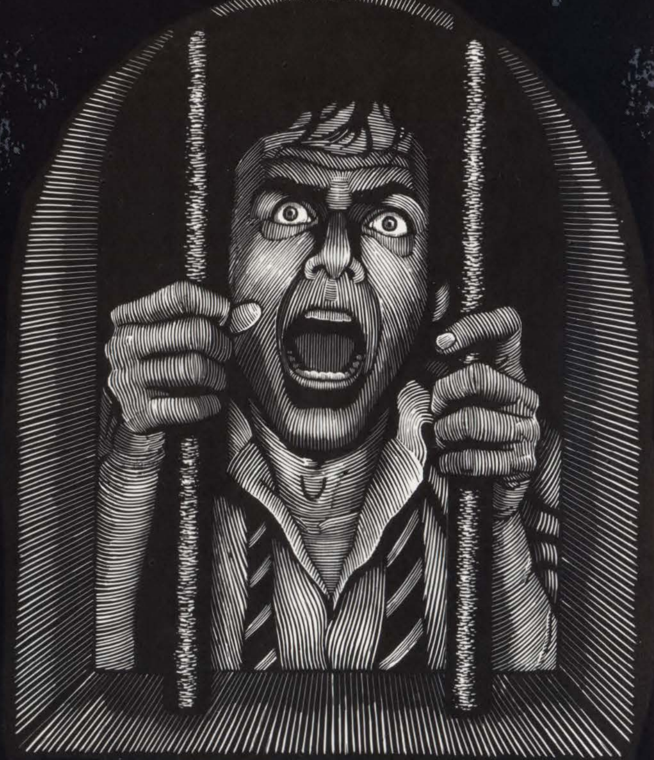
# MAINFRAME JOURNAL

For Users of IBM System/370 Architecture & Compatible Systems

September 1989

## Get Acquainted With The Network Control Program

# ESCAPE FROM THE DATA SHOP OF HORRORS.



Get your VM™ Data Center Under Control at the VM Software Seminar.

Tired of dueling with DASD demons? Stuck in the slime of security management? Up to your eyeballs in user abuse? You're not alone. There's a whole army of virtual vermin ready to apply cruel and unusual punishment to your entire VM data center—if you let them.

Yet there *is* a way out. And as thousands have discovered, it's surprisingly easy.

It's called the VM Software Seminar—and it's free. At this unique half-day event, you can learn first hand how to strengthen your control over VM operations while streamlining your workload. How to improve service to users while making the most of system resources. And how to extend these improvements across multiple environments through the sophisticated use of network data transfer technology.

## The seminar that puts you in control.

Each VM Software Seminar is packed with information on tools and techniques to maximize the value of VM operations with a minimum of data center personnel.

You'll see the comprehensive functionality of VMCENTER II,™ the world's leading VM systems management package with an unmatched record of reliable, cost-effective performance.

You'll also see a series of tools for improving performance throughout your SNA network. And you'll have a chance to exchange insights on a variety of subjects with other systems management professionals from your area.

It's the perfect seminar for anyone interested in improving VM operations. And the perfect escape from your daily dungeon to a brave new world of confidence and control. So don't delay—reserve today. And put the horrors behind you once and for all.

## Agenda

**8:30** Registration and coffee
**9:00** Seminar begins
**10:15** Break
**12:00** Free lunch

## Seminar dates and locations

- **Atlanta, GA** November 17
- **Boston, MA** October 12
- **Cherry Hill, NJ** October 31
- **Chicago, IL** October 6
- **Cleveland, OH** October 4
- **Dallas, TX** November 15
- **Denver, CO** November 7
- **Detroit, MI** November 14
- **Hasbrouck Heights, NJ** October 11
- **Hartford, CT** October 20
- **Houston, TX** November 9
- **Indianapolis, IN** October 5
- **Kansas City, MO** November 8
- **Long Beach, CA** October 25
- **Long Island, NY** October 19
- **Minneapolis, MN** October 3
- **New York, NY** October 10
- **Ottawa, ON** October 18
- **Raleigh, NC** November 1
- **San Francisco, CA** October 26
- **San Jose, CA** October 27
- **Seattle, WA** October 24
- **Tampa, FL** November 16
- **Toronto, ON** October 17
- **Washington, DC*** November 3

*Highlighting Federal Government Issues

**Reserve your place today, call**
**(800) 562·7100**
**(703) 264·8413**

SYSTEMS CENTER
FORMERLY VM SOFTWARE, INC.



A-MFJ-8909

CIRCLE #20 on Reader Service Card ▲


"Our network moves enormous loads quickly. So does our CICS."
"At CSX, we have 61 production CICS regions and seven million transactions per day. With The Monitor For CICS,™ we see problems long before our users do."
Rail transportation. Container shipping. Gas pipelines. Resorts. CSX is a $13 billion giant. With over 21,000 miles of rail, 6,000 miles of natural gas pipeline, and 5,000 miles of fiber optics, CSX needs real-time status to service its customers.
So at their Jacksonville, Florida, and Baltimore, Maryland, facilities, CSX uses CICS to track status and inventory — and relies on The Monitor For CICS to manage CICS performance. "The Supertrace feature lets us look inside an application and gauge its effects on system performance," says Jason Butler, Manager, Technical Services. "We can trace application logic and evaluate resource consumption right down to the event level."
"We've built a unique monitoring system that is PC-based and set up so that Monitor commands are automatically executed to identify poor response times. This lets me spend more time with features such as the storage display. Now when a problem arises in CICS, I can alter storage or delete ICE/AID chains rather than shutting down and cold starting the system."
The Monitor is the complete CICS performance management system that'll help you save the day. Become the hero in your CICS community! For a free, 30-day trial of The Monitor For CICS, call us today at 1-800-227-8911 or 1-703-893-9046.
LANDMARK™
THE MONITOR® FOR CICS
MINDS YOUR BUSINESS
Landmark Systems Corporation
8000 Towers Crescent Drive, Vienna, Virginia 22180-2700
CIRCLE #16 on Reader Service Card ▲

# CONTENTS

September 1989 • Volume IV • Number 9

## DEPARTMENTS




**COVER:**
*To "get acquainted" with the Network Control Program, an integral part of IBM's Systems Network Architecture,* *turn to page 26.* *Cover illustration from The Image Bank.*

*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


With CA-LIBRARIAN You Have As Much Change Control As He Does. Almost.
Our library management system has evolved to a higher plane of existence. CA-LIBRARIAN® is now the only comprehensive software package that combines library management and change control for all IBM operating environments.
You can now rely on CA-LIBRARIAN to manage the thousands of program changes taking place in your source library every year. You get on-line audit capability, automatic synchronization to ensure every change is valid within your system. Built-in security reinforcement. Plus, a complete record of code changes made on a day-to-day basis.
To date, more than 7,000 facilities have come to rely on CA-LIBRARIAN. Find out why by calling Jamie Brooks at 800-237-9273 (in N.J., 201-874-9000). And discover what it's like to have total control over your destiny.
Well, almost.
COMPUTER® ASSOCIATES
Software superior by design.
© 1989 Computer Associates International, Inc., Route 206 & Orchard Road, CN-8, Princeton, NJ 08543-0008
CIRCLE #105 on Reader Service Card ▲

## *CA Bashing: Is It Justified?*

*Bob Thomas*

When those in the audience at a recent IBM mainframe user group meeting were asked to share their experiences with CA-SCHEDULER from Computer Associates (CA), a loud chorus of boos, groans and snickers erupted immediately. The very same response greeted each and every mention of CA, its products and its support. A question about user satisfaction with CA-VOLLIE and CA-LIBRARIAN was met with, "I don't have CA's VOLLIE, I have ADR's VOLLIE."

Here at *MAINFRAME JOURNAL* we have edited out quite a few vicious attacks (in the name of journalistic fairness) from articles where the author took a circuitous route from the main thrust of the article just to blast CA. In fact, several writers have volunteered to do an exposé-type article on CA's "merger mania" and "lack of customer support."

At the same meeting, though, after all the hootin' and hollerin' died down, CA users one by one confessed that support was really pretty good, phone calls were getting returned in short order and products were being enhanced. And, by the way, the CA-SCHEDULER user presented a positive review of the product.

What the heck is going on here! Why is everybody picking on CA (other than investors, that is)? Is CA such an easy target because of the aggressive way it has gobbled up so many other software companies? Is all this CA bashing justified?

My take on the matter is that the flurry of takeover activity in such a short time frame has created some chaos at CA and its acquired companies. Support probably did diminish. Product enhancement did slow down and halt entirely in some cases. Good people did lose jobs in the name of efficiency and to eliminate duplication of function. But, judging from the responses of quite a few CA users, things are not nearly as bad as the press has portrayed. CA users are rallying to the support of their beleaguered vendor. Revenue from the sales of CA products continues to grow, so it must be doing something right. Is "CA bashing" justified? You be the judge. I invite CA users to send me your comments for a follow-up article.

## *Share Your Experiences For "Big Bucks"*

To excerpt from Ken Peterson's letter in Reader Forum (page 8), "One can only learn so much from manuals. Getting working professionals to write about their experiences helps all of us to share hard-earned information." There is a hunger for information among DP professionals in mainframe shops, especially the "Tips and Techniques" type of information. My goal is to tap into the tremendous resources available out there in "readership land" for articles that will help and enlighten others.

If you're doing something unique or interesting on your system, take time to write a brief description in article form and submit it for publication in *MAINFRAME JOURNAL*. We're not trying to win the Pulitzer Prize here so don't worry about *style; substance* is what we are after. Our editor, Carol Hoag, is an expert at "smoothing out the rough spots." For your efforts, not only will you receive the undying gratitude of the many you will unknowingly help and pats on the back at user conferences; but also, you will receive cash from *MAINFRAME JOURNAL* (no T-shirts, subscriptions or association memberships in lieu of real money, here). No, you probably won't be able to retire, but from what I can tell, our payment rates are among the highest.

*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
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.

## *Mainframe Programming On The PC*

It was interesting to contrast "Improving Mainframe COBOL/CICS Programmer Productivity" by Tiernan and Danker in the July 1989 issue with our experiences here at American Hotel Register. Like the authors, we considered the Realia COBOL compiler for the PC. We rejected it, interestingly, for the very reasons that Tiernan and Danker chose it — it acts like a mainframe compiler ported down to a PC. We wound up choosing Visual COBOL from MBP Software (Alameda, CA). This package, while fully mainframe-compatible, is a true PC platform with an amazing screen designer, an integrated, powerful debugger, superb documentation, no rough edges and very few "gotchas." We did not get any of the CICS options, since we run color CICS on 3179 terminals, which no PC vendors seem to support. OptTech Sort, which the authors purchased separately, is included automatically as part of the plain-vanilla Visual COBOL compiler. For editing, we use the venerable PC-Write from Quicksoft (Seattle, WA). While it is a fullword processor rather than a text editor, it works like a champ for straight programming.

As Tiernan and Danker pointed out, IBM's file-transfer package does not work with packed data. In addition, we could never get it to work with any signed data at all. Although the authors wrote their own applications programs to unpack and repack data, we purchased PC-Mainframe from CF Software (Des Plaines, IL). While this package is not as intuitive as Visual COBOL or PC-Write, it does work well, it is quite flexible and it allows our applications programmers to upload and download files quite easily without worrying about minutiae.

During the two years we have been using PCs for programming, they have steadily, gradually been gaining ground. New PC applications have been developed by our staff and some existing processes have been ported down to our network. Tiernan and Danker wrote that they were excited about the use of PCs in their mainframe shop. We absolutely agree.

*Jerry Ebner*
*American Hotel Register Co.*
*Northbrook, IL*

## *VSE/SP Versus VSE/AF VRM*

It may come as a surprise to many people to know that even if they are on VSE/SP 3.2, they are still running VSE/AF 2.1! Keep in mind that the SP stands for System Package. One of the components of the system is VSE Advanced Functions (VSE/AF). The Version Release Modification (VRM) of AF did not change between SP 2.1, 3.1 and 3.2, even though enhancements were made. The highest level is AF 2.1.7. At VSE/SP 4.1, IBM enhanced VSE/AF and labeled it Version 4. There never was a Version 3. More specifically, there are levels 4.1.0 and 4.1.1 that, coincidentally, match the levels of VSE/SP that contain them. I guess IBM was confused with the levels not matching before because I have found VSE/SP 3.1 manuals referring to VSE/AF 3.1, which never existed!

What makes the above more than just numbering trivia is the fact that if you are a VSE/SP 2.1.7 user and you apply PTF UD44067 (recently superseded by UD44188), you end up with the AF enhancements of a 3.2 system. Namely, you can use up to nine address spaces and 128MB virtual. This makes sense when you consider that the fix is for VSE/AF 2.1.7, so it doesn't matter whether you are on SP 2 or 3. This gives you the benefit of "Pete's Patch" along with the support of IBM since it is their code. This fix makes it very attractive to those users who want the AF benefits of SP 3.2 but don't immediately need all of the other component upgrades such as POWER, CICS, VTAM and so on.

Notes: UD43521 superseded UD42411 and UD43212
UD44067 superseded UD43521
UD44188 superseded UD44067

This PTF caused massive changes to much of the VSE system. It fixes many problems (329 APARS resolved by UD44188!) including looping systems, hardwaits, softwaits and GETVIS related. As mentioned above, it also adds support for nine address spaces and 128MB virtual.

If you are going to apply the IBM PTF and you use Goal Systems' FLEE/VSE 2.5x (Goal DCM-Systems), you *must* apply DCM-System/VSE Zaps 2.54-12 to avoid any library corruption problems. *(Editor's Note: Version 3 of Goal's DCM-Systems corrects the problem).*

*Dennis L. Bliss, CDP*
*Sr. Systems Programmer*
*SC Data Center*
*Monroe, WI*

## *Returning The Favor*

I would like to thank you for forwarding the VSAM time stamp decoding program I requested. I have received and modified it to fit our needs under VM/SP 6.0. To return the favor, I will make available on tape and/or print my copy of the source, update files, AUX files and all other files associated with this project. I have made modifications to allow the use of the program under CMS/DOS as well as directly under CMS as a module. You may have your readers contact me directly at the following address.

*Russell N. Hathhorn*
*Portland Community College*
*Portland, OR 97219*
*(503) 244-6111 ext. 4705*

## *Sharing Hard-Earned Information*

I find the articles in *MAINFRAME JOURNAL* on IBM software products and architecture to be of great value to me. One can only learn so much from manuals. Getting working professionals to write about their experiences helps all of us to share hard-earned information. Your articles on CICS and VSAM-related topics have been great! Please provide more articles on TP monitors and databases.

*Ken Peterson*
*D L Barney Software*
*Wilmington, DE*

## *CICS Conversion*

I recently converted file buffer processing in CICS from NSR to LSR. I used seven *MAINFRAME JOURNAL* articles as the basis for the conversion. The information the articles contained was technical, yet illustrated and discussed well enough that the subject was covered quite well. My thanks to *MAINFRAME JOURNAL* and the authors of those articles.

*Ray Pittman*
*Moore Business Forms*
*Denton, TX*

## VSE 15-Partition Support Now Available

A set of modifications increasing available VSE partition space by 25 percent was announced by John Baker, senior systems programmer with Policy Management Systems Corp., at the IBM GUIDE user group conference recently held in Toronto, Canada. The modifications, designed for VSE/AF, BTAM-ES and ACF/VTAM, let *non-ICCF* users execute as many as 15 concurrent jobs. According to Baker, ''This modification expands VSE operation workload capacity and gets us closer to fully maximizing data center productivity. For example, many shops now have to run multiple CICS partitions to accommodate workload. With 15-partition support, up to three additional CICS regions can be run without restricting the associated batch workload.'' These VSE system modifications have been tested by a half-dozen VSE system users including the distributor of the enhancement, Smartech Systems, Inc. ''In light of IBM's recent announcements about VSE system re-emphasis and the role it plays with the 9370 series, we anticipate a surge in VSE market growth,'' adds Eric L. Vaughan, President of Smartech Systems. Baker's enhancement makes an investment in VSE an even more attractive proposition. The VSE system has always been extremely cost-effective. Now, its increased workload capacity leverages current VSE investments and increases the possible user base.'' The enhancement is available at *no cost* from Smartech Systems. For information, contact Chris Lawrence at (800) 53-SMART or (214) 956-8324.

## IBM Peripheral Market Share Declines

IBM has lost market share in the DASD, printer and terminal/workstation arena since 1987. The largest decline was in the system printer market which dropped by seven percent. IBM also lost market share in PC usage among IBM/PCM mainframe sites by six percent. Increasing PC market share at IBM's expense was Compaq and Zenith. On the other hand, IBM's tape drive market share has experienced healthy increases over the last four years mainly due to the introduction of the 3480 tape drive. These downturns in IBM's market share reflect more than the cycle of IBM product introduction followed several years later by less expensive PCM compatible devices. These days, the PCM manufacturers are quickly matching IBM's products and in many cases offering more. *Source: Computer Intelligence (Karen Landis – IBM/PCM Mainframe Industry Analyst)*

## LEGENT Corporation to Merge With BST

LEGENT Corporation has announced it will merge with Business Software Technology (BST). LEGENT previously owned approximately 10 percent of BST. The value of the stock transaction is valued at $47 million. ''This acquisition, our first since the formation of LEGENT,'' said Chairman Mario Morino, ''is an ideal example of our expansion strategy.'' Formed in 1984, BST develops and markets the ENDEVOR family of products, which manage, control and improve the process of building, changing and operating software systems.

## Automated Data Center Seminar Series Scheduled

Starting in October, KPMG Peat Marwick will be providing a seminar series on ''Automating the Data Center.'' The seminars will be led by specialists from KPMG Peat Marwick who have extensive experience with planning and automating data center functions. Speakers will include information technology professionals from LEGENT Corporation, Goal Systems International, Business Software Technology, Nolan, Norton & Company and Unitech Systems. The seminar schedule is as follows: October 3-4, New York, NY; October 26-27, Boston, MA; December 4-5, Newport Beach, CA. Seminar tuition is $1250 and includes all learning materials, luncheons and receptions. For more information call the Executive Education registrar at (800) 762-3932.

## BBUG Is Back!

Fourteen years ago the Boole & Babbage User Group (BBUG) conference gave birth to the Computer Measurement Group (CMG). Today CMG is the premier conference dealing with computer measurement, performance evaluation and capacity planning. BBUG is returning with gusto October 15-18 in New Orleans. The theme of the conference is automated operations and performance management. For more information call BBUG '89 at (415) 451-4000.

## 1 Extended Flexibility

Authorize multiple modes of user access to the network tailored to your requirements. Also, benefit from standard features such as single signon, user customizable menus, scripting, conferencing, cut and paste, and online help.

## 2 Online Administration

Update user and application definitions dynamically with online panels. Plus, simplify administration and preserve security by defining application access privileges in rules.

