (ARs), are available. There are 16 ARs which correspond to the 16 General Purpose Registers (GPRs) that are the basis of S/370 hardware architecture. A task can now operate in two modes in the MVS/ESA environment: standard and AR-MODE. In standard mode, data movement is controlled only by the contents of the GPRs.

In access mode, a combination of the GPR and its corresponding AR is used to determine data movement. If the GPRs' corresponding AR contains a dataspace token, data movement is between the address space and the dataspace. If two GPRs are involved in data movement and their corresponding ARs contain dataspace tokens, then movement of data occurs between the two dataspaces identified by the tokens in the ARs.

Switching between standard and access mode requires a single executable instruction. The task may also operate in both modes in either 24-bit or 31-bit addressing mode. It is intuitive that the simplification of cross-memory access through new hardware will provide a significant reduction in the overhead associated with MVS cross-memory services in concert with performance improvements for accessing large amounts of processor storage data. A complete discussion of the access mode of operation in MVS/ESA is beyond the scope of this article.

Hiperspaces

The second major innovation in the MVS/ESA architecture is the exploitation of expanded storage through the hiperspace facility. Hiperspaces are a derivative of dataspaces and are created through additional options of the DSPSERV Assembler macro, the same macro used to create dataspaces. Here, most of the similarities to dataspaces end.

Hiperspace pages reside either in expanded storage or on auxiliary storage but never in central storage. Hiperspaces are defined as one of two types: cache or scrollable. Cache type hiperspaces are available only to authorized tasks and are used to retain 4K blocks of data from permanent (DASD resident) data objects.

Data in a cache type hiperspace is not guaranteed and must have a permanent DASD source. Though primarily non-migratable (CASTOUT = NO option on the DSPSERV dataspace and hiperspace creation macro), if available expanded storage frames are depleted to a critical state, pages in cache type hiperspaces will be migrated. If the creating address space is swappable, pages in the cache type hiperspace will be swapped when the address space is swapped.

Scrollable hiperspaces can be created by any task, reside in expanded storage and can be migrated. The primary differences in dataspaces and hiperspaces lie in addressability and read/write functions. Pages in hiperspaces are *not byte addressable*. The pages are moved to and from expanded storage in 4K blocks and are only byte addressable once they reside in the address space of the requesting task. Reading and writing of hiperspace pages does not rely on the use of access registers. Scroll type hiperspaces have guaranteed data since the only copy of the data may reside in the hiperspace.

Special facilities available through the CREAD and CWRITE macros for cache type hiperspaces and the SREAD and SWRITE macros for scrollable type hiperspaces provide for the transfer of pages to and from hiperspaces. The macros are available for Assembler level programs. The primary use of hiperspaces will be I/O elimination as seen in the discussion of I/O processing.

Data Windowing

Data windowing provides callable subroutine interfaces for high-level languages such as COBOL, PL/1 and FORTRAN. The subroutine calls provide access to temporary or permanent data objects in both dataspaces and hiperspaces. The callable subroutines provide seven basic functions for manipulation of data in dataspaces and scrollable hiperspaces:

- CSRIDAC - identify and access a linear object
- CSRVIEW - establish view of an object
- CSRSORT - capture current view via scrollout
- CSRSAVE - commit changes to permanent object
- CSRREFR - refresh current and scrolled view
- CSRIDAC - end options
- CSRVIEW - termination processing.

It should be noted that the new data windowing facility deals with VSAM linear data objects and is built around the initial DIV facilities available in MVS/XA. Taking advantage of both the dataspace and hiperspace facilities from high-level languages in applications development will require that both the designer and, more importantly, the programmer have a complete understanding of the DIV processing concept and capability. Taking advantage of dataspaces and hiperspaces for other than DIV processing will require Assembler-level knowledge and programming. Future high-level language capabilities may be expanded to allow for creation and access of dataspaces and hiperspaces via direct subroutine calls or better yet, an EXEC level precompile verb structure such as the CICS/VS command level and the DB2 SQL structures currently available.

I/O Processing

Elimination of I/O through dataspaces and hiperspaces is currently confined to the use of VSAM linear objects or DIV data objects as they are commonly referred. At MAP time (the linear data object equivalent of OPEN for a non-linear dataset), the Access Control Block (ACB) for the linear object can specify that I/O for the linear object or more specifically buffering, will be directed to a dataspace or hiperspace. The specific dataspace or hiperspace is determined by the token supplied in the ACB at MAP time.

Standard access methods, such as VSAM and QSAM, do not provide direct I/O to and from dataspaces and hiperspaces. The user may acquire I/O buffer areas in a dataspace or hiperspace via the DSPSERV macro's DEFINE and IOON options. Data buffers are brought into the address space and the task must then move the buffers to the dataspace or hiperspace. Use of the data in buffers residing in a dataspace or hiperspace require that the data be read back into the address space.

Another option available to the user for data buffering on non-linear objects to dataspaces or hiperspaces is the use of a user-written MVS IOS interface routine. Most users will find this an unacceptable approach.

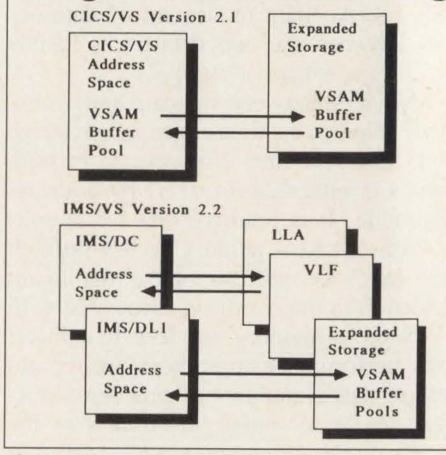
Large Virtual Buffering

Standard VSAM datasets can be mapped directly to hiperspaces through the Local Shared Resource (LSR) buffer pool facility. This capability requires the DFP Release 3.1 product and will be available only to authorized tasks. LSR processing requires construction of a buffer pool via the Build Virtual Resource Pool (BLDVRP) Assembler macro. Currently, LSR buffer pools built with the BLDVRP Assembler macro are constructed in the address space above the 16MB line (see Figure 3).

The BLDVRP macro has the capability of being provided the token of the hiperspace in which the LSR pool will be constructed. When the VSAM dataset is opened for processing, data buffers are directed to and, subsequently retrieved, from the LSR buffer pool built in the hiperspace. Again, this capability is currently limited to authorized tasks. Initial users of this capability will be CICS/VS Release 2.1 and IMS/VS Release 2.2. Use of the facility will be at the user's option through normal generation facilities.

F I G U R E 3

Large Virtual Buffering



Library Lookaside

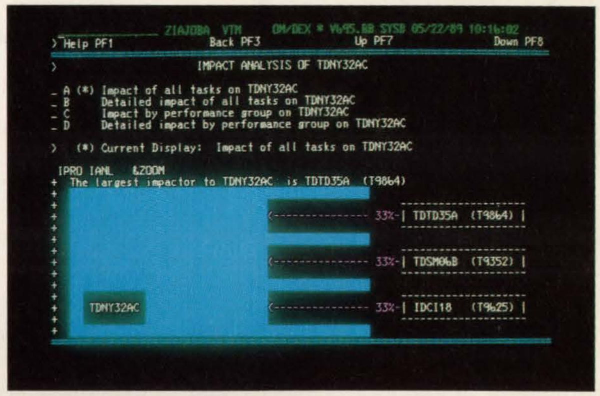
The Library Lookaside (LLA) facility is an expansion of the linklist lookaside capability introduced with MVS/XA. While the facility was confined to only linklist libraries in MVS/XA, the LLA implementation in MVS/ESA will allow for directory management of any type of partitioned dataset. Performance enhancement through the elimination of directory searches and faster retrieval of data will result from use of LLA. The negative to LLA can be the amount of real storage required by the LLA address space as the number of library directories it is required to manage increases. Careful selection of LLA managed libraries will be a concern for all installations (see Figure 4).

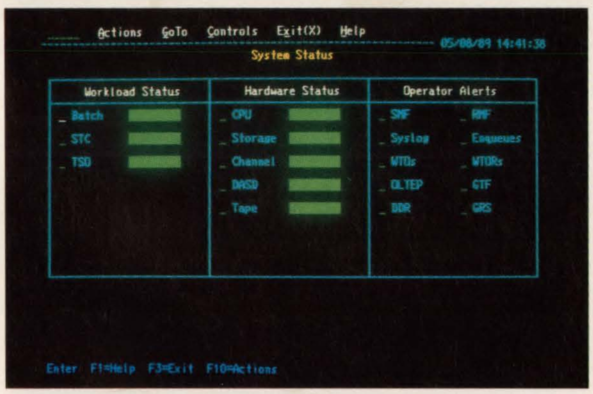
Virtual Library Facility

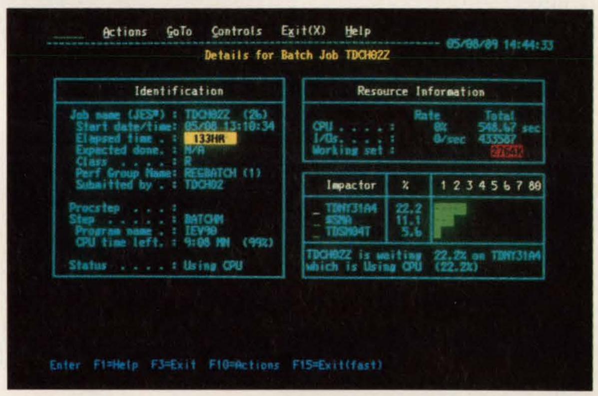
The Virtual Library Facility (VLF) will operate as an independent address space for storing *named* data objects in the VLF address space as well as a server for storage and retrieval services to subsystem dataspaces.

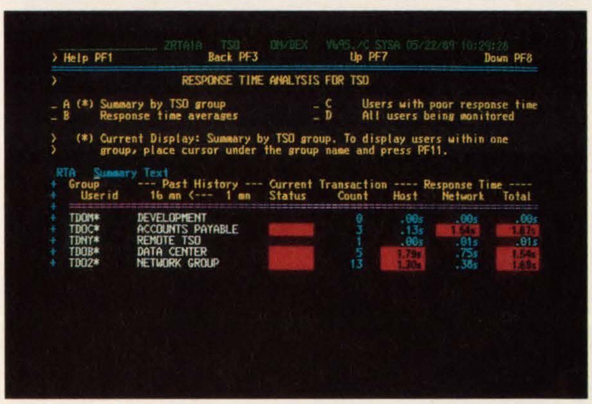
Data objects range from executable load modules to TSO CLISTs or any other data for which quick access is required. Load modules cannot be executed in the VLF address space or a VLF managed dataspace but must first be transferred to the requesting address space. The availability of TSO/E Release 2.1 provides for *compiled* CLISTs and the REXX language

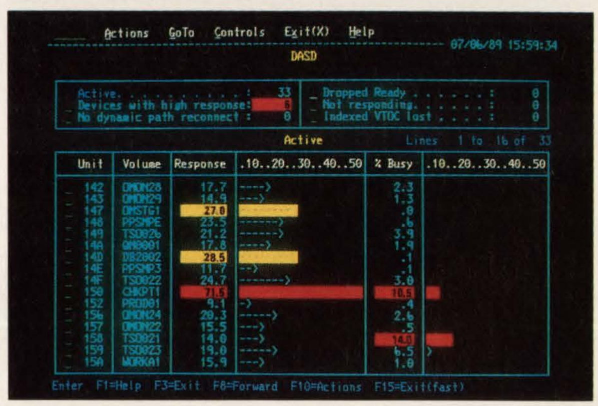
The first word in performance management is still the last.











OMEGAMON®.

The OMEGAMON family for MVS.

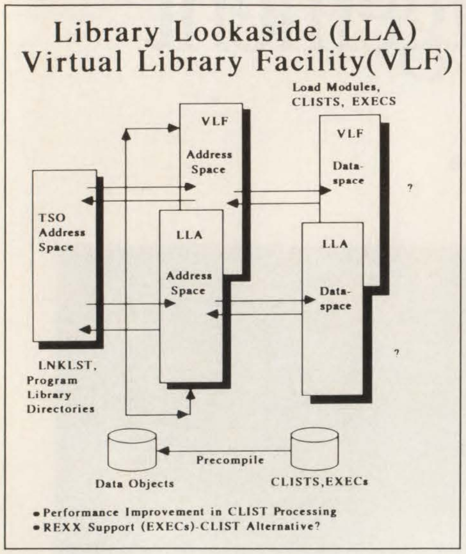
To have the last word in your data center, call Terry Forbes today at **(800) 541-8514.**

!Candle®

The one the others imitate.

CIRCLE #116 on Reader Service Card ▲

FIGURE 4



under TSO. Compiled REXX EXECs will also be candidates for VLF management.

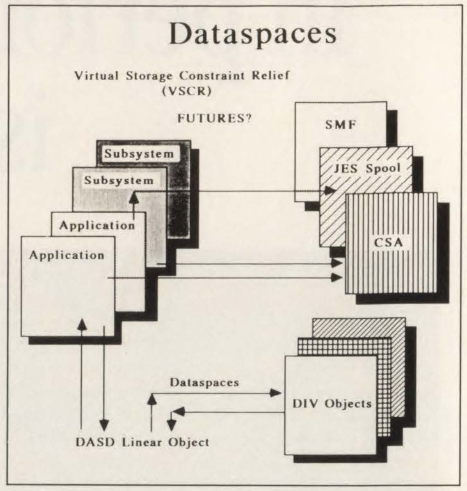
The amount of data object storage and the effects upon real storage frame availability and system paging rates will be the main concern for most installations. From the overall system perspective, VLF provides the most immediate facility for I/O elimination for datasets used by the citizens.

Additional Dataspace And Hiperspace Use

Additional uses of dataspaces will be seen with the DFP Release 3.1 product. Use of a dataspace by the MVS catalog address space will provide for storage of a large number of catalog entries in processor storage and elimination of the I/O delays for catalog record search and retrieval. The catalog dataspace will be managed by VLF acting as a server for the catalog management address space (see Figure 5).

Job Entry Subsystems (JES) will utilize dataspaces to store data that is frequently referenced or provide storage for facilities that require a large number of entries. The JES2 subsystem will utilize a dataspace for Internal Reader (INTRDR) processing and to allow the user the capability of defining a larger number of active internal readers. The JES2 dataspace will also assist in providing System Managed Storage (SMS) compatibility where sharing systems are composed of both MVS/ESA and MVS/XA operating systems.

FIGURE 5



The JES3 subsystem will utilize the dataspace capability for Job Control Table (JCT) storage which eliminates much of the need for I/O to the JESQ datasets for storing and retrieving the status of active jobs in the system. The JES3 dataspace will also be used to provide SMS support in the MVS/ESA operating environment.

A major user of hiperspace capability will be the Database 2 (DB2) subsystem in the Release 2.1 version. The DB2 control datasets (commonly referenced as SYSIBM.) are VSAM linear datasets. Thus, the placement of catalog and other information from these datasets in dataspaces, but more likely cache type hiperspaces will be easily accomplished through standard DIV mapping. This mapping to cache-type hiperspaces could eliminate most of the I/O to these linear datasets required for support of DB2 user requests, database access and database storage management.

Significant improvements in DB2 request processing performance will be seen with the majority of retrieval for these datasets coming from cache-type hiperspaces. The downside will be the amount of expanded storage required to hold the data from these datasets. Reduction in expanded storage frame availability will result in less space for paging and swapping, more paging and swapping being directed to auxiliary storage and higher page migration rates and activity.

What are the potential future uses of dataspaces? An initial consideration would be the use of a dataspace for MVS control block and data storage that is normally available in the Common System Area (CSA). This would reduce the virtual storage requirements below the 16MB line and potentially provide for larger user private areas. Though Extended CSA (ECSA) is available above the 16MB line, many tasks requiring access to the data do not operate in 31-bit mode. The major drawback to a CSA dataspace is the lack of task execution capability in a dataspace. The current architecture allows for the placement of executable code in the CSA or ECSA (both in an address space). Only data objects that were not executable tasks or code could be placed in a CSA dataspace. This could present some difficult management logic for determining placement and location of data with CSA, ECSA and a CSA dataspace.

A second option would be the use of a JES dataspace for spooling purposes. Small print datasets, particularly of the type normally used in the TSO environment, could be directed to JES spool dataspaces and subsequently retrieved using the ISPF 3.8 screen option as an example. The chaining of JES spool buffers on auxiliary spool datasets via the track-record number (TTRN) reference could be replaced with virtual page address chaining within the dataspace. Directing print to a dataspace spool facility could be at the user's option or automatic if the number of print lines generated was below a specified threshold.

A third option would be the use of dataspace(s) as a temporary storage area for SMF records. A major concern in SMF recording has been the increase in the subsystems now writing information to SMF and the amount of SMF data MVS has to handle. This is a particular concern when synchronization of SMF data with other subsystem data, such as RMF interval data, is addressed. A dataspace could be used as an intermediate storage area for the SMF writer with the transfer of the SMF records to the SMF datasets on auxiliary storage operating as an asynchronous function.

Theoretically, this would pose a delay of only a few minutes before the SMF records were written to auxiliary storage and available for processing.

As the MVS/ESA operating system continues to evolve, you will expect to see more subsystems and user-developed application systems take advantage of the capabilities of the new technology. The major impact will be central and expanded storage requirements and developing strategies for controlling the use of the dataspace and hiperspace technology.

Control Of Dataspaces And Hiperspaces

Control of dataspace and hiperspace creation will be an integral part of system memory management in most installations. As presented previously, the current limit for dataspace and hiperspace creation is 256 (combined dataspaces and hiperspaces) for non-authorized address spaces. The actual number available is 253 since MVS/ESA uses zero, one and two for its own purposes.

Real storage frame backing is not required for virtual storage in a dataspace until the storage is actually used. A user could create 253 dataspaces and not provide a significant impact upon system real storage frame availability until the addressable space in the dataspace is actually used. This apparently is not true for hiperspaces.

Creation of a hiperspace, whether cache or scrollable, apparently does not allocate the expanded storage frames until the page is actually used. Even with this technique, hiperspace allocation can have a significant impact on the performance of other tasks in the system due to the depletion of frames available for paging and swapping. Extensive creation of DREF dataspaces whose pages are backed by expanded storage and not migratable as well as cache-type hiperspaces with the CASTOUT=NO option could rapidly deplete the available expanded storage frames. Fortunately, the latter two's creation are confined to authorized address spaces. Scrollable hiperspaces can be created by any address space. Herein lies a problem. If a non-authorized user decides to create and use 100 hiperspaces at the default allocation of 1MB, chaos could result with paging and swapping operations as well as migration activity.

The number and size of dataspaces and hiperspaces can be controlled through the standard SMF User Step Initiation (IEFUSI) exit. No default IEFUSI exit is provided with the MVS system. The default storage size for dataspaces and hiperspaces is 938 4K blocks or 956K of addressable storage. Upon entry to the IEFUSI exit, a three-word parameter list is available.

The first word contains the default size allocation for a dataspace or hiperspace. The second word contains the maximum combined sizes for all user key 8-F dataspaces and hiperspaces with the default set at 256MB. The third word contains the maximum number of user key 8-F dataspaces and hiperspaces that can be created by a single user with the default set at 256. Note that the restrictions apply to user key 8-F address spaces and not to authorized or key 0-7 address spaces.

Most installations may want to consider lowering the default number of dataspaces and hiperspaces to the four or five range (this would allow one or two dataspaces or hiperspaces for creation by a non-authorized address space). This will limit the amount of real storage a non-authorized user may consume with dataspace allocations as well as limiting the number of expanded storage frames consumed by a non-authorized user for hiperspaces.

Modeling And Performance

A primary consideration in planning for MVS/ESA and modeling to project the impact of MVS/ESA migration is the effect on central and expanded storage. As with all new releases of the MVS operating system, new function is implemented. New function means increases in module sizes as well as increases in the number of modules in the operating system. The storage and data areas to support the new function implementation require real and expanded storage backing and in the case of real storage, some long-term fixing of storage pages. Initial review indicates that the base MVS/ESA operating system requires approximately 2.5 to 3 megabytes of additional real storage.

Address space resident set sizes also increase in MVS/ESA. The additional pages are apparently required to support the underlying control block structures for new functions such as dataspaces and hiperspaces. Measurement data indicates that MVS/ESA address space resident set sizes are 28K to 40K larger. This represents an increase of seven to 10 pages per address space. For planning purposes, it is recommended that the 40K or 10-page increase in resident set size be used.

The initial implementation of LLA and VLF supporting LINKLST libraries has shown varied working set sizes based on the number and size of the libraries included in the LINKLST definition. The average across several installations has shown approximately 1.2 to 1.5MB resident sets for LLA and 4.0 to 4.5MB resident sets for VLF. With the expansion of the LLA and VLF facilities to support non-LINKLST directories and datasets, these resident set sizes will obviously increase.

The use of DIV will increase both real storage requirements and expanded storage frame usage. An intuitive guideline for expected increases in the number of frames required has been approximately 40 percent of the total size in bytes of the dataset directory. The 40 percent could also be applied to the size in bytes of a dataset containing *data objects* to be loaded into storage and managed by the VLF facility. This will obviously depend upon the number of *named objects* from datasets included in the VLF management list.

Large virtual buffering and the use of VSAM LSR buffer pools will affect the number of expanded storage frames required to support the buffer pools. The amount of storage used can be controlled by the user through the buffer pool construction facilities. The user should be cautious with the size of LSR pools starting with smaller pools and increasing the size of the pools where measurement data indicates that an increase should not have an adverse effect on system paging or expanded storage migration rates.

Modeling the effects of MVS/ESA requires increases or changes in the memory usage characteristics of tasks active during the modeled period and in some cases, altering the amount of I/O expected when new facilities such as dataspaces or hiperspaces are used. The initial adjustment is to the amount of system storage utilized by MVS/ESA. As stated above, the intuitive increase is approximately 2.5 to 3.0MB.

The initial increase in workload resident set sizes should be 40K or 10 pages in the model. This increase should indicate if the currently available real storage size of the system will be adversely affected by changing to the MVS/ESA operating system environment.

Dataspaces and hiperspaces are a little more subjective. If a dataspace is a single user type dataspace (TYPE=SINGLE), then the resident set size of the owning workload should be increased by an expected real storage usage, possibly in the 20 percent to 40 percent range of the amount of virtual storage allocation in the dataspace. Why increase the average resident set size for the workload? For TYPE=SINGLE dataspaces, it would be wise to assume that the tasks in the workload are of like type and that each will create its own dataspace. Thus, each active task in the workload would incur the additional real storage frame requirements in support of its dataspace. For TYPE=ALL dataspaces that will be shared by multiple subsystems and address spaces, the increase in real storage frames requirements need only be represented once. This can be accomplished by increasing the amount of system storage by a value for the expected number of real storage frames required to back the dataspace.

For hiperspaces, either cache or scrollable, the model should be changed to reduce the number of available expanded storage frames by the expected number of frames required to support the hiperspace. For cache type hiperspaces and VSAM LSR pools, the reduction in the number of available expanded storage frames is directed by the size of the cache space or the number and size of each LSR pool created. For scrollable hiperspaces, the reduction in available expanded storage frames will be determined by the number of tasks expected to create scrollable hiperspaces and the average size, in frames, of the hiperspaces created.

Results of making the above described changes to a base MVS/XA model and the effects on workload response times and system paging rates are shown in Tables 1 and 2.

Table 1 shows the effects of additional memory requirements for MVS/ESA and

TABLE 1

MVS/ESA Memory Effect Evaluation

Measurement	3090-600E	MVS/ESA + 3MB	Resident Set +10	+ VLF 4MB	3090-600S	TSO Trans + 12,000	Add 128MB CS	Add 128MB ES
DASD Page-In Rate	40.82	48.45	113.94	139.17	12.15	551.88	24.44	34.99
DASD Page-Out Rate	26.86	31.89	74.98	91.59	8.0	344.41	15.25	21.84
DASD Swap-In Rate	60.60	67.89	145.07	159.96	56.55	540.65	87.53	63.87
DASD Swap-Out Rate	82.01	91.74	196.03	216.17	76.42	730.60	118.29	86.04
Paging Rate To ES	662.80	725.70	845.68	950.63	242.64	1.19K	93.96	1.15K
Swapping Rate To ES	1.40K	1.44K	1.63K	1.55K	1.60K	1.69K	487.20	3.04K

3090-600E
128MB Central Storage
256MB Expanded Storage

expected effects of upgrade to a 3090-600S processor and the addition of either central or expanded storage. The base machine for the BEST/1 model is a 3090-600E with 128MB of central storage and 256MB of expanded storage. The configuration is operating with MVS/XA Release 2.2. The first column shows the effect of the increase in central storage requirements for the base MVS/ESA system. We are using a 3MB increase which may be slightly on the high side. To date, this author has seen this number range from 1.5MB to 3MB.

The second column shows the effect of the increase in active address space resident set size. We used a value of 40K or 10 pages as the increase value. Recent information indicates that the increase may be as much as 50K or approximately 13 pages per active address space.

Note the significant increases in all page transfers with the resident set size increase. The fourth column shows the expected increase in all page transfers by the addition of 4MB of named objects in the VLF address space. At the point the additional memory increases have been added to the base system, the 3090-600E is totally saturated.

The next column illustrates the expected effect on page transfer activity from the upgrade of the 3090-600E to a 3090-600S. Note the page rates drop significantly with the upgrade to the 3090-600S. This can be attributed to faster completion of transactions, particularly non-trivial TSO transactions, which results in a shorter frame residency time. Storage frames are released and returned to the free queue at a faster rate resulting in fewer page faults and demand paging operations.

The next column illustrates the effect of increasing the TSOA workload's transaction arrival rate by 12,000 transactions per hour (the equivalent of adding 230 TSO users to the processor complex). At this point, the 3090-600S processor is totally saturated due to the significant increase in page transfer rates for all categories.

The last two columns show the predicted effect of adding 128MB of central storage to the processor or adding 128MB of expanded storage to the processor. We note that though the demand paging and swapping increases with the addition of expanded storage are not dramatic, the predicted transfer rates for pages to and from expanded storage are extremely high. The rate of transfer of pages to and from expanded storage shown in the last column would exceed the guideline of five percent utilization of the processor complex for page transfer.

Table 2 illustrates the expected effect on workload response times from the various upgrade scenarios. We note the '*'s in the column where the TSOA transaction arrival rate was increased to 12,000 per hour after the upgrade to the 3090-600S processor. The last two columns illustrate the expected effect on workload response times from each of the memory upgrade scenarios.

The trivial TSO response times are not dramatically affected by the 3090S upgrade or the addition of central or expanded storage. This is due to the lack of contribution to response time of both processor resources and paging. The primary benefactors of both the 3090S upgrade and the memory additions are the

TABLE 2

MVS/ESA Response Time Evaluation								
Wkload/Trans/ Response	3090-600E	MVS/ESA + 3MB	Resident Set +10	+ VLF 4MB	3090-600S	TSO Trans + 12,000	Add 128MB CS	Add 128MB ES
Batch/22.0/9.86	10.10	10.13	10.32	10.41	7.91	*	9.02	9.57
TSOBT1/5.75K/.54	.54	.54	.55	.55	.53	*	.53	.53
TSOBT2/3.15K/6.64	6.64	6.77	7.55	7.94	3.24	*	4.72	6.23
TSOC/1.87K/1.08	1.08	1.12	1.13	1.14	1.10	*	1.11	1.11
TSOAT1/28.01K/.15	0.15	0.16	0.17	0.17	0.15	*	0.16	0.17
TSOAT2/13.42K/5.91	5.75	7.10	7.82	8.16	3.23	*	4.97	6.40

3090-600E
128MB Central Storage
256MB Expanded Storage

non-trivial TSO workloads. The primary contribution to response times for these workloads is processor and paging, particularly for fourth- and fifth-period users. With either memory upgrade, the increase in transaction arrival rate for TSOA saturates the processor. We see that in this situation, the central storage upgrade would be the better option from a total TSO response time perspective.

A final word. In representing I/O reductions expected from use of dataspaces and hiperspaces in the model, the analyst should be conservative. Additional measurement and analysis is required to ascertain the effects of page frame availability on the I/O patterns of the various types of datasets. It may be some time before a base of information is available from which reasonable guidelines for expected I/O reductions can be developed.

Many of the subsystem releases that will take advantage of the new MVS/ESA capabilities will not be available until late 1989. Their effects upon storage requirements, performance improvements and I/O reduction will require more knowledge and hands-on experience to determine reasonable modeling attributes.

Summary

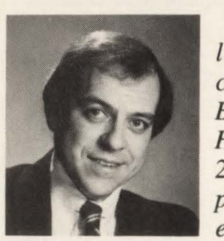
MVS/ESA will provide multiple opportunities for enhancing performance in the large systems environment. The major arena to be addressed by most installations will be central and expanded storage sizes and additional subsystems and application systems to take advantage of dataspaces, hiperspaces and the large virtual buffering options.

Establishing effective controls on the creation of dataspaces and hiperspaces will be critical to system performance. Most installations will begin using the SMF step initiation (IEFUSI) for the first time to limit both the number of dataspaces and hiperspaces created as well as the memory size for creation.

Large virtual buffering for CICS/VS, IMS/VS and other authorized tasks will experience decision points on whether to direct the LSR pools to dataspaces or hiperspaces, the size of the pools and the possible increase in paging and migrations rates compared to the performance gains received through reduction in I/O processing.

Additional RMF enhancements, available in Releases 4.1.0 and 4.1.1, will provide some of the necessary measurements on page residency and page movement from use of the new MVS/ESA facilities. Though some enhancements are available in SMF, very little additional information is available on page movement at the task level. Performance and capacity projections for MVS/ESA will be pretty much the same as for MVS/XA with the major difference being projecting memory requirements for new function usage.

ABOUT THE AUTHOR



J. William Mullen is a senior consultant for BGS Systems, Inc. He has more than 20 years of data processing experience and has concentrated on the performance measurement and capacity planning area for the last 15 years. Mullen is a past officer and director of SHARE, Inc. and CMG, serving for eight years as editor of the CMG Transactions.

DBA *from page 37*

long batch jobs to run and some paperwork to complete, submit the jobs early in the morning and then start completing your paperwork.

Submitting your long-running work at the beginning of the day will increase the chances of successfully completing it by the end of the day even if you encounter problems. Single threading your tasks will cause many time delays and will leave you wondering what you accomplished at the end of the day.

Avoid Repetition

If you find yourself repeating the same task numerous times, then that task is probably a good candidate for automation. Too often, business operations within a corporation are automated, but the data processing activities remain static and manual. Tasks such as generating forms and memos, database space calculations and implementing application maintenance should all be automated.

Some may fear being automated right out of a job. This is unfounded and untrue. Automating tasks which you repeat will allow you to complete the tasks faster and free up your time for other tasks. Automation leads naturally into the final subject of this article.

Summary

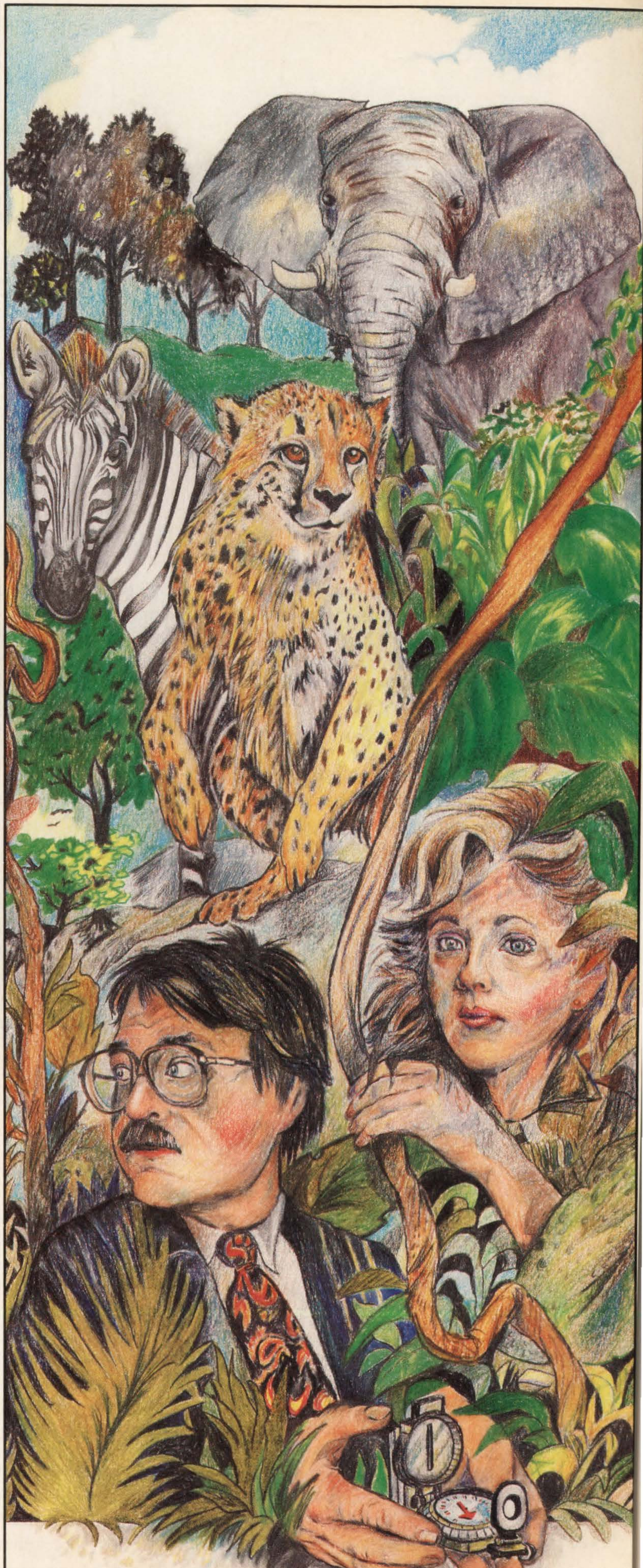
Remember it is important to complete tasks faster while not sacrificing quality which in turn allows more free time. Completing tasks slowly or withholding information will not promote job security.

No one is indispensible. There will always be someone who will come along to pick up the ball and run with it once you have dropped it. Making yourself invaluable is not as good an idea as it may seem on the surface.

At first it may seem that making yourself invaluable will promote job security. However, if you look past that initial reaction, you will find that making yourself invaluable in your current position will only ensure that you are not allowed to leave that position to do bigger and better things.

ABOUT THE AUTHOR

Robert Engle is currently a database technical specialist at the Federal Reserve Bank of St. Louis. Engle's experience in data processing includes both application analysis/development and database technical support.



DB2 And The OS/2 Database Manager

By Howard Fosdick

With some 5000 estimated licenses, DB2 dominates the mainframe database market in the same way CICS rules the market for teleprocessing monitors. However, IBM has been less successful in selling its desktop relational DBMS. The OS/2 Database Manager, an integral part of OS/2 Extended Edition, has not yet established dominance. This is due to the generally slow acceptance of Operating System/2. Still, DB2 professionals would be wise to concern themselves with the OS/2 Database Manager and its potential impacts on their shops for several reasons.

Systems Application Architecture (SAA) makes it clear that IBM's four RDBMSes will converge. Features available on one DBMS will soon become available for the others. Studying DB2's counterpart RDBMSes on other platforms portends its future directions.

Also, IBM's distributed database plans call for communications between DB2 and the Database Manager. Many IBM shops will opt for a two-tier communications architecture with DB2 acting as a repository for data downloaded to desktop machines running OS/2 Database Manager. Last, industry experts predict the Database Manager will become a dominant desktop DBMS, especially in larger accounts.

These facts force DB2 professionals to confront a host of issues that must be resolved in order to foster a happy marriage between DB2 and the Database Manager. DB2 personnel must plan for the probable use of the Database Manager in their shops from the standpoints of both applications development and technical support. DB2 users must address these issues:

- How portable are code and applications between DB2 and the Database Manager?
- How can DB2 and the Database Manager coexist in the mixed systems environment postulated by SAA?
- Will programmer skills be transferable from DB2 to the Database Manager? What about end-user training?
- How will MIS deal with the support issues involved in this mixed system environment?

No one article can address all these issues. This article presents some key aspects of the Database Manager in terms of the applications development and support issues that DB2 professionals will encounter. The versions of the products this article refers to are DB2 Version 2 Release 1 and OS/2 Database Manager Version 1 Release 2.

SQL Programming

A starting point in comparing any two relational DBMSes is their versions of SQL. Figure 1 provides such a comparison.

Like DB2, the Database Manager features Data Manipulation Language (DML) and Data Definition Language (DDL). Its DML is basically the same. However, the Database Manager offers the additional keywords EXCEPT and INTERSECT for operating on the results of SELECT statements.

The Database Manager's DDL for logical objects (tables, views and indexes) is essentially the same as DB2's, while its DDL for physical objects (tablespaces, index spaces, storage groups and databases) is totally different. The Database Manager does not generally permit control over physical storage because the desktop machine requires greater ease of use and less technical expertise than the mainframe environment. PC users do not have expertise in database administration.

Until the announcement of OS/2 Database Manager Version 1.2, the product's only security was the assignment of a single password per database. There were no SQL GRANT or REVOKE statements. Version 1.2, due in November 1989, adds these DCL statements to Database Manager SQL. However, while the Database Manager's approach to security is conceptually similar to that of DB2, significant differences exist in security administration. The Database Manager's GRANT and REVOKE statements are much narrower in scope than their counterparts under DB2.

The bottom line for the SQL comparison, therefore, is that the user-oriented SQL is largely the same, while the levels of the language that map onto real storage and security are different. Database program code that only issues DML and logical DDL will be much more portable than code that manipulates physical objects or controls security.

End users and applications programmers who work with high-level SQL statements will see little difference between the Database Manager and DB2. DBAs and systems support staff will see major differences. However, they should find it easy to adapt to the differences in Database Manager SQL because it is similar to and more simple than DB2.

More Subtle Aspects Of SQL Programming

If you intend to port applications be-

tween the Database Manager and DB2, there are many, more subtle implementation differences to be concerned about than merely the SQL language. Here are a few examples:

- Isolation levels: prior to Version 1.2, the Database Manager supported REPEATABLE READ only; now it supports CURSOR STABILITY and a new option unavailable under DB2 called UNCOMMITTED READ
- Log: the Database Manager log is assigned per database rather than per DB2 subsystem; the Database Manager does not perform log archiving like DB2
- Recovery: the Database Manager rolls back like DB2, however, disaster recovery of a database is to the time of last backup (either full or incremental) rather than to the last commit point
- Catalogs: each Database Manager database is assigned its own catalog that has a smaller number of tables than DB2's and the contents and column names differ; the basic design and usage of the catalogs are the same as in DB2
- Utilities: Database Manager utilities are similar to those of DB2 in their functions; however, there are important differences in their operations and the manners in which they are invoked.

The above comparisons are only indicative and are not comprehensive. However, they should be enough to convince you that porting code and applications between the Database Manager and DB2 *could* be a complex undertaking, depending on the nature of the application. Difficulties can be vastly reduced by knowledge of these differences during applications development and design.

The porting of code requires a lot more than simply equivalent SQL. Factors like isolation levels and locking can be critical. Structural aspects of the DBMS, such as its approaches to the catalog, security, utilities, logging and recovery are also fundamental to successful cross-system applications.

The Applications Development Environment

To accurately address applications development and support issues, consider more than the DBMS itself. The availability of similar tools across environments

FIGURE 1

Database Manager Versus DB2 SQL

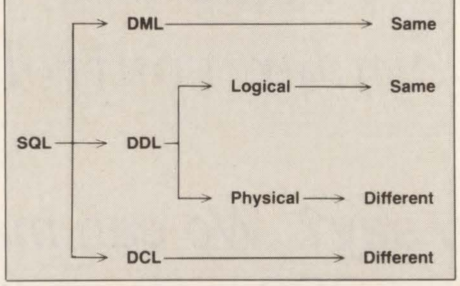


FIGURE 2

Database Manager Versus DB2 Applications Development Environment

Component Or Tool	DB Mgr.	DB2
QMF	–	X
Query Manager	X	–
Prompted Interface	X	X
Customized Interface	X	–
CSP	–	X
COBOL	X	X
PL/1	–	X
FORTRAN	X	X
C	X	X
REXX Interface	X	–
Presentation Manager	X	–
SAA Dialog Manager	X	–
ISPF Dialog Manager	–	X
ISPF/PDF Editor	X	X
TSO/ISPF	–	X
CICS	X	X
IMS/DC	–	X

probably does more to determine the transferability of programmer skills across systems than the similarities and differences of the DBMSes. For example, if a programmer generates applications with one tool with DB2 and that tool is not available with the Database Manager, a skills problem slows applications development regardless of how similar the DBMSes are. The mainframe programmer skilled in COBOL, ISPF/PDF and JCL who suddenly confronts C and the OS/2 command language on the desktop has the same problem. Applications development involves much more than compatible DBMSes.

This is, of course, the problem SAA addresses. Figure 2 shows that there is a definite disparity in the *applications development environment* between the Database Manager and DB2. While SAA is bringing great progress in this area from the viewpoint of applications developers and support personnel, there are still significant differences between the environments. Those working with multiple SAA RDBMSes need to be cognizant of these differences.

Also, recognize that the figure compares only IBM-vended tools for both environments. Any third-party software you use in either environment should be added to the comparison.

Many companies are committed to making their desktop development products compatible with the Database Manager. While tools are rare today, in the long-term application, development workstations will be available that fit well with both DB2 and the Database Manager.

Conclusions

The OS/2 Database Manager is, in spirit, DB2 on the PC. It contains the same broad sweep of features as DB2 and brings many of them, for the first time, to the desktop. In this respect, the Database Manager is an innovative product that breaks new ground.

While conceptually similar to DB2, the Database Manager is a different relational DBMS implementation. This article provides some examples of these differences but does not catalog them all. DB2 shops that obtain the OS/2 Database Manager and adopt IBM's SAA are wise to be cognizant of some of these less-frequently discussed issues.

There are many *real-world* factors important to MIS installations that deal with SAA DBMS other than DB2. Discussion among industry analysts, the press and vendors focuses primarily on SQL as the vehicle for transportability of applications and technical knowledge. This approach has merit. However, accepting it as a final answer is hazardous to MIS installations involved in real projects responsible for applications development and support. Developers must be concerned with operational characteristics of the DBMS and the larger applications development environment, as well as SQL language differences.

ABOUT THE AUTHOR

Howard Fosdick is President of the Midwest DB2 Users Group and a Vice President of the International DB2 Users Group. He has written six books and frequently lectures on technical topics and industry trends. Fosdick is an independent consultant. (312) 279-4286.

JOBTRAC™

Comprehensive Production Control Software that includes:

- Job Scheduling
- Job Restart and Recovery
- Sysout Archival and Online Retrieval
- Console Message Recognition and Reply
- System Operator Command Scheduling
- And More...

in a product that is
IN PRODUCTION QUICKLY
and is EASY TO ADMINISTER.

Goal Systems

Goal Systems International Inc. • 7965 N. High Street • Columbus, Ohio 43235 • Phone: (800) 537-4891
Goal Systems International S.A.R.L. • 88 avenue de Wagram • 75017 Paris, France • Phone: (1) 42 67 55 55 • Telex: 641.094

CIRCLE #74 on Reader Service Card ▲

with DB2!

PLATINUM *technology, inc.*
555 WatersEdge Drive
Lombard, IL 60148-9930
(312) 620-5000   FAX (312) 953-1923

For further information, in-house demonstration, or our exclusive no-obligation product evaluation call:

1-800-442-6861

CIRCLE #7 on Reader Service Card ▲

IBM Business Partner

Understanding Program Exceptions

By Harvey Bookman

Most programmers categorize program exceptions (program interrupts) under the general topic of ABENDs. Although the common usage may have made the lack of differentiation between a program exception and an ABEND acceptable, there is a big difference between the two. Program exceptions are detected by hardware while ABEND conditions are detected by software, usually the operating system. To further confuse the issue, programmers often do not understand the differences in processing between the hardware and the operating system.

When you attempt to add two packed decimal numbers together, whether the program doing the add was written in Assembler language or a high-level language like COBOL, the hardware will execute an Add Packed (AP) instruction. If one of the fields is not a valid packed field, the hardware generates a program interrupt and an exception condition occurs. This is distinct from an error such as attempting to open a non-existent file. An OPEN macro in Assembler (or OPEN verb in COBOL) generates a Supervisor Call (SVC) instruction. This passes control to the operating system (MVS, VSE and so on). If the operating system software detects a problem while attempting to open the file, it may decide to issue an ABEND instruction. This would be considered a true ABEND.

To more finely define ABENDs, the system breaks them into two different types, system ABENDs and user ABENDs. System ABENDs occur when the operating system issues an ABEND macro; user ABENDs occur when a user program issues the ABEND macro. ABENDs that occur in programs written by software vendors are not operating system ABENDs. For instance if DB2, IMS or IDMS software decides to issue an ABEND, the ABEND will be a user ABEND. Obviously, if DB2, IMS or IDMS requests a service from the operating system (such as to open a database file) and the request fails, a system ABEND may occur.

A list of the system ABENDs can be found in *OS/VS Message Library: VS2 System Codes*. User ABENDs can be found in the documentation describing a programming product such as the *VS COBOL Programmer's Guide* or by examining the program causing the user ABEND.

The basic difference between an ABEND and a program exception is that while an ABEND is always done on purpose by the program issuing it (either the operating system or the user program), a program exception will occur unexpectedly by a program (except in rare cases when a program interrupt is used in place of an ABEND or for special reasons in sophisticated code). This article will discuss the program exceptions numbered one through B (0C1 through 0CB).

One of the reasons (but certainly not the only reason) that programmers have trouble understanding dumps is because they do not realize what they represent. I have already explained the difference between an ABEND and a program exception and will now explain various program exceptions. I once was trying to explain to a colleague that most programmers do not know what an 0C1 program exception is. He quickly answered that most programmers do know that it is when you branch to a program that was not included in the linkedit. He had given a specific instance of an 0C1 program exception, not describing what it was. His answer was similar to an answer of "DB2" to the question of "What is a database?"

0C1 Operation Exceptions

An 0C1 program exception is an operation exception. It indicates that the CPU attempted to execute an instruction that was invalid for the hardware. On the IBM 370 hardware a machine language instruction is one byte long (excluding a few privileged instructions that are two bytes long), allowing 256 different values. Less than 256 instructions exist so not all of the possible values are valid. When an Assembler language program calls another program, the reference is through a V-type address constant that gets resolved at linkedit time. If the V-type constant is not resolved, it is set to zero. When the program then uses the V-type constant as an address to branch to, a branch to address zero in memory follows. This area in memory does not contain valid code and an operation exception occurs.

A program branch using an uninitialized address (such as using the value in a register that had not been properly set) may cause an 0C1. Actually, you are lucky if an unexpected value in an address does cause an operation exception. Otherwise, if the uninitialized value happens to contain a valid address of code, you may end up running a piece of code that you did not intend to. It would be an extremely difficult problem to debug since it is conceivable that you might not even know a certain piece of code was executed. This is an important point to remember: when a program gets a program exception or ABENDs, you are probably lucky. Rather than forcing you to spend hours or days finding where a problem is occurring, the ABEND code can be used to reference the reason for the problem or the PSW will point to the code that is causing the program exception. The PSW is generally of little value when a true ABEND occurs since it merely points to the ABEND macro that canceled the job step.

If an Assembler program attempts to do I/O before a dataset is opened, an 0C1 can occur. The address of the access method to perform the read or write is located in a control block (for example, the DCB). This address is filled in when the dataset is opened. If the dataset is never opened, the address will contain a zero and an I/O operation will cause a branch to address zero in memory.

The 0C1 dumps that are usually the hardest to solve are the ones that occur due to a storage overlay. It is easy to find out why the 0C1 occurred, but it may not be easy to find why the storage overlay occurred. For instance, if 100 bytes in the middle of your code in a dump contain a person's address and that area in the program had been attempting to execute, you will get an 0C1. In this case you know to look where the person's address is moved and try to find the point where your registers were not set properly. However, if a piece of code had been overlaid with two bytes of hexadecimal zeros, it would be a completely different matter. It may not be easy to determine where in the code the move of the bad data had been executed.

An 0C2 program exception occurs when a program in the problem state attempts to execute a privileged instruction. Applications almost always run in the problem state. They are not allowed to issue privileged instructions such as those that change the PSW and storage keys. A program in the problem state may attempt to issue a privileged instruction that was not coded in it, since an unitialized register may cause a branch to an area in which memory contains the value of a privileged instruction. An 0C3 is among the rarest of events on a 370 machine. It can only occur when a program issues an Execute (EX) instruction that has another EX instruction as its target.

0C4 Protection Exceptions

An 0C4 is one of the most common program exceptions. It occurs when a program attempts to store data into an area of memory (and also fetch data from an area of memory for some instructions) that is not allocated to the program. The specific technical reasons for the interrupt can get complex since the hardware recognizes key-controlled protection when the access key is not equal to the storage key, low-address protection in the first 512 bytes of memory and segment protection. Basically a protection exception indicates that although the memory requested for use exists in the hardware (either in real or virtual memory depending on the program executing), the hardware and operating system are not allowing the program to use it.

The basic difference between an ABEND and a program exception is that while an ABEND is always done on purpose by the program issuing it, a program exception will occur unexpectedly by a program.

An 0C4 should not be confused with an 0C5 which is an addressing exception. An addressing exception occurs when the address used by an instruction does not exist in the hardware configuration. 0C5s were more common before virtual storage. Programs used to run in real storage and, although addresses of up to 16MB were valid in the hardware, most machines had a considerably smaller amount of memory. With virtual storage, a program usually has access to the maximum range of addresses allowed in a 370 instruction and 0C5s rarely occur. When invalid addresses are used, the address will usually exist but will not be valid for use by the program running. This will cause an 0C4.

The most common reason for a COBOL program to get an 0C4 is that a subscript or index exceeded its maximum value. However, there is a greater chance that parts of the program's data or code will be overlaid than that an 0C4 will occur.

The 0C4 will only occur if the index or subscript has a positive or negative value large enough to exceed the boundaries of the program (for standard COBOL programs), exceeds the size of the WORKING-STORAGE and TGT (for COBOL CICS programs and COBOL II programs compiled with the RENT option) or exceeds the length of a GETMAINed area. In all of these cases, the 0C4 will not occur if the area referenced happens by chance to fall into another program or another storage area within the job step.

Another frequent cause of an 0C4 is the case when the USING clause in a CALL statement does not match the USING clause in the PROCEDURE DIVISION statement of the called program. This can cause the address of a data item to be missing or the length of a field to be incorrectly defined. In either case, the called program code may attempt to use an area not properly allocated to it and an 0C4 may ensue.

You should realize that when an 0C4 occurs, the instruction causing the protection exception may have already been partially executed. For example, if a Move Character Long (MVCL) instruction is executed, the protection exception may not occur until a multiple of 256 bytes had already been moved. This means that the registers pointing to the *from* and *to* fields,

F I G U R E 1

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01    TABLE1.
      05   FIELD1 OCCURS 3 TIMES                    PIC S9(3) COMP-3.
PROCEDURE DIVISION.
      MOVE ZEROS TO TABLE1.
      ADD +1 TO FIELD1 (1).
```

F I G U R E 2

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01    FIELD1.
      05   FIELD2                                   PIC S9(3).
01    FIELD3                                        PIC S9(3) COMP-3 VALUE +123.
PROCEDURE DIVISION.
      MOVE FIELD3 TO FIELD1.
      ADD +1 TO FIELD2.
```

as well as the length of the data to be moved, may not contain the values which they contained when the instruction began to execute.

0C6 Specification Exceptions

An 0C6 program interrupt is a specification exception that occurs when certain instructions are specified incorrectly for execution. Examples are: a branch instruction attempts to branch to an odd numbered address; an odd-numbered register is specified in an instruction that requires an even-numbered register; an invalid floating-point register is used; or a decimal division or multiplication is executed where the length of the first field is not greater than the length of the second field.

0C6s rarely occur in COBOL programs. With 360 hardware they occurred more frequently. Instructions like Load Halfword (LH) and Load (L) fullword required the operands to be respectively on halfword and fullword boundaries. When COMP fields were used and slack bytes were not properly defined, 0C6s sometimes occurred.

0C7 Data Exceptions

The 0C7 is the most widely received of all of the program exceptions. It is a decimal data exception and occurs when data is in an invalid format for the instruction referencing it; specifically the numeric digits are not in the range of zero through nine or the sign is not one of the valid values of A, B, C, D, E or F. They most often occur when a program attempts to use a field that had not been initialized.

Although many 0C7s occur through oversights, a large number of 0C7 program exceptions occur through lack of knowledge. One way to get invalid data into a COMP-3 field is to initialize a group level rather than the field itself. After the code in Figure 1 is executed, each of the three occurrences of FIELD1 will contain hexadecimal values of 'F0F0'. The fields do not have valid signs ('0') and each field contains invalid digits ('F'). A data exception will occur if any of the occurrences of FIELD1 are used in arithmetic operations.

Another piece of code that will cause an 0C7 is in Figure 2. A field defined with a PIC of COMP-3 is moved to a group level and no conversion takes place. After the MOVE instruction is executed, FIELD1 (and, therefore, FIELD2) will contain a hexadecimal value of 123C40. This is not a valid display numeric field and will cause a decimal data exception if it is used in an arithmetic operation.

Program Exceptions 0C8 Through 0CB

I will briefly mention other program exceptions. An 0C8 is a fixed-point overflow exception, occurring when a fixed-point signed binary arithmetic instruction or an instruction that shifts left a signed number causes an overflow. An 0CA is a decimal overflow exception and is similar to an 0C8, except that an 0CA applies to decimal rather than binary numbers. It occurs when at least one non-zero digit is lost during the processing of a decimal instruction. When either a decimal overflow or a fixed-point overflow occurs, execution of the instruction that caused the problem is completed. The result is correct except that the information that overflowed is lost.

An 0C9 is a fixed-point divide exception that is caused when a divide by zero occurs in a binary arithmetic operation or when a Convert To Binary (CVB) instruction causes a result that exceeds the 32-bit register size. A decimal divide exception (0CB) occurs when a zero is used as the divisor in a decimal divide instruction or when the quotient is greater than the size of the receiving field.

The first step that is required to read a dump is to understand why the dump was produced. It is often needless to analyze the detailed data in a dump because the reason for the ABEND was quite explicit. For instance, if a program gets a user ABEND of 200 and it is clearly known through documentation that this error indicated a control card was missing from a file in the JCL, there is no reason to spend time analyzing the data in the dump. If the control card in question was present and the error still occurred, only then would it be necessary to investigate the dump.

Too often a programmer attempts to find the reason for a user ABEND by searching through *OS/VS Message Library: VS2 System Codes* or, even worse, by searching through *MVS/370 Message Library: System Messages*. Programmers must understand the differences between program exceptions, operating system ABENDs and user ABENDs. Additionally, programmers must learn to properly use the two manuals referenced above. The popular expression, "I don't have to memorize it; I can look it up in the manual," is true if you are looking in the proper manual. Experience has shown that the average programmer does not understand the purpose of what should be the most commonly-used manuals. Knowledge will make the difference between fumbling through a dump page after page and effectively analyzing the information in it.

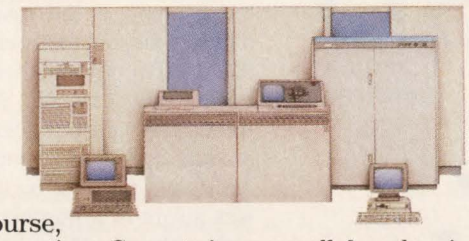
ABOUT THE AUTHOR

Harvey Bookman is President of Bookman Consulting, Inc., a software development company specializing in programmer proficiency testing. He is responsible for the development of the expert system testing service, TECKCHEK™. Bookman Consulting, Inc., 67 Wall Street, Suite 2411, New York, NY 10005, (212) 819-1955.

Our laser printers give new meaning to "sharing"

Until now, sharing has been pretty one-sided: several workstations had to share one laser printer. But Printer Systems has redefined the idea of sharing. Now a laser printer can share across several dimensions.

Different platforms

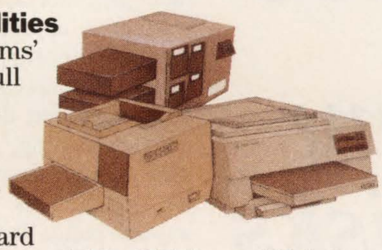


Printer Systems laser printers can be shared by several different kinds of computers. Like IBM. And DEC. And, of course, PCs. All at the same time. Coax, twinax, parallel, and serial connector ports are provided on all our laser printers.

Different applications

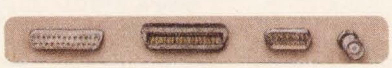
And you can get some pretty fancy things out of these different computers, without a lot of programming. Like barcodes. And FlexFonts from BitStream, with access to over 2,000 fonts. So now your mainframe can do what you thought you needed a PC for.

Different personalities



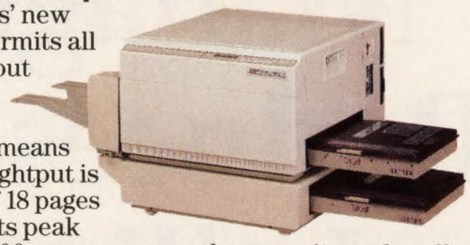
With Printer Systems' new laser printers, full emulation of the IBM 3812, the DEC LN03 Plus, and the HP LaserJet II is a given. In fact, our printers not only come standard with all the same fonts as IBM's, DEC's, and HP's laser printers, but our font matrices are precisely identical to theirs. So ours print exactly the same as theirs.

Real compatibility



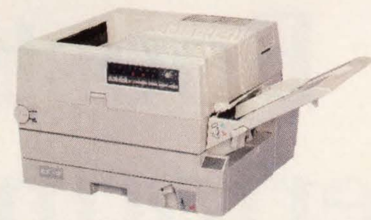
Complete compatibility is a given, too, right down to the resolution. Like the IBM 3812, you can have 240 dpi for IBM applications. Or 300 dpi like the DEC LN03 Plus and the HP LaserJet II. In the same machine. Switched–not scaled–by the machine itself depending upon the application.