## 3 High Performance

Benefit from the efficient movement of data between terminals and applications with the minimum path length, rapid execution using the SRB dispatching mode, and the top performance of the internal storage manager.

# Great Reasons To Make *New* NC-MultSess Your Session Manager

**To fully appreciate NC-MultSess, try it free for 30 days. Call us today at (800) 348-3523 or (412) 256-2900 and we'll send you the product.**

**Westinghouse**
Management Systems Software

# Strategic Planning For MIS: The Hidden Resources

By Robert J. Rennie, Ph.D.

The notion of information systems planning seems complex due to the rapidly changing technologies and business needs upon which such plans are based. Despite the inherent difficulties in developing them, the need for MIS plans is unquestionable. Organizations must view information systems and related technologies as significant corporate resources which can contribute directly toward improving the bottom line. These resources must then be developed, allocated and used with the same unity of purpose and attention to detail as other forms of business capital. This requires the use of strategic and tactical planning processes.

A popular concept at this time is the Rolling Plan for Technology. The term *rolling* refers to the fact that the original plan is not replaced each year, rather it is updated and extended one year at a time. This feature provides continuity from year to year. Such plans are typically designed to provide for three to five years.

The first 12 to 18 months of the plan are detailed and specific, enabling them to serve as an operational or tactical plan. This near term or tactical plan speaks to immediate operational requirements and pending priorities relating to hardware acquisition, operating systems installations and applications development projects. Often, the MIS service level agreements are interfaced at this level of the planning process.

Features of the tactical plan include clear descriptions of the work to be completed, assignments of responsibility, due dates, manpower requirements, costs and measures of quality. The overall fit with long-term plans and the immediate business objectives of the organization are also included in the plan.

The remainder of the plan is developed with a strategic orientation and is, thus, less specific. It provides a complete picture of the future vision of the organization's technological position. The longer term or strategic plan is designed to specify concepts and architectures which will be implemented to support the attainment of the organization's long-range objectives and goals. Features include hardware designs, software development methodologies and tools, systems and communications architectures and philosophical positions on MIS issues, as well as business-based forecasts of requirements.

Although most large MIS organizations have the on-board talent and vendor support necessary to develop quality MIS plans, the products of these in-house experts are often viewed suspiciously by top management. There is some truth to the old adage about an expert being someone who is 50 miles from home with a briefcase.

Upper management concerns are often based in fears of inbred thinking and the natural homeostatic tendencies of groups. That is, the group norm is to perpetuate the status quo, rather than to avidly pursue improvement.

A supposedly unbiased, ''fresh'' view is what top level managers are buying when they hire outside consultants to review or assist in the development of MIS plans. Generally, consultants offering planning services are quite good. Unfortunately, such services are usually expensive, purchased at the expense of other IS projects and often leaving in-house talent feeling undervalued and unappreciated.

There is an alternative that can produce exceptional results. High quality plans which are consistent with the organization's goals and objectives and meet the ''fresh eye'' criteria of top management are achievable through the use of Quick Response Teams (QRTs).

# Remote printing on a PC without a single compromise.

New 3270 & Token Ring Support


Until now you've had to rely on a S/36 or AS/400® to deliver remote printing. Now a PC with BARR software and adapter sustains print speeds of 6,000 lines-per-minute and line speeds of 128,000 bits-per-second. Only BARR maintains all this performance with the reliability and ease of use PC users expect.

BARR RJE software drives up to five printers from a single PC. What's more, you can enter data, print, and receive output all simultaneously with out interruption. BARR's advanced multi-tasking software easily manages even the most complicated tasks, including LAN access, tape support, file transfer, and special forms printing. In addition, BARR offers one year of friendly, dedicated customer support with each purchase.


Communications adapters and software are available for the IBM PC, PS/2, and compatibles.

Try BARR for 30 days. We've helped thousands save millions. Call 800-BARR-SYS.

| Protocols | | |
|---|---|---|
| SNA RJE – SDLC or Token Ring | | |
| RJE + 3270 | HASP | 3780 |

| Host Systems | | |
|---|---|---|
| JES2 | DOS/POWER | CDC NOS |
| JES3 | RSCS | VS1/RES |

BARR

BARR SYSTEMS INC.,
4131 NW 28 Lane, Gainesville, FL 32606
800-BARR-SYS, 904-371-3050, FAX: 904-371-3018



*CIRCLE #91 on Reader Service Card ▲*

The QRT is a hybrid with roots in the task, special ad hoc and professional network consulting arrangements with which most of us have had our share of experiences. The QRT is assembled for the express purpose of supporting the development of the MIS plan.

In general, the concept is to recruit from five to 10 IS professionals with whom to network on the development of IS plans. These members should not be industry specific; in fact, greater diversity of experience among group members increases the value of the group. Ideally, the team would be composed of the top IS person in each of the participating organizations.

Quick response is used to describe the team because the meetings need to be assembled on fairly short notice as critical issues and concerns are dealt with in the planning process. Regularly scheduled meetings fail to provide the immediate feedback for which these teams are developed and demand too much time from team members.

The recommended structure of the process includes the following six distinct steps:

1. *Correspond* and meet with potential QRT members
2. Hold *brief kick-off meetings* with QRT, describe protocols and sort out non-disclosure/non-compete agreement issues
3. Provide team members with *background information and an agenda via mail* with a communication outlining the final product expected of the group
4. Hold a *formal full-day session* with the QRT including:
   - Background/introductory presentations
   - Presentation of draft IS plan
   - QRT work time (provide clerical support but leave the team unattended to avoid bias)
   - Conceptual draft report of findings and QRT work assignments for final product
5. Confirmation/proof and *publication of final plan and the QRT report*
6. *Formal presentation* to corporate management with representative(s) of the QRT presenting.

The in-house staff and the QRT will most likely have differing opinions concerning certain areas. This situation is best resolved by formalizing the final plan according to the views of the in-house staff and having the QRT express opinions on these issues in team members' own reports.

The final products from this effort should include an MIS plan that has been developed by the in-house experts with the benefit of advice and commentary from the QRT and a separate QRT report that includes an analysis of the plan, an analysis of the process used for the development of the plan, a general critique of the current operation and a quick description of the QRT, its members and the QRT process.

Not all of the benefits are for the organization under study. The QRT members also benefit from exposure to the rest of the team and the collective thinking of the group. Additionally, once the team is established, the process can be refined and applied to the other members' organizations.

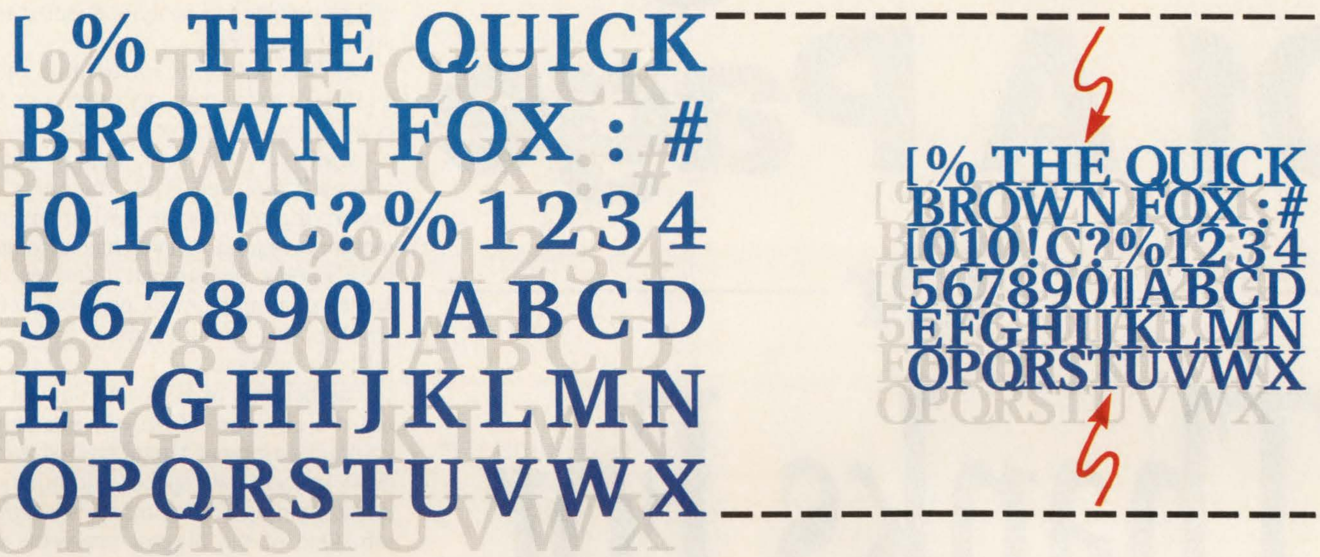
Perhaps the greatest hidden resource in strategic planning for MIS is the network of expert professionals who can provide significant input through a well orchestrated peer review process. The QRT process taps that resource for the benefit of all concerned. QRTs are significantly cheaper and can prove to be more realistic than typical consultants.

QRTs are a cost-effective way to develop a comprehensive MIS plan for meeting the needs of your organization while providing your corporate management the benefits of new ideas and a fresh perspective of technologies for IS in your business. This is the essence of strategic planning.

*ABOUT THE AUTHOR*

*Robert J. Rennie, Ph.D., is Director of Data Processing for the Brevard County School District (Kennedy Space Center Area) in Florida and a Principal of XENTREK Systems, Inc., an MIS planning and management consulting firm. Dr. Rennie has held various senior level MIS positions over the past 14 years. He is a frequent author and speaker on the management of technology. Brevard County School District, 3205 S. Washington Ave., Titusville, FL 32780, (407) 631-1911.*

DATA COMPRESSION FOR DB2




# With INFOPAK, this simple concept is finally simple to use

Using data compression to improve DASD storage efficiency is a technique that has traditionally been simple in concept but difficult in execution. INFOPAK from InfoTel Corporation has changed all that.

With INFOPAK you can recover as much as 70% of the space on your present disks. You can do it without compression tables and without the data analysis required to generate them. You can do it without interfering with application programs, since INFOPAK is totally transparent. And you can do it whether you're running DB2, VSAM, IMS, IDMS, or DATACOM/DB.®

INFOPAK achieves its initial compression using regular load and unload utilities. No modifications to existing programs or JCL are required and INFOPAK may be installed by your database administrator in a few easy steps.

In addition to improved DASD utilization, INFOPAK delivers important performance benefits. As significant compression yields significant I/O reductions, you will see marked improvements in response time and throughput.

Get a realistic look at the improvements INFOPAK can make in your DASD space utilization. Ask for a free TESTPAK program. Run it against your existing data bases to determine the DASD space gains you can expect.

Contact InfoTel today!

## TEST IT FOR YOURSELF!

**Ask about our free TESTPAK program! Evaluate the data compression and performance advantages of INFOPAK against *your own* data files.**

InfoTel

**InfoTel Corporation**
14906 Winding Creek Court
Suite 101D
Tampa, FL 33613
Phone 800/543-1982 or
813/264-2090
FAX 813/960-5345

*CIRCLE #186 on Reader Service Card* ▲

*MVS Performance Notebook*

# The Age Of A Page Or Thanks For The Memory

By Mark Friedman

Memory is back! An integral part of IBM's aggressive promotion of the benefits of the 3090 processor and MVS/ESA is memory in all its forms: real memory, central and expanded storage, virtual memory and auxiliary storage. The performance benefits of the MVS/ESA architecture involving a new type of memory called a data space (as opposed to an address space) are touted as *data in memory*. IBM has also announced cuts in the price of both central and expanded memory to spur 3090 purchases and upgrades.

All of this leads me to conclude that memory is back. Not that it ever really went anywhere, but since main memory is probably the least exciting and most misunderstood component of your mainframe system, it probably just seems that way. First of all, memory is passive — it does not spin or execute. Second, techniques for measuring memory utilization are also poorly implemented in MVS. Both contribute to the fact that memory concerns are often ignored.

But now, memory *is* back. MVS/ESA in particular will exploit increasingly larger memories, buffering frequently-accessed data stored permanently on slow speed I/O devices in high-speed electronic storage. This is an absolutely essential step to maintain performance as processors continue to get larger and faster.

Trendiness aside, memory management has always been an intrinsically interesting performance topic. Memory is shared among all active users of the processor and thus is subject to the same kinds of contention as other shared resources. However, memory resource allocation decisions tend to be made so frequently that precise performance data is prohibitively expensive to obtain. Without precise information, memory performance is difficult to predict and analytic queuing models are not much help.

MVS is notorious for treating memory badly. From a capacity planning standpoint, MVS treats memory like a sponge — whatever you have, that is what gets allocated and used. This, of course, is the underlying philosophy behind any virtual memory system. MVS has also always been poorly instrumented for measuring and evaluating memory performance. The result is that sizing memory in MVS is a difficult problem.

The benefits of ESA and data-in-memory are obvious. However, managing an ESA environment supercharged with address spaces, data spaces and hiperspaces all competing for memory is a complex job. One of the biggest problems is that a single memory-intensive application can affect all users in the system.

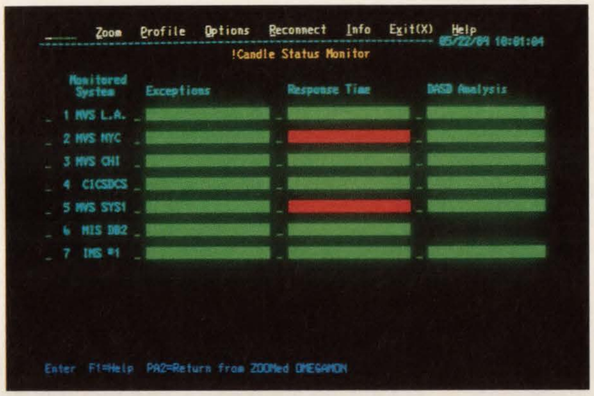
Prioritization schemes exist in MVS to protect shared resources from domination by a single, unfavored user and memory is no exception. There is a simple prioritization scheme for active users (known as storage isolation) and a more complicated one for temporarily inactive users (known as logical swapping). Both storage isolation and logical swapping can be difficult to implement because the appropriate measurement data is lacking. Implementing storage isolation, for instance, without a thorough understanding of both MVS and the workload often does more harm than good.

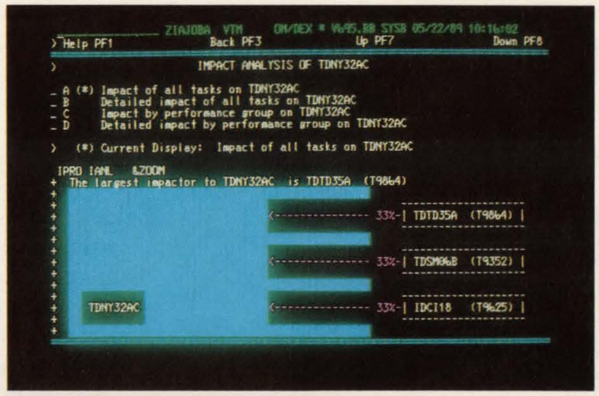
## MVS Memory Performance

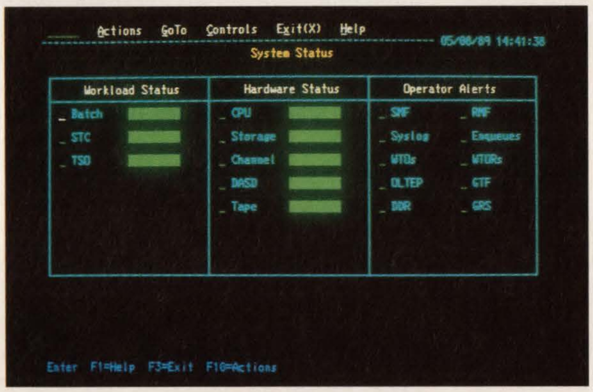
The characteristic of virtual memory systems is that "they do not degrade gracefully," as one wag in an IBM think tank put it recently. Indeed. A real memory bottleneck leads to paging and paging has well known performance ramifications. A performance analyst's nightmare is that memory performance is acceptable up to the point where the bottleneck is manifest and then system performance degrades rapidly.

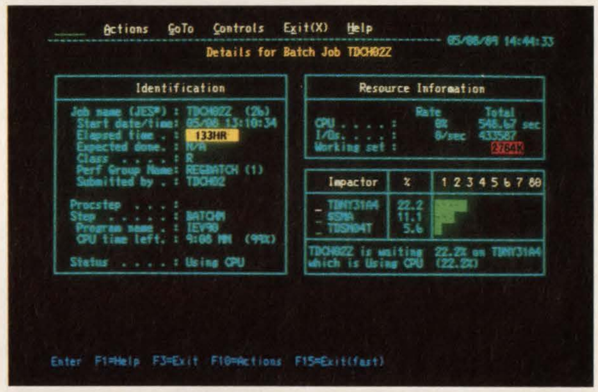
A basic lack of performance data contributes to the problem. Nevertheless, I find that you can *improvise* a bit and come up with suitable memory performance indicators to solve most tuning and capacity planning problems. And, if you are clever

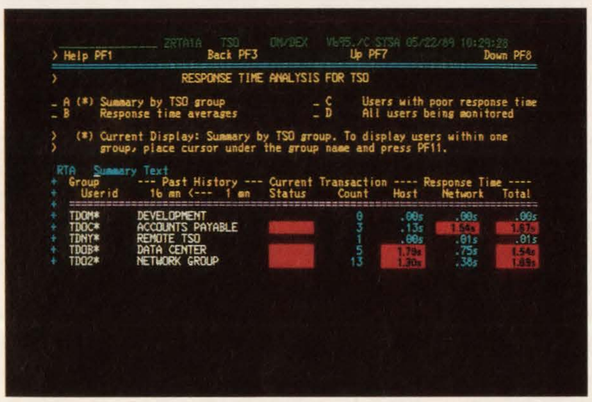
# The first word in performance management is still the last.

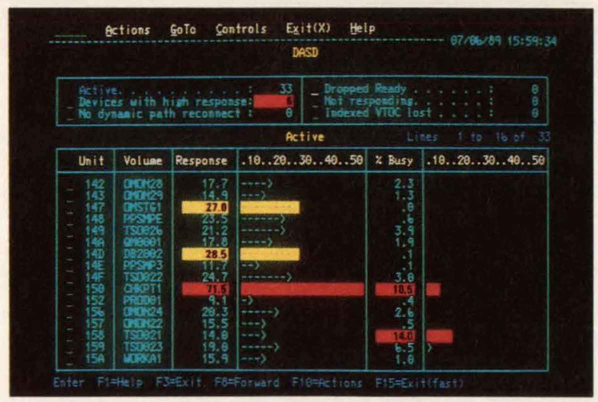



# OMEGAMON®.

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 ▲***

and understand how MVS manages memory, there are valuable performance measures that you can acquire shedding additional light on memory performance and capacity problems.