Introducing the Intelliprint 218®



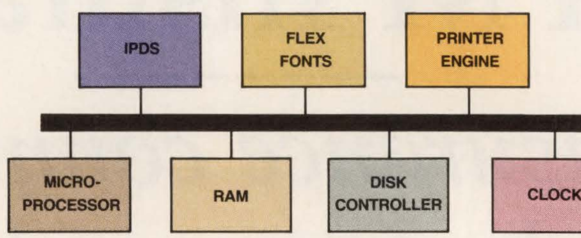
Printer Systems' new Intelliprint 218 permits all this sharing without giving up performance. Its 32-bit RISC technology means that its real throughtput is the rated speed of 18 pages per minute. And its peak duty cycle of 50,000 pages per month means it can handle however much you demand of it.

And the Intelliprint 106®



Sometimes, you can have too much of a good thing. So Printer Systems developed the Intelliprint 106. It, too, is based on 32-bit RISC technology, so its 6 page per minute rated speed is its real throughput. And its duty cycle of 3,000 pages per month is ideal for lower-demand situations.

Open architecture on a PC bus



Printer Systems has also redefined "sharing" by putting a PC AT inside the Intelliprint 218. That provides all the functionality of a dedicated workstation. And, like any other workstation, it affords the flexibility to add options, like a LAN interface, for instance.

Wait, there's more

In fact, there's a lot more. Printer Systems has more up its sleeve, like spooling. And real-time, automatic switching between interfaces. And different graphics emulations. But we can't tell you about all of it yet. At least not here. Why not give us a call at 1-800-638-4041? We'd be delighted to fill you in. So you can start sharing the way sharing was meant to be.

Compatible in every way

Printer Systems Corporation, Corporate Headquarters: 9055 Comprint Court, Gaithersburg, Maryland 20877
(301) 258-5060, (800) 638-4041, Fax: (301) 926-7333, Telex II: 710 828 9621

CIRCLE #50 on Reader Service Card ▲

Dataset Management Reduces DASD Utilization, Boosts CPU Performance For GPM Insurance

Insurance company tracks tapes and disks with EPIC/VSE

By Rachael Payne

In today's competitive environment, successful organizations increasingly depend on information technology to sharpen their ability to anticipate trends, service customers and reduce costs. Even details like dataset management can be automated to improve productivity and contribute to the bottom line. Government Personnel Mutual Life Insurance Company (GPM) in San Antonio, TX is a case in point. By taking control of its data processing resources (including tape and disk management), GPM increased management control over its disk and tape resources, saved money and decreased DASD and CPU utilization in the bargain.

To meet its burgeoning DP requirements, GPM operates an IBM 4341 Group II under VM/SP. Its main production guest machine is under VSE/AF. The company is currently installing VSE/SP 2.1.7. Its DASD configuration includes 22 spindles of IBM 3350 drives which are being phased out and 16 addresses of IBM 3380 drives. GPM's data center operates 24-hours-a-day, six-and-a-half days a week and, according to Systems Programmer Roger E. Moore, it manages a rapidly growing tape and disk library, including more than 1800 active reels of tape. To better manage this vital resource, GPM uses EPIC/VSE from Tower Systems (Costa Mesa, CA), a tool designed to consolidate management of datasets and reduce manual intervention in controlling both tape and disk dataset resources.

More Throughput

EPIC/VSE was installed in 1987 when the growth of the insurance company's datasets began to threaten the ability of the staff to control them properly. It had been using Tower's TD-FAST but was forced to upgrade because the former product does not support 3380 disk drives. Although GPM's evaluation committee was aware of a number of other commercial packages, it selected EPIC/VSE primarily because of its compatibility with TD-FAST. No catalog changes were required other than the EPIC automatic conversion utility. ''The conversion process from TD-FAST to EPIC was virtually painless,'' Moore reports. ''Both systems used the same catalog, so all that was required was essentially a backup and restore and we were in business.''

EPIC/VSE eliminates time-consuming and error-prone manual dataset management procedures. It offers users a unified, interactive approach to managing, securing and reporting on all tape and disk dataset resources. Designed to manage both tapes and disks using a single, integrated catalog, EPIC/VSE's tape facilities improve response time at the operator console. Its disk facilities promote improvements in DASD space and channel utilization.

GPM is gratified that EPIC/VSE is helping reduce its DASD requirements, Moore notes. It is more efficient in allocating DASD space than an operator using manual calculations. ''Personally, I find I normally over-allocate by as much as 50 percent to accommodate future growth of files. With the ability of EPIC to control secondary allocations and to truncate track allocations when files are closed, we take only the space we need,'' Moore points out. Without its present dataset management system, Moore says GPM would have to add 25 percent to the volume of disk storage required for interim work files. For example, if an application program defines a data file that had 10,000 tracks allocated to it, but only writes two records to that data file, EPIC/VSE modifies the VTOC entry for that file so it physically takes only as much room as it needs. The remainder of the disk space is released to the pool. "It allows us to define pools of disk work space and that has cut down the amount of disk work area that would otherwise be required." Moore adds.

He emphasizes that EPIC/VSE's disk space allocation is based on an entry in the EPIC Data Set Name (DSN) catalog for its primary and secondary allocations, if desired. "This eliminates the error-prone guesswork of calculating the number of blocks or tracks needed for a disk file," he says. Moore explains further that if a programmer underestimates the number of records required, the program allocates an additional 50 percent of the primary allocation as a default if no secondary allocation parameter has been specifically defined in the DSN catalog. If too many records are estimated, automatic file truncation returns unused file space to the disk pool for immediate availability. This provision reduces the pressure for additional disk devices.

GPM computer operators also like EPIC's Enhanced Subdataset feature allowing them to put multiple, individually named datasets on a single reel of tape. When they call for any one of those datasets, EPIC automatically identifies the correct tape and positions itself at the appropriate point on the tape. GPM operators use it extensively for tape backups and occasionally for restores.

Although EPIC allows data centers to dispense with the need to affix physical labels to tapes, GPM still maintains that practice. "No one really looks at the labels anymore but we still do it," Moore admits. "Old habits are hard to break." On the other hand, GPM takes advantage of EPIC's vaulting option for off-site storage of tapes. Paper labels for these vault-bound tapes are occasionally useful. Labels help archive and personnel confirm that they are dealing with the correct tape. As new generations get created, the vaulting mechanism tells operators which reels to bring to the primary data center and which to archive.

Reduced Rerun Time

Measuring the cost-effectiveness of such automation is notoriously difficult, Moore concedes. Nevertheless, the consensus at GPM is that EPIC has delivered substantial benefits. In justifying the product, GPM first considered the operations rerun time that could be saved by prohibiting errors of the incorrect tape going into an application. Decreased rate of errors resulted in improved productivity.

"Throughput has been tremendous," he adds. "We derive a high level of productivity without a lot of manual intervention. EPIC has certainly helped control both the growth of personnel and the expansion of disk space requirements. It was obvious to us from the beginning that it generated dollar benefits in excess of its costs," Moore says. EPIC/VSE helps GPM concentrate on the total management of datasets, regardless of their storage media, and helps keep track of the location of datasets. Features, such as dynamic allocation, increase programmer productivity, he adds. According to Moore, it all comes down to confidence in the system. The new tape management system is simply trusted more. "The system detects whether operators accidentally mount or scratch wrong tapes," he says.

As its business needs evolve, GPM expects its tape and disk library to grow proportionately with its client base. The insurance company will expand its range of services and this action will put even more pressure on dataset management. GPM is well-positioned to manage that growth.

"EPIC/VSE is important to the integrity of the data center. It allows us to utilize disk space much more efficiently, makes sure that correct tapes are mounted and helps us better maintain our catalogs. It will not limit our growth; our dataset requirements are provided for," Moore explains.

Moore would like to comment on Tower's technical support, but cannot. "When a software product works well, you never have any problems with it and it does everything it's advertised to do, there's not much to say. EPIC is definitely an asset to GPM," he points out. Finally, EPIC has been well accepted by the GPM computer operators. "What would the operators say if we took EPIC away from them?" Moore muses. "I don't think the answer would be printable. We'd have a small mutiny here," he concludes.

GPM increased management control over its disk and tape resources, saved money and decreased DASD and CPU utilization in the bargain by taking control of its data processing resources. Seated from left are Manny Raneir, programmer analyst, and Gary James, supervisor of operations. Standing from left are Bill Sendelbach, manager of operations, Don Havel, programmer analyst, and Roger E. Moore, systems programmer.

ABOUT THE AUTHOR

Rachael Payne is a New York-based free-lance writer specializing in technical issues.

VM In The Development Center

X Program Writing In XEDIT

By Michael Seadle, Ph.D.

This is the second in a series of articles discussing how application development teams can use VM to improve their productivity. The first article (June 1989 issue) presented an overview of development issues and focused on how to make effective use of NOTE, VM's electronic mail facility. This article continues the theme by examining VM's edit environment.

Profile XEDIT

Writing programs in VM requires a full-screen editor: XEDIT. You should configure XEDIT so that the full-screen display begins just below the lines showing file name, line number and cursor position. You may also want to move the scale line with its column numbers from mid-screen to the top. To do this, you need to create a PROFILE XEDIT:

```
/* REXX profile for XEDIT */
SET SCALE ON 3
SET CURLINE ON 4
```

If you FILE at this point, your next edit session will use these new commands.

Typing errors for common commands occur frequently. You can anticipate certain transpositions and tell XEDIT to treat them as synonyms. For example:

```
SET SYNONYM FIEL 4 FILE
```

This command tells XEDIT to treat the four-character string FIEL as if it were FILE. You can include any number of synonyms, but you should be careful not to transform one command accidentally into another.

AUTOSAVE

XEDIT has an option, AUTOSAVE, that saves your file at regular intervals during each session. Shops in which the system crashes with appalling frequency need to set this option on. More stable installations may prefer not to. All editing in XEDIT takes place in memory. XEDIT normally updates a file only when told to do so. This has some advantages. You might have a bright idea about moving code around, get half way through and realize it will not work. In XEDIT mode, a simple QQUIT command leaves your original unaltered.

Effective use of this option requires some common sense. If you spend eight hours editing a file without SAVEing it, you leave yourself seriously vulnerable if the system crashes. Likewise when you feel a bright idea coming on, you would do well to SAVE before starting and not SAVE again until you are sure you want to keep what you have done.

NULLS

XEDIT allows inserting forgotten words or letters into a line using the INSERT key. This is easiest if you include:

```
SET NULLS ON
```

in the PROFILE. If you set NULLS OFF, you must move the cursor to the end of the line, hit ERASE-EOF, then move back to where the insert belongs. The disadvantage of NULLS ON is that anything you put at the right-hand edge (such as comments) will fall to the left unless you explicitly enter spaces.

You can avoid this disadvantage by adding:

```
SET FULLREAD ON
```

to the PROFILE. Then XEDIT will read every position on the screen, including nulls which it converts automatically into the missing blanks. FULLREAD has higher overhead because it causes more data transmission. It may cause a performance problem, particularly for remote terminals. You should monitor how well your system performs with FULLREAD ON. Instead of including it automatically in a system PROFILE XEDIT, you might want to train staff to activate it only as needed from the command line.

PF Key Settings

You will probably want to define the PF keys to fit familiar conventions, perhaps ones from prior edit environments. XEDIT allows you to set as many PF keys as the terminals have and to override any defaults. You should, however, observe some exceptions to this freedom. PF3 is one. VM uses PF3 again and again to QUIT a process. Moving QUIT to another PF key may ultimately slow the staff's adaptation to VM. Also, you should never set a PF key to QQUIT, even if you think users will use that command frequently. QQUIT exits an edit session immediately without saving the file and with no user-friendly ''are you sure'' message. The extra effort of typing the command (or its abbreviation, QQ) is worthwhile to avoid serious loss from accidentally hitting the wrong PF key.

You should define the PF keys so that the staff can easily examine more than one file at once in split-screen mode. These are the same commands as you used with NOTE. Just add them to the PROFILE:

```
SET PF21 SCREEN 2
SET PF22 SCREEN 1
```

You can, of course, use any PF key, not just 21 and 22. The first command splits the screen horizontally; the second unsplits it. The horizontal splits are the most common because they show a full 72 columns, but staff members may also want a vertical split for comparing, say, two versions of the same program:

```
SET PF23 SCREEN 2 V
```

You can let them move the cursor manually from one split-screen area to another with the arrow or tab keys or you can set up another PF key to speed the process:

```
SET PF24 SOS TABCMDB
```

TABCMDB jumps the cursor from command line to command line.

Some uses of XEDIT require all upper case — coding COBOL, for example. Others, such as documentation, must have mixed case to be readable. You may also wish to write your REXX EXECs in upper case, but put the comments in mixed case to make them easier to distinguish. Defining two more PF keys simplifies switching from mixed to upper case and back:

```
SET PF16 CASE U
SET PF17 CASE M
```

Staff members may want to look at data files whose line length exceeds the screen width. XEDIT's default lets the extra data wrap around onto the following lines. Most people find this difficult to read. You could have them type a VERIFY command or you can define another PF key:

```
SET PF18 VERIFY 1 72
```

You may even want to implement this

command automatically every time they XEDIT a file. In that case you add:

```
SET VERIFY 1 72
```

to the PROFILE.

XEDIT's default settings include PF keys for paging through a file one screen at a time. This will suffice for small files but not for anything more than 60 lines (three screens). Staff members will need to be able to move to one end of a file or the other quickly. A PF key for the TOP command is simple:

```
SET PF13 TOP
```

But BOTTOM puts the last line of the file at the current line which is often so high that only one line displays. A multiple command solves this situation:

```
SET LINEND OFF
SET PF13 BOTTOM # '-17'
SET LINEND ON #
```

By temporarily turning off the normal end of line character (#), XEDIT equates PF13 to a string that includes the #. PF14 then treats BOTTOM and −17 as two commands and it executes −17 after BOTTOM as if they were on separate lines. The result is that staff members see a full-screen display of the final 18 lines of data.

From the command line, staff members can move any distance up and down a file by typing + or − the number of lines. To repeat the action, they can use the default setting for PF9, "=", that automatically reexecutes the last command. If they wish to recall the command to change it, they can use the default setting for PF6, "?". Moving through a file this way requires several keystrokes. Instead, you can define extra PF keys for commonly used screen shifts. A half-screen shift, forward and backward, would be:

```
SET PF19 '+10'
SET PF20 '-10'
```

VM allows several PROFILE XEDITs. You may put a standard edition on the public access Y disk and also set up one specifically for the project team on another shared minidisk. Individual staff members can ignore both of these and keep a unique PROFILE XEDIT on their own A disks. VM will use whichever PROFILE XEDIT it finds first in the search chain, beginning with the A disk and proceeding down the alphabet. A shop can have as many different PROFILEs as staff members. Remember, however, that such a PF key Tower of Babel makes training and troubleshooting decidedly more difficult.

XEDIT Macros: XC

XEDIT allows you to write additional commands called macros. Two of the most popular macros simplify the process of copying lines from one file to another in split-screen mode. The first copies either one line (XC) or a group of lines (XCC. . .XCC). The second (G) inserts whatever XC or XCC copied. Both are available on public VM workshop tapes. The versions below offer a more visibly structured format:

```
/*  XEDIT prefix area copy macro  */
PARSE ARG PREFIX CALLTYPE PLINE OPERANDS
100.MAINLINE:
  PARSE SOURCE . . . . . MACNAME .
/*  Error checking  */
IF PREFIX NOT = 'PREFIX' THEN SIGNAL 200.NOTPREFIX
IF CALLTYPE = 'CLEAR' THEN EXIT
IF CALLTYPE = 'SHADOW' THEN SIGNAL 210.SHADOW
IF CALLTYPE NOT = 'SET' THEN SIGNAL 200.NOTPREFIX
/*  Is the command XC or XCC?  */
SELECT
WHEN MACNAME = 'XC' THEN DO
IF OPERANDS = '' | DATATYPE(OPERANDS,'W') THEN
'COMMAND : 'PLINE 'PUT' OPERANDS
ELSE SIGNAL 330.INVALIDOP
END
/*  For XCC, determine whether it is the first or last of the
    pair. If first, set up the start of a block. If last, close
    and save the block.  */
WHEN MACNAME = 'XCC' THEN DO
IF OPERANDS NOT = '' THEN 220.INVALIDOP
'COMMAND EXTRACT /PENDING BLOCK' MACNAME '/'
/*  Was there a prior XCC?  */
IF PENDING.0 = 0 THEN  /*  No prior XCC  */
'COMMAND : 'PLINE 'SET PENDING BLOCK' MACNAME
ELSE DO  /*  There was a prior XCC  */
'COMMAND : ' PENDING.1 'SET PENDING OFF'
'COMMAND : 'MIN(PENDING.1, PLINE) 'PUT',
ABS(PLINE - PENDING.1) +1
END
END
/*  Something went wrong if the command is neither
XC or XCC  */
OTHERWISE 'COMMAND EMSG INVALID SYNONYM -'
  MACNAME
END
EXIT
/*  Various error message routines  */
200.NOTPREFIX:
'COMMAND EMSG USE ' MACNAME ' AS A PREFIX
  COMMAND'
EXIT
210.SHADOW:
'COMMAND EMSG DO NOT USE' MACNAME 'ON A
  SHADOW LINE'
EXIT
220.INVALIDOP:
'COMMAND EMSG INVALID OPERAND -' OPERANDS
EXIT
```

XEDIT supplies the values for PREFIX, CALLTYPE and PLINE. The macro itself checks to make sure the command was used in the prefix area and not on a shadow line (one which the X or XX. . .XX function made invisible). With XC, it does an immediate write to storage (PUT) for the number of lines indicated by the operand. For example:

```
XC3 ===== FIRST LINE TO COPY. . .
    ===== SECOND LINE TO COPY. . .
    ===== THIRD LINE TO COPY. . .
```


BlueLine
SOFTWARE INC.
Unlock your lockfile with
VLOCK/VM!

will hold all three lines for copying. If you use XCC instead, the macro determines whether a pending block exists or should be started. It simply does the counting for you. This form:

```
XCC =====  FIRST LINE TO COPY. . .
    =====  SECOND LINE TO COPY. . .
XCC =====  THIRD LINE TO COPY. . .
```

has the same result as above.

The PUT command loads the block you copied into a temporary file which lasts the length of your XEDIT session.

XEDIT Macros: G

To retrieve the block, you need another XEDIT macro called G:

```
/*  XEDIT macro for inserting stored blocks  */
PARSE ARG PREFIX CALLTYPE PLINE OPERANDS
100.MAINLINE:
/*  Error checking  */
IF PREFIX NOT = 'PREFIX' THEN SIGNAL 200.NOTPREFIX
IF CALLTYPE = 'CLEAR' THEN EXIT
IF CALLTYPE = 'SHADOW' THEN SIGNAL 210.SHADOW
IF CALLTYPE NOT = 'SET' THEN SIGNAL 200.NOTPREFIX
/*  Get the stored block and forget any operands  */
'COMMAND :' PLINE 'GET'
IF OPERANDS NOT = '' THEN,
'COMMAND EMSG OPERANDS IGNORED'
EXIT
/*  Various error message routines  */
200.NOTPREFIX:
'COMMAND EMSG USE G AS A PREFIX COMMAND'
EXIT
210.SHADOW:
'COMMAND EMSG DO NOT USE G ON THE SHADOW LINE'
EXIT
```

This macro reads the same XEDIT values and does the same error checking as XC, then simply inserts the stored text. For example, if you used G here:

```
G =====  LAST LINE BEFORE THE COPIED BLOCK
  =====  FIRST LINE AFTER THE COPIED BLOCK
```

the result is:

```
=====  LAST LINE BEFORE THE COPIED BLOCK
=====  FIRST LINE TO COPY. . .
=====  SECOND LINE TO COPY. . .
=====  THIRD LINE TO COPY. . .
=====  FIRST LINE AFTER THE COPIED BLOCK
```

You should put the G macro in a file called G XEDIT and the XC macro in a file called XC XEDIT. It is important that you include a synonym command for XCC in the PROFILE XEDIT so that XCC commands find the macro under its XC name:

```
SET PREFIX SYNONYM XCC XC
```

You could instead also separate XC and XCC into two macros.

XEDIT macros are nothing more than REXX programs which use XEDIT facilities and run during an XEDIT session. Staff members might want to write macros that enhance existing commands such as the search and replace. If, for example, they want to replace a variable name only under certain conditions, a macro can check for that condition before executing the replacement. Macros can run either from the prefix area, as in the examples above, or from the command line. They can be as long and elaborate as the writer wants.

With XEDIT, the choice is yours.

This article is adapted from chapter ten of VM Applications Handbook *(McGraw-Hill, 1989) Gary McClain, editor.*

ABOUT THE AUTHOR

Michael Seadle, Ph.D., is head of user services at Eastern Michigan University, Ypsilanti, MI. He has worked as an applications developer, database administrator and VM systems programmer. Also, he helped establish and is an active member of the VM Enthusiasts of Michigan, a VM users' group.

TSO/E REXX An Overview

By Zeida Heavener

New to the TSO environment, REXX may replace CLIST; it is easy to learn, yet powerful.

After being available to the VM/CMS user community for several years, the Restructured EXtended eXecutor language (REXX) has recently become available to the MVS/TSO community. Although the words *restructure* and *extended* may lead you to believe that REXX is a new version of an older language, REXX was designed by IBM during the late 1970s and early 1980s as a new language without any requirement for compatability with previous programming languages.

It is important to note that IBM has made a statement of direction regarding REXX: "TSO/E REXX is the implementation of the SAA procedures language on the MVS system." Among other things, this means that REXX will probably be around for a while and it will probably be widely used.

REXX programs, which are sometimes referred to as REXX EXECs, can now be run interactively under TSO/E or in batch under MVS. REXX EXECs will probably be used extensively in the TSO/E environment where they will perform tasks that have traditionally been performed by CLISTs.

The TSO/E REXX lanaguage is an interpreted language. Among the strong points of the language are its ease of use, wide range of built-in functions and extensive debugging capabilities.

The following paragraphs are an overview of the features within REXX that I consider most interesting and useful. This overview will not include specific details such as instruction syntax, but it will give you an idea of what tools are available to the REXX programmer.

Ease Of Use

The REXX language is essentially free format. Instructions can start at any column and blank characters or lines are not meaningful. There are no headings or sections to a program. Programs usually have one instruction per line; however, REXX allows multiple instructions on one line and an instruction can span several lines. Punctuation is not required except to indicate continuation of an instruction or when there is more than one instruction per line. Variables do not need to be declared or data types identified (the meaning of data depends only on its usage).

An Example Of A REXX EXEC

Following is a simple REXX EXEC that divides two numbers supplied by the user.

```
/*      REXX      */
say 'please enter miles traveled.'
pull miles
say 'please enter gallons of gas used.'
pull gallons
mileage = miles / gallons
say 'gas mileage for the trip was' mileage
exit
```

The first line of the program shown above is a comment. Comments begin with "/*" and end with "/*". Comments can be on one or more lines or on part of a line. IBM recommends that all REXX EXECs running under TSO begin with a comment containing the word "REXX" in the first line. This is called the REXX identifier and distinguishes it from a CLIST.

The SAY instruction is used to display a message on the terminal and PULL reads user-entered input.

There are three variables in the program shown above: miles, gallons and mileage. Something peculiar about REXX variables is that before a REXX variable is initialized, it contains its own name translated to uppercase. For example, the instruction "say x" will display "X" unless the value of x has been altered by a previous instruction. By means of an assigment, the value of a variable can be changed to contain decimal, floating point or character string data.

Use Of Mixed And Lowercase Characters

The REXX language processor will translate all alphabetic characters to uppercase before they are processed. To prevent this from happening, data within an EXEC can be enclosed in single or double quotes. This will ensure that the information is processed exactly as typed.

When reading input from the terminal or arguments passed from another EXEC, the PARSE instruction can be used to prevent uppercase translation from taking place.

Additional Operators

In addition to the conventional arithmetic operators (+,-,*,/,**), there are two new operators in the REXX language:

% divide and return a whole number

// divide and return the remainder only

In addition to the conventional comparison operators (=,<,>,7), REXX provides two ways of testing for equality: equal (=) or strictly equal (==). Two expressions are ('strictly equal') when everything, including blanks and case, are exactly the same. For example, the following expressions are all true:

```
'ABC'   =   'abc'
' ABC'  ==  ' ABC'
'ABC' 7==   'abc'
```

The NUMERIC Instruction

The NUMERIC instruction of the REXX language is used to set parameters which in turn determine the way in which arithmetic operations are carried out.

The DIGITS option of the numeric instruction controls the precision, or number of digits, to which arithmetic operations will be evaluated. There is no upper limit, meaning that one could ask for an arithmetic operation to be carried to 1000 or more decimal places. However, the IBM manual does warn about the excessive resource consumption of such programs. The default of the DIGITS option is nine decimal places.

The FUZZ option of the NUMERIC instruction controls how many decimal digits of error will be ignored during a numeric comparison operation. This value defaults to zero which means that under normal conditions, all digits are significant.

The FORM option of the NUMERIC instruction controls the form of exponential notation that will be used by REXX for the result of arithmetic operations and built-in functions. It can be SCIENTIFIC or ENGINEERING.

Program Flow

There are many instructions in the REXX language which can be used to control program flow. Among them is the conventional IF . . . THEN . . . ELSE instruction, a variation of it called the SELECT instruction and many DO . . . END instruction combinations. A few of these are the DO UNTIL which tests the expression after the loop executes at least once and the DO WHILE which tests the expression before the DO executes the first time. In addition, there is a DO FOREVER which will go on forever unless it encounters a LEAVE instruction.

Unlike most procedural languages, there is no GO TO instruction in REXX — the closest thing to it is the SIGNAL instruction. I think that the developers of the REXX language wanted programmers to write structured code only, so they left out the GO TO instruction.

The SIGNAL instruction works like the GO TO instruction; it is used to transfer control to another portion of a program identified by a label. However, the use of the SIGNAL instruction within REXX is discouraged and can lead to problems. This is because all DO loops and IF clauses are terminated when the SIGNAL instruction is executed. For example, if a SIGNAL instruction is executed from within a DO loop and it causes a branch to a different instruction within the same loop, the program will end with an error message as soon as it encounters the END statement. This is because the DO loop was terminated as soon as the SIGNAL instruction was issued, even though you did not leave the loop. I recommend that the SIGNAL instruction be used with caution and only in situations such as branching to an error routine when an error is encountered.

REXX supports the use of subroutines. A subroutine is internal if it is contained within the same EXEC. If an EXEC calls another EXEC, this makes the called EXEC an external subroutine. Unless specified by the PROCEDURE option, an internal subroutine will share all its variables with the main program. This means that the value of a variable is what was last assigned, regardless of whether the assignment was made in the main program or in the subroutine. External subroutines do not share variables with their callers and information must be passed to them via arguments. All REXX routines are recursive, which means that a routine can call itself.

Built-In Functions

REXX provides a great variety of built-in functions. These have been classified as arithmetic functions, comparison functions, conversion functions, formatting functions, string manipulating functions and miscellaneous functions. Two examples are:

WORDS returns the number of blank delimited words in the input string; for example, X = WORDS ('the sky is blue') will cause X to be set to 4.

WORD returns the nth blank delimited word in a given character string; for example, X = WORD ('the sky is blue',2) will cause X to be set to 'sky'.

These are just two examples from a set of more than 50 built-in functions available. In addition, new functions can be developed by the user if the need arises for a function that has not already been supplied.

The INTERPRET Instruction

As the name implies, the INTERPRET instruction of the REXX language is used to evaluate and execute an instruction. The following EXEC illustrates how this instruction can be used:

```
/*  REXX calculator  */
arg x
interpret say x
exit
```

This four-line EXEC, when invoked with an arithmetic expression, can be used as a calculator. The first line of the EXEC is a comment, the second line reads an argument into the variable x, the third line will execute the instruction contained in the variable x and "say" the result and the fourth line will end the EXEC. For example, if the name of this EXEC were CA, then entering 'CA 5.5 + 2.7 + 9.2' will cause the EXEC to return the answer of 17.4. Unfortunately, this calculator does not do hex arithmetic. However, it will convert from hexadecimal to decimal and vice versa when invoked with the proper built-in function.

The Data Stack

The data stack is an expandable data structure used by REXX programs to store information. It is easiest to think of the data stack as a long array of elements; these elements can vary in width and data types. There is no limit to the number of elements or the size of the elements placed in the data stack.

Elements can only be retrieved from the "top" of the stack (only one element is retrieved at a time) and elements can be placed (one at a time) on either side of the stack.

The PUSH instruction places an element on top of the stack thereby creating a LIFO stack. The QUEUE instruction places an element on the bottom of the stack, thereby creating a FIFO stack.

One of the most important functions of the stack is that it can be used to pass data between several programs or subroutines. A program gets access to the data stack whenever REXX relinquishes control to it. This means that the stack provides a way of passing large amounts of information between REXX programs and subroutines.

Another function of the data stack is for processing commands at program termination. Whenever a REXX EXEC terminates execution, if there are any elements left in the stack, they will be treated as commands to be issued. Therefore, it is important to empty the stack unless you intend to process the remaining commands at program termination. This is a frequent cause of errors in REXX programming. If you have an EXEC that runs fine except for some error messages at the end, chances are that something was un-

intentionally left in the stack.

The REXX language supports the use of multiple stacks. New stacks are created with the NEWSTACK instruction, stacks can also be deleted with the DELSTACK instruction.

The EXECIO Instruction

The EXECIO instruction of the REXX language can be used to perform I/O operations to a dataset. With a single read operation, an entire dataset can be placed in the stack or in an array. A single write operation will take the entire content of the stack or array and place it in an output dataset. Read-and-write operations can also be issued that will process only a record at a time.

Tracing Commands

The TRACE instruction of the REXX language provides the user with a wide variety of tracing options. The user can choose from tracing intermediate results, only final results of instruction evaluation, only commands, only errors and so on.

In addition, the interactive debug facility permits the user to pause between instructions as (s)he traces. During a pause the user can insert instructions or display fields.

Addressing Different Host Command Environments

The REXX language allows you to issue TSO, ISPF or MVS commands from within an EXEC. When the REXX language processor encounters a command, it passes it to the host command environment for processing. The ADDRESS instruction of the REXX language can be used to change the host command environment, so that a command can be routed to the proper environment for processing.

Running REXX EXECs In Batch

REXX EXECs that are too time consuming to run interactively can be run in batch under TSO or in a non-TSO address space.

REXX comes with a powerful programming interface that allows EXECs to call or be called by programs written in Assembler or a high-level language and residing in a load library.

Summary

Its simplicity and similarity with the English language make REXX an easy programming language to learn.

Its rich selection of built-in functions, its ability to address different environments and its powerful mathematical capabilities make REXX a powerful tool for data processing professionals.

In addition, REXX is fun.

ABOUT THE AUTHOR

Zeida Heavener is a member of the Tech Support team at American Bankers Insurance Group in Miami, FL. During her 15 years in data processing, she has been active in various professional organizations and user groups such as GUIDE International where she was project manager from 1984 to 1987.

GIANT IS GROWING

We're Your Free Global Information Access NeTwork

The single source to find all your data processing needs:

- Software Products
- New Hardware Products
- Data Processing Services
- Consultants
- Education
- All Hardware Environments
- All Operating Systems
- Used Computers and Computer Equipment

We cover the entire range of hardware and operating system environments:

Mainframes, Minicomputers, PCs. IBM (and PCM), DEC, Apple, Data General, and others. MVS, VM, VSE, UNIX, PC-DOS, and more. CICS, IMS, DB2, TSO, CMS, ISPF, etc. System Support Software, Application Software. Education, Consultants, and Conversion Specialists. Uninterruptible Power Systems and Surge Protectors. Modems, Cables and other Connectivity Solutions. Everything/anything related to data processing!

Personal Service: (414) 546-8496
Call us direct and we will search our data base to find solutions that fit your business needs. Personal assistance is available Monday through Thursday, 9:00 AM to 1:00 PM Central Time. Personal Service is available FREE of charge.

FAX Support: (414) 546-8499
We also support facsimile, just send us a description of your needs and we will search our data base and return to you a list of items that fit your requirements. FAX Support is available FREE of charge, 7 days a week, 24 hours a day.

Dial-up Access
2400 baud — (414) 546-8480
1200 baud — (414) 546-8492

The network is available FREE of charge. Monday through Saturday. Use any of the popular PC communications packages that emulate a VT100 terminal, such as PROCOMM Plus™, CROSSTALK™, KERMIT and many others.

Terminal Type	**VT100**
Line Speed	**2400 or 1200**
Stop bits	**2**
Data bits	**8**
Parity	**None**

Used Computers and Computer Equipment

GIANT now includes used computers and equipment. If you're in the position to sell excess computer equipment, call us today for details. If you're looking to purchase used equipment, call our Personal Service number, access the GIANT Data Base from your PC, or send us a FAX.

Software Manufacturers!! Hardware Manufacturers!! Service Suppliers!!

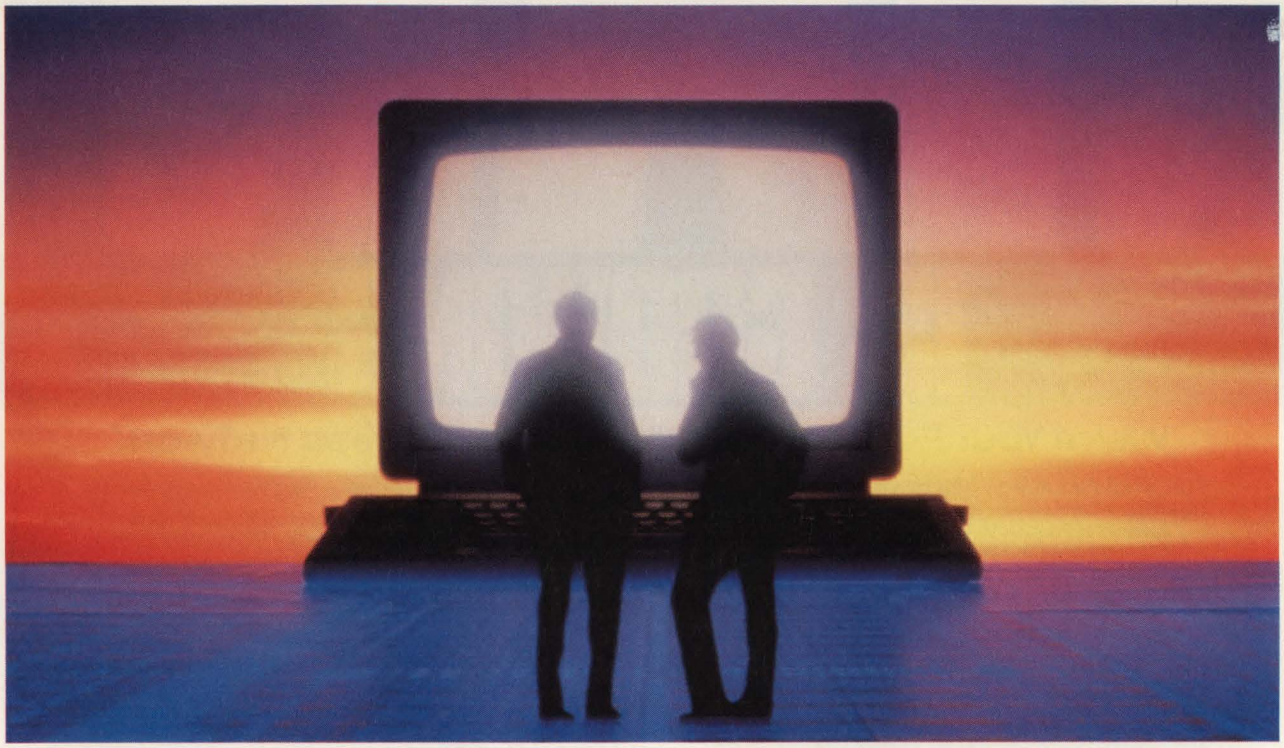
Advertise your products and services on GIANT today. Put your message into every data center across the country . . . everyday. Join with GIANT and become part of the GIANT team: we're making the marketing media of the future a reality today! Give us a call and we will send our GIANT information kit to you immediately.

PSI
Productivity Systems, Inc.
P.O. Box 20958
Milwaukee, WI 53220
414/321-8688

CIRCLE #135 on Reader Service Card ▲

DOCS™

THE BEST-KEPT SECRET IN VSE CONSOLE AUTOMATION.



Today's data centers are serious about managing their VSE operations. Most realize the system console is the logical first step. But what many don't realize is that the solution for VSE console automation is already in use at hundreds of data centers around the world.

SINGLE IMAGE CONSOLE

DOCS from SMARTECH SYSTEMS features a Single Image Console Facility under VM (SICF/VM) that lets you control multiple VSE systems from one DOCS console. Multiple VSE consoles are consolidated on one CRT, creating a more efficient operation. *And you always have access to local or remote consoles* because DOCS is not dependent upon an online system.

The VM OPERATOR console can be consolidated and managed from the DOCS/VSE console, which means you view VM operations in full-screen 3270 mode using one less CRT. You can redisplay VM operator messages online and list VM hardcopy data because DOCS logs *all* messages to the hardcopy file with date and time stamp. You even have access to CMS minidisks and can execute REXX programs directly from the VSE console.

MESSAGE SUPPRESSION AND ROUTING

DOCS message suppression and routing capabilities let you customize your console to display only the messages you need to see. You can suppress messages at selected terminals, or route messages based on partition, device or user profile.

...AND MORE

DOCS offers more than *fifty additional features*, making it the most comprehensive VSE console automation solution available. Features such as auto-reply, programmable function keys, security features, unattended operations capabilities and more.

NO RISK, MONEY-BACK GUARANTEE

Buying DOCS is risk free because it comes with an unconditional six-month money-back guarantee. *And all it takes is a simple 30-minute installation.*

So if you are looking for the secret to VSE console automation, find out what hundreds of our customers already know. For more information on your console automation needs, call 1-800-53-SMART. (Outside the U.S., call 214/956-8324.)

SMARTECH SYSTEMS, INC.

Turning high technology into *SMART TECH*nology.℠

10015 W. Technology Blvd., Dallas, TX 75220
FAX: 214/357-6338 Telex: 9102503110

International Representatives:

France: Futurs Systemes et Technologies • Poitiers, France
Tel: 33-49-01-7374, FAX 33-49-01-7351

Australasia: Mycroft Systems, Ltd. • Auckland, New Zealand
Tel: 64-9-817-7673, FAX: 64-9-817-3640

United Kingdom: Rendeck U.K. Ltd. • Essex, England
Tel: 44-0702-333555, FAX: 44-0702-333055

For additional international representative information, contact SMARTECH Systems, Inc.

IBM Business Partner

CIRCLE #65 on Reader Service Card ▲

An EDP Control Audit With Teeth

It Began With An Unlikely Partnership

By Avery C. Cloud

A partnership between the Internal Audit and Information Services (IS) might seem a little like the lion lying with the lamb. But this unusual partnership was struck between these two departments at North Carolina Baptist Hospital/Bowman Gray School of Medicine in Winston-Salem, NC to create an effective model for performing EDP control audits.

It was determined that Internal Audit and IS really wanted many of the same things. For example, both wanted to make sure that the information resources of the company were being properly managed and protected. IS managers have a Duty of Trust to ensure that adequate measures are being taken to protect the electronic data of the company. In like manner, the EDP auditor of Internal Audit has the responsibility of making sure that this Duty of Trust is being honored.

It is in the best interest of IS managers to have the EDP auditor verify that the information resources of the company are under sufficient control. This verification can protect IS managers from legal liability in the event of data loss or corruption. Successful suits have been filed against IS executives for not adequately protecting electronic data. For this reason it only makes sense that the two departments should work together to develop processes and systems that effectively uncover control deficiencies.

The remainder of this article describes an innovative model designed to allow auditors and data processing managers to be more effective in analyzing the sufficiency of information resource EDP controls, thereby measuring their level of protection against data processing catastrophes. The audit model was dubbed the Clayton Flexaudit, suggestive of some of its notable flexabilities that will be described later.

Describing The Three Main Problems

What Is Control?

Everyone knows that IS is supposed to have adequate control of the company's information resources; however, confusion looms over the definition of control. Diverse data processing backgrounds often lead to disparate notions of exactly what the term *control* really means.

A consistent definition for control is needed throughout the entire company if an effective control system is to be created and managed. When all players define the game the same way, success is more likely. This consistency of perception will create consistent behavior and help everyone to make congruent information control decisions. So the first step is to define exactly what information resource control means.

What Is The Standard?

The primary objective of any audit is to test the performance of some activity against an appropriate standard. If it is an accounting audit that is being performed, the standard against which a company's accounting practices is tested is called Generally Accepted Accounting Principles (GAAP). A programming and systems audit uses the department's written procedures and published standards as the yardstick to measure performance. Unfortunately, the EDP control audit usually does not have any official standard by which information resource controls may be measured. So it followed that the second step was to determine what would be an effective standard by which information resource controls can be measured.

How Should The Measurements Be Made?

Typically, control audits are strictly qualitative in nature. The auditor develops a checklist of what he believes should be in place. He interviews IS employees and does walk-throughs to check compliance with his list of rules. He notes failures and what needs to be improved. After all is said and done, the auditor produces a written narrative describing what he found.

There are several inadequacies in this approach. First, the auditor has not shown why more control is necessary. There has been no quantitative measure of need. The danger here is that without knowing exactly how much control is needed, resources could be wasted in the creation of excess controls. The converse danger also threatens in that, without knowing the goal, it could be easy to fall short of enough controls.

The auditor's checklist is usually made up of some generally accepted principles of information resource controls. The weakness is in the generality of these qualitative measures. Just because some

F I G U R E 1

Applications And Their Need For Control

Application	Security	Privacy	Integrity	Avail.	Recover.
Payroll	H	H	H	M	H
General Ledger	M	M	H	M	H
Chargeback	L	L	M	M	M

H = High M = Moderate L = Low

practice is good at a few data centers does not necessarily qualify it for blanket application in all data centers.

Finally, such qualitative measures do not lend themselves very well to reporting improvement or degradation trends over time. The only way that this can be done is by comparing the items in last year's report to the ones in this year's and looking for changes.

Addressing The Three Problems

Defining Control

Control can be described as the effective management of information resources based on the level of need that each information resource has for control. Therefore, *need* should be the standard for control instead of some nebulous idea or textbook checklist. Effective control means that the specific information resource is neither under-controlled, which introduces unacceptable risk or over-controlled, which introduces unnecessary cost.

This basic definition of control was teased apart to discover its individual elements. The concept of control was perceived as being made up of five individual elements. These elements are the following:

- Security: protection of data from destruction or unapproved tampering
- Privacy: prevention of unlawful or unapproved disclosure
- Integrity: managing the accuracy of data
- Availability: accessibility of data when it is needed
- Recoverability: the ability to recreate, restore or recover damaged or lost data.

All issues relating to information resource control can be tucked neatly under at least one of these five element headings. Control is made up of these five elements and each of these five elements is made up of its own special blend of software, hardware and procedures. For example, security software is usually purchased to control access to the computing resource and specific information. The use of this software is usually coupled with authorization procedures and an appropriate security organization. These are some of the components of the security element of control. The other elements also have their own component structure.

Need As The Standard

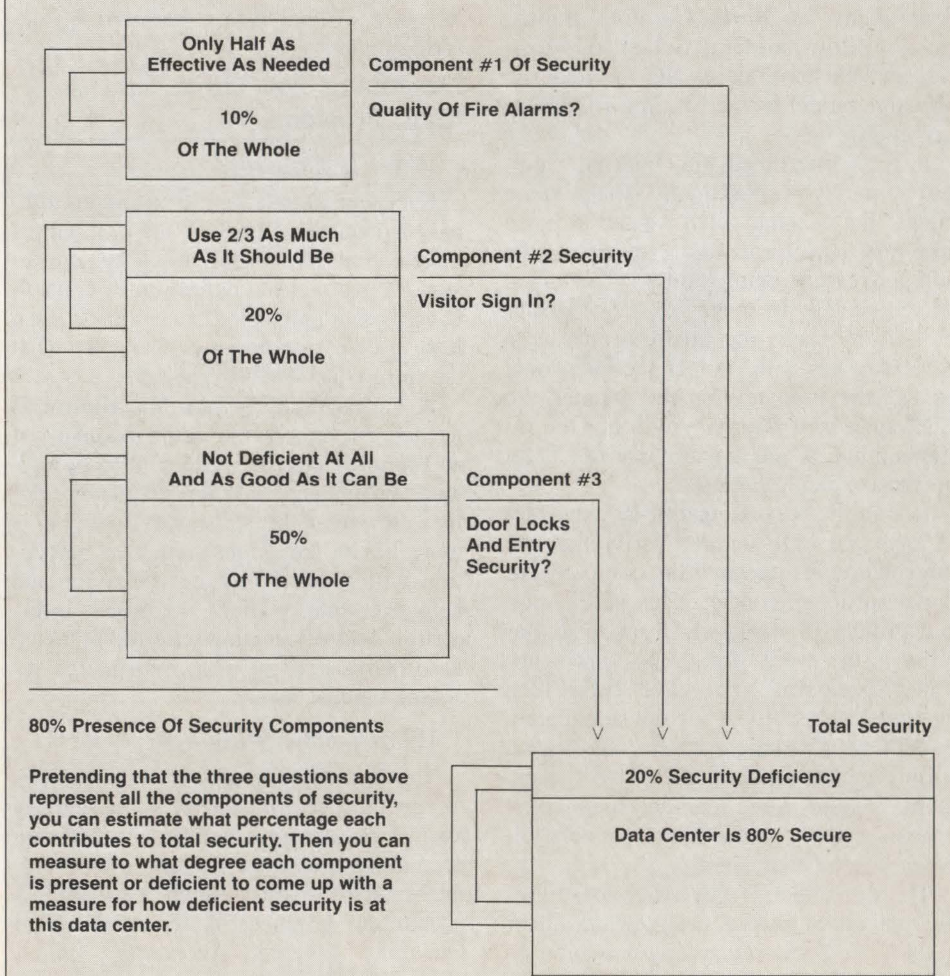
Information is being adequately controlled if each of these five elements is present at a sufficient level to fit each information resource's specific need. To provide these five elements, the data center and applications functional areas each plays a complementary role. The first thing that must take place is that the data center staff must create an environment replete with the necessary software, hardware and procedural tools to effect these five elements. Secondly, the applications functional area must then employ the tools that are provided by the data center, as well as some of their own internal tools, to adequately control the information resources that they are responsible for.

The auditor should measure the performance of each of these functional areas separately. The environment that the data center has created can best be measured against an industry standard, adjusted by the preferences of the management of the company. This functional area is referred to as the objective control function. Different industries have different degrees of

F I G U R E 2

Graphic Of Assessment Logic

The Theory Behind Assessing Element Need And Deficiency



dependency on their computing resources and, therefore, have different degrees of need for objective controls. Financial institutions have a high need for control because they are so dependent on electronic data for day-to-day operations. On the other hand, the retail industry is generally less dependent and, so, has a lower need for control. After a number of quantitative and qualitative permutations, a standard was selected for medical centers in general and adjusted for the preferences of IS and Internal Audit. Both departments decided that the optimum level of objective controls would be about 75 percent.

The next step was to determine the method to test how well the applications staff was employing the tools that the data center had provided. The most effective approach was to test each application at the element level; that is, to see how well each of the five elements is being employed. Security, privacy, availability, integrity and recoverability must all be present at levels sufficient for each application's specific needs.

A little experimentation proves that each application has a different level of need for each of the five elements of control. Figure 1 lists some typical applications and their relative need for each of the five elements.

Once again using *need* as the standard, you can test how sufficient each element is relative to the applications measured need for each element. This technique ensures that control efforts are focused in the right areas.

For example, if the payroll application has a sufficient degree of security, privacy and integrity but is lacking in availability and recoverability, resources should be focused at improving the latter and not the former. This strategy protects the company against spending money on excessive controls, allowing it to get more control for its money.

Coming Up With A Measure

To apply quantitative measures to the assessment, a simple mathematical concept is employed to the problem. The concept simply put means that any item is complete only when its parts are complete. There are two items in the problem that have to be measured for completeness. One is *need* and the other is *sufficiency*. All of the issues that constitute need for security, privacy and each of the other elements are listed. Then, all of the issues that make each of the five elements sufficient are listed. After that, what percentage each item contributes to 100 percent *need* or 100 percent *sufficiency* of each element is determined. A graphic of this concept is presented in Figure 2.

A scoring system is created that weights each item so that when the maximum score is assigned and multiplied by its weight, the full potential percentage contribution will be produced. When all the weighted *need* scores of each element are added together, you come up with a number that represents the amount of need each application has for that element. The sum of the sufficiency weighted scores tells you how much each element is present in the application. Survey forms are created to perform both the objective and subjective portions of the audit. Audit assist spreadsheet templates are created to perform the calculations and create graphs.

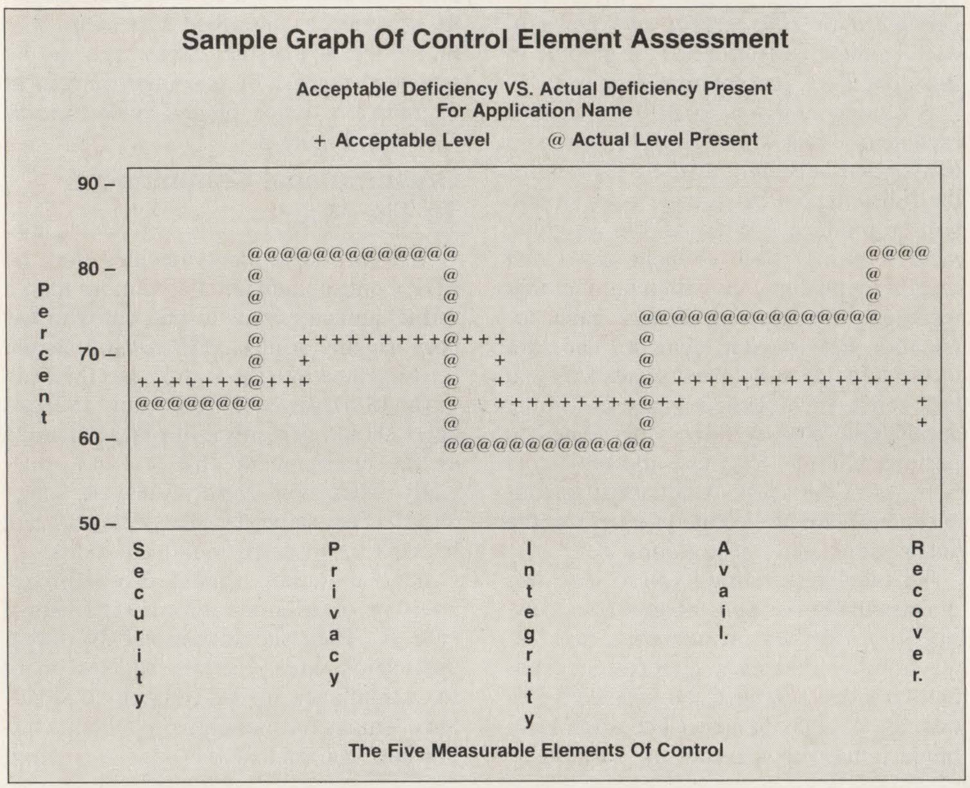
This technique allows you to set down quantitatively the percentage of security, integrity, privacy, availability and recoverability that each application needs and compare it against the measured sufficiency of each element. The same technique is used to measure the sufficiency of objective controls. The difference is that need is assessed by finding an industry standard, as described previously. The nice thing is that the assessment can be presented graphically and improvement or degradation trends plotted over time as shown in Figure 3.

Organizational Commitment To The Model

The two departments decided that the EDP Control audit should be more than a formal annual review to find out who has been naughty or nice. EDP controls should be integrated into the culture and thinking of the IS department. To do this, IS managers should perform informal self-audits as the environment changes and especially when new applications are introduced. This same flexible audit tool can be used for these periodic self-audits.

Internal auditors should also be proactively involved in the information control process. They should periodically target applications and functions and test them for compliance to the company's standards. This is far more effective than waiting until the end of the year to attempt

F I G U R E 3



the massive job of evaluating all applications and functions. The opportunity for thoroughness is greatly diminished when the auditor attempts to audit too much at once.

If you have questions, write to the following address: Clayton Flexaudit, PO Box 58, Lewisville, NC 27023.

ABOUT THE AUTHOR

Avery C. Cloud, primary designer of the EDP control audit model, is Director of Technical Services at North Carolina Baptist Hospital/Bowman Gray School of Medicine of Wake Forest University in Winston-Salem, NC. Avery has been involved in most arenas of data processing. He has experience in disaster recovery planning, change management and operations as well as extensive programming and technical experience.

A Tactful Way To Program

Blind programmer uses adaptive equipment to perform his daily tasks

By Joanne Kimbler Cooper

You are tired of staring at your terminal screen. Maybe your eyes burn or simply feel strained. So you look away for a moment or close your eyes to give them a rest. What if, when you tried to open them and once again look at your screen, you could not see? Would you still be able to code, test, read and type — or do any part of your job?

Phil Obregon is a 32-year-old senior programmer at the University of California Irvine Medical Center in Orange, CA. He does his job just as any other programmer would — except for one difference. Obregon is blind.

Background

"I don't ever remember seeing," Obregon recalls. "I lost my sight completely at two years old due to cancer. I had malignant tumors in each eye."

Obregon started his education at a special school where he learned to read Braille. Braille is a system of lettering devised for use by the blind in which raised dots are read by touch. From third grade through college, he attended regular school. In college, he studied general educaton with an emphasis on data processing.

His interest in computers surfaced after college when Obregon took a nine-month course sponsored by the California Department of Rehabilitation. "I learned COBOL PL/1, FORTRAN and Assembler," he states.

After completing his schooling, Obregon began looking for work. "I looked for almost nine months. I had such a hard time. People aren't aware that blind people can do a job. My biggest problem was having to sell myself to employers," he explains.

There was one incident with a potential employer that sticks out in his mind. "Everyone who interviewed me at this company recommended that I be hired. But someone in upper management had worked with a blind person who had not been productive. Through politics, I was not hired," Obregon remembers.

The brick wall that Obregon came up against during his interviews is what he calls the *fear factor.* "I would always be asked, 'Do you need Braille equipment?' I was amazed at how many companies said thanks and referred me to others who would be able to buy the equipment they thought I needed. I don't need special equipment purchased for me. I can use the standard terminal that everyone else uses. And I can program like anyone else," he points out.

The Optacon

What Obregon does use is a different means for his programming. "I already have what I need to do the job," he continues. "It's a different medium but the end result is the same. It is a tactical medium — a small, electronic device called an Optacon."

The Optacon from Telesensory Systems (Mountain View, CA) is actually an array of pins and a small camera. "I run over any piece of printed material and the image is transferred over the array of pins. For example, if I am holding the Optacon over the letter 'A' it will give me a raised image of the letter 'A'," Obregon explains.

"I purchased the Optacon II a couple of months ago. Compared to the original Optacon, it is a little smaller only weighing two pounds and it has one battery. It can be directly connected to a computer and does not have to be hand-held. I use it at work and have the older model at home. My top speed is probably between 90 and 100 words per minute. But many times I am not actually reading, I'm just scanning printouts for specific information," Obregon points out.

The Optacon II converts print or computer output into an enlarged, vibrating tactile form. To read print, Obregon would move the camera across a line of print with one hand. The index finger of the other hand is placed on the Optacon's tactile array, which is approximately one inch long and a half inch wide. As the camera is moved across the letter, the Optacon sends a letter electronically and the image is simultaneously reproduced on the tactile array by vibrating rods. Obregon is able to perceive the vibrating image with his index finger. Any graphic image viewed by the camera or received electronically is perceived by the user. The Optacon is not limited to reading letters only.

"Besides the device I use, there are other devices," Obregon continues. "One of the big things today is a speech synthesizer. By hooking up the synthesizer to computers, you can get voice output instead of tactical output. It puts the words together and tells you what's on the screen."

"I have another Telesensory Systems' product called Vert Plus," he says. Vert is synthetic speech output that works with IBM PCs and most compatibles. It can be instructed to read whole words and numbers or to announce each letter and number individually while sounding like a human voice.

"A lot of people use strictly speech. I have not been using it very long so it is slow for me. I need to take time to learn it. Speech is supposed to be a lot faster because you are just reading straight text. But with programming you are not really reading every character. Even in manuals, I just look up certain things and don't read the whole book" Obregon explains.

"It is easier for me to just pick up my Optacon. I know what I want to see and go right to it. With speech, you have to learn how to navigate. However, the Optacon is tiring because I have to hold the camera up to the screen all day long. Someday, I will have to switch to speech. It would be less tiring. But I will never give up the Optacon because it gives me the freedom to read anything — mail, newspapers, magazines," he says.

The Job

It took Obregon nine months before he got his first job at the University of California Irvine Medical Center as a programmer in 1979. Richard Sechrest, Director of the Information Systems Department, recalls hiring him. "It was a complete unknown to us. And just like any difference, there was some reluctance in hiring him. I hired him right from school and I was assured by the institution that he was good. It was an entry-level position so I took a chance. Phil is an extremely accomplished technical individual," Sechrest states.

Obregon explains, "I have the same job description and responsibilities as any other applications programmer." He works in a DOS/VSE environment on an IBM 4381. He uses DYL-280-II, a 4GL from Sterling Software's Dylakor Division (Chatsworth, CA), along with the Optacon for 95 percent of his programming.

Jim Thompson, principal programmer at the University of California Irvine Medical Center, comments, "I've worked with Phil eight years or so. He is very good at what he does. Basically, his duties are the same as mine. We program, do user requests, reports and new systems. I use my eyes and he uses his Optacon. There's no difference."

According to Sechrest, Obregon spends a lot of his time maintaining and changing payroll applications. "There are so many different unions, regulations and policies as well as eight different employee groups, resulting in the payroll programs constantly changing."

"Phil is one of the sharpest technicians here," Sechrest continues. "He has a fantastic memory and can remember things he worked on years ago. We had a programmer here who left a mess. Phil rewrote all of the programs in one-tenth of the time it originally took to program."

"In my experience, I would not be reluctant to hire another handicapped person. And we don't give Phil special treatment. He is a rare and unique individual," Sechrest adds.