A good place to start a survey of memory performance is with the basic design goals of a virtual memory management operating system, looking at MVS in particular. In MVS, page stealing is implemented by evaluating the relative age of pages in the system, which is represented by a value called the Unreferenced Interval Count (UIC). This article concentrates on the UIC — how it is measured, what it is used for and what it means to you.

## Virtual Memory Concepts

A virtual memory system is a hierarchically-managed memory system consisting of a primary memory store (main memory), an auxiliary memory store (virtual memory) and a mapping scheme from virtual to real addresses. An executing program has full access to the range of virtual addresses supported without regard to real memory constraints.

The mechanisms that establish a virtual addressing environment are straightforward and familiar. The program's virtual address space is divided into segments and pages. Tables are constructed which show the correspondence between virtual addresses and real memory frames. During program execution virtual addresses are translated into real addresses.

Whenever a virtual address is accessed in an executing program that does not have a counterpart in real memory, a *page fault* occurs. The operating system intercepts the access, interrupting program execution until the address is resolvable and execution can resume. The page containing the virtual address needed by the program must be accessed from auxiliary storage and copied into main memory before program execution can resume. During page fault resolution, the program is delayed.

### *Partitioned Memory Schemes*

Virtual memory management schemes were developed to provide cost-effective utilization of main memory in a multiprogramming environment. In the first multiprogramming environments, such as OS/MFT and OS/MVT, main memory was carved into fixed-size memory partitions. As jobs entered the system, they were assigned to a real memory partition based on the REGION= parameter on the JOB card. An executing job tied up its memory partition until it completed.

Main Storage Occupancy (MSO) as a measure of memory usage makes sense in a partitioned real memory operating system such as OS/MFT. It can be used in job accounting and chargeback to account for memory usage because each time a job is run it will request the same region size.

Because MSO was used in job accounting in OS/MFT and OS/MVT, it was carried over to MVS. In a virtual storage operating system, a program uses only the amount of real storage that it requires, so region size is relatively meaningless. Since real memory is allocated on demand and programs compete for memory allocations, depending on the mix of jobs in the system, a program will use different amounts of real storage each time it runs. Consequently, MSO is not a repeatable measure of resource consumption that can be used in job accounting in MVS.

The basic problem with real memory partitions is that they are static and waste memory. Once a program is running in a main memory partition or region, it occupies that region for the duration of its run. However, the program may not need all the memory in the region. Given a set of jobs, devising a single fixed partition scheme that is optimal is an impossible job. The obvious solution of allowing dynamic partitions turns out to be even more impractical than a fixed scheme.

I hope there are no readers running OS/MFT or OS/MVT (you never know), but sites with PR/SM, for instance, encounter the same type of problems with fixed real memory partitioning schemes. PR/SM allocates real memory just like OS/MFT. It is a form of multiprogramming with a fixed number of tasks, except that in this case, the tasks are different copies of the operating system.

When you define a PR/SM partition, you allocate a fixed amount of both central and expanded storage to the partition. The storage is dedicated to the partition. There is no way, for instance, that you can dynamically reassign some lightly-used memory from one partition to another one that is going through a period of memory constraint.

That is the beauty of a virtual memory management system. Because memory is allocated on demand in small, fixed-size chunks, it is unnecessary to make these kinds of irrevocable resource allocation decisions up front. Since the real memory demands of a program or set of programs (including, perhaps, the operating system) are themselves dynamic, the dynamic management technique is much more efficient. That is why, other things being equal (which, of course, they are not), comparing a PR/SM configuration to a comparable VM configuration with guest operating systems, the PR/SM configuration will require substantially more real memory.

Before the Amdahl people have time to say, "What about MDF?" I should mention that a recent version of MDF supports a form of dynamic memory reconfiguration. Memory dedicated to one domain can be quiesced, varied off-line and added to a different domain without taking down MDF. A similar enhancement was made to OS/MFT to create OS/MVT.

Dynamic reconfiguration suffers from the fact that during some period of time while the memory is being quiesced in anticipation of being moved, it does not belong to anybody. Configuration flexibility is enhanced, true, but overall throughput still suffers. Periodic dynamic reconfiguration is still not as efficient as complete dynamic on-demand allocation.

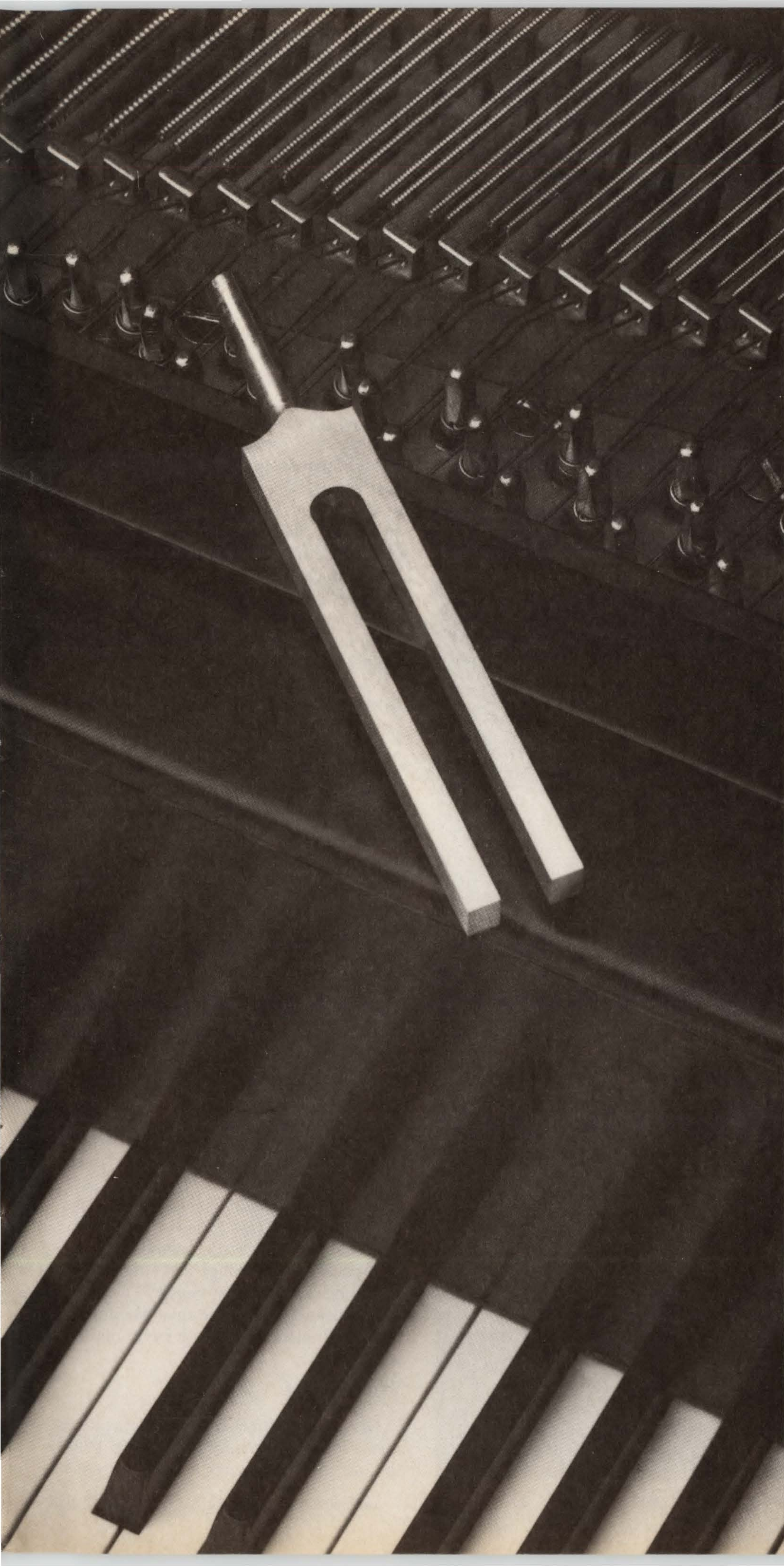
So my advice to Amdahl customers is that the additional flexibility of MDF is nice to have, but you should still try to buy plenty of extra memory.

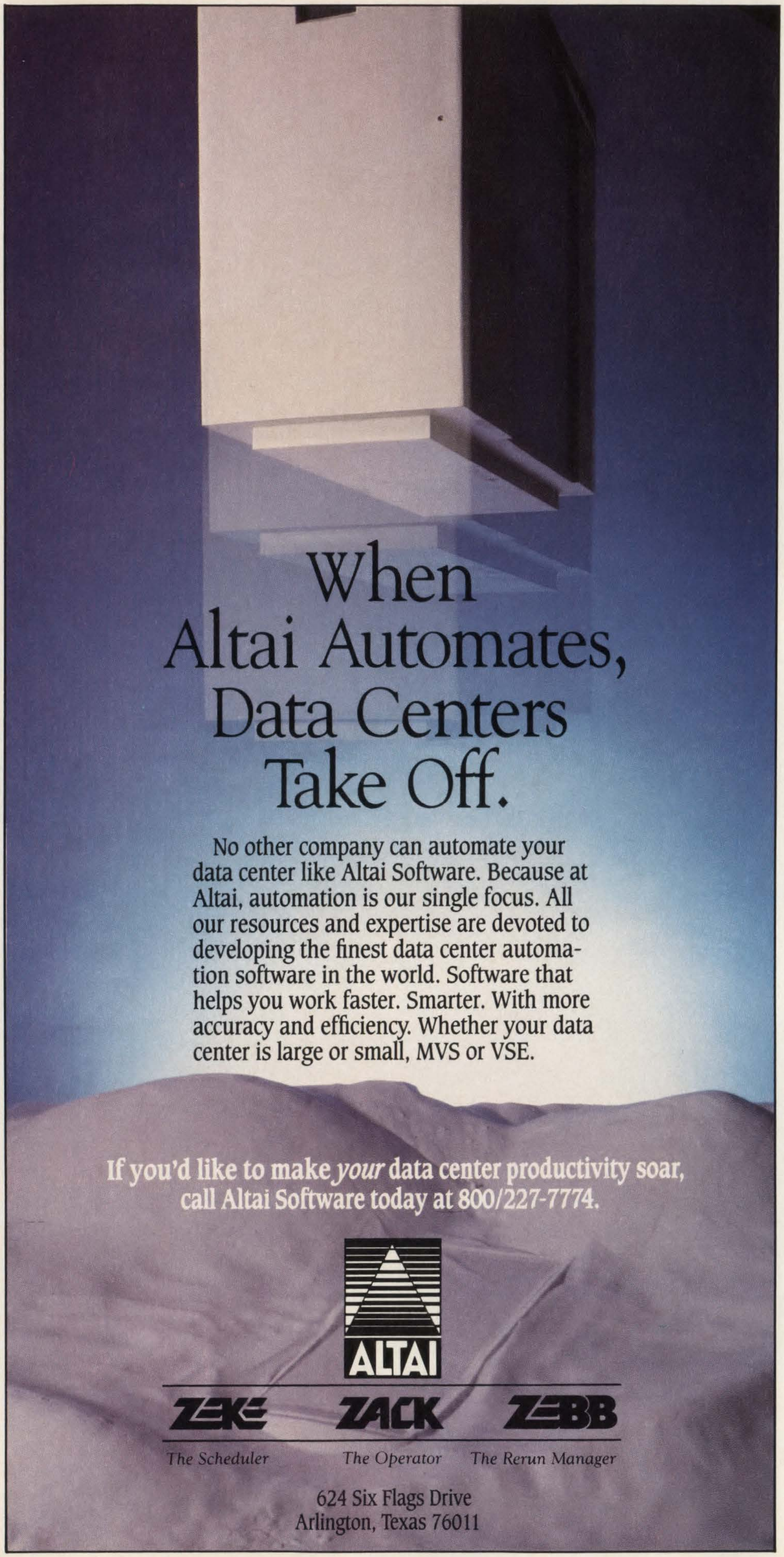
### *Page Size*

Page size is an important performance parameter in a virtual memory system. Normally, the value chosen represents a tradeoff between the speed with which pages can be brought into central storage and the costs associated with over-allocating main storage. The point on over-allocation needs further elaboration. It was just pointed out that virtual systems are efficient because they do not over-allocate storage.

However, remember that a reference to a virtual storage address that is not currently backed by central storage triggers a page fault, which then causes the entire page that contains the needed address to be brought into real memory. If, for instance, only the one address is needed, there is a considerable over-allocation of main memory. But it is too inefficient to have to return to auxiliary storage each time a new instruction or data area is accessed. So a compromise is reached.

The 4096 byte page size in MVS is a compromise established in the mid-70s when the premier paging devices were fixed head drums and disks with track sizes of under 20K bytes. In the early '80s,








IBM effectively established the moveable head 3380 as the premier paging device by increasing the page size for bulk paging operations, such as swapping, by a factor of twelve. IBM's own studies indicate that moving twelve-page *swap sets* back and forth results in a 30 to 40 percent over allocation of main memory compared to 4K page operations. Over isolation of TSO period one and two working sets often results in even higher degrees of wasteful over-allocation of main memory.

With the availability of expanded storage on the 3090 processor, a bus-attached, high-speed, semiconductor memory device, 3380 bulk paging (especially in the form practiced under extended swapping), has been largely abandoned. Actually, expanded storage also has performance characteristics which favor bulk paging operations that VM is currently taking advantage of, while MVS is pursuing a return to a more classic, demand paging approach.

Virtual memory addressing schemes work well because in actual practice programs typically use only a fraction of their full addressing capability at any time. Program segments devoted to error and exception handling, for instance, or initialization and termination do not need to be permanently resident at all times in order for the program to execute efficiently.

Studies of memory addressing patterns show that there is a high degree of *locality* in address references in many programs, a tendency for memory references to be clustered together in both time and place. There is also evidence that many programs are subject to *phases* in which address references are localized and there are *phase shifts* in which the address reference pattern may change drastically.

It is, of course, important that the benefits of a dynamic memory management scheme are not canceled out by the additional overhead compared to a fixed partitioning scheme. The overhead of dynamic address translation is minimized by moving address translation to the hardware. Today, processor technology that incorporates pipelining and translation lookaside buffers for on-board CPU cache has reduced much of the tangible impact of virtual memory management on the processor.

## The Age Of A Page

One practical problem in implementing virtual memory management results from the fact that real memory resources must be made available on demand. A dynamic policy of memory allocation on demand virtually ensures that memory will eventually saturate and all real memory frames will be allocated. Whenever the memory store is fully allocated, an address reference that causes a page fault requires that the new page requested will replace a currently allocated page in memory.

So one problem in virtual memory systems is determining which page to replace. Optimally, one seeks to replace a page in memory that will not be needed again soon. Of course, the operating system has no way of telling which pages will be referenced again soon. The best you can do appears to be to use information on past behavior to attempt to predict the future. A page that has not been referenced in a long time (an old page) is generally unlikely to be referenced again soon and is a candidate for page replacement by a new page. This page replacement policy is known as LRU, which stands for Least Recently Used. The LRU pages are the first candidates chosen for page replacement.

Parenthetically, even when there is information on usage patterns available to make informed decisions about, for instance, page-fixing, memory utilization patterns are apt to be dynamic enough that a static allocation method will be out-performed by a dynamic allocation algorithm enough times that it will not be worth the time and effort (and maintenance) to pursue. At a certain level of complexity, it is simply easier and more effective to let LRU manage memory dynamically than attempt to do it all yourself.

Determining which relatively inactive pages to remove from active storage in favor of a newer page requires historical usage data on main memory accesses. Typically, the available usage statistics are crude. In MVS there is a hardware reference bit that is automatically turned on whenever a location in a page frame is referenced. There is also a change bit that gets turned on whenever data is stored into a page frame.

### *UIC Update*

MVS maintains a Page Frame Table (PFT) with an entry (the PFTE) for each real frame in the system. A one-byte field in the PFTE called the Unreferenced Interval Count (UIC) is used to keep track of the age of the pages in central storage. Once an interval, SRM runs the PFT and maintains the UIC values for all the pages in the system. The UIC update routine executed by SRM is shown in Table 1.

In the UIC update routine, SRM examines the hardware reference bit for the corresponding page frame. If the bit is turned on, it means the page has been referenced in the last interval. The bit is reset to zero and the UIC value for the page is set to zero.

If the reference bit is off, it means that the page has not been referenced in the last interval. The current value in the UIC field is incremented by one, up to a maximum of 255 — the maximum value for a one-byte counter field. Thus, the UIC value represents the time interval over which the page was unreferenced. It is the age of the page.

As it updates the UIC values for all the pages in the system, SRM remembers the age of the oldest page in the system and stores it in the Resource Control Table (RCT). This value is the system high UIC,

**TABLE 1**

**The UIC Update Algorithm**

```
BEGIN:
current-interval-timestamp = TIME ();
interval-in-seconds =
        SECONDS (previous-interval-tamestamp) —
        SECONDS (current-interval-timestamp);
previous-interval-timestamp =
        current-interval-timestamp;
oldest-page = 0;
DO i = 1 to #-of-frames;
        IF pfte-allocated(i) THEN DO;
                IF reference-bit(i) THEN DO;
                        reference-bit(i) = FALSE;
                        pfte-uic(i) = 0;
                ENDDO;
                ELSE DO;
                        IF pfte-uic(i) < max-uic THEN
                                pfte-uic(i) =
                                        pfte-uic(i) + interval-in-seconds;
                        IF pfte-uic(i) > oldest-page THEN
                                oldest-page = pfte-uic(i);
                ENDDO;
        ENDDO;
ENDDO;
WAIT (interval-in-seconds);
GOTO BEGIN;
```

the age of the oldest page in the system, and it is widely used and reported as an indicator of main memory contention.

The UIC update routine incorporates one final wrinkle to make it more efficient. Under normal circumstances the UIC update routine runs once every second. However, if the system high UIC indicates that the system is not memory constrained, there are old pages in the system. It is probably not that important to discriminate on a 3090 processor executing 10 to 100 million instructions per second between a page that has not been referenced in 100 seconds and one in 101 seconds. Both are old pages that are candidates for replacement.

Consequently, SRM varies the rate with which it runs the UIC update routine according to the system high UIC from the last update run. While the exact details have not been published, it appears that the UIC update routine is still run once a second when the system is memory constrained, but it runs as little as once every 20 seconds when the system is relatively unconstrained (system high UIC > 60).

From the algorithm in Table 1, you can see that the UIC update routine scans the PFT sequentially. The time it takes to run the UIC update routine is proportional to the number of memory frames. The larger memory is, the longer the UIC routine takes to run.

### *Large System Effect*

IBM describes any operating system routines that run sequentially so that the time they take varies directly in proportion to the size of the machine as having a *large system effect* on these routines. The GETMAIN and FREEMAIN memory management routines in MVS/370 scanned sequentially chains of control blocks describing allocated and free storage. The larger the system was, the more storage blocks would be created and the longer these routines would take to run. You might say these routines suffered from a large system effect. Rewriting GETMAIN and FREEMAIN so that they did not have to scan control block chains sequentially was required for MVS/XA to support larger systems.