Obregon explains, "With DYL-280-II, a lot of the batch programming is automatically generated. For example, for report layouts you tell it which fields you want to print, the order and specific column headings and it will automatically generate the print line and calculate spacing and centering. With COBOL, all of this would have to be coded and it would take a lot longer. DYL-280-II cuts programming time way down and that is the biggest reason I use it."

"With COBOL, I would have to write my own routine to read a record, see which is higher or if the record matches. DYL-280-II has automated this. You give the keys you're looking for and if one file doesn't have a matched record, you give the specifications and let DYL-280-II generate the code for you. I just worry about manipulation without worrying about a read or write," Obregon adds.

Not Just A Programmer . . .

Obregon starts his day by most people's standards at about 4 a.m. "I'm a morning person. I workout first thing to get my blood circulating," he explains.

He lives in a condo and takes the bus to work. "I have a cane that I carry in front of me. It encounters objects before I do and when it doesn't work, it's usually my fault," Obregon continues.

When Obregon is not working and not playing games on his PC at home, those who know him will tell you he has most likely gone skiing for the weekend. "I prefer downhill skiing to cross-country but I've done both," he says matter-of-factly. "I also water ski but haven't done it in years."

He started skiing about five years ago with his brother. Presently, he skies on his own and through the National Handicapped Sports and Recreation Association. "I have someone skiing behind me giving me directions. There are other techniques you can use. Blind racing skiiers will have guides in front of them. I try to plan trips at least twice a year — locally at the Mammoth Ski Resort as well as up in the Sierras," he adds.

"I love it," Obregon says. "It is such a free feeling. Being blind, I don't get that feeling with running, although I ran in high school and college. You have to hold on to someone the whole time."

In Conclusion

"The technology is here," Obregon states. "I want to let the world know that the technology is available for me to do the exact same job that a sighted person can do. The only difference is how I get the information from the screen to my brain."

ABOUT THE AUTHOR

Joanne Kimbler Cooper is a free-lance writer living in Powell, OH. She has contributed to various technical and business publications.

Converting The REXX Way

By Christine Gogots

Most VSE-to-MVS conversions involve outside services or software. BancPLUS Mortgage Corp. in San Antonio, TX chose a somewhat unique method for its conversion — REXX.

The Decision To Convert

''We are a large CICS shop,'' explains Charles Lee, Technical Support Manager at BancPLUS Mortgage Corp. ''We normally process 200,000 transactions a day and we were up against the limits of VSE. The decision to move to MVS was made in order to overcome the VSE storage and I/O constraints.''

John Deptula, Senior Vice President of MIS, relates, ''When VSE started to experience some degradation, we knew we would have to upgrade our CPU or convert. It was cheaper to convert.

''One of the difficulties to overcome was trying to relate the benefits of a conversion to MVS to upper management. The easiest way to explain the need was that users were not getting their reports on time every day and there was not always a sub-second response time for CICS as there had been in the past.''

''Plans for the conversion began in November 1987 on an informal basis,'' Lee remembers. ''We put a plan in place around February 1988 and made the final decision to go with the conversion in May.''

''When we first started on this path, there were several options,'' Deptula says. ''We could hire an outside firm to do the conversion, buy a turn-key conversion tool that an outside company would provide, buy the conversion tool from IBM or do everything ourselves. IBM did provide an SE to help with planning and coordination.''

Lee had used REXX as a prototype language prior to the conversion decision. ''Writing large programs in Assembler takes a lot of time. With REXX, we can prototype programs or a series of programs within a short period of time. We use CMS for program development and submitting JCL for execution. We looked at several conversion packages but didn't find any that would properly convert our CMS job flow and include file commands,'' he recalls.

Deptula comments, ''We are all VM bigots here. We wanted to stay with VM because there are a lot of benefits to us internally. All of our applications personnel were familiar with VM and REXX. VM is a great product and you don't have the overhead of TSO.

''After looking at the presentations, we found that our best course of action was to use REXX. After a small test of that technique, we found it to be pretty successful. Besides the experience we would gain from doing the conversion ourselves, we wouldn't have to pay high fees to an outside firm. We would come out far better and control the conversion a lot better using REXX.''

Why REXX?

''I had quite a bit of experience with REXX,'' Lee explains. ''I had written a number of procedures to help monitor the system and help the applications programmers. For example, to compile a program, you just say 'compile' and a REXX EXEC adds the necessary job control statements to the COBOL source program and submits the job to VSE or MVS for execution.''

On a whim, Lee put together a prototype EXEC to convert a job stream. ''I was surprised at the speed. My original thought was once the prototype was done in REXX and I had the logic down, I would convert it to Assembler. After watching it run, I didn't need to do that and continued development in REXX. The basic conversion routines were completed in a couple of weeks.

''Because REXX is an interactive, high-level language, it was easier to change than COBOL or Assembler would have been. It has a lot of facilities embedded in the language which makes it pretty easy to do some complex routines. This was the most surprising element of using it. I could convert a 1000-statement job stream in 15 to 20 seconds.

''For the conversion process, I had to

use three distinct facilities. The first was a procedure to make technical adjustments to the CICS programs. During the migration, we went from CICS 1.6 to 1.7."

There was one procedure to make modifications to the COBOL source programs, which took about 600 lines of REXX code. There were six procedures to scan and modify the JCL. It took 1500 lines of REXX code for the JCL modifications.

Lee states, "It would have taken a lot more lines of code and many more man hours to develop these procedures with Assembler. You can do an enormous amount of work with one REXX statement, while it would take hundreds of lines of Assembler code. It packs a pretty powerful punch.

"The price you pay for this is speed. REXX is slower than any compiled language would be. Depending on how a program is written and what its function is, REXX can be so slow that it is unacceptable. However, we did not run into this problem.

"The key element of the conversion was to go with REXX. If we would have done it any other way, it wouldn't have come out the way it did. And with REXX, it took less than 200 hours of manpower to develop the basic procedures."

Preparing For The Conversion

In January 1988, BancPLUS Mortgage Corp. installed a 3090-120 E running VM/XA SF and VSE Release 1.3. Previously, a 3083 had been running VM/SP since 1981. "By installing the 3090, we were positioning ourselves for an MVS conversion if needed," Lee states. By early July, the initial MVS system was installed. The department has 20 applications programmers and four technical support people. Ninety-nine and nine-tenths percent of the programs are written in COBOL.

Lee relates, "We spent from that July through December familiarizing our technical support group with the system. We didn't have 100 percent understanding of some facilities we were getting in MVS. None of us had any experience with MVS so it was a brand new world for us. After it was installed and we saw how it really worked, there were some pretty significant adjustments to be made. Having written the conversion procedures in REXX allowed us to make these adjustments quickly.

"One life saver was getting the MVS Express system from IBM. Express is a marketing tool that IBM uses. It is a pre-generated version of MVS tailored to your installation's hardware configuration. We installed it at 9 a.m. and had it running by 11 p.m. We had to do a lot of refinement after it was installed, but it was a good base system. It cut two months from the technical support staff's effort."

In January 1989, the applications programmers became involved in the conversion process as well. "The technical support staff ran the EXECs to convert the COBOL source programs," Lee explains. "They also converted the JCL and modified what the EXECs didn't process properly. Then the applications staff ran the EXECs to convert the JCL and modify any statements the EXECs did not process properly. Last, the programs were tested to make sure they functioned the same under MVS as VSE."

This time was also spent ordering and installing third-party vendor products. These include CA-ONE, a tape library management system from Computer Associates International (Garden City, NY), The Monitor, a CICS peformance monitor from Landmark Systems (Vienna, VA) and FAVER, a backup/restore facility from Goal Systems International Inc. (Columbus, OH).

"We first had FAVER under VSE. For compatibility we felt this product best fit our needs for VSAM file migration. We also have VMFM, a VM product from a local company, Cestrian Software. This product helped us quite a bit with the conversion process, in conjunction with REXX, in managing CMS files and display panels. It is a product that complements some of the VM/CMS facilities and offers a variety of functions including a CMS file processor and panel manager. I used it quite a bit for panel definitions on EXECs," Lee says.

About five percent of the JCL changes were done manually. Lee points out, "Computer Associates' System Manager, which we were running under VSE, includes facilities to allow 'Go To' statements in JCL. MVS doesn't have those facilities so the biggest part of the manual changes were to alter the structure of the JCL logic and to accommodate the differences in tape and DASD file management."

"IBM suggested Xamo, a conversion assistance migration offering, which was a six-month extension of the normal two-month test allowance," Deptula says. "We didn't want the double software costs of both VSE and MVS. With an eight-month window, we wanted total control of the conversion and we wanted to get it done. The key to the whole conversion was planning. That is what really helped make it successful."

IBM's Involvement

"We wanted someone from the outside questioning what we were doing," Deptula says. "Jose Kypuros, an IBM SE, was our check and balance."

Kypuros explains, "I was there to overview. Charles and I came up with a schedule and we made sure we stuck to it. My task was to supply information when problems came up."

Kypuros says he used a lot of his IBM experience for the different areas of the conversion. "I went to school for VSAM, POWER to JES, VTAM and CICS. If I saw something being done wrong during the conversion, I would explain how MVS works and why it should not be done that way. I also emphasized the differences between VSE versus MVS."

The Conversion

Deptula comments, "Each MVS conversion is the same but it is unique. With REXX, we could handle it better. We started it on a Saturday morning and had the whole conversion done by Sunday. With someone else's tool, we could not have done it in that time frame.

"We didn't want our users to be affected by the conversion. The goal was to have VSE running on Friday and MVS up and running on Monday with the only difference being improved response time. We didn't change any applications during the conversion unless it was absolutely necessary in order to run under MVS."

There were three weekend dates that were choices for the conversion time. According to Deptula, "Normally, work is done on Saturdays so in order to do the conversion, production would have to shut down. April 8 was chosen, knowing that if it did not work, April 15 or 22 could be used to try again.

"We had several 'go, no-go' checkpoints during the conversion. By Sunday evening, we reached the last checkpoint and jointly decided that everything, from our viewpoint, was problem-free. We then cut over to MVS," Deptula points out.

On Monday, the only problem was slow response time. By Tuesday, it was back to sub-second response. "We did not tune anything on MVS until it was up and running and we could see what was slowing it down," Deptula adds.

"We also had one CICS problem," Lee adds. "An application had not been converted properly. We also encountered a few JCL errors when running some of our batch jobs. But ever since the first week, MVS has been solid."

Benefits Of The Conversion

"After the conversion, we had a session with the users," Deptula says. While it was difficult to explain the benefits of MVS, the benefits of getting reports on time could be demonstrated. "We met our goals of having a 'transparent' conversion and management was very pleased with the conversion."

"Batch runs have improved astronomically," Lee says. "We used to leave CICS up until 8 p.m. and were pretty pinched to get the production cycle run by 7 a.m. the next day. And we would run over at month-end. Now, we finish production between 3 and 3:30 a.m. CICS response time is approximately the same as before the conversion."

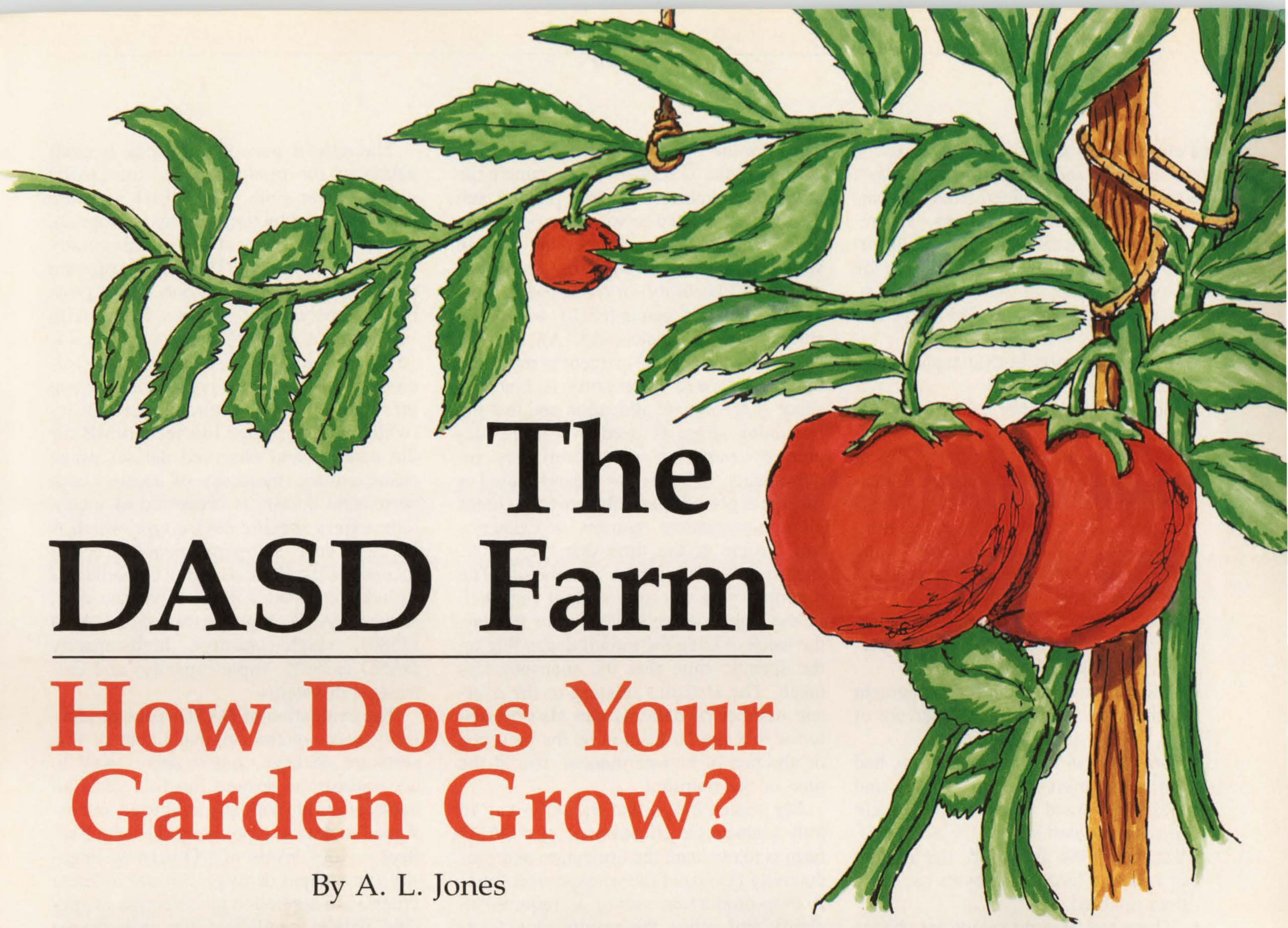
The system and applications are now being tuned for the MVS environment. He continues, "We took our VSE applications and brute-forced them into MVS. We can't expect them to run at their optimum. But not trying to redesign them during the conversion was one key element to our success."

"Using REXX for the conversion provided a tremendous benefit because we learned so much more about our operation that we had taken for granted over the years," Deptual says. "We found some jobs that should not have been running the way they were. We were able to see how certain things operate and identify what was not smart and how we could run a job a better way."

"The biggest problem with conversions is either there is a poor job of planning upfront or there is a real good job of planning but those involved don't stay with the plan," comments Kypuros. "The eagerness of BancPLUS toward staying on their schedule made them an MVS shop quickly and cleanly."

ABOUT THE AUTHOR

Christine Gogots is a free-lance writer specializing in VSE-to-MVS conversions.



The DASD Farm

How Does Your Garden Grow?

By A. L. Jones

It has become fairly standard vernacular in the DP industry today to refer to the collection of Direct Access Storage Devices in a DP shop as the DASD farm. For most installations, this label is quite appropriate. The typical DASD farm continues to grow and grow and grow — and it does not even need water! The total acreage is increasing, but how about such factors as the acres in production, the yield per acre and the value of the crop? Will current directions toward crop rotation help? This article takes a tongue-in-cheek look at some of these issues and may present some surprises for fellow 4H (Hollerith, Hollerith, Hollerith & HAL) club members.

"Today, large processor complexes are experiencing capacity growth of 30 to 40 percent per year in their DASD farms." This quote exemplifies the current perception of storage directions within the DP industry. But is this projection true? Or, perhaps more appropriately, *should* this projection be true?

While the following information is not a complete and detailed evaluation of these questions, it is a preliminary evaluation based on a somewhat unconventional method of looking at DASD utilization. Most studies of DASD usage evaluate statistics and trends relating to the *allocation* of DASD (a land-use survey). An added perspective addressed here is the *productivity* of DASD (a crop-production survey). The scope of this evaluation is not broad enough to be viewed as definitive, but I sincerely hope that a consciousness regarding DASD productivity will be raised and that additional, more scholarly studies of this subject might be prompted.

Just for fun and in recognition of common reference to DASD farms, I will frequently make use of "agricultural" language. Since most DP personnel are citified, the following basic definitions of the language are offered:

- Farm — the collection of direct access storage devices at a DP installation; this may include standard rotating DASD, cached DASD and Solid-State Disk (SSD) storage devices
- Section — a subset or subdivision of the farm; this might be a particular device-type subsystem or simply a portion of the farm that exhibits certain characteristics
- Acreage — the number of megabytes (MB) or gigabytes (GB) of storage space associated with a farm or a section
- Crop — the unit of production associated with I/O subsystems, namely, input or output operations (I/Os) processed to or from storage
- Yield — the amount of crop (or number of I/Os) produced; this may be related to time (I/Os per second) or space (I/Os per acre).

The results presented are based on the evaluation of several DP installations and should represent common DASD farms. They may or may not reflect the characteristics existing at another specific installation.

The *Farmer's Almanac* And Others

The MVS DASD usage survey conducted in 1988 by IBM and detailed in the *IBM Technical Report,* December 1988, is perhaps the most authoritative and complete evaluation of DASD usage characteristics and trends in print today. It is based on data obtained from 106 participating computer installations. The report contains a wealth of valuable information, but only a few key findings in the IBM report pertain to the topic and limited depth of this article. These include the following:

- The median total DASD capacity online was about 150GB
- The median number of datasets online was about 23,500
- Average overall DASD space allocation was 67 percent; average allocation for 3380A-, D- and J-type DASD was 69 percent while average allocation of 3380E-type DASD and 3380K-type DASD was 67 percent and 60 percent respectively
- Of all datasets, 72.4 percent are sequential and these account for 44 percent of allocated space
- SYS1 datasets account for 2.1 percent of the datasets and use 6.7 percent of the DASD space, on average
- About 40 percent of all datasets had not been referenced in 15 days and these accounted for about 20 percent of the allocated space
- Between 1984 and 1988, the amount of installed DASD gigabytes has more than doubled
- "There has been no significant change in the last eight years in the percent of DASD volume space that is allocated to datasets."

The IBM report discusses past and present characteristics of DASD utilization but does not project future trends. The 1988 *DISK/TREND Report* does, however, and the projected growth in shipment of new storage capacity alone is 23, 25 and 23 percent respectively in the years 1989, 1990 and 1991. A projection such as this tends to confirm the opening quotation of this paper. DASD capacity growth has been significant (around 20 percent per year) during the most recent years and is projected to continue in the immediate future.

Land Use VS. Crop Production

The problem with studies such as those discussed is that they evaluate only the instantaneous allocation characteristics of a DASD farm. They are primarily based on Volume Table of Contents (VTOC) snapshots taken at DP installations and simply reflect the allocation and distribution characteristics of volumes and datasets existing at the time of the snapshot. They are the DP equivalent of a land-use survey by the USDA — they identify the acreage available, the acres planted and the acreage devoted to various crops. They do not, however, evaluate the way in which the allocated space (or acreage) is used for I/O activity (or crop production).

The statistics generated by traditional studies can be misleading. An average space allocation of 67 percent is reported, but the unstated impression is that this space is *permanent* allocation and that the remaining space is used in varying degrees by *temporary* (short-term) space requirements. What is not clearly stated is that the reported space allocation includes all non-permanent datasets in existence and on-line at the time that the VTOC snapshot is taken for an installation. The reported space allocation might very well be the highest space allocation for the day, the week or even the month depending on the specific time that the snapshot was taken. The statistics relating to the average number of datasets can also be distorted in a similar fashion by the presence or absence of non-permanent data at the time of the snapshot.

My contention is that the only way to truly evaluate the effectiveness of a DASD farm is to evaluate the utilization and productivity (yield) of the acreage over time. A crop-production survey is required to clarify and refine the results of a land-use survey.

Survey Implementation

In order to establish a foundation and framework for a crop-production survey, it is necessary to understand the basic land-use characteristics at a specific farm. Packages which will perform the required analysis are available from several vendors. At StorageTek, three proprietary packages are employed which are available to our current and potential customers at no charge: SYB1VTOC, ACSDASD and CPIO.

SYB1VTOC essentially performs the function of VTOC analysis. In short, this package accesses all DASD devices which are on-line at the time of execution and summarizes the allocation and dataset count statistics for an installation by device type and dataset organization.

ACSDASD analyzes the last-reference date for datasets at an installation. The program presents an aging distribution report of dataset count and total tracks by dataset organization based on the last-reference date found relative to package execution time.

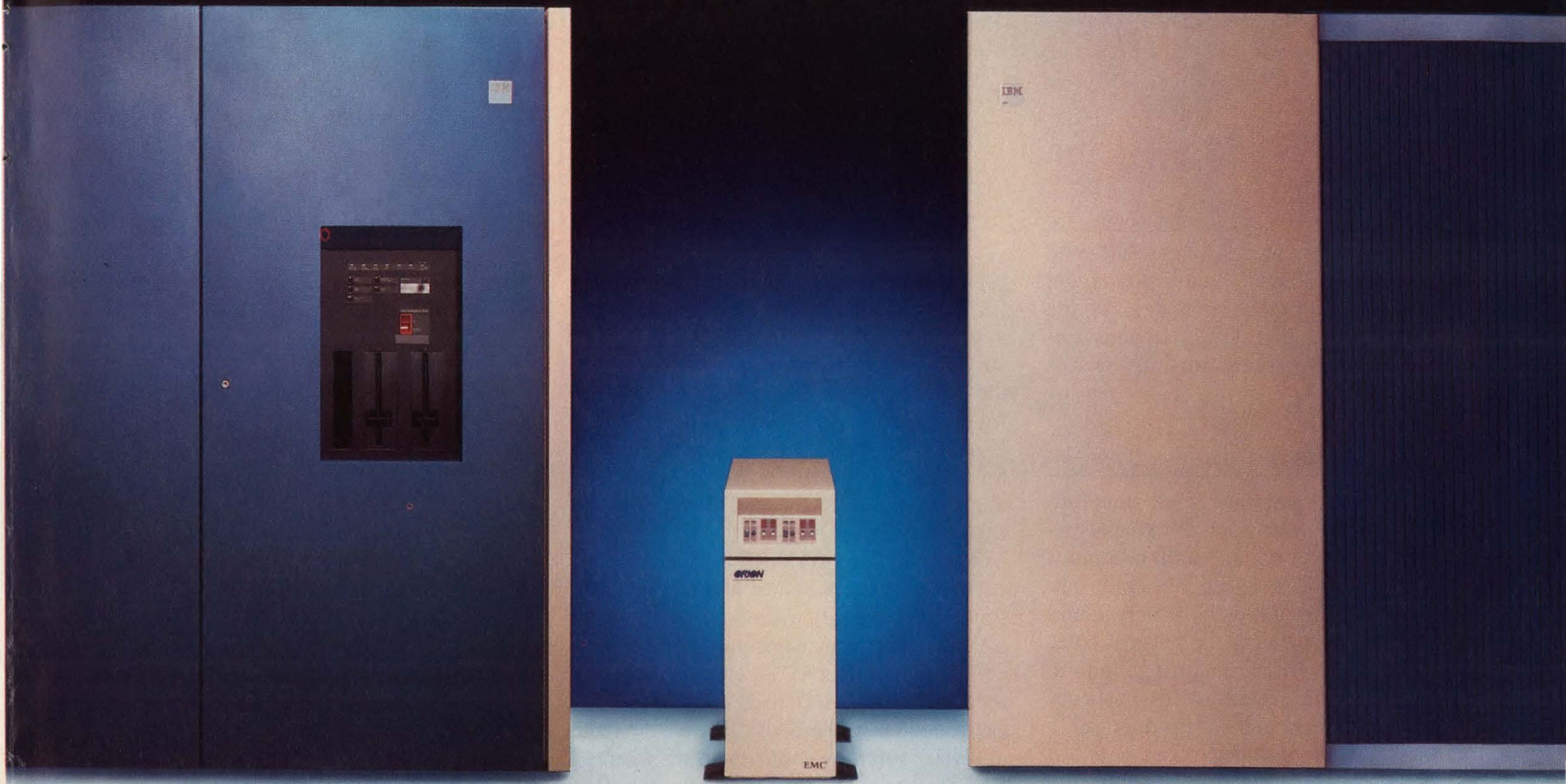
The newest part of this study is evaluation of the productivity of the DASD acreage over time. The CPIO software provides this function for any appropriate period of processing at a DP shop (usually one to two weeks). The CPIO software package is based on the concept of cost-per-I/O analysis and analyzes I/O activity (as recorded in SMF record types 4, 14, 15, 17, 34 and 64) for each dataset referenced during the analysis period. Using the characteristic of throughput density (which is the average I/O/second/MB for the dataset) and observed dataset usage characteristics (frequency of access), each permanent dataset is suggested as a candidate for a specific device-type which is the most cost- and performance-effective media for the dataset. Device candidates include expanded storage, solid-state disk, cached DASD, high-performance DASD (3380J, single-capacity), high-capacity DASD (3380K, triple-capacity) and cartridge subsystems.

The evaluation of DASD space productivity was not the original intent of this software package, but it serves well to accomplish the desired function since all datasets are individually evaluated and assigned to device-type categories which reflect various levels of I/O activity intensity (throughput density). Several different criteria are applied to the selection of specific dataset candidates for each device type but, for simplicity, the following interpretations and labels will be used for the presentation of the following results:

- Expanded Storage (ESTOR) and Solid-State Disk (SSD) candidates represent high activity storage — the classification of V-HI will be used in subsequent results
- Cached DASD (CACHE) and High-Performance DASD (HPDASD) candidates represent high activity storage — the classification of HIGH will be used in subsequent results
- High-Capacity DASD (HCDASD) candidates represent moderate activity storage — the classification of AVG. will be used in subsequent results
- Automated Cartridge System (ACS) candidates represent low activity sequential storage — the classification of LOW will be used in subsequent results.

Non-permanent datasets are tracked for high-water-mark space utilization and reported separately in three categories including SORTWORK temporary datasets, non-SORTWORK temporary data-

EMC's ORION:
The Cost-effective Solution to Your I/O Performance Problems.



EMC's ORION Solid State Disk Subsystem

Maximum Performance

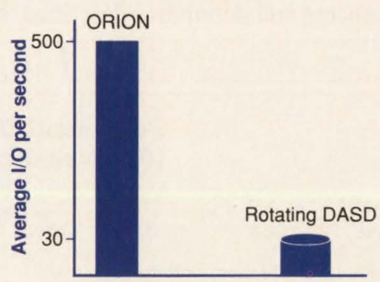
EMC's ORION is the fastest solid state disk subsystem available for your mainframe computer. ORION features a technologically advanced design enabling you to receive unprecedented performance gains.

An integrated 3880 type storage director, in addition to features inherent in solid state technology, gives ORION an access time of 0.1 millisecond — a performance milestone.

What's more, EMC protects your investment by making ORION compatible with all IBM 370 and IBM PCM computers. Therefore, performance boosts are realized well into the future, when CPU upgrades become necessary.

EMC's ORION — the I/O solution you have been waiting for to maximize your system's performance and productivity.

ORION: High Performance/Low Cost



Minimum Price

EMC's ORION Solid State Disk Subsystem is the lowest cost solid state solution to your I/O performance problems. Its state-of-the-art design gives you a substantial return on your investment for years to come.

ORION's small footprint and low power requirements eliminate costly computer room renovations. Ease of installation and low cost of ownership make ORION an expedient and economical solution to your I/O performance problems.

EMC's ORION — the most economical performance boost for your current and future mainframe systems.

EMC²
The System Enhancement Company.

For more information, call:
1-800-222-EMC2
In Mass., call (508) 435-1000

CIRCLE #143 on Reader Service Card ▲

FIGURE 1

Configuration & Utilization Summary
(A Modified Land-Use Survey)

DP Site	Total GB (acreage)	% Allocation (acres planted)	% Utilization (acres producing)
1	238.8	72.4	57.9
2	287.5	65.0	51.3
3	65.0	79.0	62.8
4	145.0	67.5	39.1
5	55.7	75.0	42.9
6	145.0	72.5	48.5
7	132.5	58.4	47.6
8	160.0	??	32.5
Average	153.7	70.0	47.8

sets and transient (deletable) datasets with permanent DSNames. Non-permanent space usage is reported separately by the analysis software and will be treated separately in the remaining sections of this article.

Survey Results

The survey results included here are derived from a detailed activity analysis for several days of data at eight computer installations. The installations which are included are geographically dispersed and represent the following industries: telecommunications, manufacturing, public utilities, banking, retail stores, grocery stores, engineering and oil production.

The results should not be considered to be conclusive since they are based on the detailed evaluation of only eight computer installations; however, it is felt that they do represent a viable preliminary evaluation. Partial evaluations for perhaps 20 additional DP installations were reviewed but the absence of one or more key data elements precluded the incorporation of these partial findings; however, it is worth noting that even the partial results agreed, fundamentally, with the detail findings.

A Modified Land-Use Survey

Figure 1 presents a configuration and utilization summary derived from this survey. It is interesting to observe how closely the average DASD acreage and the average percent of space allocation (which is equivalent to the acres planted or seeded) agrees with the averages reported in the IBM DASD survey. Survey installations exhibited an average size of 153.7GB while the IBM survey showed 150GB. Our survey showed an average allocation of 70 percent while the IBM survey showed 67 percent. If nothing else, this closeness of results should indicate that the installations included in this survey are representative of larger population averages.

The modification (or addition) this survey makes to the common DASD usage survey is the inclusion of a measure for the percent of DASD space actually used for I/O (crop) production. The "% Utilization" shown in Figure 1 is equivalent to a measure of the acres actually producing crop over the analysis period at each installation.

The percent of DASD space actually used for I/O processing at each installation in this survey varied from a low of 32.5 percent to a maximum of 62.8 percent and averaged 47.8 percent. It is fully recognized that I/O to SYS1 datasets is not reflected in the SMF data used by the analysis software but, even if average SYS1 space usage (as reported in the IBM survey) is added to this figure, the total is still just over 50 percent.

An Evaluation Of Yield Per Acre

Figure 2 presents an analysis of I/O activity to permanent datasets (which by definition, in the software used, is simply those datasets not deleted during the analysis period). The acres (GB) used by these permanent datasets are subdivided into sections of activity classification based on the concept and definitions described earlier in *Survey Implementation*. The results for each classification vary from installation to installation. The averages, however, should be representative of even a larger sample of installations and can be summarized in general terms as follows:

- Approximately one-third of the total I/O activity goes to approximately 0.3 percent of the total available DASD space; this is high activity data
- Approximately 30 percent of the total I/O activity goes to less than six percent of the DASD space; this is high activity data
- Approximately 15 percent of the I/O goes to 30 percent of the space; this is average (or moderate) activity
- Approximately two percent of the I/O uses more than six percent of the space; this is low activity sequential data. If the average figures for datasets which have not been referenced in more than 15 days (zero percent of the I/O and 20 percent of the space) are added to these values, you find that two percent of the I/O uses approximately 25 percent of the available space.

If these findings for each of these sections are simplified into a basic ratio of percent I/O (the yield) to percent GB (the acres), you can observe a basic yield per acre comparison as follows:

Section Classification	Yield Per Acre
V-HI	100
HIGH	5
AVG.	0.5
LOW.	0.1

Obviously, the crop productivity differential between various sections of the DASD farm is 1000 to 1. It is beyond the scope of this article to deal with this issue, but it should be obvious that one type of farmland (device type) cannot effectively support each of these sections; with such highly varied yields, the crops are obviously different. Rice should not be

FIGURE 2

Permanent Data Activity Analysis
(An Evaluation Of Yield Per Acre)

	ACTIVITY CLASSIFICATION							
DP	— V-Hi —		— High —		— Avg —		— Low —	
Site	%GB	%I/O	%GB	%I/O	%GB	%I/O	%GB	%I/O
1	0.1	16.5	3.1	35.4	32.0	25.6	12.8	1.7
2	0.2	25.6	3.6	25.0	28.3	10.1	6.8	2.4
3	0.9	36.9	16.4	37.3	34.8	8.0	8.1	1.5
4	0.4	33.0	5.3	30.6	28.6	15.4	1.1	0.3
5	0.2	59.5	3.4	20.5	25.0	9.7	1.9	1.3
6	0.6	52.8	3.9	15.2	25.9	10.6	12.3	3.1
7	0.2	18.1	5.2	41.3	31.7	15.8	3.5	1.1
8	0.1	21.1	5.1	45.7	20.3	11.8	4.1	3.7
Avg. =	0.3	32.9	5.8	31.4	28.3	13.5	6.3	1.9

© 1989 Christian Holtz

F I G U R E 3

Non-Permanent Data Activity Analysis
(An Example Of Crop Rotation)

DP Site	Sortwk (&&) %GB	Sortwk (&&) %I/O	Non-Sort (&&) %GB	Non-Sort (&&) %I/O	Deleteable %GB	Deleteable %I/O	Total Non-Permanent %GB	Total Non-Permanent %I/O
1	0.5	0.8	0.3	11.5	8.5	8.2	9.9	20.5
2	0.1	0.0	0.3	2.3	11.9	34.4	12.8	36.7
3	1.2	0.7	0.8	15.2	(no type 17)		2.6	15.9
4	0.8	2.7	0.3	4.7	2.4	13.1	3.7	20.5
5	0.3	0.2	2.2	5.4	1.8	3.1	12.4	8.7
6	0.7	2.0	0.5	10.3	5.8	5.7	5.9	18.0
7	0.8	2.3	0.8	14.1	5.7	6.9	7.0	23.3
8	1.1	4.1	0.5	7.9	1.4	5.3	2.9	17.4
Avg. =	0.7	1.6	0.7	8.9	5.4	11.4	7.1	20.1

(Type of Non-Permanent Data: Sortwk (&&), Non-Sort (&&), Deleteable)

****NOTE: Percent GB for individual types is the average plus two standard deviations; percent GB for the total is the maximum observed concurrent usage.***

planted in a desert and wheat should not be planted in a rice paddy.

An Example Of Crop Rotation

Figure 3 presents an analysis of non-permanent data activity. Non-permanent data includes true temporary datasets (with DSName = &&xxxx) and datasets which have permanent DSNames but are deleted during the analysis interval. For analysis purposes, true temporary datasets (&&) are subdivided into SORTWK datasets (DDname = SORTWKnn) and non-SORTWK datasets.

Non-permanent datasets utilize DASD space on a dynamic basis, acquiring and releasing space as needs dictate. They are a current and frequently-used type of data which exemplifies the merit and potential of varying the use of storage space as needs vary. They exemplify crop rotation in action.

Figure 3 presents information both by type of non-permanent data and in total. Space usage for this data is tracked by a second-to-second basis and the high-water-mark usage during each of 1000 intervals is used to compute the maximum and average usage with standard deviation for each type. Interesting observations from this data include the following:

- Contrary to common opinion, SORTWK datasets tend to invoke much *less* I/O than non-SORTWK datasets
- All temporary datasets (&&) tend to use much less DASD space than commonly believed — maximum interval space usage for these datasets tends to be about two percent of the available space
- Total non-permanent space usage (based on the absolute maximum concurrent usage by all types) averages 7.1 percent of the available space; this space processes 20.1 percent of the I/O on average.

If you look at this data in a fashion similar to that used for the permanent data in Figure 2, you see that the yield ratio is about three to one. Non-permanent data, using a crop-rotation concept is much more productive than more than 80 percent of all DASD space. Recent developments with system managed storage, as introduced by IBM with DFSMS, can be viewed as a form of crop rotation; the potential of dynamic storage management within and between various device types is exciting, particularly in view of these findings.

Poor Richard's Summary

Figure 4 presents a summary picture of the findings of this article. Some liberties have been taken to factor in the 6.7 percent average space utilization by SYS1 datasets. Values are modified to convenient increments. Annotations summarize the observations in the following paragraphs; comments regarding potential device types are based on the cost-per-I/O concept.

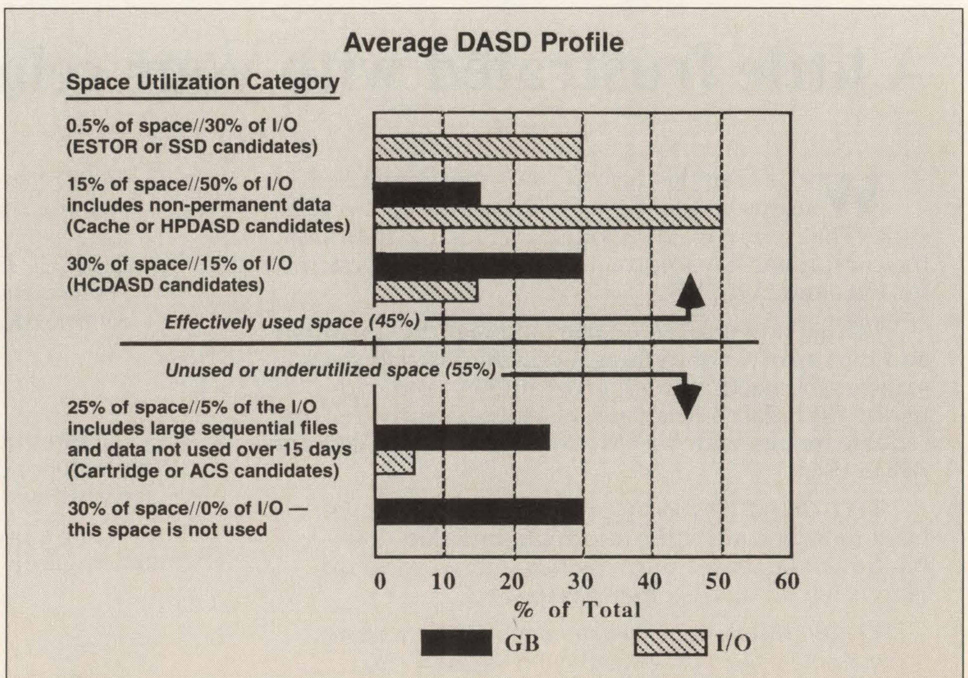
Approximately one-half percent of the DASD space generates 30 percent of the I/O activity. Because of this intensity of I/O activity, this data calls for a high performance device-type such as expanded storage or an SSD.

Approximately 15 percent of the DASD space generates 50 percent of the I/O. The high amount of activity to this data suggests the use of a cached DASD subsystem or high-performance single density DASD (for example, the IBM 3380J). All non-permanent data is included in this category because the I/O-to-space profile for such data is closer to the profile for this category than any other.

Approximately 30 percent of the DASD space generates only 15 percent of the I/O activity. This includes moderate activity data and low activity data which must be on a direct-access device. High-capacity (for example, double or triple density) DASD is the suggested storage media.

Approximately 25 percent of the DASD space generates a mere five percent of the I/O activity. This data grouping will in-

F I G U R E 4



clude large sequential files and data not referenced for more than 15 days (or any appropriate period). This type of data is a strong candidate for processing or archival with cartridge or an ACS.

Approximately 30 percent of existing DASD space generates no I/O activity and, in most cases, it is not allocated. This space is essentially not used.

Overall, only about 45 percent of an average DASD farm is utilized in a productive and effective fashion. Approximately 55 percent of the available space is either unused or underutilized and, therefore, is not needed or belongs on an alternative storage media.

A Farmer's Conclusion

Standard DASD usage surveys report that average space allocation is in the range of 65 to 70 percent and has remained fairly constant for several years. The findings in this article, however, are that "effective" DASD usage is only about 45 percent today and I believe that "effective" usage is declining as time passes. It is expected that faster CPUs and larger shared DASD complexes will force a reduction in effective space usage, at least with current technology.

What is the trend in "effective" DASD utilization? Is the projected growth rate for installed DASD capacity realistic or warranted in view of the fact that only 45 percent of currently installed DASD is used effectively? What if more data is placed on the proper device? What if the current trends toward system managed storage enable more effective utilization of existing DASD? What if —? Only time will answer these questions and the answers must be deferred to future studies.

ABOUT THE AUTHOR

A. L. Jones is a senior systems engineer for Storage Technology Corp. He has 22 years of experience in data processing which includes business applications programming, database management system design and development, technical services and computer performance and capacity evaluation. Storage Technology Corp., 2270 S. 88th St., Louisville, CO 80028-4358, (303) 673-3030.

VSE/SP Conditional JCL

By Mark Hanna

Conditional JCL is one of the enhancements provided with VSE/SP Release 2 and subsequent releases. With conditional JCL you can execute or bypass job steps according to the success or failure of previous ones. Conditional job control can be used to eliminate many situations: a VSAM file tape backup fails, the IDCAMS delete/define works and the restore step fails because there was not a tape created during the backup. A situation like this would result in the file being destroyed. This article presents the new features of conditional job control, provides examples for their usage and explains the new JCL statements.

Before conditional JCL, VSE job streams had to be planned carefully. Multiple job steps were placed within a VSE job so that if one canceled, the remaining job steps would not execute. If the job steps were in separate VSE jobs, then the next job on-job step would begin execution. Several techniques were devised by VSE shops to handle job flow after job step abnormal termination or cancel. A technique I have installed for several customers was an Assembler language routine used to flush remaining job steps in a job. This routine could be requested to cancel or dump. It also turned on the pause bit in the partition communications region so at end-of-job, the partition would stop with the READY FOR COMMUNICATIONS message. By issuing a cancel macro the job steps that remained after the canceling job step would not be executed. VSE job control would skip to the /& (end-of-job) statement. If you have a requirement to set return codes from a high-level language, chapter four of the IBM manual, *VSE/SP Migration Volume 2*, contains a sample Assembler and COBOL program that you can use to create your own callable subroutine to set a return code. Subsequent job steps can test the return code to decide execution flow. Table 1 lists the six new JCL statements and their meaning.

VSE conditional JCL does the following:

- Provides job-step to job-step communication
- Provides logic to alter job execution and job control statement sequence
- Provides checking of return codes set by IBM components and user application programs
- Utilizes six new JCL statements (the first five listed below are conditional JCL statements).

```
. /. label
. // ON
. // IF
. // GOTO
. SETPARM
. // PWR
```

Normal VSE job control processing of JCL statements takes place in sequential order as the statements are submitted to the system. A job begins with a // JOB statement and ends with a /& (end-of-job statement). Within a job there may be one or more job steps. With conditional JCL this normal top-to-bottom processing sequence can be altered conditionally. Sequence is altered depending on return codes set in previous job steps within the same VSE job. The return codes are passed to job control by programs or when abnormal termination occurs or the job is canceled.

Return Codes

Central to the use of conditional job control is the ability to set and check return codes. Condition codes may be set by Assembler language programs in the range from zero to 4096. The return code is usually set by the EOJ or DUMP macros. There are several return codes used by IBM programs. The return code standards for IBM products and components are listed in Table 2. The IBM VSE components and products that set and use return codes are listed in Table 3.

The return codes passed to job control are tested by IF or ON statements. Testing the return codes allows job processing decisions to be made on whether to skip or execute particular job control statements. Sequence is altered by the GOTO in conjunction with /. label statements.

/. Label Statement

The label statement is used to identify a point to which control may be passed skipping job control statements. The name used for label will be referenced by the label operand of the GOTO statement. The label may consist of one to eight alphanumeric characters and the first character must be alphabetic. A symbolic parameter is not allowed as a label. The first two characters "/." identify this as a label statement.

When used as a job control statement, the label statement must be coded on the same level as the GOTO that referenced it. The same level is defined as having both the label and GOTO inside a procedure or both outside a procedure. In other words, you may not attempt to GOTO a LABEL in a different procedure when using nested procedures.

For example, if a label statement was needed with a value of EXIT it would be coded:

```
/. EXIT
```

ON Statement

The ON statement for job control is a global condition check. When coded in a job stream, it checks the return code condition immediately after each job step. When this statement is processed in a job stream, it is valid for the rest of the VSE job. If several IF statements exist in a VSE job, they are stored and the most recent is checked first. When a GOTO label is specified on the ON condition, the GOTO target label must be on the same level as the GOTO statement. Checking will be done in the same level it was specified or in all lower levels. Therefore, an ON condition specified within a procedure is in effect until the end of the procedure. An ON condition specified outside a procedure is in effect until end-of-job.

An example of an ON statement that would check for a cancel condition and branch to the end-of-job statement would be coded:

```
// JOB
// ON $CANCEL GOTO JOBEND
.
/. JOBEND
.
/&
```

Table 4 shows the default ON state-

TABLE 1

Six New VSE/SP JCL Statements	
Statement	**Meaning**
/. label	Defines a locaiton within a job stream
ON condition action	Serves as a global condition for the entire job
IF condition THEN	Controls job step flow depending on certain conditions being true
GOTO label	Alters job step execution sequence conditionally or unconditionally
PWR R| H	Allows POWER commands to be issued conditionally or unconditionally
SETPARM	Defines a symbolic parameter and/or assigns a value to it

TABLE 2

IBM Return Code Standards	
RC	**Meaning**
0	Function executed
4	Function executed, but end result may not be as expected
8	Some functions not (or partly) executed, but processing continues
12	Function could not be performed
16	Severe error; job stream flushed

TABLE 3

IBM Components Support Of Return Codes		
Product	**Component**	**Programs and Remarks**
VSE/AF	Librarian MSHP Assembler	Llbr and LNKEDT: RC=12 not used MSHP: Only 0 and 16 supported Assembly: Only 0 and 128 supported
VSAM		IDCAMS
CICS		DFHEnp1$: n=A,C,P,R depending on language translator
EREP		Supported
FORTRAN		FFORTRAN: RC=99 for severe error; user program can set on STOP statement
DISOSS		Supported
COBOL R3 PI/I RPG II NCP/SSP DITTO Network PPs VTAM SDF/CICS SORT		Supported for compiler only No support No support No support No support No support for networking products No support not needed No support No support

ments return code value and action. These are the values that are in effect when a VSE job begins. These values are overridden when you issue an ON statement. If you want to override job control's check of all return codes, you would code the following: ON $RC > 0 CONTINUE. This will nullify the default actions and allows you to code IF statements to check the return codes in the job stream.

In Table 4 CONTINUE means that normal flow and processing will continue if the specified condition is true. $EOJ refers to the end-of-job statement /&. $ABEND means an abnormal termination. $CANCEL means a job has been canceled through the AR CANCEL or a program issued a CANCEL command.

TABLE 4

Default ON Conditions
ON $RC < 16 CONTINUE
ON $RC > 16 GOTO $EOJ
ON $RC = 16 CONTINUE
ON $CANCEL GOTO $EOJ
ON $ABEND GOTO $EOJ

TABLE 5

Possible ON Conditions		
condition	| OR AND &	condition
$RC $ABEND $CANCEL	comparator	number

Table 5 lists possible ON conditions and their meaning. Note that the conditions may be combined with a comparator. There are three ON conditions: $RC, $ABEND and $CANCEL. $RC specifies the return code of the preceding job step, $ABEND is for job step abnormal termination and $CANCEL occurs when the operator or the program issues a cancel request.

Referring to Table 5, the $RC condition allows one of six comparators against a return code. The comparators are: EQ or =, NE or ¬=, GT or >, GE or >=, LT or < and LE or <=. Two $RC conditions may be continued using the logical operators (OR or |) and (AND or &). The operators must be preceded and followed by a blank character. When $RC is used with logical operators the specified action takes place at end-of-job-step when:

- The conditions are connected by OR or | and one or both of them are true
- The conditions are connected by AND or & and both of them are true.

Multiple job steps which worked prior to VSE/SP2 may fail because the default action for return codes greater than 15 is cancellation of the remaining job steps. While this will be a rare occurrence, you can avoid the problem above by coding the ON statement.

IF Statement

The IF statement can only be used as a job control statement. The ON condition, just discussed, provides global conditional checking where the IF statement provides local condition checking. IF is only valid at the point in the job where it occurs. After the IF test, if the condition

TABLE 6

Possible IF Conditions

condition	| OR AND &	condition
$RC	comparator	number
$MRC	comparator	number
pname	comparator	string

checked were true, the next statement is executed. If the condition were false, the next statement is skipped unless it is a / / JOB, /& or / + statement. The IF statement can be continued. The conditions that can be checked are: $RC, $MRC and parameter name.

Table 6 lists possible IF conditions and parameters. The IF statement can check Return Code of preceding job step ($RC), Maximum Return Code from all preceding job steps in this job ($MRC) or a symbolic parameter value. The condition pname specifies a parameter value for comparison. The number condition specifies an integer from zero to 4095. The condition string specifies up to 50 characters. The symbolic parameter name tested (pname) must have been previously set with the SETPARM statement.

The IF statement uses the same six comparators as discussed under the ON condition above. It also uses the same logical operators and logical operator rules. If parameter name is used in the comparison, it will be a character string of zero to 50 characters. Special characters may be enclosed in quotes. Parameter name can be a symbolic parameter set by the SETPARM statement. It can also be used to check for a NULL parameter.

In the following example of an IF statement, the statement checks for a maximum return code of less than or equal to 10 and the return code of the previous job step less than eight. If the statement is true, the GOTO to label CKEOJ is executed. If the tested condition is false, the GOTO statement is bypassed.

```
// IF $MRC <= 10 AND $RC < 8 THEN
// GOTO CKEOJ
```

GOTO Statement

The GOTO statement causes all JCL statements or commands (except / / JOB, / & and / + statements) to be skipped following the GOTO statement up to the specified label. The specified label must be coded on a label (/ .) statement. An example of the GOTO statement would be:

```
// GOTO EXIT
```

If a / + is found during statement skipping, the rest of the job is skipped. You can specify $EOJ as a label to specify that all statements up to end-of-job are to be skipped.

SETPARM Statement

The SETPARM statement is used to assign a value to a symbolic parameter that may be used later in job control. SETPARM is accepted only within a job. For use in conditional JCL processing, the SETPARM value may be tested by an IF statement. A parameter may be a character string or a predefined return code. After the SETPARM statement is processed in a job, the symbolic parameter is treated as if it were the specified string or return code. It is in effect for only that one job. The symbolic parameters to be defined may consist of one to seven alphanumeric characters the first of which must be alphabetic. A symbolic parameter may be: a character string, $RC or $MRC.

Character string is a symbolic parameter value of up to 50 characters. $RC is the return code of the preceding job step and $MRC is the maximum return code from all preceding job steps in this job. To give the name RTNCDE1 to a return code, the following SETPARM statement could be issued:

```
SETPARM RTNCDE1=$RC
```

Later in the job stream it could be tested by issuing the following IF statement:

```
IF RTNCDE1>0 THEN
GOTO ENDJOB
```

/ / PWR Statement

The PWR statement, while not a conditional JCL statement, can be used to control or alter job processing. It can cause a VSE job in the POWER queue to be released or held. Syntax checking of the statement is done by VSE/POWER not the job control programs. Therefore, symbolic parameters are not allowed. The operand to specify the job to be held/released cannot be a single character (which would specify a class), the word ALL or an asterisk. It has the same syntax as POWER commands issued at the operator console.

In the following example, if a zero return code was returned from the previous job step, the job BKUPFILE would be released from the reader queue. Otherwise, a branch would be taken to the label NOBKUP skipping the PWR command:

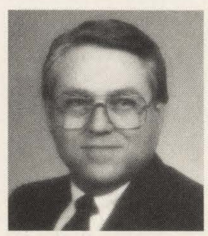
```
// IF $RC>0 THEN
// GOTO NOBKUP
// PWR PRELEASE RDR,BKUPFILE
/. NOBKUP
```

Conditional JCL Rules

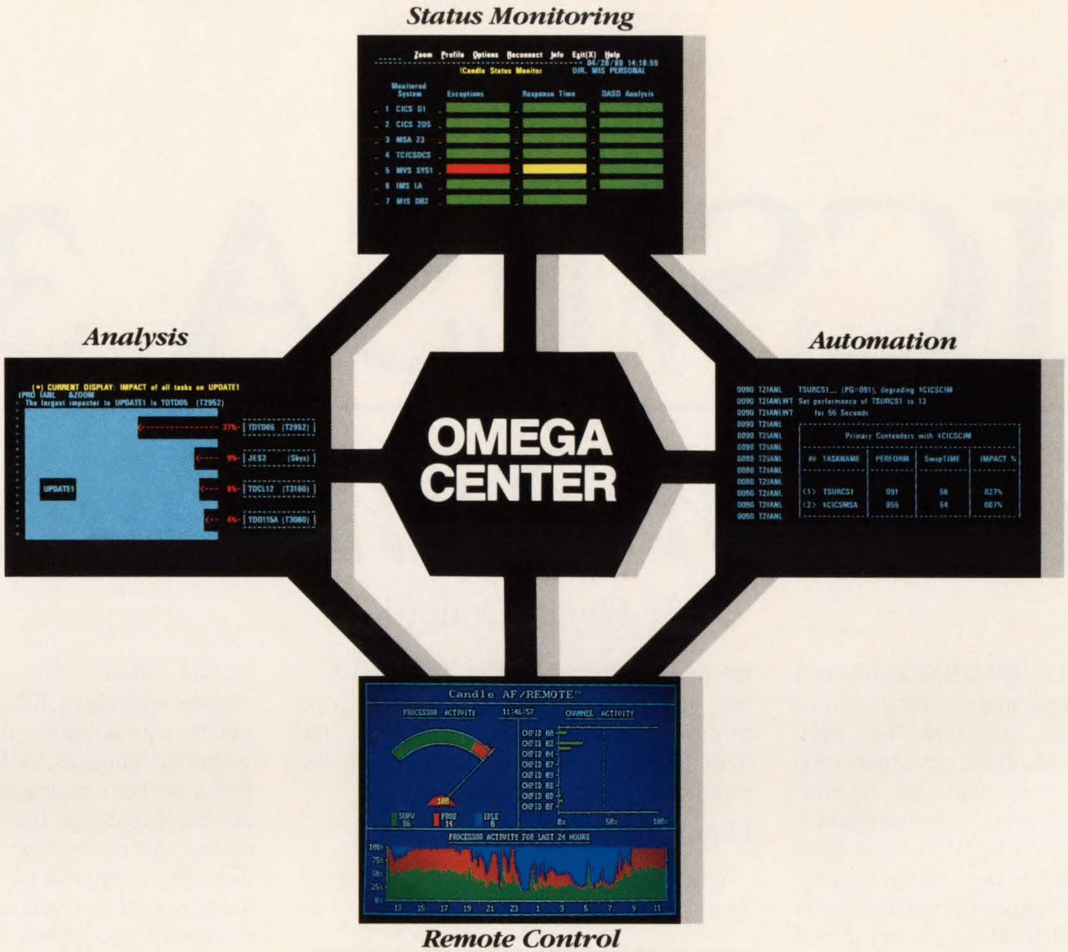
IBM has established the following rules for using conditional job control statements:

- The statements of a job stream can only be skipped in a forward direction
- A target label statement must be on the same level as the GOTO statement, that is, both outside a procedure or both in the same procedure
- If a label is not found before end-of-procedure or end-of-job, a skip to end-of-job is performed
- No check for duplicate labels is performed; a skip is always performed up to the first matching label found
- ON-conditions are checked first whenever a job step has been executed
- If there are several ON statements defined in a job stream, the conditions and actions of these statements are stored and processed in a last-in-first checked sequence; therefore, the last one to be issued is carried out
- If an ON $CANCEL condition is in effect and a job stream is canceled, either through the use of the AR command or a program request, the action specified is performed and processing continues; otherwise, the job is flushed
- An ON $ABEND condition is raised when a job step ends abnormally. If an ON $ABEND condition action has been specified, the action is performed and processing continues; otherwise, the job is flushed.

ABOUT THE AUTHOR



Mark Hanna is an IBM Business Partner. His consulting firm, Hanna & Associates, specializes in CICS, VSAM, COBOL, VM and VSE installation and support. He is the author of CICS Concepts & Facilities *and co-author of* Introduction To VSE & VSE Job Control *written with Suzan Hanna. Hanna has been in data processing since 1967. Hanna & Associates, P.O. Box 3325, Edmond, OK 73083-3325, (405) 340-1457.*



For data centers that need more than piecemeal solutions.

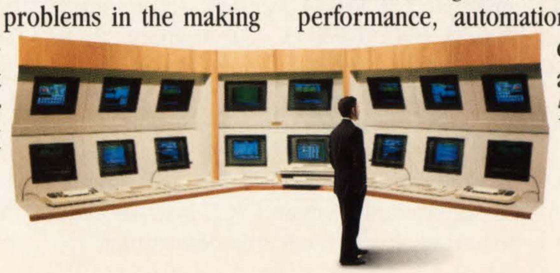
SAA lets you support more sites, more platforms, and more environments than ever before. But there's a catch: you still have to manage performance effectively across the entire Enterprise system and, at the same time, ensure the highest levels of end user service. Suddenly, partial solutions just won't work.

The task can be overwhelming . . . unless you enter the Center — Candle's OMEGACENTER,™ an integrated service level management system for today's forward-looking data centers.

OMEGACENTER is a completely integrated solution that simplifies the coordinated operation of the four essential components of Enterprise Performance Management: early detection, in-depth analysis, automation, and remote control.

At-a-glance Simplicity

The Status Monitor™ detects problems in the making and alerts you immediately with intuitive graphics. It provides single-point monitoring of up to 34 systems or subsystems anywhere on your VTAM network.



Analysis — Once the Status Monitor spots a problem, you can zoom directly into any OMEGAMON® for in-depth analysis.

Two-Way Control

Automation — Candle's AF/OPERATOR™ has the only effective interface to OMEGAMON. This bi-directional connection makes it easy to automate a wide variety of performance solutions — at machine speed and without the need for human intervention.

Remote Control — When human assistance is required, AF/REMOTE™ swiftly notifies designated personnel so they can solve the problem promptly from a remote ASCII terminal or PC.