The UIC update routine suffers from a large system effect. Since it does not have to be executed more often than once a second, even on the largest processors, IBM has been able to avoid addressing the performance implications of the UIC update routine. And when there is a lot of memory and, by implication, less memory contention, SRM slows the frequency that the routine is run to save cycles.

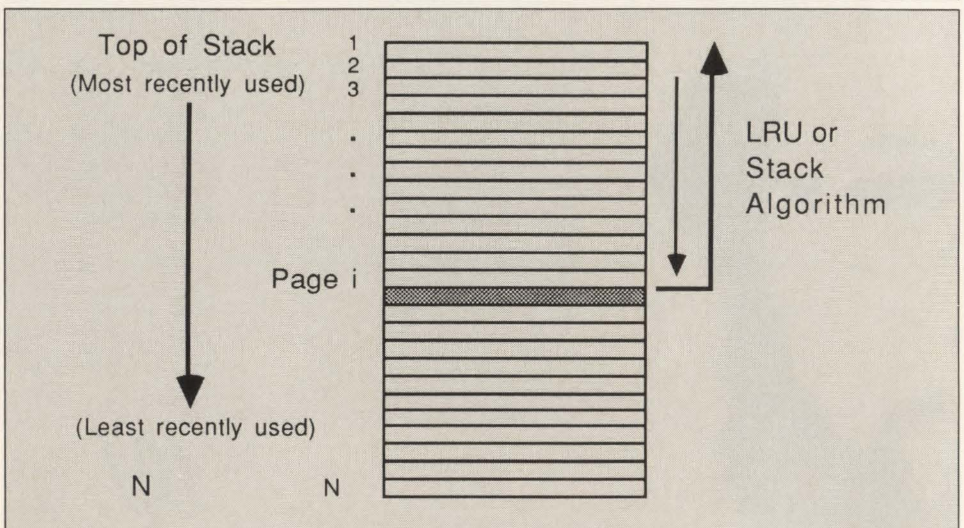
Unfortunately, SRM is not the only application that scans the PFT. RMF derives memory allocation and usage statistics from the PFT. The RMF measurement routine, which is used by both Monitor I and Monitor II reporting programs, scans the PFT. Monitor II executes a PFT scan on demand, Monitor I only once a sampling interval. Did you ever notice that the number of samples on the reports that use memory and paging data from the RMF type 71 record is much less than the number of samples on other reports for the same interval? Did you ever wonder why?

After the UIC update routine runs, the age of each page in the system is known. Notice, however, that almost immediately after the routine runs, the UIC information is no longer absolutely current. If a page with a UIC > 0 is referenced, its UIC is not reset to zero until the next time the update routine is scheduled.

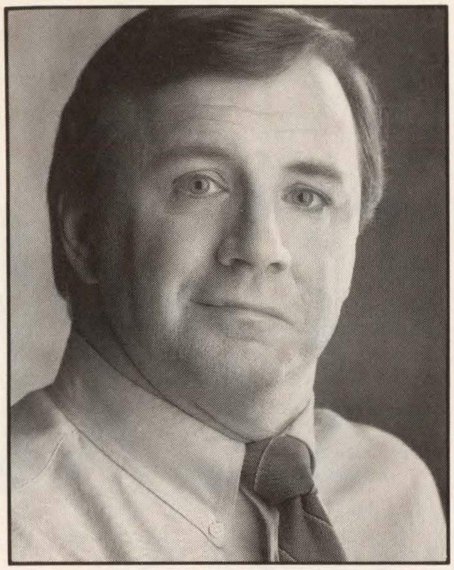
Notice also that the UIC mechanism does not distinguish the age of a page any finer than the interval between UIC update runs. Discriminations finer than one

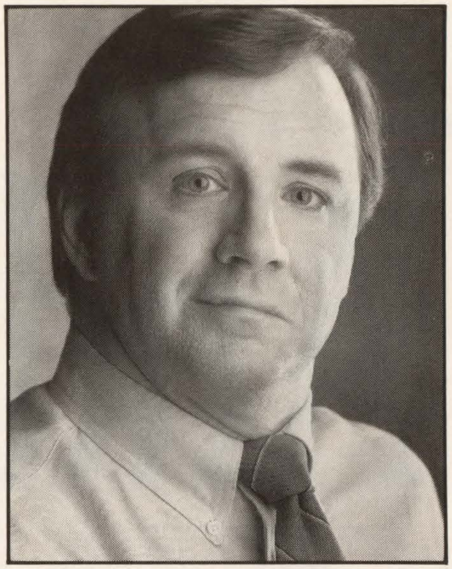
*See* Age Of A Page *page 102*

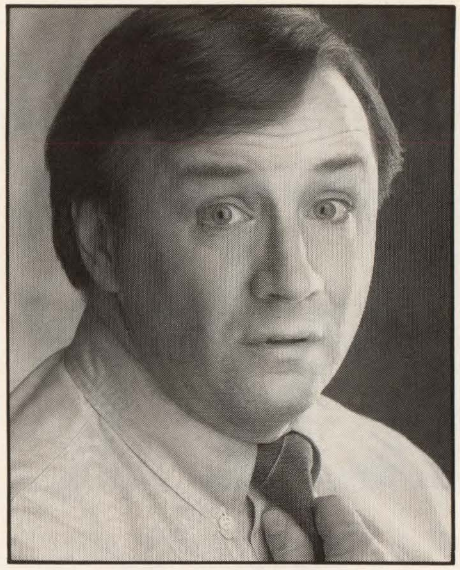
**FIGURE 1**




*A reference to Page i, moves Page i to Top of Stack and ages all pages 1 to i-1 by one.*







**You know your CICS from your IMS…**

**and your VTAM from your DASD.**

**But how about your CAR-RT SORT from your ZIP+4?**

**We make it EZ as ABC.** When it comes to running a DP center, it's a jumble out there. The last thing you need to worry about is managing huge mailings for your company—*and* doing them inexpensively.

Fortunately, there is a solution. Group 1 Software® gives you the power and flexibility to address all mail-related demands through comprehensive software and services—from list management and postal discounts to demographics and personalization. We even offer a variety of consulting services from our expert staff of mailing professionals who will assist you in your data center.

Group 1 is the industry leader in postal and list management software, offering *total solutions* for high-volume mailing needs, plus an unmatched command of ever-changing postal regulations in the U.S. and Canada.

We're so sure Group 1 Software will save you time and money, we'll perform a free benchmark test using *your* mailing list. Just return the attached coupon for more information. We'll also send you a free 42-page publication on business mailing preparation.

Stop losing Z's over high-volume mailings. Call or write Group 1 Software today.

**1-800-368-5806**

**Yes! I want to take charge of my mailings—PDQ!**
Please send me more information about Group 1 Software.
6404 Ivy Lane, Suite 500, Greenbelt, MD 20770-1400

Name Title

Company

Address

City State ZIP

( )
Phone

GROUP1®
GROUP 1 SOFTWARE
The Direct Marketing Software Company

MJ99

*CIRCLE #161 on Reader Service Card* ▲

*ISPF Tips And Techniques*

# ISPF And Text

## *Finding It, Changing It And Formatting It – Fast*

By Jon E. Pearkins

Text formatting is, according to our readers, a job for which the ISPF Editor is well suited. This article briefly presents ISPF's text formatting capabilities, as well as a summary of relevant line commands (see Figure 1).

**FIGURE 1**

**Text Formatting Command Tips**

- **TE** — Full-screen power typing
- **TF** — Format paragraphs to even length lines
- **TS** — Split a line at the cursor
- **UC** — Capitalize lower-case letters
- **LC** — Capital letters become lower case

### FIND ALL

There are two tricks that can help you deal with text when you want to do a global search or a global change. The ISPF Editor is a full-screen editor and cannot operate in any other manner. Most other editors also have a *line mode,* typically a relic from the days of terminals that used paper instead of a screen. The actual commands may vary, but each provides an equivalent of the FIND *string* ALL command that displays every line containing the specified string through the entire file.

The ISPF Editor Primary Command FIND ALL displays a screenful of lines beginning with the first occurrence of the string and prints a count of the number of occurrences of the string through the entire file in the upper right corner of the screen. Remember, this count is occurrences, not lines, so it would add three to the count if it found three occurrences of the string in a single line.

However, what about the original problem: displaying all the lines in the file that contain the specified string? David Levine, data administration analyst for Sony Corporation of America in Park Ridge, NJ, explains, "There is a convenient way to find all occurrences of a string in a dataset without having to PF5 (re-find) your afternoon away. Enter EX ALL on the Command Line. This will cause all lines of the dataset to be 'X'ed out'. Then, on the Command Line, enter F string ALL and all occurrences of the string will be displayed with intermixed 'X'ed out' lines. Then just PF8 (down) your way through the dataset if more than a screenful are found. To return the dataset to normal, enter RESET on the Command Line."

### *Formatting Text*

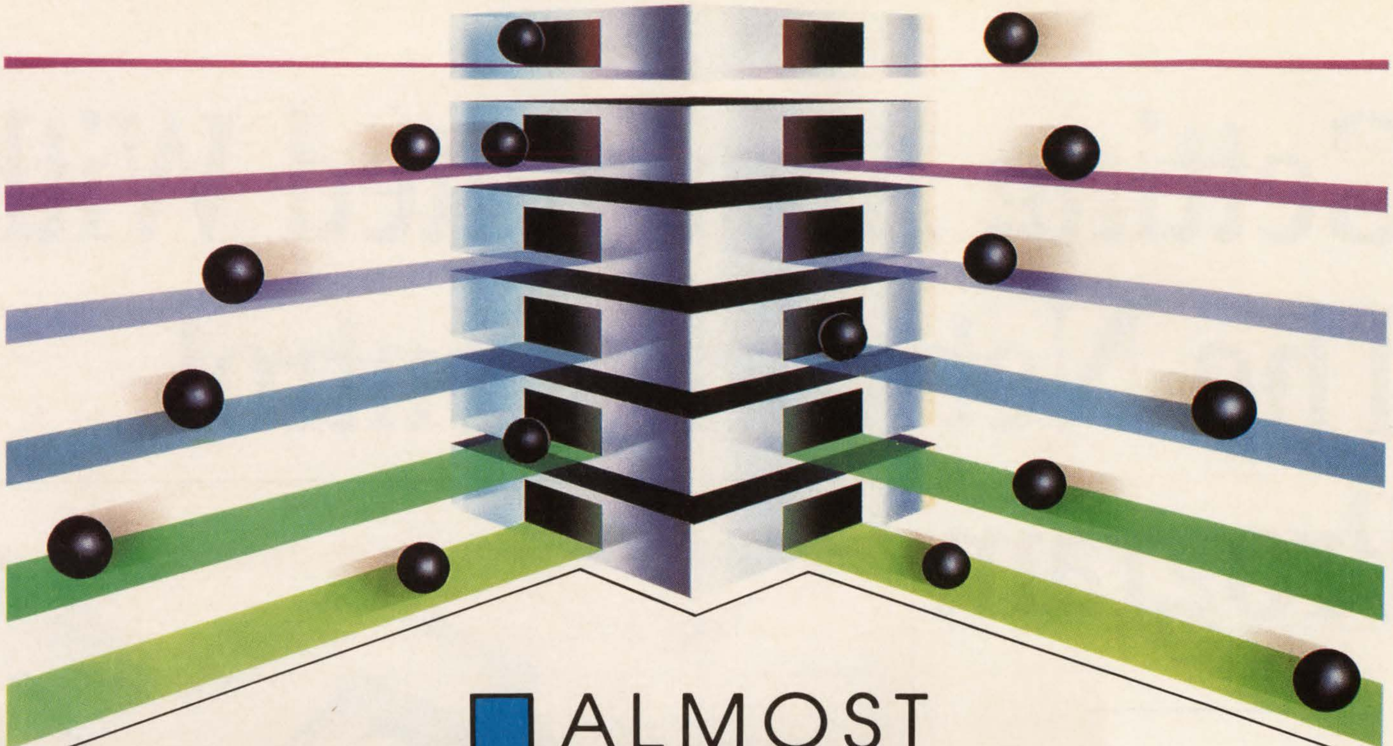
In the absence of a word processor, the ISPF Editor can be a productive tool for getting text into a readable form. The T Line Commands are your main tools (see Figure 1).

Text Enter (TE) entered on the first line of the screen will give you a full-screen on which to type, without having to worry about where one line ends and the next begins. IBM calls it *power typing*. If you are a writer brought up on a typewriter, you already have a sixth sense that tells you when you are nearing the end of the line. But, if you are like the vast majority, the keyboard locks up frequently as you discover that you have tried to type past the right margin of the screen. With TE, you only have to worry about filling up the screen. Whenever you hit ENTER, the text is broken up into lines on word boundaries, never in the middle of a word, within the columns boundaries set by the BNDS Line Command.

Text Format (TF) is used to provide standard line lengths for text. When used in conjunction with the RIGHT Primary Command, you can do practically anything. In the normal case of block format paragraphs, you would enter TF64 on the first line of each paragraph to re-format the text in each paragraph so that it fills up columns one to 64. TF does not produce even margins (justification) or do hyphenation, but it merely breaks each line at the end of the last word that will fully fit on the line. It even puts two spaces at the end of each sentence.

In most situations, TF works well. One of the clues TF uses to define the end of a paragraph is the last line with the same indentation as the beginning of the paragraph — the line where the TF command was placed. In block form, you would have a blank line between each paragraph, so it works there. In indented paragraph form, even if you do not double space between paragraphs, the indentation of the next paragraph would signal the end of the previous paragraph to TF. It even works in block form when you

*See* ISPF *page 94*



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.**

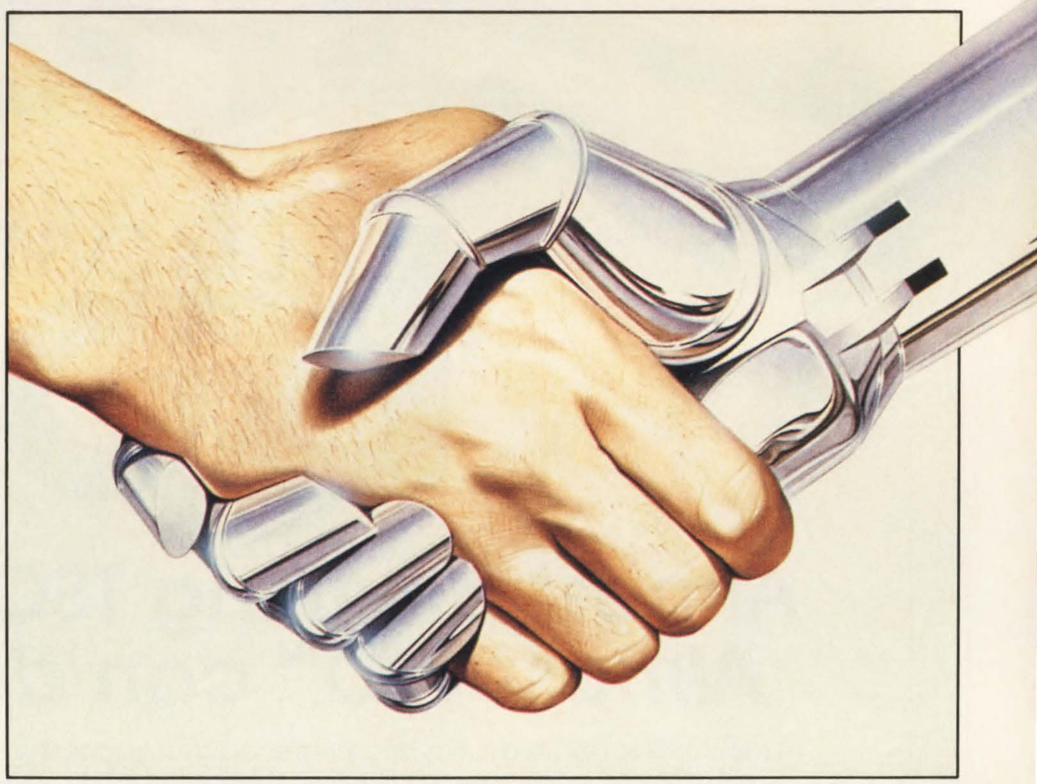
## Applied Software, Inc.

**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* ▲

# Getting Acquainted With The Network Control Program



By Beverly J. Weaver

If you are like many of us in technical support, your first introduction to the Network Control Program (NCP) was when you were asked to add new line definitions to the NCP generation source and create a new NCP load module. We all have copied existing line definitions, changing the names to protect the innocent and hoped for the best. Did you ever wonder what was really going on out there in the communications controller when your gen was finished and the module you created was loaded?

Maybe you support software which runs on the host computer such as the operating system, database systems, job entry subsystems or related program products. Would you like to know something about the programs running in the communications controller?

## What Is NCP?

The Network Control Program is an integral part of IBM's Systems Network Architecture (SNA). Along with VTAM, it allows the lines connecting users in remote locations to access the applications on your host system. For an introduction to VTAM, see "How VTAM Works" in the April 1989 issue of *MAINFRAME JOURNAL*.

Devices are designated as *local* if they are channel-attached to the host and controlled directly by VTAM; devices are *remote* if they are attached by lines (usually telephone lines) to the communications controller. Remote devices are handled by the NCP at the request of the controlling VTAM system.

NCP runs in the communications controller (IBM 3705, 3720, 3725 or 3745), which is often called a Front-End Processor (FEP) because it moved a large amount of the work off the host for devices on remote lines. Before VTAM and NCP were introduced, each application and/or the access method running on the host provided the line-handling software.

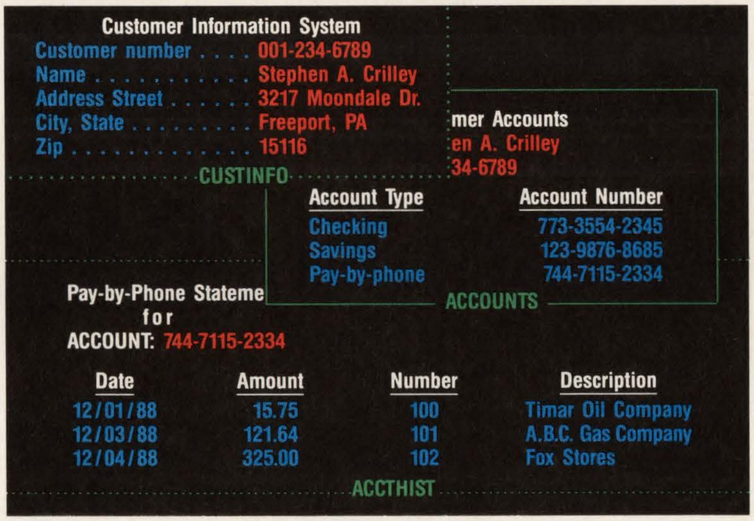
The main function of the NCP is to send and receive data on lines to remote SNA devices such as 3270-type controllers, SNA/RJE stations or distributed processors. These are connected to the communications controller using serial lines. Serial lines allow only one bit at a time to travel across them. Because no control bits can be sent at the same time with the data, control information is sent before and after the data as headers and trailers. How that control information is formatted and used defines a line protocol. The primary protocol used in an SNA environment is Synchronous Data Link Control (SDLC). NCP also supports Binary Synchronous Communication (BSC) protocol for certain 3270-type controllers.