Because you'll never find peace with piecemeal solutions, Candle has integrated all the essentials. OMEGACENTER — where performance, automation, and productivity join forces. For details, call Terry Forbes or your Candle account representative at (800) 541-8515.

Candle®

CIRCLE #120 on Reader Service Card ▲

CICS/ESA 3.1

It's Been Announced!

By Phyllis Donofrio

On June 20, 1989, IBM announced the new and improved version of CICS that everyone has been waiting for. At least, those customers who have submitted more than 200 requirements against the product from user groups such as SHARE and GUIDE have been waiting for it. That is how many requirements have been satisfied by CICS/ESA 3.1. Not only did IBM listen, but also it acted. Requests such as VSAM file definition via RDO and UCTRAN control at the transaction level are just two examples. Of course, there are still outstanding requirements that have not been incorporated — maybe in 3.2? The result, however, is an innovative version that incorporates significant new technologies and exploits new architectures of ESA. This article will attempt to present a few of the enhancements available in the new version such as new code base, storage management enhancements, performance enhancements, new CICS dispatcher, serviceability and support and migration/coexistence issues.

New Code Base

In CICS/ESA, much of the existing 2.1 code has been entirely rewritten into a different base, providing a new look for CICS. The Statement of Direction in October 1988 announced the promise of the *new* CICS without any specific date. The announcement of CICS/ESA 3.1 on June 20, 1989 contained more detailed information. With the proposed ship date of June 1990, IBM is providing a description of facilities and functions to prepare customers for its delivery. In addition to enhancements, the 3.1 version removes some previously-supported functions and CICS users must prepare alternatives or removal of these facilities if they wish to take advantage of this release. In addition to the warnings of removed function, the Statement of Direction further details the migration and coexistence issues.

Domains

CICS/ESA 3.1 contains a domain-based architecture. Domains are IBM's technique to *encapsulate* the program code and its control blocks. As defined in the new manual, *CICS/ESA Version 3.1 Release Guide (GC33-0655),* "a domain is a functionally separate element of CICS code that has sole control over its own resources." This technique isolates the *function* from the *use*. For many years customers have asked CICS to provide storage isolation within CICS systems and, therefore, avoid those dreaded storage violation failures. This new internal architecture separates CICS into functional domains by isolating the control blocks within the domains. A few functional areas that have been restructured in this version are the dispatcher, the loader, dump processing and monitoring. If data within the domain is required by another domain, a *domain call* is made and the results are returned to the requestor. This process uses a *strictly defined interface* between a domain and the requestor, affording protection for critical elements from incorrect modification.

> ***Version 3.1 incorporates significant new technologies and exploits new architectures of ESA.***

Of course, many new modules are distributed Object Code Only (OCO). Programmers who in the past enjoyed accessing and manipulating CICS internals may find it difficult, if not impossible, to continue to do so. These domain-based modules will, no doubt, consider this information proprietary and not manipulatable. The domain architecture will also allow IBM to enhance and service the product with less impact to the customer. Customers who requested storage isolation and integrity, yet continued to *tailor* CICS for their particular needs, will find it difficult to have both. Most installations, however, have "seen the handwriting on the wall" and removed source code modifications from their applications.

Storage Management

There are a number of enhancements to storage control in the 3.1 version. Some use ESA facilities to exploit the new architecture, while others provide Virtual Storage Constraint Relief (VSCR) that many installations have been waiting for.

While CICS has utilized the Dynamic Storage Area (DSA) for storage management for quite a while, the 3.1 version will contain an Extended DSA (EDSA) for storage above 16MB. The EDSA, like the DSA, is created during initialization and satisfies program requests for storage when possible above the 16MB line. New parameters in the SIT, DSASZE and EDSASZE replace the previous OSCOR entry to specify the size of the dynamic storage area, both above and below the 16MB line. Installations that were previously constrained by the size of the DSA and the high rate of program compression that could occur will now have substantially more capability with an EDSA.

Another new facility called Progressive Program Compression enhances program control. While the storage manager domain provides statistics of free storage to the loader domain, the loader domain uses a Least Recently Used (LRU) algorithm to delete programs and maps gradually, and with less impact on the system, reduce the possibility of Short-On-Storage (SOS). This process should make both the storage and program control within CICS more evenly managed during system operation.

A new attribute for programs and maps has been provided. If a program or map is infrequently used, it can be defined *transient*. When the use count for these resources reaches zero, storage is released and can be made available to other requests. This further enhances storage management techniques and gives the customer more control over what resources use the storage and for how long.

Additional extensions are provided to programming interfaces, including the EXEC CICS commands. These include several functions that were previously only available using macro-level programming. This, of course, is IBM's way to encourage installations to replace macro-level programs with supported commands.

Customers will receive VSCR from several changes: CICS supplied transactions, the File Control Table and code and SNTTEs. A major change has been made to DL/1. A new address space, Database Control (DBCTL) moves many DL/1 resources out of the CICS address space, much the same way that DB/2 *resides* outside of CICS. For the DL/1 user, this can free more than 1MB of virtual storage from within the CICS region.

Performance

In this release, domain-based monitoring provides substantial improvements in monitoring overhead. IBM states that the cost of performance monitoring has been reduced by 50 percent.

CICS/ESA 3.1 now uses the MVS program loader from a separate Task Control Block (TCB) in CICS. This provides the ability for CICS to take advantage of Library Lookaside (LLA) and Virtual Lookaside Facility (VLF) in ESA. CICS will now be able to load frequently-used resources, such as programs and maps, directly from the LLA dataspace (or hiperspace). A high rate of program load may now be satisfied directly from storage (main or expanded) rather than from DASD. This facility, in addition to Progressive Program Compression, could significantly improve CICS performance and program load time for many installations.

Other performance improvements include a user-selectable trace, allowing the installation to turn off trace during normal production hours, and the support of secondary extents in the CICS program library. A new dynamic transaction routing facility gives the MRO/ISC environment the ability to control the direction of transactions, potentially balancing the load between CICS systems.

New CICS Dispatcher

With the domain-based architecture, multiple CICS functions can be placed under separate MVS TCBs (much the same way that the first user of subtasking, VSAM, works in 2.1). This not only exploits multi-processor CPUs for installations that are running them, but also allows IBM to pursue this direction when functions and domains are added. N-way CPU exploitation via the new CICS dispatcher could significantly improve transaction performance.

Serviceability and Support

This will, undoubtedly, be one of the most controversial topics of CICS/ESA 3.1. As the product becomes more OCO and the domain approach removes internals from the applications that were previously available, problem determination techniques will have to change. This version contains several changes to existing facilities to provide *replacements* for accessing system internals. (What will third-party vendors do?)

TRACE

First Failure Data Capture is a new facility that utilizes trace points in CICS code to trap an error when it occurs, even if the trace flags for that function are off. This makes it possible to turn trace off (and the overhead of tracing) and still have information available pertaining to the failure.

Trace has been enhanced with several options. Installations can select tracing for specific CICS components, transactions or terminals. A CICS-supplied transaction, CETR, allows inquire and set of all new trace facilities. CETR provides screens for:

- Current status of the internal, auxiliary and GTF trace
- Current trace selection flags for each function and domain
- Standard, special and suppressed trace settings for terminals and transactions.

Trace entries can be printed via system and transaction dump formatting via DFHTUP or (for GTF traces) via the CICS-supplied DFHPDX formatting facility.

DUMP

Dump has been substantially changed

and has removed all support of the previous formatted, snap, storage violation and CICS region dumps. *All* CICS dumps are now produced via the MVS/ESA SDUMP facility and are routed to a SYS1.DUMPxx dataset. A new SIT option specifies retry intervals; however, if the dump is unsuccessful a CICS message is produced. Installations will be *required* to ensure that a SYS1.DUMPxx dataset is available. System programmers who have not previously used DFHPDX with IPCS to format CICS/OS/VS 1.7 or CICS/MVS 2.1 SDUMPs will be unable to escape the inevitable. Start using the facility now to avoid sudden *migration shock*. A complete explanation for installation and utilization of DFHPDX and IPCS was contained in the article, "CICS Dump Processing With DFHPDX," (August 1989, *MAINFRAME JOURNAL)*.

Migration/Coexistence

One major factor in any installation's decision to migrate to a new release is the protection of the business investment already made. No customer would discard existing applications and immediately begin rewriting programs in preparation for any new version or release. In that light, IBM has provided some migration aids and coexistence techniques to allow customers to begin using enhanced function while retaining (for some time) unsupported facilities.

Although no installations should be surprised, CICS/ESA 3.1 does not support COBOL or PL/1 macro-level applications. IBM made that perfectly clear in the October 1988 Statement of Direction. This release will be the last to support Assembler macros, probably since many installations have error routines such as NEP or PEP written in Assembler. Customers are encouraged to rewrite these routines using the command level interface. IBM has also provided DFHMSCAN, which can be used in pre-CICS/ESA 3.1 systems to locate and begin conversion of macro code. This facility is available via PTF for both Versions 1 and 2 and can be ordered through the normal service process.

Some additional function has also been removed such as TCAM (DCB) and BTAM. In the next release, exits will no longer have access to control blocks and applications will not have addressability to the CSA. Customers are encouraged to utilize RACF or another external security manager, since CICS internal security will be removed.

Installations wishing to run CICS/ESA 3.1 and retain some CICS systems with unsupported function are advised to keep CICS/MVS Version 2 to support those systems. CICS/ESA 3.1 can function ship via MRO/ISC to macro applications in a Version 2 system and use CICS transaction routing to access BTAM terminals from the 3.1 applications. Of course, this environment will require multiple CICS licenses (Version 2 and Version 3). In addition to support issues, installations will need to be aware of the financial impact of paying for both licenses.

Speaking of financial impact, customers may be amazed to note the license charge on CICS/ESA 3.1. As they say, "No Free Lunch," and the enhancements provided in this version are indeed, not free or even cheap. Thankfully, this announcement was made well before the intended ship date. Customers that plan to install when the product becomes available will need to reflect the increase in their budgets.

Summary

CICS/ESA 3.1 has many additional enhancements that cannot be fully explained in this short article. I would encourage customers who are interested in this new version to read the announcement letter in great detail. Even though this version does not ship until 1990, several manuals are now available and can be used to get a more thorough understanding of the changes and impact. *GC33-0655 Release Guide* and *GC33-0656 Migration Guide* contain a much deeper explanation of the new facilities. This version is truly a *new* CICS as promised in the Statement of Direction and will provide many new challenges and opportunities for those in CICS technical support.

ABOUT THE AUTHOR

Phyllis Donofrio has been involved in CICS technical support for more than seven years, including the installation of both CICS/VS 1.7 and CICS/MVS 2.1. She is the author of "CICS Debugging, Dump Reading and Problem Determination" from McGraw-Hill and is the Problem Determination Chairman of the SHARE CICS Project. Ms. Donofrio is currently an independent consultant for CICS and other MVS technical support and can be reached at P.O. Box 1521, Lutz, FL 33549.

Automated Service Level Management

And Its Supporting Technologies

By Jack Noonan

Technologies that aid the data processing professional in the management of service delivery are the focus of this article. The management of service delivery or Service Level Management (SLM) is a process that includes establishing service levels, managing exceptions to agreed upon service levels and planning capacity changes for future service requirements. In many environments, service delivery commitments are documented in the form of Service Level Agreements (SLAs).

SLAs usually contain the following elements:

- Availability: the time frame in which a service is available for use (on-line up from 6 a.m. to 9 p.m.)
- Delivery Time: the duration of a service activity (response time not to exceed three seconds, compiles not to exceed 30 minutes)
- Volume: number of concurrent uses of a service (no more than 300 concurrent users, no more than 50 transactions per second)
- Cost: the cost established for each unit of service (dollars per transaction or dollars per job)
- Quality: number of errors incurred during service delivery (number of reruns per shift).

Underlying effective SLM are some supporting technologies. Specifically, structural components of an SLM system consist of monitoring, analysis and automation together with the ability to do all of these from a remote location.

Monitoring

Monitoring is the foundation SLM technology for determining if a problem exists or if a service commitment is not being met. The monitoring component should provide status on availability, delivery times and volumes. Delivery times such as transaction response times are important when monitored, while the network user actually experiences them. In other words, response time should be monitored in an end-to-end fashion, displaying the actual response time experienced at the terminal. This is important in order to eliminate any confusion over the *real* response time.

The user interface of a monitoring system plays a strong role in the focus and alignment of all participants in the SLM process. This interface should be easy to understand and intuitive to use. A traffic-signal-like implementation seems appropriate to this application. A green status signal can indicate good service delivery or at least the absence of a service delivery problem. In addition, a yellow status signal can warn of potential service delivery exposures and a red status signal can indicate that a service commitment is not being fulfilled. Status displays of this type can then be used to focus all SLM participants on the single goal of keeping the lights green.

The statusing capability should be able to integrate status from any location within an enterprise. This will support the cooperative processing and distributed data characteristics of new applications. The statusing capability should also provide diagnostic navigation to additional information wherever the monitor may be located within the enterprise.

Analysis

Analysis to determine where the problem is located, what workload is causing the problem and what resources are in contention will facilitate in the correction or prevention of service delivery problems. This analysis applies to all service delivery exposures. However, in this article I will focus on transaction response-time problems.

When analyzing transaction response time it is important to understand where the transaction is spending its time; specifically, how much time was spent in the host and how much time was spent in the network? Doing this analysis first should improve problem-solving effectiveness. Notice that I said effectiveness, not efficiency. Location analysis will help in the selection of the correct problem to solve, but not necessarily help solve the problem faster. If a transaction spends one second in the host and 10 seconds in the network, the primary target for response-time improvement should be the network, because a 100 percent improvement in host response time would net less than a 10 percent improvement in total response time.

This same logic also applies to the workload impact analysis or the analysis of who is causing the problem. Why tune DASD to improve transaction response time if the problem is caused by a batch job that was inadvertently released on prime shift? There is no tuning trick known to man that will solve a response-time aberration of this type. The solution for this can be accomplished by any systems operator as long as the offending job can be identified. Once this job is identified, it can be canceled, swapped out for later execution or placed in a higher performance group to speed its completion.

Finally comes resource analysis. Depending on the resource in contention and the enterprise requirements, many things can be done, ranging from re-prioritizing or relocating resources to increasing resource capacity. Resource analysis has been with us for many years and is well understood by a large body of analysts. I have little to add to this subject except to say, do not do resource analysis first. Perform location analysis first, then workload impact analysis and, last, resource analysis.

Automation

Automation is the next logical extension to an SLM system. As data processing systems increase in size and complexity, automation becomes an imperative, especially when service delivery problems arrive at machine speed. The automation of data gathering, problem analysis and problem resolution can be combined to make SLM a winning experience.

Automating the data-gathering function has specific appeal for recurring short duration response-time problems. These problems are usually identified by an irate user complaining about slow response time. A better alert would be a red status indicator, but no matter how the problem is identified, a diagnostician will be needed. Just as the diagnostician arrives to solve the problem, all is well; the system returned to an acceptable response time or the status indicator turned green. As this scenario is repeated, all parties become frustrated. If, when the exceptional condition was recognized, the system had automatically gathered the appropriate information, the diagnostician would have had the information needed

Structural components of an SLM system consist of monitoring, analysis, automation and remote control.

to solve the problem. The automation of the data-gathering function has applicability to any exceptional condition where a diagnostician is not immediately available.

Automatic data gathering can become complex with many conditional paths. It becomes difficult to identify where automatic data gathering ends and automatic analysis begins. Automatic analysis could be described as conditional data gathering with a conclusion. No, automated analysis will not eliminate the need for the analyst or diagnostician. Automation will only free the diagnosticians from reinvention, allowing them to focus on the unsolved mysteries where intuition is king and the activity is still an art.

For those problems where automated analysis is applicable, automation of the solution can become no more than the execution of a command string. If the problem can be resolved through a terminal, then the solution can usually be automated.

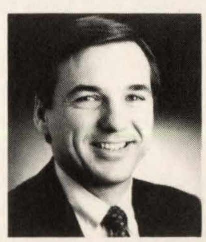
As you have learned, automation is applicable to the SLM process. It does, however, require unique and complex interactions between the monitoring, analysis and automation technologies. These interactions require a tight technology coupling to minimize the automation overhead and to simplify the automation effort.

Remote Control

Remote Control is the fail-safe component of an SLM system. No matter how much or how little the SLM system is automated, something will inevitably go wrong. When this happens, an expert is needed immediately. If the expert is off-site, a technology is needed to deliver the problem to the expert. This technology should have the capability to remotely alert the expert of the presence of a service problem and additionally provide the needed remote system access and controls to solve the problem. The ever-increasing decentralization of computing systems, along with the always decreasing availability of experts, positions remote control technology as a required component of any effective SLM system.

In summary, automated service level management consists of an integrated status management system to focus and align the organization, a tightly-coupled automation system to automatically analyze and correct SLM problems and, last but not least, a remote management system to facilitate remote diagnosis and repair.

ABOUT THE AUTHOR

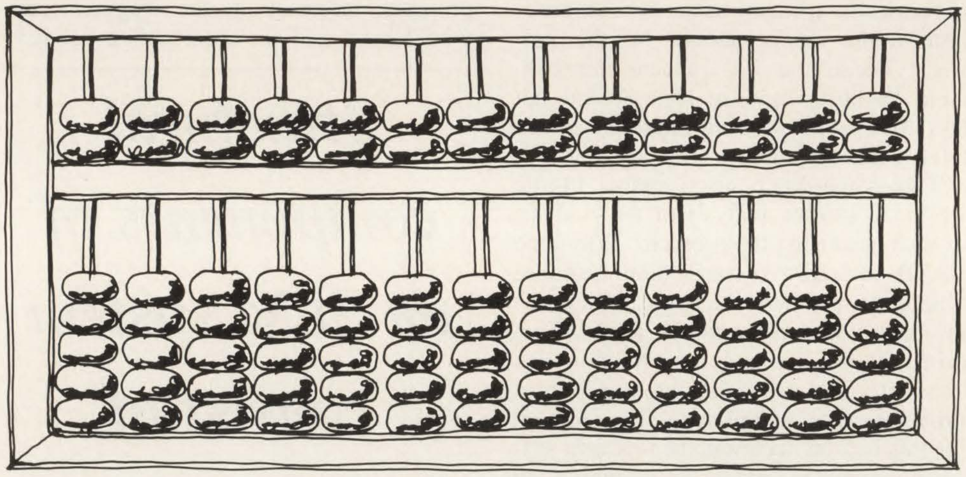


Jack Noonan is Vice President of the Product Group for Candle Corporation, headquartered in Los Angeles, CA. The Product Group is responsible for planning, research and development of Candle products, most recently in the automated operations, VM and DB2 markets. Candle Corp., 1999 Bundy Dr., Los Angeles, CA 90025, (213) 207-1400.

Tuning VSAM Index Control Intervals

Calculating VSAM's Key Compression Rate

By Michael D. Sachais



Once you know the key compression rate, use that information to calculate an appropriate index Control Interval (CI) size. This article will teach you how to print and read index records (similar to reading a dump) to determine VSAM's key compression rate.

Printing An Index Record

Figure 1 is a printout of a portion of an index record from the index component of a cluster named MY.KSDS.FILE. The name of the index component is MY.KSDS.FILE.INDEX. The index printout was generated using the following IDCAMS command:

```
PRINT INDATASET(MY.KSDS.FILE.INDEX) DUMP COUNT(4)
```

The IDCAMS PRINT command is used to print VSAM clusters and their components. The above PRINT command tells IDCAMS to print four records (COUNT(4)) from the index component MY.KSDS.FILE.INDEX in DUMP format. When printing the index records of a cluster, you should always print the records using the DUMP parameter of the IDCAMS PRINT command. This will print the index record in a format you can analyze. You should also print between five and 15 index records (using the COUNT parameter). This will give you a sufficient sample of the index records for your analysis. Only four index records will be printed in the above example because I know that there are only four index records in the index component of this cluster.

Another parameter of the PRINT command which should be used is the SKIP parameter. The SKIP parameter when used with the PRINT command allows you to print records in different parts of a component. For example, the following PRINT commands may be used to print six index records in an index component that contains 50 index records.

```
PRINT INDATASET(MY.KSDS.FILE.INDEX) DUMP COUNT(2)
PRINT INDATASET(MY.KSDS.FILE.INDEX) DUMP COUNT(2) SKIP(20)
PRINT INDATASET(MY.KSDS.FILE.INDEX) DUMP COUNT(2) SKIP(40)
```

In the first PRINT command there is no SKIP parameter. Therefore, two index records (COUNT(2)) will be printed starting with the first index record in the index component. No records will be SKIPPED. In the second PRINT command the first 20 index records in the index component will be SKIPPED over. Then, two (COUNT(2)) index records will be printed. In the last PRINT command the first 40 index records in the index component will be SKIPPED over, then two (COUNT(2)) index records will be printed.

Using these three PRINT commands together allows you to get a sample of the index records from the beginning, middle and end of the index component. This will give you a more accurate calculation of the key compression that is occurring in the cluster. This is helpful if the key compression varies in different parts of the cluster. It is, therefore, recommended that you use the SKIP parameter when PRINTing index records.

For a further explanation of the IDCAMS PRINT, SKIP and COUNT commands see the IBM *VSAM Catalog Administration: Access Method Services Reference* manual.

Analyzing An Index Record

Analyzing the information in an index record is similar to reading a dump. It consists of translating hexadecimal numbers into meaningful information. Like reading a dump, reading index records gets easier the more you do it. Because the information stored in an index record is recorded in hexadecimal values, I recommend using a calculator which can calculate in both hex and decimal to aid in your analysis. This will expedite the translation process of the information in the index record. I personally find it much easier to work with decimal values than hexadecimal values.

The following sections describe the information stored in an index record that you will use to calculate the average length of the key entries in an index record.

New Index Record Identifier

The NEW INDEX RECORD IDENTIFIER is not actually part of an index record, but rather a label printed out by the IDCAMS PRINT command. It can be used to identify the individual index records that have been printed by the IDCAMS PRINT command. Each index record that is printed will be preceded by the label "RBA OF RECORD — XXXXXX," where XXXXXX is the RBA of the index record. In Figure 1, the index record you will be analyzing is identified by the label RBA OF RECORD — 0.

Length Of The Index Record

The first two bytes of information starting at offset X'00' in an index record is the length of the index record. This will always equal the index CI size minus seven. In Figure 1, the two bytes at offset X'00' are 05F9. The hexadecimal value 05F9 converted into decimal is 1529. Figure 2 illustrates a partial LISTCAT for the KSDS cluster named MY.KSDS.FILE. In this figure you can see that the INDEX CISIZE for the cluster MY.KSDS.FILE is 1536 bytes (CISIZE in the INDEX ATTRIBUTES subsection). The size of the index record is, therefore, 1529 (1536 − 7) bytes.

Length Of The Key Entry Control Information

The one byte of information located at offset X'02' in an index record is the length of the control information in each key entry. The length of the control information can be three, four or five bytes. You may recall control information consists of the number of characters compressed from the front of the key, the length of the key and the pointer to the associated data control interval.

In Figure 1, the one byte at offset X'02' is 03. This means that the control information in each key entry is three bytes long.

Length Of The Key Entry Vertical Pointer

The one byte of information located at offset X'03' in an index record is the length of the vertical pointer in each key entry. The length of the vertical pointer is dependent on the number of CIs in a Control Area (CA). The length of the vertical pointer can be one (indicated by X'01'), two (indicated by X'03') or three (indicated by X'07') bytes long.

In a Sequence Set Index (SSI) record, a vertical pointer will be a pointer to one of the data CIs in the CA governed by the SSI record. In an index set index record, a vertical pointer will be a pointer to one of the index CIs to which the index record points. Remember, SSI records will contain the highest key in each of the data CIs in the CA it governs. Index set index records will contain high keys in the index records on the preceding level of the index structure. Therefore, the pointers in the different types of index records will point to the proper type of CI.

In Figure 1, the one byte at offset X'03' is 01 indicating that vertical pointers in the index record will be one byte in length.

Index Record Type Indicator

The one byte of information located at offset X'10' in the index record indicates

If you have article ideas, comments or suggestions concerning MAINFRAME JOURNAL, write or call: Bob Thomas, editor-in-chief, MAINFRAME JOURNAL, PO Box 551628, Dallas, TX 75355-1628, (214) 343-3717.

F I G U R E 1

Partial Printout Of An Index Record From The KSDS Cluster MY.KSDS.FILE.

```
RBA OF RECORD - 0

000000   05F90301  00000000  00000600  00000000  0100006A  009704F5  95949392  91908F8E
000020   8D8C8B8A  89888786  85848382  81807F7E  7D7C7B7A  79787776  75747372  71706F6E
000040   6D6C6B6A  69686766  65646362  61605F5E  5D5C5B5A  59585756  55545352  51504F4E
000060   4D4C4B4A  49484746  45440000  00000000  00C2D5D2  C2404040  40404040  40404040
000080   4040F1F1  F2F0F040  40404040  40404040  F9F9F940  F0F1C901  2643C2D5  D2C24040
  "
  "      (MORE KEY ENTRIES)
  "
0005E0   40404040  40404040  40F1F0F0  404040C4  F9F94040  40C40027     00
```

the type of index record (index set or sequence set) and on what level the index record exists. SSI records always have a level indicator of X'01'. The next higher level of index has a level indicator of X'02' and so on.

When calculating the average key length, you should only analyze the information in SSI records which are determined by a level indicator of X'01'. The purpose of tuning index CIs is to ensure that the index CI is large enough to address all of the data CIs in the CA governed by the index record. SSI records are the only type of index record pointing to CAs. Therefore, SSI records are the only type of index records that need to be analyzed when calculating the average key entry length.

In Figure 1, the one byte at offset X'10' is 01 indicating this index record is part of the sequence set. Therefore, it may be used to calculate the average key entry length.

Beginning Address Of The Unused Space In The Index Record

The two bytes of information starting at offset X'12' in the index record gives you the address within the index record where the unused space in the index record begins. If you subtract X'18' (the length of the index record header) from this number, you can calculate the length of the free CI pointer list within the index record.

If the address of the unused space is X'0018' (the end of the header information), you can conclude that there is no free CI pointer list in the index record because the free CI pointer list is stored between the header information and the unused index record space.

In Figure 1, the two bytes starting at offset X'12' are 006A (decimal 106). If you look at address X'006A' (located on line 000060 in the third column), you can see where the unused space (binary zeroes) in the index record begins. You can count that there are seven bytes of unused space in the index record (the last byte of binary zeroes is preceded by the byte X'C2').

To calculate the length of the free CI pointer list, subtract X'18' from X'006A'. This gives you a result of X'52' (decimal 82). This number tells you that the free CI pointer list is 82 bytes long.

The Free Control Interval Pointer List

The free CI pointer list begins immediately following the header information (at offset X'18' in the index record) and ends immediately before the start of the unused space in the index record. Its length depends on the number of pointers in the list and can be calculated by subtracting X'18' from the address of the start of the unused space in the index record.

If the unused space begins at offset X'0018', there is no free CI pointer list in the index record. This means all of the data CIs in the CA are utilized and the size of the index CI is large enough to store the keys of all the data CIs in the CA.

Free CI pointers are stored in the list in descending order and used from right to left beginning with CI 0. Therefore, the first (leftmost) pointer in the list represents the last CI in the CA that will be used.

The last pointer in the list is located immediately before the unused space in the index record and will point to the next data CI to be used in the CA. It is important to note that if all the records in a data CI are deleted, the CI is considered "free" and the pointer to that CI is re-added to the free CI pointer list. Therefore, the pointers in the list will not necessarily be in numerical order.

To begin analyzing the free CI pointer list, you first need to determine the number of data CIs in the CA governed by the index record. This will help you to determine the number of used and unused CIs in the CA. The number of data CIs in the CA can be obtained from the CI/CA value in the DATA ATTRIBUTES subsection of the LISTCAT.

The number of data CIs in the CA to which the index record addresses in Figure 1 is 150. This is shown by the CI/CA value in the DATA ATTRIBUTES subsection of the LISTCAT in Figure 2.

Now that you know how many data CIs there are in the CA, you need to determine the number of unused and used data CIs in the CA.

To determine the number of unused data CIs in a CA, you need to determine the number of pointers in the free CI pointer list. This will tell you the number of unused data CIs in the CA.

Subtracting X'18' (the length of the index header) from the beginning address of the unused space in the index record will allow you to calculate the length of the free CI pointer list. Dividing the length of the list by the length of each pointer in the list (given at offset X'03' in the index header) will allow you to calculate the number of pointers in the list. This is also the number of unused data CIs in the CA.