When the NCP has collected data from remote devices, it is transferred to VTAM across a channel, which is a parallel link. A parallel link allows sending more than one bit at a time across it along with additional control information. Channel interfaces are used to connect other devices to your host including DASD controllers, tape controllers and local cluster controllers (3274, 3174).

## Other Software In The Communications Controller

It is possible that you may be running

# TPX® DOES WINDOWS.




## *Casting New Light on Session Management and Application Integration.*

Any session manager can increase productivity through reduced logons and logoffs, simplified network access, broadcasting, etc. But there's only one session manager that goes beyond these fundamental capabilities to address the growing demands of MVS and VM based SNA networks. Only one session manager with full function, dynamic windows and application integration.

That session manager is TPX from LEGENT.™ We're used in over 1800 data centers worldwide. More than all other session managers combined.

TPX Windows are dynamic and let you work with as many sessions as you like on your 3270 terminal, instead of just displaying a limited number of sessions in static mode. And the configuration of TPX Windows is easily changed as your needs change.

TPX Windows makes it easy to integrate applications. You can view multiple applications concurrently and then use simple CUT and PASTE commands to move data from one application to another, eliminating repetitive data entry. You can also integrate applications using a simple, but powerful language to create "seamless" windows for end users.

TPX can save you even more time and money with capabilities such as virtual printer management, mailbox facilities, data compression, online administration, panel management, and more.

Isn't it time you took a closer look at the leading session manager? For more information or a free 30 day trial, call 800-323-2600, or 412-323-2600 in PA.

TPX with Windows. Session Management never looked so good.

LEGENT



Two Allegheny Center
Pittsburgh, PA 15212

CIRCLE #80 on Reader Service Card ▲

other software in your communications controller. You might be running the Emulation Program (EP) instead of NCP. EP makes it possible to emulate an older Transmission Control Unit (TCU) like the 2703. This allows older access methods to control devices directly. For example, if you have Basic Telecommunications Access Method (BTAM) devices connected to IMS or CICS, then these devices would use EP lines in your communications controller. Just remember that all the lines that VTAM controls will be NCP lines; the EP lines will be controlled by other access methods.

You can run both the NCP and EP code in the communications controller. The NCP has its own program modules and buffers; the EP programs use separate buffers to send and receive data for the EP lines. Because they are effectively partitioned from each other, a module with both NCP and EP is called a Partitioned Emulation Program (PEP).

## The Load Module

You generate the executable module which will run in the communications controller by doing the NCP gen. Source code is created which defines the hardware installed on the communications controller, the lines to be used, the VTAM systems which are connected and so on. For older versions of the NCP, each source code statement was a macro and was run through a STAGE1 and STAGE2 process similar to a system generation. This produced the executable code to be loaded and run in the communications controller.

Currently the source is a set of *definition statements* which are run through the NCP/EP Definition Facility (NDF) to create the load module. NDF is a big improvement over the old STAGE1/STAGE2 process because it is a much faster way to create the load module. NDF also provides additional abilities to scan the source statements; errors that might have taken hours to find in old NCP gens could be caught in a minute or two with NDF.

After the load module is created, it can be loaded using a batch job or by issuing a VTAM command:

```
V NET,ACT,ID=NCP01,LOAD=YES
```

## The NCP Generation

In the NCP generation source are all the statements which describe the network devices attached to the NCP. In addition, there are statements to describe channel connections to host systems, the communications controller hardware that

# ACF/VTAM Tuning For ACF/NCP Channels

Channel tuning statistics can be used to find the bottlenecks in both the front-end processor and the IBM host. ACF/VTAM tuning statistics can be a valuable tool to increase channel throughput rates, improve ACF/VTAM storage utilization and decrease host CPU cycles. You can easily improve the performance of an existing network with the proper analysis of tuning statistics that result in minor network changes.

In this brief article, I will explain how to start collecting tuning statistics, examine what the ACF/NCP channel statistics mean and outline some basic rules that can be applied when analyzing these statistics.

## Starting And Stopping ACF/VTAM Tuning Statistics

To start collecting ACF/NCP channel tuning statistics, include the TNSTAT parameter in the startup parameters for ACF/VTAM. The default setting for tuning statistics is NOTNSTAT. The keyword is specified in the ACF/VTAM ATCSTR00 file found in the VTAMLIST dataset.

The TNSTAT keyword has the following format:

```
TNSTAT, [CNSL | NOCNSL], [TIME=nn | 60]
```

Both operands are optional. The first operand controls whether the channel statistics will be displayed on the console. CNSL presents the information on the console, while NOCNSL (no console display) is the default. The second operand shows how often ACF/VTAM will produce a tuning statistic record. TIME=60, the default value, tells ACF/VTAM to generate a record every 60 minutes.

Tuning statistics can be turned off and on using the ACF/VTAM Modify command entered through the ACF/VTAM operator's console. TNSTAT must be included in the startup parameters to use this operator command.

The command to stop collecting statistics looks like this:

```
F NET, NOTNSTAT
```

To restart the collection process, you enter a command like this one:

```
F NET, TNSTAT,CNSL=YES,TIME=60
```

You will find the output of the tuning statistics in different places depending on the operating environment. In MVS, the output is written to the SMF dataset. In VSE, the output is written in the ACF/VTAM trace file. In VM, the output can be found in the VTAM TUNSTATS file on ACF/VTAM's A-disk.

Record formats are different depending on the type of device attached to the channel. Refer to the ACF/VTAM customization manual for specific record formats. In all three environments, the CNSL parameter controls whether the information is also displayed on the operator's console. In VM, the operator's console is the same as the ACF/VTAM virtual machine console.

The tuning information is interval data. All fields are reset to zero when the data is written to the tuning statistics dataset. To do proper tuning over a period of time, tools need to be developed that allow you to report the data in a meaningful manner. The tool would produce reports that perform statistical functions such as calculating averages, means and maximum values for selected channels or intervals.

## What Information Is Available?

Figure 1 is a sample display of ACF/VTAM tuning statistics for the interval ending at 12:40.

ACF/VTAM statistics produced for channels contain a lot of information. You need to be able to use this information to make changes to the ACF/VTAM or ACF/NCP definitions that result in reduced CPU cycles through channel coattailing, improved host storage utilization or increased channel throughput.

In this context, coattailing is the sending and receiving of data from ACF/NCP in one channel program. When ACF/VTAM sends data to ACF/NCP, it fol-

# 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 ▲*

is installed, the paths which will be used to route data to other parts of the network and various other facilities.

This source should appear as a major node in your VTAM source definition library with a name that matches the name of your NCP load module. When you browse the source for your own NCP gen, you will see some of the following statements: PCCU, BUILD, HOST, PATH, GROUP, LINE, CLUSTER, TERMINAL, PU, LU, GENEND.

## PCCU Statement

The PCCU statement is used to describe a VTAM system and its connection to this NCP. Under normal circumstances you would have a separate PCCU statement for any VTAM system which will be activating the NCP. This statement is for VTAM use only. NCP does not use the parameters on the PCCU statement; however, the spelling and syntax are checked during the NCP gen process.

## BUILD Statement

The BUILD statement describes a lot about the load module to be created and the hardware on which it will run. On the BUILD you specify what type of gen this will be: EP, NCP or PEP. You will describe whether you are running this NCP on a 3705, 3725, 3720 or 3745 and what version of the NCP you will be running. Certain time-out values are specified here that relate to the whole NCP gen.

The BUILD statement is a long statement with many possible parameters, but once it is in place you will not need to change it very much. One exception might be the NEWNAME parameter, which specifies what the name of this particular load module will be. If you use a rotating naming convention for your load modules, you will need to update this value each time you do a new gen.

## HOST Statement

The HOST statement describes any host systems that are connected to this NCP with a channel. There will be at least one channel-attached host unless this is a remote communications controller which is loaded across an SDLC link.

The HOST statement describes how VTAM will issue commands to READ and WRITE data across the channel interface, including the number of buffers that will be set aside on the host when the READ command is issued, the size of those buffers and the number of buffers in the NCP which will be used each time VTAM issues the write channel program with a read channel program. Data queued by ACF/NCP for the host is sent after ACF/NCP finishes receiving the data from ACF/VTAM using the same ACF/VTAM channel program. Path Information Units (PIUs) are held or queued at ACF/NCP based on the gen's DELAY parameter. Proper DELAY values promote coattailing. Coattailing saves CPU cycles by cutting down the number of attention interrupts raised from ACF/NCP.

To improve proper host storage utilization, you want to make sure ACF/VTAM allocates enough input buffers to satisfy the data the ACF/NCP has available to send. The MAXBFRU and IOBUF parameters are used to determine how much storage will be allocated for each read channel program. DELAY can be used to regulate how long ACF/NCP saves PIUs prior to sending them to ACF/VTAM. The ACF/NCP DELAY parameter plus the ACF/VTAM buffer pool definitions impact host storage utilization.

To increase the throughput on an ACF/NCP channel, you can use a combination of proper read buffer allocations and the DELAY parameter. Table 1 compares variables described by Figure 1 and lists some guidelines that you can follow when analyzing channel statistics.

**FIGURE 1**

**ACF/NCP TNSTAT Display**

| | |
|---|---|
| (1) TIME=124042308 | (7) RDATTN=1 |
| (2) DLRMAX=1 | (8) RDBUF=196 |
| (3) ATTN=31 | (9) PCNAME=006-L |
| (4) OPIU=180 | (10) CHRD=135 |
| (5) DATE=89123 | (11) IPIU=196 |
| (6) CHWR=178 | (12) SLODN=0 |

*(1) Time (hhmmssth format) when the record was written to disk or displayed on the console.*

*(2) Number of dump and/or load requests during the interval.*

*(3) Number of times ACF/NCP raised an attention interrupt to notify the host to initiate a read channel program.*

*(4) Number of output path information units sent to this ACF/NCP during the interval.*

*(5) The date in Julian (yydd) format when the record was written. For example, 89123 represents Wednesday, May 3, 1989.*

*(6) Number of write channel programs executed during the interval.*

*(7) Number of times a read attention was raised by ACF/NCP at the end of an ACF/VTAM read channel program. This tells ACF/VTAM to issue another read to receive the remaining ACF/NCP data.*

*(8) Number of ACF/VTAM buffers that were used to read ACF/NCP data during the interval.*

*(9) The channel link name. -L indicates that this name was generated by ACF/VTAM.*

*(10) Number of read channel programs executed during the interval.*

*(11) Number of input path information units received from this ACF/NCP during the interval.*

*(12) Number of times that ACF/NCP went into slow down during the interval.*

**TABLE 1**

| If | Then |
|---|---|
| ATTN > CHRD | The ACF/NCP DELAY parameter is too low. |
| SLODN > 0 (consistently) | The DELAY parameter is too high. |
| RDBUF = IPIU | IOBUF is too large. |
| RDBUF > (2 * IPIU) | IOBUF is too small. |
| RDBUF < (MAXBFRU * CHRD) | MAXBFRU is too large. |
| OPIU > CHWR | ACF/VTAM is using CPU cycles doing blocking. |
| RDATTN > (.10 * CHRD) | MAXBFRU or IOBUF is too small. |
| RDATTN = 0 (always) | MAXBFRU or IOBUF is too large. |

## Summary

The ACF/VTAM tuning statistics are a valuable tool that can be used to improve the performance of an existing network channel. Tuning statistics collected over a period of time can be used to forecast network growth and determine workload characteristics to allow you to better service your end user. I strongly recommend that you take a look at what can be done in your network using this resource.

*Lloyd A. Hagemo, Jr.*
*Product Development Manager*
*BlueLine Software, Inc.*

sues the WRITE command to send data to the NCP.

## PATH Statement

The PATH statement is used to tell the NCP how to find other NCPs or VTAM systems in the SNA Network. A unique subarea number is assigned to each VTAM and another to each NCP. This number is used as part of the network address for each resource in the network in a similar fashion to an area code in a telephone number. After the proper subarea is reached, the remainder of the network address is used to locate the proper resource. This second portion of the address is called the element address.

PATH statements list every subarea in your network that might ever be a destination for data passing through this NCP. For that destination subarea you must define the closest subarea (adjacent subarea) which lies on the path and the name of the communications link (the transmission group number) to be used to reach the adjacent subarea.

## Defining Remote Lines And Devices

To define lines and the devices on them, you need several statements which have hierarchical relationships. For SDLC lines you will have the following statement types: GROUP, LINE, SERVICE, PU, LU. For BSC you will see: GROUP, LINE, SERVICE, CLUSTER, TERMINAL.

## GROUP Statement

First you must have a GROUP statement that describes a group of lines with similar characteristics. For NCP type lines, there are two possible line control protocols: BSC and SDLC. In addition, you may have GROUP statements for lines which are controlled by the Emulation Program (TYPE=EP). Each type of line must be in a separate group; all SDLC definitions must follow the BSC definitions and EP line definitions in the gen source.

For example, all the switched (dial-in or dial-out) NCP lines that use SDLC line protocol could be grouped together and those which were non-switched (dedicated/leased) would be in another group. All BSC devices would be defined in groups separate from the SDLC devices.

## LINE Statement

All circuits are defined using the LINE statement. Characteristics of the line are listed here; for example, the address of the interface that is used to connect the line to the communications controller whether it is a half-duplex or full-duplex line and so on.

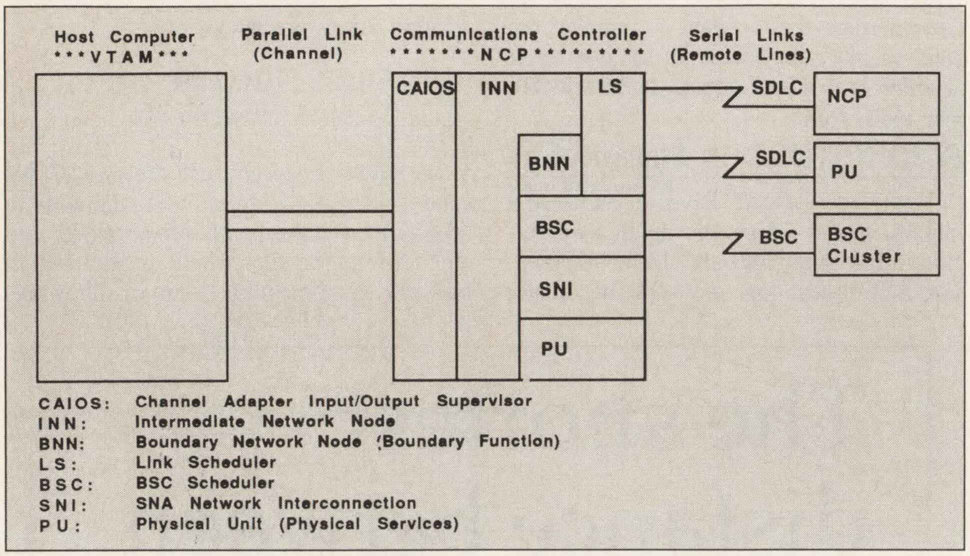
## SERVICE ORDER Statement

Next you would have a SERVICE statement that names the devices on this line in the order they should be serviced. For BSC lines, this includes the cluster controller and every terminal device on the line. For SDLC, only the cluster controller names are listed.

## BSC CLUSTER And TERMINAL Statements

For BSC lines you use the CLUSTER and TERMINAL definition statements to define the 3270-type cluster controller and the terminal devices which are attached to the cluster controller. Some of the things

FIGURE 1




you will see on the CLUSTER macro include the general polling characters and the type of control unit. On the TERMINAL statement there are polling characters to be used to obtain information from the device and addressing characters to be used to deliver data to the device.

## SDLC PU And LU Statements

For SDLC lines the PU statement describes the cluster controller, including the station address that is used to send and receive data for this cluster controller. There is also information to describe the largest number of bytes which can be transmitted to this controller at one time (MAXDATA).

The LU describes the microcode in the cluster controller representing the terminal device. There is a local address associated with each LU and other information that describes how the terminal looks to the network. Many of the parameters on the LU are for VTAM to use. Examples are the logon mode table name and mode table entry that should be used (MODETAB and DLOGMOD) to describe this device during logon processing or the Unformatted Systems Services Table (USSTAB) that should be used to translate character-coded logon requests.

## Sift-Down Effect

Many of the parameters that belong on the LU statement may be placed on any related statement at a higher level. In other words, the logon mode table name (MODETAB) may be placed on the LU or on the PU above it, on the LINE above the PU or on the GROUP above the LINE. The value will *sift down* to statements below unless specifically overridden.

Sift-down is effective on all the statements used to define both BSC and SDLC lines except the SERVICE macro. This allows a shorter gen source listing, but it can make the gen source confusing to look at. To determine what value of a parameter is being used for the LU definition, for example, you might have to look at four statements — the LU, the PU, the LINE and the GROUP. If you did not find the value specified in the gen, then you would look up the default in the NCP definition reference manual. To make this a bit easier for you, NDF supplies reports which list exactly what values will be used for each parameter and from which statement those values were obtained.

## NCP Activation

You can activate the NCP by having the NCP in a predefined set of major nodes which will automatically be activated when VTAM is started or you could issue the following command:

V NET,ACT,ID=NCP01

In either case, VTAM will need to access the source code from your NCP gen. VTAM will read the PCCU statements to determine what channel address is to be used to communicate with the NCP, then use that address to send commands to start that communication. If necessary, the NCP load module will be transmitted across the channel and the controller will be IPL'd.

## Talking To VTAM

VTAM on the host computer controls the channel connection to the NCP. VTAM issues READ or WRITE channel commands to receive or send data. The NCP never issues a READ or WRITE across the channel.

When something is going from the host to a remote device, VTAM will write the data out to the NCP. When the NCP gets the data from VTAM, the destination address fields in the SNA header are used to deliver the data to the proper resource.

If the NCP has collected data that needs to go to the host, the NCP can use an attention interrupt to tell VTAM that a READ command should be issued. VTAM will then issue the READ to pick up the data.

## NCP Functions

The NCP provides several functions, all directly or indirectly related to sending and receiving data for remote lines. These include Channel Adapter Input/Output Supervisor, Intermediate Network Node Function, Boundary Function, Link Scheduler, BSC Scheduler, SNA Network Interconnection and Physical Services.

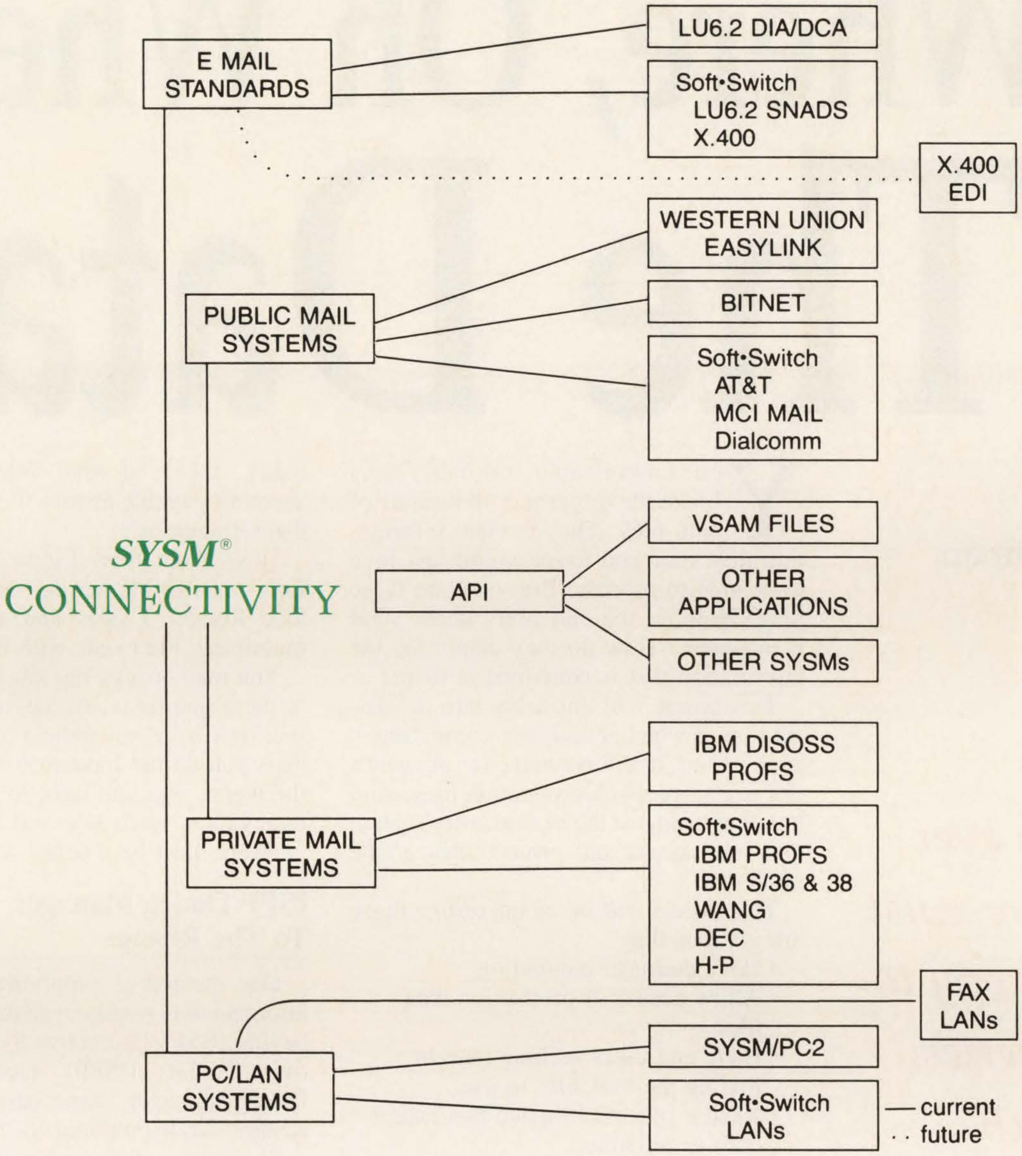
These components are shown in the communications controller in Figure 1. For information to travel from an application on the host to a remote printer, for example, it would flow across the channel to the NCP, through the CAIOS modules, through INN to determine where to send the data next, then to the Boundary Function for transformation of headers (and so on), then to the Link Scheduler to deliver across the SDLC line to the PU. The PU in the cluster controller is responsible for delivering the data to the printer.

## Channel Adapter Input And Output

The NCP code provides the necessary functions to create attention interrupts across the channel to the host and to understand the commands and data that VTAM sends across the channel interface. These programs are used to assist the channel adapter, which is the hardware used to connect the channel to your communications controller. A communications controller may have several channel adapters installed; each would be attached to a different VTAM system. VTAM talks to the NCP across one channel address for all the NCP lines; this three-digit operating system address is called the Native Subchannel Address. This same address is used to load the NCP with the executable module or to take a storage dump of the communications controller.

*See* NCP *page 95*

# Electronic Mail Connectivity Where You Need It!




***SYSM*** provides connectivity solutions for:

- EasyLink, PROFS, DISOSS, TSO, VM, BITNET, LU6.2 DIA/DCA
- The Soft•Switch path to PROFS, WANG, DEC, HP, LU6.2 SNADS, X.400, IBM S/36 & 38, AT&T, MCI Mail, Dialcomm and LANs.
- An Application Program Interface that enables user-written programs to interact with ***SYSM***.
- Future connections to X.400 protocols, EDI, and facsimile.

To see how connectivity can solve your corporate communication needs, call 1-800-338-6692 today to arrange for a FREE trial.

*SYSM*® is a registered trademark of H&W Computer Systems, Inc. Soft•Switch is a registered trademark of Soft•Switch, Inc., EasyLink is a service mark of Western Union.

**H&W COMPUTER SYSTEMS, INC.**
P.O. BOX 15190
BOISE, IDAHO 83715-0190
208-385-0336 • 800-338-6692

**CIRCLE #89 on Reader Service Card ▲**

# Oh Where, Oh Where Is The Data?

By Ira Hoffman

*Have you ever thought there must be a better way to handle queries, research documentation or trace JCL? There is!*

Your mainframe probably has hundreds or perhaps thousands of data files. They contain information that your end users would just love to be able to process. But what are these files? How do the end users know what is available? How do they determine the information that is contained in them?

This article will not delve into the discussion of whether end-user computing is good or bad. It will not argue for or against end users accessing production files. And it will not suggest the best security system to allow access and protect files at the same time.

This article *will* be of interest to those installations that:

- Have end-user computing
- Allow access of production data files
- Have end users seeking help in finding the best files to use
- Need a process to make this task as easy as possible.

## Has This Ever Happened To You?

Jimmy Cooke, a FOCUS user in Finance, asks you what fields are on the accounts payable history file and what the dataset name is.

How about this? Linda MacIntosh, a Lotus user in Marketing, wants to download Region 3 sales and asks you if a mainframe file exists with that data.

You may or may not know the answers to these questions. At the very least, you will now have to handle a query that perhaps you do not have to (or want to). At the worst, you will have to research documentation, trace JCL and so on.

There must be a better way! There is!

## ISPF Dialog Manager To The Rescue

One method of supplying this type of information is readily available to all shops having IBM's Interactive System Productivity Facility (ISPF). Using the ISPF Dialog Manager, some easily developed screens can handle *most* of these queries.

The information and screens that follow show the implementation of a system dubbed PRODuction FILEs (PRODFILE). Right off the bat, let me state what this system is and what it is not.

**FIGURE 1**

```
------------------------------------------------------ Utility Selection Menu ------------------------------------------------------
OPTION ===> P
   1   LIBRARY      - Library utility: Print index listing or entire dataset
                      Print, rename, delete, or browse members
                      Compress dataset
   2   DATASET      - Dataset utility: Display dataset information
                      Allocate, rename, or delete entire dataset
                      Catalog or uncatalog dataset
   3   MOVE/COPY    - Move, copy, or promote members or datasets
   4   DSLIST       - Dataset list: display all entries in specified catalog -
                      edit, browse, delete, rename, print, compress and so on
   5   RESET        - Reset statistics for members of ISPF library
   6   HARDCOPY     - Initiate hardcopy output
   P   PRODFILE     - Display production file information
   S   SCRIPT       - Format text on-line
```

## What PRODFILE Can Do

In most large shops having lots of production files, you will probably see the 90/10 rule. The end-user community, using its downloading capabilities or 4GL software, probably accesses 10 percent of all available files 90 percent of the time.

Of course, how the end-user community actually does this is a function of the installation in question. Sooner or later the end user ends up accessing a file. But how many days did it take to find out if such data was even available? How much of your time (whether you are Info Center, Tech Support, User Support and so on) is used? Is the file eventually used the *best* file that could be used? For example, is the user reading sales data for *all* regions just to process and report on data for a *specific* region (that happens to be available as an Extract File)? Is the end user, via his 4GL, sorting the file because the sort sequence is not known to him?

PRODFILE will not handle every question about every file. But based on past experience, *in your installation,* you can determine just what production files are used most frequently by your end users — the major files. PRODFILE will provide answers about these files.

PRODFILE is easy to develop, will save you from many phone calls and will increase your end-user's productivity.

For example, by choosing a better file (an Extract File or one in the desired sequence), your end-user's job will *execute more efficiently* and *save money* (has a nice ring to it).

### FIGURE 2

```
------------------------------------------ PRODFILE – Primary Menu ------------------------------------------
Command = = =>

  1   INFORMATION      – What PRODFILE is – what it does, what it does not do
  2   CHANGES          – New files, changed files and so on
  3   MAIN FILES       – The major files available for use
  4   EXTRACT FILES    – The extract ("mini") files available for use
  5   ADMINISTRATOR    – Functions available for PRODFILE Administrator only

                       Enter the desired option on the Command line

                           Depress PF1 For Assistance
                           Depress PF3 to exit from PRODFILE
```

### FIGURE 3

```
------------------------------------------ PRODFILE – Information ------------------------------------------
PRODFILE is an ISPF-based system that offers information about two kinds of production data: the "major"
files and "extract" files. These are the two types of mainframe data files most likely to be accessed by an
end-user using a 4GL language.

In both cases PRODFILE gives certain information for each file – the sorting sequence, the file definition
and so on. For the extract files the actual criteria that created the file is stated.

PRODFILE is NOT a data dictionary. It contains information about a very small percentage of all files, but
those files that collectively are accessed a high percentage of time by the end-user community.

                          Depress PF3 to exit from this screen
```

### FIGURE 4

```
------------------------------------------ PRODFILE – Changes ---------------------------- ROW 1 OF 3
Command = = =>                                                                    Scroll = = => PAGE

                       Enter the desired option on the Command line

                           Depress PF3 to exit from this screen
                           Depress PF7 to scroll backward
                           Depress PF8 to scroll forward

     DATE                             ACTION
    7/22/87   Customer Master (preferred) file now available for use

    9/10/87   Sales for Region 7 file now available as extract file

   11/13/87   Format for Sales forecasts file will change eff. 3/1/88
```

### FIGURE 5

```
------------------------------------------ PRODFILE – Main Files -------------------------- ROW 1 OF 24
Command = = =>                                                                    Scroll = = => PAGE
                           Depress PF1 for Assistance
                           Depress PF3 to exit from this screen
                           Depress PF7 to scroll backward
                           Depress PF8 to scroll forward
                   Place an "S" next to a file to get detail information about that file

                  NAME                                          DATASET NAME
 _  Accounts Payable Current file                               PD.AP0100C
 _  Accounts Payable Daily Transactions                         PD.AP0100T
 _  Accounts Payable History file                               PT.AP0100H
 _  Customer Master                                             VD.AC0909C
 _  Customer Master Inactives                                   PT.AC0909H
 _  Customer Master Preferred                                   TT.AC1919C
 _  Expense Payable Current file                                PT.EP0780C
 _  Expense Payable History file                                PT.EP0780H
 _  Inventory Master                                            VP.NN8920V
 _  Payroll Master file                                         PT.PR5300C
 _  Payroll Master Terminations file                            PT.PR5210H
 S  Sales (all regions)                                         PD.SA1019D
 _  Sales forecasts                                             PD.SA1021C
```

## What PRODFILE Cannot Do

Do not think of this system as being competition to a data dictionary. It is not trying to do that. Using available software (ISPF), it attempts to try to provide the maximum information with the minimum resources.

## A Look At PRODFILE

What follows is a general description of the PRODFILE system. After the last screen and its description are some additional comments (implementing the system, file access/security considerations and so on).

Look at the implementation of PRODFILE. Since PRODFILE can best be described as a utility, I have chosen to include it in the ISPF option 3 panel — Utility Selection Menu (see Figure 1).

After selecting option "P" for PRODFILE, the PRODFILE Primary Menu appears (see Figure 2).

This screen offers four selections for all end users and an additional one for the person designated PRODFILE Administrator.

The Information screen (option 1) gives an overview of what PRODFILE is, what it does and what it does not do (see Figure 3).

Option 2 from the PRODFILE Primary Menu (Figure 2) is Changes. This brings up an informative screen telling the end user what has happened to those files about which the PRODFILE system supplies information. This information may include new files that are available, changes to existing files and so on (see Figure 4).

The next option of the PRODFILE Primary Menu, option 3 of Figure 2, is the nuts and bolts of the system. Main Files brings up a series of panels, in sequence by an *English description* of the file, not in sequence by the dataset name. Alongside each file is the first piece of information — the dataset name. This is shown in Figure 5.

In Figure 5, the end user is interested in the file that shows Sales (all regions) data. Placing an "S" next to that file will bring up the detail screen for that data file. The Main Files Detail Information is shown in Figure 6.

This screen shows detail information about the Sales (all regions) file. This includes the sort sequence. The sorting order of the file is of significant interest to the end user. It may indicate that the file is already in the desired sequence. The 4GL language may be coded to read the file to a desired logical record, then stop (reading). Reading less data will save processing time and reduce costs.

The screen also shows the Extract Files — those files that contain segments of information found in the major file. Of course, there may not be an Extract File available, but if there is, PRODFILE is an excellent way to let the end-user community know about it.

In the example Extract Files are, in fact, available. Four different files are shown — Sales for Region 1, 2, 3 and 7. The end user is interested in data only for Region 3; therefore, he places an "S" next to that line (as shown in Figure 6). This brings up the Extract Detail Information panel (see Figure 7).

This screen gives all the detailed information for the Extract File, in this case the Sales for Region 3 data file. The Extract Criteria section states the method that created the file. In general, Extract Files contain data that is identical to the major file. This may not always be so. An Extract File might start out with the same data (records), but may have them in a different (and possibly better) sequence. It may have fewer fields than the major file or differ in some other way. But whether it is the same or different, its existence is documented via PRODFILE.

The next screen in the system lists the Extract Files. By choosing option 4 on the Primary Menu (Figure 2), a screen similar to the Main Files panel (Figure 5) but showing *only* the Extract Files is displayed. Once again the sequence is an *English descriptive name,* not the dataset name. The Extract Files panel is shown in Figure 8.

Since all Extract Files are listed, you see for the second time the four sales files (Sales for Region 1, 2, 3 and 4). From this screen an end user can get the same information as from the detail screen for a major file that has Extract Files (see Figure 6).

The last panel in the system, option 5 on the Primary Menu, invokes the PRODFILE Administrator screens. The contents of these panels are not relevant to this discussion. Functionally, they give the administrator the ability to maintain the system — add, change and delete information about the major and Extract Files.

## Implementing The System

There are a number of sources that can be used to determine the major and Extract Files. They include the following:

**FIGURE 6**

```
----------------------------------------- PRODFILE – Main Files Detail Information ----------------- ROW 1 OF 4
Command = = = >                                                                        Scroll = = = > PAGE
                                   Depress PF1 for assistance
                                   Depress PF3 to exit from this screen
                                   Depress PF7 to scroll backward
                                   Depress PF8 to scroll forward
Name:               – Sales (all regions)
Dataset Name        – PD.SA1019D
Sorting Sequence    – Region, Customer Number, Date
File Definition     – VPD019 in file definition book
Comments:           – this file contains information for the fiscal year
   (March 1 through February 28). It is purged each year in March. Care should be taken when using this
    file between March 1 and the date of the purge that the data from the correct year is used.
Extracts of this file – place an "S" next to any one for further info.
        NAME                                                   DATASET NAME
_  Sales for Region 1                                          MD.SA10191M
_  Sales for Region 2                                          MD.SA10192M
S  Sales for Region 3                                          MD.SA10193M
_  Sales for Region 7                                          MD.SA10197M
```

**FIGURE 7**

```
------------------------------------------ PRODFILE – Extract Detail Information ------------------------------------------
Name:               – Sales for Region 3
Dataset Name        – MD.SA10193M
Major File Name     – PD.SA1019D
Sorting Sequence    – Customer Number, Date
File Definition     – VPD0191 in file definition book
Comments:           – this file contains up to 900,000 records, depending on the time of year.
Also, see comments for major file regarding the purge. This file is also purged sometime in March and care
should be taken when using this file between March 1 and the date of the purge that the data from the
correct year is used.
Extract Criteria – this file contains the data for ONLY Region 3 found in
     the major file – PD.SA1019D
                                   Depress PF1 for assistance
                                   Depress PF3 to exit from this screen
```

**FIGURE 8**

```
-------------------------------------------------- PRODFILE – Extract Files -------------------------- ROW 1 OF 14
Command = = = >                                                                        Scroll = = = > PAGE
                                   Depress PF1 for Assistance
                                   Depress PF3 to exit from this screen
                                   Depress PF7 to scroll backward
                                   Depress PF8 to scroll forward
                        Place an "S" next to a file to get detail information about that file
              NAME                                                     DATASET NAME
_  Inventory Master category "CP"                                      PD.IN11237D
_  Inventory Master category "CT"                                      PD.IN14337D
_  Inventory Master category "FP"                                      PD.IN14237D
_  Inventory Master category "FQ"                                      PD.IN11337D
_  Sales for Region 1                                                  MD.SA10191M
_  Sales for Region 2                                                  MD.SA10192M
_  Sales for Region 3                                                  MD.SA10193M
_  Sales for Region 7                                                  MD.SA10197M
_  Travel Expenses Region 1                                            BT.TR98101D
```

- Asking the end-user community individually, at a user group, via the newsletter and by other means what files should be in the system
- Looking at system performance data (SMF, JARS and so on) to determine actual file usage
- On-going requests, suggestions and the like from the end users, MIS development and others for new/changed major and Extract Files.

### File Access/Security Considerations

The PRODFILE system does not give access to any file. It gives descriptive information about production files in a "user friendly" way. However, there may be concern in your installation that even the knowledge of the dataset name of a restricted file is in itself a breach of security. This should be resolved with the appropriate individuals in your organization prior to making those files part of the PRODFILE system.