In Figure 1, subtracting X'18' (the length of the header) from X'006A' (the beginning address of the unused space in

F I G U R E 2

Partial LISTCAT For The KSDS Cluster MY.KSDS.FILE.

```
CLUSTER ---------- MY.KSDS.FILE
  DATA ---------- MY.KSDS.FILE.DATA
    ATTRIBUTES
      KEYLEN---------------------41   AVGLRECL----------------156   BUFSPACE---------------9728   CISIZE-------------------4096
      RKP -------------------------0   MAXLRECL----------------156   EXCPEXIT-------------(NULL)   CI/CA---------------------150
      SHROPTNS(2,3)          SPEED   UNIQUE              NOERASE   INDEXED          NOWRITECHK   NOIMBED           NOREPLICAT
      UNORDERED              REUSE   NONSPANNED
  INDEX ---------- MY.KSDS.FILE.INDEX
    ATTRIBUTES
      KEYLEN---------------------41   AVGLRECL------------------0   BUFSPACE------------------0   CISIZE-------------------1536
      RKP -------------------------0   MAXLRECL---------------1529   EXCPEXIT-------------(NULL)   CI/CA----------------------23
      SHROPTNS(2,3)       RECOVERY   UNIQUE              NOERASE   NOWRITECHK          NOIMBED   NOREPLICAT         UNORDERED
      REUSE
```

the index record) yields a result of X'52' (decimal 82). This is the length of the free CI pointer list (82 bytes). To calculate the number of pointers in the list (unused data CIs in the CA), divide X'52' (82 decimal) by one (the one byte at offset X'03' is 01 indicating that the free CI's pointers (vertical pointers) are one byte in length). This yields a result of X'52' (82 decimal). There are 82 pointers in the pointer list, which means that there are 82 unused data CIs in the CA to which this index record points.

You can now use the number of unused data CIs to calculate the number of used data CIs in the CA. To calculate the number of used data CIs in the CA, simply subtract the number of unused data CIs from the number of data CIs in the CA.

Using Figure 2, the number of used data CIs in the CA can be calculated as follows:

68	=	150	–	82
Used Data CIs		Data CI/CA		Unused Data CIs

There are 68 out of 150 used data CIs within the CA. Therefore, only 45 percent of the CA is being utilized; 55 percent is being wasted.

Calculating The Average Key Entry Length

The information you have just calculated can be used to easily calculate the estimated average length of a key entry in the index record. It is important to understand that the values you will calculate are only estimates. The amount of key compression will vary between keys and index record. Therefore, you cannot calculate an exact average key entry length for the keys in each of the index records of the cluster although an estimate of this value should suffice in calculating a proper index CI size.

To calculate the average key entry length in an index record, perform the following steps.

1) Determine the Beginning Address Of The First Key Entry

The first key entry in the index record is located immediately after the last byte of unused space in the index record. The starting address of the unused space, you may recall, begins at offset X'12' in the index record for two bytes. To locate the first key entry, scan through the unused space until you reach the first non-zero byte. This character will be the first character of the first key entry. Record the address of this byte.

In Figure 1, the starting address of the unused space is X'006A'. Scanning through the contiguous string of binary zeroes (X'00'), the first non-zero byte reached will be X'C2' at offset X'71' in the index record. This is the beginning address of the first key entry.

2) Determine The Number Of Used Data CIs In The CA

The number of pointers in the free CI pointer list is used to determine the number of used data CIs in the CA. The calculation used to determine the number of used data CIs in the CA was discussed in The Free Control Interval Pointer List section earlier in this article.

In this section, it was determined that 68 data CIs in the CA were used. Record this number.

3) Calculate The Amount Of Space Used By The Key Entries

The amount of space used in the index record to store the key entries can be calculated as:

Total Space Used By Key Entries	=	Index Record Length	–	Beginning Address Of Key Entries

By subtracting the beginning address of the key entries in the index record from the length of the index record, you eliminate the space in the index record used by the header, free CI pointer list and unused space. This leaves you with the space in the index record used to store the key entries.

In Figure 1, the length of the index record (the first two bytes of information starting at offset X'00') is X'05F9'. When converted to a decimal value, the length of the index record is 1529 bytes. The beginning address of the key entries is X'71' (decimal 113) which was determined in step 1 above. Therefore, the space in the index record used by the key entries is 1416 bytes (1529 – 113).

4) Calculate The Average Key Entry Length Per Data CI

Now that you know the space used by the key entries (step three) and the number of used data CIs (step two), you can calculate the average key entry length per data control interval as:

Average Key Entry Length	=	Space Used By Key Entries	/	Used Data Control Intervals

The average key entry length is only an approximation of the length of a key entry in the CA governed by the index record. To get a better estimate of the average key entry length in all the CAs in the cluster, you need to perform steps one through four on various other index records. I recommend analyzing between five and 15 SSI records in the index component of the cluster and calculating an average from the results you get in your calculations.

For example, if you analyze five index records and the average key entry lengths you calculate are 8, 12, 10, 10 and 9, use the average of these numbers (10) as the average key entry length. When averaging the numbers to arrive at a final average key entry length, always round up. It is better to overestimate the key entry length than underestimate it. Use the following guidelines to round your numbers:

- If the calculated average key entry length contains a decimal value greater than or equal to .5, round up to the nearest whole number and add one; for example, if the average key entry length is calculated as 9.6, use 11 (9.6= => 10 + 1) as the average key entry length
- If the calculated average key entry length contains a decimal value less than .5, round up to the nearest whole number; for example, if the average key entry length is calculated as 9.1, use 10 (9.6= => 10) as the average key entry length.

Using Figure 1, the average key entry length for the CA governed by the index record would be calculated as:

20.82 = 1416 / 68

Following the rounding rules, this number would be rounded to 22 (20.82= => 21 + 1). For the sake of simplicity in this example, analyze only one index record and use the average key entry length of 22 in any further calculations.

5) Calculate The Minimum Acceptable Index CI Size

Knowing the average key entry length per data CI will enable you to calculate the minimum CI size needed to store the key entries for all the data CIs in the CA.

The minimum index CI size required can be calculated as:

Minimum Index CI Size	=	Average Key Entry Length	X	CI/CA

The minimum index CI size must be rounded up to the next valid index CI size.

The number of data CIs per CA (CI/CA) is determined by the size of the data CIs and the CA and may be obtained from the LISTCAT CI/CA value. From step four you determined the average key entry length is 22 bytes. The minimum index CI size required for the cluster MY.KSDS.FILE can be calculated as:

3300 Index CI Size	=	22 Avg Key Entry Length	X	150 Data CI/CA

The calculated index CI size of 3300 needs to be rounded up to 3584 which is the next valid index CI size. From the above calculations you can conclude that an index CI size of 3584 should be large enough to store the keys of all the data CIs (150) in the CAs of the cluster.

In the next article you will learn how to test and choose what index CI you should use to DEFINE a given cluster. You will then be able to accurately tune the index CI size for a given VSAM KSDS cluster which in turn will minimize DASD space utilization by the cluster and minimize BATCH and CICS processing times when processing the cluster.

ABOUT THE AUTHOR

Michael D. Sachais is the author of "VSAM Tuning and Advanced Topics," VanNostrand Reinhold Publishers. He has also taught numerous classes on VSAM and VSAM tuning. His experience includes system design, application programming and system tuning in an IBM mainframe environment. 2750 Harrow Dr., Atlanta, GA 30341, (404) 454-9846.

Q **We are currently running VSE/SP 2.1.5, CICS/VS 1.6.0 and VSE/VSAM 1. 3. We have one application running under CICS that logs transaction records to a VSAM ESDS dataset, which has an alternate index associated with it. The problem we experience is that when CICS crashes for one reason or another, apparently the VSAM buffers are not written to DASD, leaving the base cluster and alternate index out of sync. I cannot seem to get a SHAREOPTION on the datasets to force the physical I/O for each write, because I cannot find the right combination of FCT entries that will allow the path to be opened without getting a VSAM open error 168.**

Is there a combination of entries in the FCT that will allow these files to open with the SHAREOPTION(4) or is there another way to guarantee that these files stay in sync, in spite of a system crash?

A CICS/VSE users are exposed to this type of integrity problem when working with VSAM alternate indexes. The problem you encounter is not due to the shareoptions you have chosen, nor can it be solved simply by selecting a different set of CICS FCT parameters. The problem stems from interaction between CICS and VSAM function and the fact that a single request from CICS to update a VSAM file record may result in updates to an entire set of AIX records "under the covers." With a system failure during this process, AIX records can be out of sync with the base cluster and CICS recovery actions will be unable to correct the condition. CICS Emergency Restart will run fine and transaction backout will operate normally, but the AIXs will remain out of sync.

Incidentally, this response assumes you use CICS logging and recovery functions, although the synchronization problem would exist even if you did not. You describe your VSAM file as ESDS with an AIX attached to it and CICS has different problems backing out changes to ESDS files. In fact, additions made during normal processing cannot be deleted by Emergency Restart and user design needs to consider if these records are to be marked as logically deleted during recovery and coding added to recovery exits as needed.

Typically, AIXs are defined as part of the "upgrade" set, which means that an update being made to the base cluster record may be reflected automatically in one or more of the alternate indexes. When you add a record to a VSAM cluster, most if not all of the alternate indexes will also be updated — in some cases with record additions and in others, with updated index records.

This AIX updating occurs before the base cluster record is written to disk and, as a result, if an error occurs while the updating is in process, the index records may point to keys which do not yet exist in the base cluster. Before CICS issues the VSAM WRITE, it writes a record to the log indicating the requested change for the use of Emergency Restart. When Emergency Restart is invoked to back out changes for in-flight transactions, VSAM will not recognize a need to correct alternate index records if the original base cluster record is still intact. This results in indexes and the base cluster being out of sync and the result is a loss of integrity.

One solution used by many customers is to rebuild alternate index datasets when a crash occurs. In this mode of operation, when a CICS or system crash occurs and Emergency Restart is anticipated, the base clusters are first VERIFYed and then the AIXs are rebuilt. This step brings the AIX and base cluster back into sync and Emergency Restart can begin. Another option is to define all AIXs as NOUPGRADE, which means VSAM will not keep them updated as the base is updated. Where UPGRADE is still desired for major batch updating, then you may wish to define a NOUPDATE path into the base which is used for no other purpose than to tell VSAM to turn off the UPGRADE option for this invocation of the base.

In the latter case, you become responsible for maintaining the AIXs, either through BLDINDEX or user program.

Another approach which eliminates the integrity problem is to process the AIX and base cluster as two separate files. In effect, you take on the responsibility for keeping the cluster and its indexes in sync, which can entail considerable effort. In this case, the files are treated separately by CICS and, for KSDS files, are fully recoverable. ESDS again provides the challenge, needing user code additions to standard CICS recovery functions and possible changes to application design to ensure full recoverability.

Physical write to the dataset will occur regardless of the selected shareoptions. Shareoption 2 should be used for both the VSAM base cluster and the AIX. Ensure only one (base or AIX) has update processing options defined or you will encounter OPEN errors with Shareoption 2. CICS/DOS/VS supports record updating through either base or AIX, not both. Update through the base cluster *only* is strongly recommended.

(Answer provided by Bob Cornell and Dan Janda of IBM)

Q **When would you use the REPRO/MERGECAT function of Access Method Services?**

A A good time to use REPRO/MERGECAT is if there is performance degradation due to too large a catalog (with multiple aliases pointing to it). Another time to consider using REPRO/MERGECAT is if you have a critical (such as on-line) application and the catalog entries are in a large multi-aliased catalog. If that catalog had to be recovered, the downtime to the application could be costly. There are, of course, other reasons to use REPRO/MERGECAT, but I feel that performance, availability and recoverability should be the key considerations. Here are a few general rules regarding catalogs:

- Non-system related datasets should be in user catalogs
- Catalogs should be backed up at least once daily along with an integrity check
- Have several user catalogs versus one large user catalog
- Periodically reorganize catalogs
- Monitor performance (note that when the number of index levels is greater than two, performance will start to degrade).

(Answer provided by Mike Bowbin of Davis, Thomas & Associates, Minneapolis, MN)

Orion Solid State Disk Battles Poor Response Time

By John Kador

Historians are fond of noting that people who ignore the past are condemned to relive it. They do not take into account that if MIS managers run out of computer capacity to process payroll, they will *not* have the opportunity to relive the experience. Unemployment lines are filled with MIS managers who failed to recognize the problems of capacity performance and management.

One of the biggest problems MIS managers face is the seemingly insatiable demand for memory and storage. The trend toward on-line processing and ballooning information storage requirements created by the growth of databases and end-user computing has caused runaway growth in the demand for DASD. According to leading industry-watchers, the DASD capacity growth rate has averaged 35 percent for the past five years and that growth is likely to continue. Unchecked, DASD demand can eat up DP budgets and cause monumental headaches for both MIS staffs and end-users.

Computer managers usually respond to the need for more DASD by getting more disk drives. But rotational disk systems impose a large measure of inefficiency. An increasingly popular alternative to rotational disk devices is solid state disk devices. These devices, like the ORION Solid State Disk Subsystem from EMC Corporation (Hopkinton, MA), are hardware alternatives to traditional rotational devices.

EMC competes with STK (formerly Storage Technology), National Advanced Systems (NAS), Memorex/Telex and Amdahl Corporation (see Table 1). Although EMC is currently the junior member of the fraternity selling solid state devices, it enjoys the highest rate of growth, gaining 10 percent of the market from the bigger players in its first year. To understand why this is so, it is useful to take a brief look at the marketplace in which it competes.

TABLE 1

Solid State Storage — Worldwide Market Share by MEG.

STK	42.0%
Memorex/Telex	21.5%
Amdahl	13.5%
NAS	13.5%
EMC	9.5%

There were 120,000 MEGS of solid state storage sold worldwide in 1988. The percentage figures represent market share by MEG sold.

Source: International Data Corp., Framingham, MA

STK offers its own solid state subsystem. NAS and Memorex/Telex sell a solid state device manufactured by Hitachi. Amdahl markets a device manufactured by Fujitsu. EMC manufactures and field supports its own design. The point is, EMC's competitors all had existing disk controllers. They took the existing controller designs and added new devices which emulated solid state storage.

In June 1989, EMC and STK agreed that the ORION/VL solid state disk technology will become the basis for the STK 4080 Solid State Disk. Under this agreement, EMC will provide STK with the VL technology on an OEM basis for use in the 4080. EMC will continue to market only the ORION/ST. Marketed under the STK label, the 4080, for the first time in STK history, will offer both battery and hard disk.

The ORION/VL, the first octal director solid state disk device, was introduced in late 1988. Each director has a dedicated path to data. Thus, users can simultaneously access eight different CPUs or channels. Previously, users seeking eight paths to data were forced to use a four director unit with channel switching. While channel switching on four directors provides eight paths to the CPU, the paths cannot be accessed simultaneously.

Starting From Scratch

EMC, by contrast, did not have an existing disk controller to modify. Instead, the company had to start from scratch. "If it already had its own disk controller (like the other companies), it probably would have done the same thing," suggests Dave Valente, an analyst with International Data Corporation (Framingham, MA). "But it didn't and as long as EMC was going to spend the money, the company decided to do it right."

One result is all circuitry is on the board level. The benefits of this compactness are immediate. By integrating the controller and memory into the same physical backplane, EMC reduced overhead and footprint requirement. First and most obviously, the environmental differences between the EMC box and the competition are staggering. The ORION looks like a PC in a tower configuration versus a couple of refrigerators. Installation is said to require less than one hour. More significantly, performance also is boosted, due in part to the physical proximity of the circuitry on the board (see Figure 1).

A university reduced average response time from 4.8 seconds to 1.1 seconds. Before ORION, 90 percent of transactions were satisfied in 4.8 seconds; after ORION, 90 percent of transactions were satisfied in 1.1 seconds. The number of screens processed per day increased 21 percent after ORION was added. This performance analysis was conducted by

Indiana University on an IBM 4381-P13 under VSE for a number of CICS applications. Measurements were made by a commercial CICS monitor over a three-month period. The university used an ORION/ST 64MB single director.

How does a data center know when the addition of a solid state disk would be appropriate? Because a solid state subsystem is fast, the prime indication is a need for immediate response times for critical applications. Solid state disk systems virtually eliminate I/O bottlenecks caused by high page or swap rates. Users who define a solid state disk as their primary paging devices report faster access to heavily accessed shared data. Besides faster response times, it is also important to have consistent response times to maintain high levels of user productivity. Occasional lightning fast response times may actually degrade productivity because they make sluggish response times more objectionable by contrast.

Three Generations

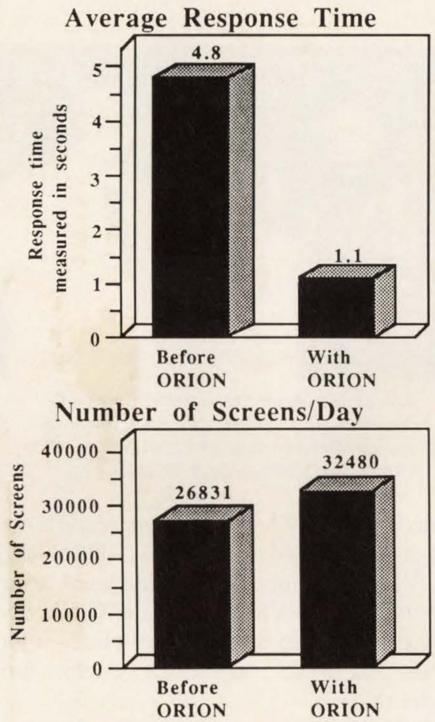
The ORION series is representative of the third generation of solid state storage devices. The first generation, circa 1983, centered around the Intel 3805 and STK 4305. These units were limited in capacity. They emulated IBM 2305 fixed head architecture. They were used almost exclusively for paging applications. Environmentally, these units required large amounts of floor space and electrical power. They vented high levels of heat, as well.

The second generation of solid state disk subsystems, circa 1985, was built around the STK 4380 subsystem. Memorex and NAS introduced the Hitachi system. These systems demonstrated increased capacity and first offered hard disk and battery backup options. These options first made the systems useful on a large scale for applications other than paging (for example, database indexes or load libraries). The footprints of second-generation systems averaged about 20 square feet, a considerable improvement over first-generation designs.

The third-generation solid state subsystems, offering still higher capacities, are more compact and more reliable. They are based on 1MB chip technology and, as a result, draw less power, generate less heat and are more reliable across-the-board. Users are increasingly putting heavily accessed data on third generation systems with hard disk and battery backup.

FIGURE 1

ORION Reduces Average Response Time; Increases Average Screens/Day Processed



Source: Indiana University, Terre Haute, IN

Headache For Dresser Industries

System performance was a big headache for MIS Director Richard Ford at Dresser Industries, a construction and mining equipment manufacturer in Broadview, IL. Due to a combination of factors, the company was forced to downgrade from a 4381 Model 14 to a slower 4381 Model 3 during a time when CAD/CAM users were complaining of poor response times. In May 1988, Dresser installed ORION as a preferred paging device on VM/SP systems. There were immediate across-the-board improvements, recalls Ford. Specifically, the number of users in resource queues across the 16MB and 32MB configurations decreased by 17 percent and 35 percent, respectively.

Improvements in paging and queuing were expected. But ORION unexpectedly caused a reduction of CPU utilization, as well. According to Ford, the "16MB unit retrieved 66 percent of our CPU and the 32MB unit saw a 15 percent reduction in total CPU utilization. The bottom line is that all complaints regarding response time have vanished and our CAD/CAM users are happy with system performance."

DASD Discipline

System performance problems are the purview of DASD capacity management, a discipline that attempts to deal with performance problems using either software or hardware or a combination of both. Some software packages help clean up inefficiencies in the way files are stored, netting additional usable space. This may be accomplished by pooling space, compressing data, providing a non-specific volume allocation facility for some datasets, building into each volume a cushion of unused space for secondary allocations of existing datasets or by fine tuning existing files in a variety of ways. Some packages dynamically allocate space, archive data on the basis of logical statements or predict how much space a given dataset will take up. Others apply principles of storage hierarchy management or systems managed storage to avoid the use of expensive storage media for data that is seldom used. EMC's ORION solid state subsystem is another alternative for users to address performance problems.

The ORION subsystems boast fast response, small footprints and low power consumption. ORION solid state subsystems are 100 percent compatible with all IBM S/370 architecture CPUs and PCM computers including Amdahl and NAS systems. Because data recovery is an important issue, EMC offers battery and Winchester hard disk backup options to guard against loss of data due to power failure.

The ORION system offers access times of 0.1 millisecond, transfer rates of 1.0 to 4.5MB/sec per director, logical volumes of one to 16, DASD images of 3370 FBA and 3380 CKD and either one or two directors. Maximum capacity for the system is 512MB (single director) or 448MB (dual director). Pricing is calculated in terms of megabytes of storage at $1400 per megabyte plus maintenance of $4.20 per megabyte (for 16 to 48 megabytes). The cost per megabyte decreases as customers install larger systems. The cabinet for the initial configuration is provided at no cost. Additional cabinets are $7500.

For more information, contact EMC Corporation, 171 South St., Hopkinton, MA 01748-9103, (617) 435-2541 or (800) 222-EMC2.

ABOUT THE AUTHOR

John Kador is a free-lance writer specializing in the business applications of mainframe software.

Integrity Solutions, Inc.

Integrity Solutions, Inc. (ISI) is a leading supplier of data integrity software for IBM mainframe organizations with VSAM batch and CICS/VS environments. With more than 1200 products installed at 600-plus data centers worldwide, the company is the largest provider of data recovery and journal management software to this marketplace.

ISI pioneered VSAM data recovery with the introduction of the DATA RECOVERY SYSTEM (DRS) in 1981. Founders Jerome Nickerson and Karl Schulz bring more than 40 combined years of experience in the forefront of this technology to the development of DRS. The original DRS product was written for Central Bank of Denver, CO to assure the integrity of its Hogan Banking System's data. The marketing rights for DRS were retained by the developers and DRS was successfully marketed to additional banks and financial institutions that first year. Today, the company's products are installed in virtually every type of industry including banking/financial, insurance, government, retail, manufacturing, education and service.

As the needs of the marketplace have changed, DRS has grown to meet these requirements. In addition, new products were developed to address evolving needs of the customer base. The JOURNAL MANAGEMENT SYSTEM (JMS) provides automatic journal archiving for CICS/VS journals and logs, the INTEGRITY CONTROL SYSTEM (ICS) automates the recovery process and MAX/SPF brings powerful programmer productivity assistance to TSO/ISPF users.

All of ISI's data integrity products are designed to work together to provide system-wide data integrity. However, each product may also be purchased separately to address specific user requirements and then integrated with other products when needed to form custom-configured systems.

DRS/UPDATE and DRS/RECOVER provide alternate methodologies for fast and accurate forward or backward data recovery between backups for VSAM files updated by CICS/VS. DRS uses backup files (either installation- or DRS-provided) and CICS/VS journal records as input to the recovery process. DRS/RECOVER is unique in providing 24-hour availability of VSAM files to CICS/VS while the backup process takes place. Both VSE and MVS operating systems are supported.

From left to right are Jerome Nickerson, President, Toni Hippeli, Vice President of Corporate Development and Karl Schulz, Vice President of Research & Development.

DRS/BATCH provides comprehensive recovery for VSAM batch environments by transparently journaling VSAM updates. With DRS/BATCH, batch job restart and rerun time is reduced and redundant backups can be eliminated. DRS/BATCH assures that your batch processing is completed on time and within your "narrowing batch processing window."

JMS/SWITCH, which was introduced in the fall of 1986, ensures the integrity of CICS/VS journals and logs by automatically archiving them with no computer operator intervention. It prevents accidental overwrites. JMS/SWITCH also manages the job submission process and its extended features significantly enhance the on-line management of the entire journaling environment.

ICS/MANAGER, introduced in July 1989, is the newest component in a system-wide data integrity system. It acts as a data integrity umbrella and may be optionally combined with any or all of ISI's other data integrity products to automate the recovery process and assure even faster and more accurate recovery. ICS/MANAGER automatically registers and manages all critical backup and journal information on a global control dataset. It serves as a single repository of all essential information needed for an error-free recovery and it automatically generates and submits customized recovery JCL when needed. Future releases of ICS/MANAGER will manage recovery across additional operating environments such as DL/1, IMS and DB2.

MAX/SPF is a new productivity enhancement tool for TSO/ISPF programmers. It provides faster and easier access to frequently-used datasets and PDSes with a unique customized DATASET NAME LIST. It also consolidates commonly used ISPF/PDF functions into one single, more flexible facility. MAX/SPF edits all VSAM and SAM files in traditional full-screen mode or COBOL Copybook layout option.

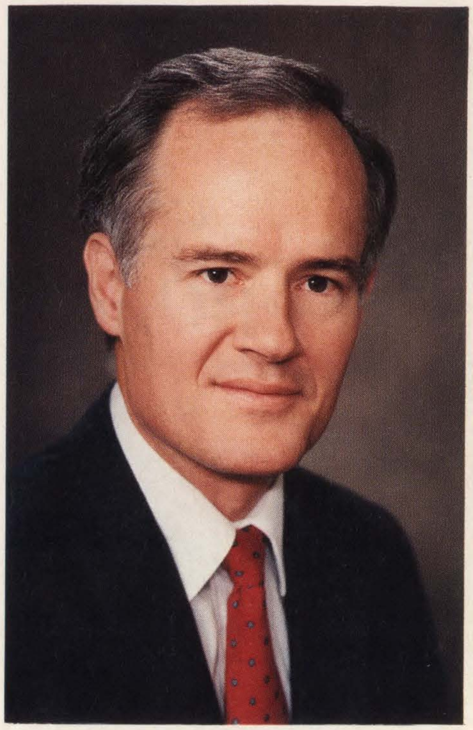
ISI's corporate offices are located in Littleton, CO, a suburb of Denver. All domestic sales are made from this location. ISI has also established a strong international marketing organization. European distribution is handled through Integrity Solutions, International (ISIL) in London, established in 1984. ISIL manages distributors in the United Kingdom, France, Italy, Spain, Portugal and West Germany. In addition, ISI has distributors in Australia, Thailand, Singapore, Malaysia and Hong Kong. During 1989, ISI plans to expand its distributor network to other Pacific Rim countries, including Japan.

ISI is committed to setting the industry standard for data integrity. This commitment means continuing to provide the highest level of excellence in all aspects of the company's operations by anticipating customer needs, developing leading-edge software solutions that address these increasingly sophisticated requirements and providing the highest level of customer service at all times. It is currently involved in setting the standards in the emerging electronic vaulting and mirror image processing environments as they become integral components of the disaster recovery marketplace. ISI has been designated an authorized IBM remarketer for these new technologies.

Vendor Profile is a regular forum whereby a vendor is given the opportunity to introduce the company and its products to MAINFRAME JOURNAL readers.

The Role Of The Help Desk In The DP Center

By Ronald J. Muns



Ronald (Ron) J. Muns is Director of The Help Desk Institute and President of Bendata Management Systems, Inc. of Dallas.

In the 1990s a common question will exist for information processing-driven organizations across the country: what is the Help Desk's role in the DP center? It is not surprising that this issue should arise particularly since information processing, as a competitive weapon, has expanded to include corporations of all types and sizes. The new information processing tools supported by the Help Desk are diverse and often complex. Besides mainframes, terminals, PCs, data networks and applications, there are expanding uses for specialized workstations, Local Area Networks (LAN), Wide Area Networks (WAN), expert systems and robotics. These new technologies will be ''mission critical'' to many organizations in the 1990s.

In order to better comprehend the role of the Help Desk, it is important to understand how the Help Desk function evolved, what its primary goals are and the areas that can hamper the DP center in realizing all of the benefits of a Help Desk.

The Help Desk function has always existed in the DP center in some fashion. Years ago the user simply called the operator or programmer. In the past few years, however, two things occurred that formalized the Help function. One was IBM's development of CSA maintenance agreements. These agreements reduced customer maintenance cost if customers established centralized control of problem management and reporting. The second event was the explosion of end-user computing tools resulting in enlightened end users calling for new computing capabilities and demanding assistance. The Information Center was eventually created to facilitate improved computer literacy and the automation of routine, clerical and analytical end-user tasks. Today, the Help Desk is that and more. As the DP sales force, it represents the DP center's competitive services to the user market and acts as the user liaison communicating the market's needs to the DP center.

A fair way to evaluate the future Help Desk's role is also to consider what it is *not*. The Help Desk is not a resource pared away from the DP center. It is not simply a must-have function to earn maintenance discounts. And, it is not available to pacify end users who only want to control their technological destiny. The Help Desk offers a new range of services with genuine economic value to the organization.

As a broad statement of purpose, the Help Desk's objective provides an efficient means toward resolving end-user problems. Consequently, the end user becomes more productive and the DP technical staff continues working on the never-ending backlog of program development activities. The user community is better equipped with the tools, training and support needed to solve business problems. In the day-to-day shuffle, the Help Desk ensures that a user's problem receives a timely resolution.

Conflict between the DP center and the Help Desk often revolves around the interpretation of the Help Desk's purpose and role. This is not surprising due to basic differences in the persons best suited for the Help Desk as opposed to the DP center.

Conducting a needs assessment better enables an organization to understand the role of the Help Desk and to better realize its potential. A needs assessment includes surveying data center supervisors and management and interviewing key users in the community. Questions include: who and where are the customers? what kind of information do they process? how do they use the information they process? with whom do they share information? what is their current level of automation knowledge? what problems are being experienced? how are they receiving help? what do users like best/worst about their current support?

The Help Desk must sell itself and its role within the organization. A brochure or manual on services, policies and procedures, for example, are effective marketing approaches with great benefits for both the DP center and user community. Publishing newsletters and forming users groups, as well as holding open houses to familiarize new users with available services, are all powerful sales tools. The image of the Help Desk will increase within the DP center and end-user community if personnel will provide high level analytical reports, charts and graphs of call history. This data coupled with verbal and written analysis will allow the Help Desk to have a strategic impact on the DP center and information processing throughout the organization.