### Additional Detail Information

The actual data requirements of the end-user community will dictate the specific detail information actually shown in the Main Files Detail Information (Figure 6) and Extract Detail Information (Figure 7). It is worth mentioning that some potentially useful information such as record length (LRECL), blocksize, device type (tape or disk), number of records and so on is available. This information may be obtained by processing SMF data, system utilization data (for example, JARS), tape management system and so on.

The developer of PRODFILE is faced with the task of determining the best source for this data and actually making it available through PRODFILE.

*ABOUT THE AUTHOR*

*Ira Hoffman is the Cost Containment/End-User Productivity Manager for Zayre Stores, Framingham, MA. He has more than 20 years experience in data processing, specializing in supporting and increasing productivity. 144 Allen Rd., Berlin, MA 01503, (508) 838-2702.*

# Where Did All The Cycles Go

## *A Study of CICS Processing Patterns*

By Ted C. Keller

Have you ever wondered what happens to all the CPU cycles expended in a CICS region? If you are doing any kind of serious planning, you probably know which transactions are consuming the largest percentage of the processor. You might even know how much overhead is associated with the task control (KCP) and terminal control (TCP) tasks. But do you really know where processing is taking place? Is the bulk of CICS processing time being spent in application code, CICS modules or system modules such as access methods?

Several years ago I developed a monitor which, among other things, could track which modules in a CICS region are using the CPU and where in each module the CPU was being utilized. My original intention was to find inefficient code in application programs which might be tuned to reduce CPU usage. However, what I discovered was the following:

- In most regions the amount of processing being performed in application code (the actual COBOL, Assembler and so on program statements) was much less than I had expected
- By far, the largest portion of processing was being spent in CICS management modules
- The amount of processing spent in CICS facilities such as CMF or trace was significant
- Not nearly as much time was being spent in some CICS system modules (such as PCP or FCP) as I would have guessed
- A relatively small amount of time was being spent in operating system services such as access methods.

In this article, I will discuss processing profiles for two different regions. The first region, which I will call Region A, contains a mix of macro- and command-level programs. The second region, Region B, runs almost nothing but COBOL command-level programs. Both regions include a wide variety of transactions and make some use of MRO services. There is little paging in these regions and all paging requirements are being serviced from expanded storage. Both regions were run under MVS/XA on a 3090 Model 600E which would have been about 55 to 65 percent busy overall.

In both of the profiles which follow, I will present CPU usage as percentages of CPU demand. *CPU demand* is the measure of the time CICS is either using or attempting to use the processor. These percentages communicate the portion of CICS' path length associated with each service.


EYEWITNESS®
CICS RADAR®


STABILIZE

FIGURE 1

**Processing in REGION A**

| Component | Percent |
|---|---|
| **CICS management modules** | |
| Monitor | 26.1% |
| Trace | 0.0% |
| Storage control | 11.9% |
| Program control | 4.7% |
| File control | 2.4% |
| Task control/terminal control | 3.9% |
| Execute interface (command level support) | 6.6% |
| Misc CICS management modules | 17.0% |
| Total CICS management modules | 72.6% |
| **Application code** | |
| User written PCP exits | 2.4% |
| All other application code | 10.1% |
| Total application code | 12.5% |
| Access methods and Other | 14.9% |
| Total | 100.0% |

## Profile of Region A

Most of the applications in Region A were originally written in macro-level Assembler over a decade ago. Newer modules added over the past several years have been primarily command-level COBOL. Applications in this region access several files in remote regions, mostly through the use of common command-level subroutines. The region currently drives CPU demand as high as 90 percent. Because of high levels of CPU demand, trace is not run in Region A. (I will discuss the impact of the trace facility later.) Plans are currently underway to move some processing to another region.

The period chosen for detailed analysis had a CPU demand of 88.5 percent. In this period, CICS management modules were responsible for 72.6 percent of the processing. Another 14.9 percent of processing was spent in miscellaneous MVS access methods or services. A mere 12.5 percent occurred in application code. Figure 1 gives a detailed picture of where processing occurred.

Figure 1 shows that the largest single user of CPU is the monitor. In this example, the monitor was a non-IBM monitor. In separate benchmarks conducted with this monitor and the IBM CMP monitor running together, both monitors used similar amounts of CPU. Both monitors were configured to collect standard performance data.

The largest amount of CPU consumed by either monitor was spent collecting information on CPU utilization. Both monitors are driven off a similar interface (EMP/TRACE) and receive control after each logical CICS event. Whenever CICS detects control switching from one task to another, the monitors will issue the MVS SVC commonly called *dispatch*. This SVC will update the CPU usage and paging statistics for the address space. Both monitors use this information to update the CPU and paging statistics for each task. In most systems, somewhere between 40 and 65 percent of the CPU time used by CMP or other similar monitors is associated with the dispatch SVC.

*In both monitors, the collection of CPU and paging statistics can be bypassed.*

In both monitors, the collection of CPU and paging statistics can be bypassed. In this case, monitors will collect only *dispatch* time. This is the time after tasks receive control from CICS until they return control to CICS. It includes CPU time used by the task plus the time spent waiting for higher priority MVS tasks, page-faults and other factors which can interrupt processing. Dispatch time is a valid measure of the time a task was attempting to use the processor, but it does not indicate how much CPU was actually used by the task. Depending on the need for an accurate measurement of CPU utilization and paging, it is possible to reduce CICS CPU usage by 10 to 15 percent by not collecting statistics on CPU usage and paging.

This may present somewhat of a dilemma — when you are driving the CPU hard, you might be most interested in determining which transactions are using the largest amount of CPU. Unfortunately, the single most significant user of CPU is likely to be the code measuring CPU usage.

Under Releases 1.7 and 2.1, the CMP monitor can be tailored to collect only selected data. It is not necessary to collect statistics on data which will not be used. Tailoring can help reduce CPU consumption by CMP. The non-IBM monitor may also be tuned. In separate tests, CPU usage

FIGURE 2

**Processing in REGION B**

| Component | Percent |
|---|---|
| **CICS management modules** | |
| Monitor | 28.5% |
| Trace | 11.7% |
| Storage control | 11.9% |
| Program control | 1.9% |
| File control | 6.8% |
| Task control/terminal control | 4.7% |
| Execute interface (command level support) | 6.4% |
| Misc CICS management modules | 17.6% |
| Total CICS management modules | 89.5% |
| **Application code** | |
| User written PCP exits | 1.5% |
| All other application code | 5.5% |
| Total application code | 7.0% |
| Access methods and Other | 3.5% |
| Total | 100.0% |

in the non-IBM monitor was reduced to 16 to 20 percent by restricting the collection of certain data.

A second interesting feature of this run is that about 15 percent of processing was performed by access method modules and other miscellaneous operating system services. There are many files defined in this region with heavy I/O activity. One particular file services about 20 million browse requests per day. Even with such heavy I/O activity, only a minor portion of processing is spent in access method code.

Of the 12.5 percent of processing actually spent in application program code, 2.4 percent was spent in one of two PCP exits. Another 4.3 percent occurred in common command-level sub-modules used to access files and 1.2 percent in common service modules such as date routines and table look-ups. Less than 5 percent of the processing is actually done in mainline application code. Compared to everything else, the processing by COBOL or Assembler application code was almost inconsequential. In fact, more time was spent in command-level service modules (6.6 percent) or in program control (including the user exits) than in mainline application code. It appears that the potential for reducing CPU utilization by tuning language-specific processing is quite limited.

One final observation about this region is that storage control was one of the larger consumers of processing. Of course, this is not because the applications are constantly requesting storage management services, but it is rather a reflection of the fact that most other CICS service modules request and release CICS storage of some kind.

## Profile Of Region B

Region B is quite different from Region A. Almost all of the modules in Region B are command-level COBOL. Local Shared Resources (LSR) are used heavily in Region B and both data and index read-hit ratios are 90 percent or better for most subpools. The typical transaction in this region will issue 15 to 20 random I/O requests, use .1 seconds of CPU and still complete in less than .3 seconds. About 1 percent of the transactions are relatively long running, issuing 300 to 10,000 random I/O requests with response times from one to 20 seconds. This exceptional level of service is primarily due to good LSR read-hit ratios. VSAM subtasking (VSP) is being used to throttle long-running tasks and allow other work to be dispatched. (Because of an incredible LSR hit ratio of well over 95 percent for the files accessed by the longest running tasks, tasks can remain in control of the processor for prolonged periods of time unless VSP is employed. CICS will allow a task to continue processing if an I/O request can be completely satisfied directly from LSR.)

The period studied had a CPU demand of 45.0 percent — it was attempting to use a processor a little less than half of the time. As can be seen in Figure 2, a staggering 89.5 percent of the processing occurred in CICS management modules. Only 7.0 percent was actually spent in any kind of application code and 3.5 percent in access methods or other operating


# Response Time is Money.

Poor CICS response time is expensive. The longer users wait, the less they get done, the more your bottom line suffers, and the more you get blamed. But good response time can also be expensive - when it's purchased through more hardware or overworked systems staff.

**No guesswork**
Candle helps keep CICS response time and your budget at a minimum with software that precisely pinpoints the causes of poor CICS performance. Our end-to-end response time feature even tells you whether the problem is in the host or in the network. And that improves the response time of your staff.

**Faster solutions**
The OMEGAMON® family of products for CICS keeps your user service levels on target by detecting availability threats and slowdowns immediately, analyzing the cause for you, and recommending the solution. A single Status Monitor™ screen keeps you on top of your entire CICS network - across CPUs and across geographic locations. OMEGAMON eliminates the time-consuming process of analyzing irrelevant resource data by doing the analysis for you.

Our products also help you budget. They give you trending and capacity planning information so you can forecast future needs and make maximum use of existing resources. With software, not by throwing more iron at your problems before it's really needed.

**Total support**
As for our response time, Candle's support team is available round-the-clock, around the world, every day of the year to answer your questions and help you get the best performance from your CICS system. And Candle Education helps you improve your own performance.

Is your response time costing you money? It's worth your time to find out by calling your Candle Account Representative or Terry Forbes today at **(800) 843-3970.**

!Candle®



***CIRCLE #120 on Reader Service Card ▲***

system services. Part of the reason for such a small percentage of time showing in access methods is that VSAM subtasking was being used. Most of the CPU utilization associated with VSAM processing was being performed by the VSAM subtask — a separate MVS task. The processing observed by my monitor is only for the main CICS task. Additional processing is occurring in the VSAM subtask. It is contributing to the total CPU used, but it is not directly impacting processing in the main CICS task. Unless total CPU utilization in the processor complex becomes excessive, VSAM subtasking can help relieve CPU constraint in a CICS region.

Comparing Figures 1 and 2, you can see many similarities between Regions A and B. The monitor facility again was the dominant consumer of processor resource. Storage control also utilized a significant portion of processing.

File control showed a significantly larger portion of processing in Region B (6.8 percent versus 2.4 percent). The difference in CPU usage was observed to be in the portion of file control communicating with the VSAM subtask. If VSP were not being used, the main CICS task would show about 12 to 18 percent more CPU usage in access method code. With VSP, CPU usage in file control is 4 to 8 percent higher. Thus, VSP appears to reduce CPU demand for the primary CICS MVS task by about 8 to 12 percent.

Both program control and the PCP exits received quite a bit more activity in Region A. Transactions in Region A tend to LINK to a large number of application subroutines. Programs in Region B tend to be somewhat larger and less dependent on LINKed sub-modules.

The biggest difference between the two regions appears in the processing performed by the trace facility. Region B shows 11.7 percent of activity in the trace module. When trace is running, it can usually be expected to add 12 to 20 percent CPU consumption by CICS. As you may recall, trace was not running in Region A.

## Conclusions

The two profiles shown above are representative of activity in each of these regions. Both are built from periods of about 10 minutes each. The numbers shown in each profile are typical of results observed in many periods for the two regions. Percentages for larger users of the CPU (especially for monitor code) varied by as much as 2 to 4 percent from sample to sample in each region, but overall patterns of activity always remained approximately the same. Other CICS regions have been profiled and have shown results similar to those observed for Regions A and B.

Based upon the processing patterns I have observed in many CICS regions, I will offer the following generalizations.

Micro-level tuning of application code will have little impact on overall CICS CPU utilization. Since it appears that 10 percent or less of CICS processing will occur in application programs, tuning efforts which make the program code as much as 30 percent faster will only reduce total CPU utilization by 3 percent or less.

*See* Cycles *page 84*

# What You Should Know About The TGT

By Harvey Bookman

The COBOL Task Global Table (TGT) is a table used in every COBOL program. It can be considered a *control block* or a *data area*. It is unfortunate that in our fourth-generation society, control block and data area are terms that a COBOL programmer is not expected to know. The explanation of the TGT in the IBM manuals does not encourage the reader to explore further. Other than a complete field-by-field description in the appendix of the *IBM OS/VS COBOL Compiler and Library Programmer's Guide,* the manual describes the TGT as "consisting of switches, addresses and work areas whose information changes during execution of the program" and "being used to record and save information needed during execution of the object program."

IBM has changed its documentation of the TGT in COBOL II. The *VS COBOL II Application Programming: Debugging* manual explains how to find the TGT and the use of a few of its fields. Unfortunately, the manual does not have a field-by-field description. These field descriptions are located in the *VS COBOL II Diagnosis Reference* manual. This manual is one in the LY series of IBM manuals (the same series as the program logic manuals) usually unavailable to application programmers. The remainder of this article will describe the TGT in VS COBOL. The differences that apply to COBOL II will be *italicized* in brackets [].

Without some sort of education or experience to show otherwise, the description in the manual leads you to believe that the TGT is not an area worth studying. While some fields in the TGT are of no use to an applications programmer, others are of considerable importance. The fact is the TGT contains a wealth of information that is available nowhere else. Knowledge of its information also aids a programmer in other areas of COBOL understanding such as reading a Procedure Map. The Procedure Map is Assembler code generated in the listing produced by the COBOL compiler. Understanding the TGT and its intricacies can literally save a programmer many weeks of time during his or her programming career.

Take for instance the problem of finding the value of an index at the time that a COBOL program ABENDs. The index values (other than those explicitly defined as data fields in a program) only exist in the TGT. From my personal experience, most COBOL programmers do not know how to find an index in a dump. After a dump occurs, a programmer will usually rerun a program with a DISPLAY of any index that is in question. This method of debugging will take many times as long as simply reading a dump.

It is absurd that a programmer will make use of a subscript value simply because it is located by using the Data Division Map of a COBOL program, but (s)he will not make use of index value because it is found in a "less convenient" place, the TGT. With the proper understanding, the value of an index in a dump can be located in about the same time it takes to locate the value of a data field. Other fields in the TGT can be located just as easily.

The TGT is included in the listing whenever a COBOL program is compiled with the PMAP, CLIST or DMAP options. [*LIST or OFFSET options.*] In VS COBOL it is included right after the Data Division Map in the listing. The displacement of each field from the beginning of the program is shown directly after the field in the TGT listing. The TGT physically exists right after WORKING-STORAGE and before the actual program code in VS COBOL.

[*In COBOL II the TGT appears near the end of the listing. When the NORENT option is used in a compilation, the TGT in use by the program will physically exist as part of the load module, located after the program code and before WORKING-STORAGE. (The order of the various parts of a COBOL program have been rearranged in COBOL II.) The displacement of each field from the beginning of the program as well as the relative displace-*

*ment from the beginning of the TGT is shown before each field.*

*When the RENT option is used to compile a COBOL II program (making a program reentrant, which is necessary when a program is written to run under CICS), the TGT (as well as the WORKING-STORAGE) exists in a GETMAINed area. In MVS and CMS it is put into subpool 0 storage while in CICS it is put into CICS managed storage. The displacement of each field relative to the beginning of the TGT is shown before each field in the listing. (The displacement relative to the program beginning is not shown since the TGT is not located adjacent to the program code.)*

*A program compiled with the RENT option will take up more storage when it runs than a program compiled with the NORENT option. This is because the program's TGT and WORKING-STORAGE exist both within the program as well as the copy in storage that is actually used. The copy in the program is used by COBOL to initialize the GETMAINed TGT.*]

Once the physical beginning address of a COBOL program is located in a dump, it is added to the displacement of any specific field in the TGT (relative to the beginning of the program) to locate the field. [*When the RENT option is used with COBOL II, the TGT used will be located in storage outside of the load module. When a COBOL II program is running, Register 13 normally points to the TGT. If a COBOL II subroutine is running at the time of an ABEND, Register 9 usually points to the TGT. When you think you have located a TGT, you can verify this fact since the TGT always starts with a hexadecimal '00108001' and contains the alphabetic constant 'C2TGT+48' at a hexadecimal offset of '48'.*] It is even easier to find fields in the TGT when an installation has COBOL debugging tools that format the TGT whenever a dump is produced.

In the remainder of this article, some of the TGT fields will be discussed in detail. The terms printed in capital letters refer to specific TGT field names in the figures. Examples of a TGT from a VS COBOL compile, a TGT from a COBOL II program compiled with the RENT option and a TGT from a COBOL II program compiled with the NORENT option are shown in Figures 1, 2 and 3.

The TGT is composed of two parts. The first part is a fixed-length portion containing fields which always appear in every program such as the Register Save area. Following this is a variable-length portion containing fields of varying lengths. The variable fields exist or not depending on specific program requirements determined by the COBOL compiler for each program.

When examining the TGT in a COBOL program, all fields in the variable portion are listed even if they do not exist in the specific program. You can determine fields that are not used by checking whether or not the field after them has the same displacement. For example, suppose that INDEX CELLS are listed in the TGT at a displacement of 330 and in the following field in the TGT, the SUBADR CELLS are also listed at displacement 330. There are no INDEX CELLS used by the program. [*This is clearer in COBOL II where fields in the variable portion that are not used are not listed.*]

## Index Cells

Each implicitly defined index in a COBOL program takes up a fullword in the INDEX CELLS. COBOL stores the value of an index in binary as if it were defined as a PIC S9(9) COMP in the program. Indexes are stored as displacements into the table on which they function. For a one-dimensional table, to determine the value of an index at the time of an ABEND, divide the value in an INDEX CELL by the length of its corresponding table entry and add one.

The INDEX CELLS appear in the same order that indexes are defined in the program; the first fullword for the first implicit index defined, the second fullword for the second implicit index defined and so on.

The indexes in a program are not numbered in the Data Division Map. If you do not have a PMAP where they are numbered, it can be rather clumsy to find an index value in a program with many indexes. You must first find each field in the Data Division Map that has a usage of INDEX-NM and has no BL or BLL cell listed in the BASE column. (Explicitly defined index cells have a base cell associated with them.) Number the implicit indexes sequentially until you get to the index that you wish to locate. [*For each field in the Data Division Map that has a usage of INDEX-NAME, the BASE column of the Data Division Map lists each index sequentially as IDX=0001, IDX=0002 and so on. You no longer have to count the INDEX CELLS yourself.*] The displacement into the INDEX CELLS where the index value begins can be calculated by subtracting one from the sequential index number and multiplying by four.

## BL Cells [*Base Locators For WORKING-STORAGE, Base Locators For Files*]

Each BL cell is a fullword that contains the address of either a record in a file or a part of WORKING-STORAGE. The pointers to records appear first. Following them is one BL cell for each 4096 bytes of WORKING-STORAGE that the program contains. [*One set of Base Locator cells is now used for files and another for WORKING-STORAGE. File cells are indicated in the Data Division Map as BLF and WORKING-STORAGE cells as BLW.*]

To find any field in WORKING-STORAGE at the time of an ABEND, locate the field in question in the Data Division Map and find its BL cell listed next to it. Find the appropriate BL cell in the TGT (remembering that each BL cell takes up four bytes). Add its value to the displacement of the field listed in the Data Division Map.

It is a common misconception that the registers at the time of the ABEND always point to the DATA DIVISION field or fields that caused the problem. While the registers often point to the DATA DIVISION fields causing the ABEND (but not always since fields are sometimes moved into work areas — see TEMPORARY STORAGE below), they certainly do not always point to all the various fields in WORKING-STORAGE and the FILE SECTION that a programmer might have to examine. The BL cells can be used to find any field in WORKING-STORAGE.

## BLL Cells [*Base Locators For LINKAGE-SECTION*]

The BLL cells are similar to the BL cells. Each is a fullword address. The major difference is that while BL cells refer to WORKING-STORAGE and the FILE SECTION, BLL cells refer to areas in the LINKAGE SECTION. The first two are reserved for COBOL's internal use. [*Only the first BLL cell is reserved for COBOL's internal use.*] The first cell is used for declaratives associated with label processing and the second for a sort work area. The first BLL cell used in the program will therefore be referred to as BLL = 3 in the Data Division Map. [*The first BLL cell in the Data Division Map is*

**F I G U R E 1**

MEMORY MAP

| | |
|---|---|
| TGT | 004C8 |
| SAVE AREA | 004C8 |
| SWITCH | 00510 |
| TALLY | 00514 |
| SORT SAVE | 00518 |
| ENTRY-SAVE | 0051C |
| SORT CORE SIZE | 00520 |
| RET CODE | 00524 |
| SORT RET | 00526 |
| WORKING CELLS | 00528 |
| SORT FILE SIZE | 00658 |
| SORT MODE SIZE | 0065C |
| PGT-VN TBL | 00660 |
| TGT-VN TBL | 00664 |
| RESERVED | 00668 |
| LENGTH OF VN TBL | 0066C |
| LABEL RET | 0066E |
| RESERVED | 0066F |
| DBG R14SAVE | 00670 |
| COBOL INDICATOR | 00674 |
| A(INIT1) | 00678 |
| DEBUG TABLE PTR | 0067C |
| SUBCOM PTR | 00680 |
| SORT-MESSAGE | 00684 |
| SYSOUT DDNAME | 0068C |
| RESERVED | 0068D |
| COBOL ID | 0068E |
| COMPILED POINTER | 00690 |
| COUNT TABLE ADDRESS | 00694 |
| RESERVED | 00698 |
| DBG R11SAVE | 006A0 |
| COUNT CHAIN ADDRESS | 006A4 |
| PRBL 1 CELL PTR | 006A8 |
| RESEVED | 006AC |
| TA LENGTH | 006B1 |
| RESERVED | 006B4 |
| PCS LIT PTR | 006BC |
| DEBUGGING | 006C0 |
| CD FOR INITIAL INPUT | 006C4 |
| OVERFLOW CELLS | 006C8 |
| BL CELLS | 006C8 |
| DECBADR CELLS | 006DC |
| FIB CELLS | 006DC |
| TEMP STORAGE | 006E8 |
| TEMP STORAGE-2 | 006F0 |
| TEMP STORAGE-3 | 006F8 |
| TEMP STORAGE-4 | 006F8 |
| BLL CELLS | 006F8 |
| VLC CELLS | 00710 |
| SBL CELLS | 00714 |
| INDEX CELLS | 00714 |
| SUBADR CELLS | 0071C |
| ONCTL CELLS | 0071C |
| PFMCTL CELLS | 0071C |
| PFMSAV CELLS | 0071C |
| VN CELLS | 0071C |
| SAVE AREA =2 | 0071C |
| SAVE AREA =3 | 0071C |
| XSASW CELLS | 00728 |
| XSA CELLS | 00728 |
| PARAM CELLS | 00728 |
| RPTSAV AREA | 00728 |
| CHECKPT CTR | 00728 |

LITERAL POOL (HEX)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 007C0 | (LIT+0) | 0001005C | 002C1000 | 00001C00 | 00008000 | 00004000 | 00000050 |
| 007D8 | (LIT+24) | 5C5C5C5C | 00481400 | 00001C00 | 00000001 | | |

DISPLAY LITERALS (BCD)

007E8 (LTL+40) ' '

| | |
|---|---|
| PGT | 00728 |
| DEBUG LINKAGE AREA | 00728 |
| OVERFLOW CELLS | 00728 |
| VIRTUAL CELLS | 0072C |
| VIRTUAL EBCDIC NAMES | 00758 |
| PROCEDURE NAME CELLS | 007B0 |
| GENERATED NAME CELLS | 007B0 |
| DCB ADDRESS CELLS | 007B8 |
| VNI CELLS | 007C0 |

*BLL=0001, relative to zero.*] In Command Level CICS, BLL 3 points to the EIB, BLL 4 to the COMMAREA and BLL 5 points to itself.

Using the addresses in the BLL cells is similar to using BL cells. The appropriate BLL cell is located in the TGT and its value is added to the displacement of a field you want to find. A BLL cell with a value of zero indicates a parameter that had not been passed to the program (in batch) or a cell that had not been initialized (in CICS). When fields referred to in a program use a BLL cell that has a value of zero, an 0C4 program check will likely occur. In CICS where initializing BLL cells is the responsibility of the programmer, improper values in BLL cells frequently cause storage violations and 0C4s. [*Initializing BLL cells is no longer the responsibility of the programmer in COBOL II.*] Programmers often do not realize that a BLL cell is required for each 01 level defined in the LINKAGE SECTION and an additional BLL cell is required for each 4096 bytes that an 01 level requires after the first 4096 bytes.

## TEMP STORAGE [*TEMPORARY STORAGE*]

These four temporary storage areas are used by COBOL as internal work areas in which to hold and manipulate data. TEMP STORAGE is used for arithmetic calculations. Some examples of TEMP STORAGE usage are when two fields with different amounts of decimal places are added together, when calculations are done with fields that have different usage modes (COMP and COMP-3) or when calculations are done using display numeric fields.

TEMP STORAGE-2 is used for editing data and manipulation of non-arithmetic data (for example, when a numeric field is moved to a numeric edited field or when a STRING statement is executed). [*TEMPORARY STORAGE-2 is used to process the data that had been processed in both TEMP STORAGE and TEMP STORAGE-2.*] TEMP STORAGE-3 is rarely used. It is used to align synchronized numeric fields. [*TEMPORARY STORAGE-3 is now used to hold the parameter addresses when a CALL is executed. The parameter addresses had been placed in the PARAM CELLS of the VS COBOL TGT.*] When a SEARCH ALL appears in a COBOL program, TEMP STORAGE-4 is used. [*TEMPORARY STORAGE-1 is used for SEARCH ALL processing.*]

```
                    LITERALS                        007C0
                    DISPLAY LITERALS                007E8
                    PROCEDURE BLOCK CELLS           007EC
REGISTER ASSIGNMENT

  REG 6          BL  =5
  REG 7          BL  =4
  REG 8          BLL =4

WORKING-STORAGE STARTS AT LOCATION 000A0 FOR A LENGTH OF 000A0.
```

The temporary storage fields often come into play when a dump occurs. For instance, if fields defined as display numeric are added together and one is not numeric, the 0C7 program check that occurs will not point to the non-numeric field in the DATA DIVISION. The fields will have been PACKed into temporary storage (which never causes a data excep-

**FIGURE 2**

*** TGT MEMORY MAP ***

| TGTLOC | |
|---|---|
| 000000 | 72 BYTE SAVE AREA |
| 000048 | TGT IDENTIFIER |
| 000050 | TGT LEVEL INDICATOR |
| 000051 | RESERVED — 3 SINGLE BYTE FIELDS |
| 000054 | 32 BIT SWITCH |
| 000058 | POINTER TO RUNCOM |
| 00005C | POINTER TO COBVEC |
| 000060 | POINTER TO PROGRAM DYNAMIC BLOCK TABLE |
| 000064 | NUMBER OF FCBs |
| 000068 | WORKING STORAGE LENGTH |
| 00006C | POINTER TO PREVIOUS TGT IN TGT CHAIN |
| 000070 | ADDRESS OF IGZESMG WORK AREA |
| 000074 | ADDRESS OF 1ST GETMAIN BLOCK (SPACE MGR) |
| 000078 | FULLWORD RETURN CODE |
| 00007A | RETURN CODE SPECIAL REGISTER |
| 00007C | SORT-RETURN SPECIAL REGISTER |
| 00007E | MERGE FILE NUMBER |
| 000080 | RESERVED — 4 HALFWORD FIELDS |
| 000088 | PROGRAM MASK OF CALLER OF THIS PROGRAM |
| 000089 | PROGRAM MASK USED BY THIS PROGRAM |
| 00008A | RESERVED — 6 SINGLE BYTE FIELDS |
| 000090 | LENGTH OF THE VN(VNI) VECTOR |
| 000094 | ADDRESS OF IGZEBST TERMINATION ROUTINE |
| 000098 | DDNAME FOR DISPLAY OUTPUT |
| 0000A0 | SORT-CONTROL SPECIAL REGISTER |
| 0000A8 | POINTER TO COM-REG SPECIAL REGISTER |
| 0000AC | CALC ROUTINE REGISTER SAVE AREA |
| 0000E0 | ALTERNATE COLLATING SEQUENCE TABLE PTR. |
| 0000E4 | ADDRESS OF SORT G.N. ADDRESS BLOCK |
| 0000E8 | ADDRESS OF IGZCLNK DYNAMIC WORK AREA |
| 0000EC | RESERVED |
| 0000F8 | POINTER TO ABEND INFORMATION TABLE |
| 0000FC | POINTER TO FDMP/TEST FIELDS IN THE TGT |
| 000100 | ADDRESS OF START OF COBOL PROGRAM |
| 000104 | POINTER TO VNs IN CGT |
| 000108 | POINTER TO VNs IN TGT |
| 00010C | POINTER TO FIRST PBL IN THE PGT |
| 000110 | POINTER TO FIRST FCB CELL |
| 000114 | WORKING STORAGE ADDRESS |

*** VARIABLE PORTION OF TGT ***

| | |
|---|---|
| 000118 | BACKSTORE CELL FOR SYMBOLIC REGISTERS |
| 000120 | BASE LOCATORS FOR SPECIAL REGISTERS |
| 000128 | BASE LOCATORS FOR WORKING-STORAGE |
| 00012C | BASE LOCATORS FOR LINKAGE-SECTION |
| 00013C | BASE LOCATORS FOR FILES |
| 00014C | VARIABLE NAME (VN) CELLS |
| 000154 | INDEX CELLS |
| 00015C | VARIABLE LENGTH CELLS |
| 000164 | ODO SAVE CELLS |
| 00016C | FCB CELLS |
| 000180 | TEMPORARY STORAGE-2 |

| | |
|---|---|
| TGT | WILL BE ALLOCATED FOR 00000198 BYTES |
| FCB00001 | WILL BE ALLOCATED FOR 00000100 BYTES |
| FCB00002 | WILL BE ALLOCATED FOR 00000100 BYTES |
| DCB00003 | WILL BE ALLOCATED FOR 00000060 BYTES |
| FCB00003 | WILL BE ALLOCATED FOR 00000100 BYTES |
| DCB00004 | WILL BE ALLOCATED FOR 00000060 BYTES |
| FCB00004 | WILL BE ALLOCATED FOR 00000100 BYTES |
| DECLVECT | WILL BE ALLOCATED FOR 00000050 BYTES |
| FWA00001 | WILL BE ALLOCATED FOR 00000050 BYTES |
| FWA00002 | WILL BE ALLOCATED FOR 00000050 BYTES |
| WRK-STOR | WILL BE ALLOCATED FOR 000000B0 BYTES |
| SPEC-REG | WILL BE ALLOCATED FOR 00000031 BYTES |

CONSTANT GLOBAL TABLE FOR DYNAMIC STORAGE INITIALIZATION AT LOCATION 0006A8

INITD CODE FOR DYNAMIC STORAGE INITIALIZATION BEGINS AT LOCATION 000D18 FOR LENGTH 00019A

tion). A temporary storage field will be in use at the time that the fields are added together and the ABEND occurs.

Knowledge of the way temporary storage fields are used enables a programmer to more easily follow a Procedure Map and resolve the problems causing an ABEND. In the PMAP temporary storage fields are shown as TS=x, TS2=x, TS3=x and TS4=x where x is the displacement (relative to 1) into the temporary storage area. [*In the Procedure Map temporary storage fields are indicated as TS1=x, TS2=x, TS3=x and TS4=x where x is the displacement (relative to 0) into the temporary storage area.*]

## VLC [*Variable-Length Cells*]

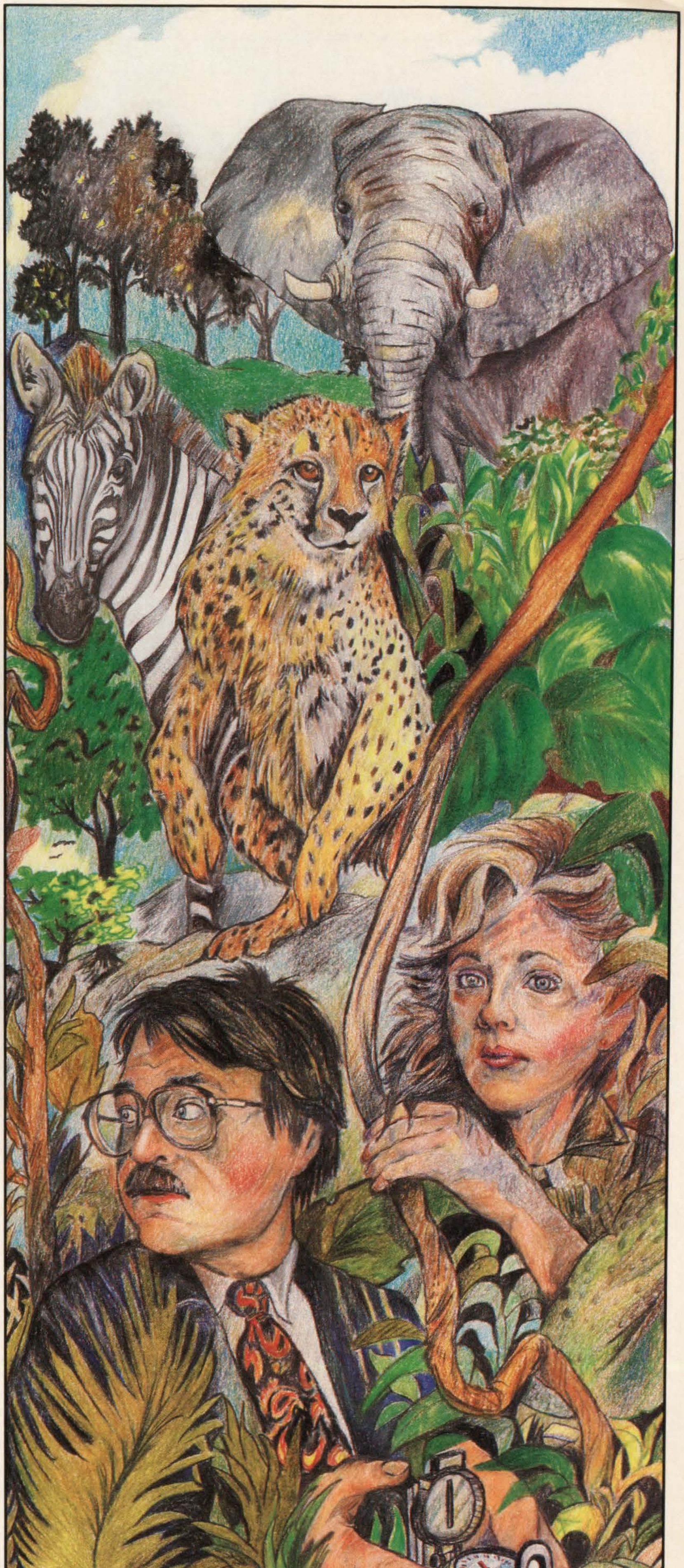
Fields in the DATA DIVISION that are defined with the OCCURS . . . DEPENDING ON clause are variable in length. At the time of an ABEND it may be of value to determine the current length of one of these fields. Each VLC contains the length of one variable-length field in a program. Every cell is a halfword in length since an elementary field may be up to 32K. [*Each cell is a fullword since an elementary field may be up to 16MB.*] The cells are in the same order as the OCCURS . . . DEPENDING ON fields are in the DATA DIVISION.

## Save Area [*72-Byte Save Area*]

Although COBOL's Save Area can often be found elsewhere in a dump and COBOL programmers normally are not concerned with it, it should be noted that the SAVE AREA is the first field in the TGT. In storage, the TGT directly follows WORKING-STORAGE. [*The TGT does not directly follow WORKING-STORAGE in COBOL II.*] If the end boundary of WORKING-STORAGE is inadvertently exceeded, the SAVE AREA is the first area to get overlaid. The overlay is possible when a subscripting or indexing problem occurs, particularly when the problem occurs in a table defined near the end of WORKING-STORAGE. If this happens, many types of ABENDs can occur after the program issues a GOBACK or a STOP RUN to finish processing. I have seen it happen a number of times and it is a valuable piece of information to keep in mind whenever an ABEND occurs after a program seemed to have completed. If you think this situation has happened, inspect the SAVE AREA (and perhaps the fields following it as well) and carefully examine the data. Determine if it might contain information that actually belongs in WORKING-STORAGE.

## FIB Cells [*FCB Cells*]

A File Information Block (FIB) exists for each VSAM file in use by a COBOL program. When a program begins execution, each FIB cell points to an FIB that was made part of the COBOL program during compilation. As each file is opened, a File Control Block (FCB) is constructed. Its address overlays the corresponding FIB address in the TGT. It is easy to distinguish whether an FIB cell is



**F I G U R E 3**

*** TGT MEMORY MAP ***

| PGMLOC | TGTLOC | |
|---|---|---|
| 000648 | 000000 | 72 BYTE SAVE AREA |
| 000690 | 000048 | TGT IDENTIFIER |
| 000698 | 000050 | TGT LEVEL INDICATOR |
| 000699 | 000051 | RESERVED — 3 SINGLE BYTE FIELDS |
| 00069C | 000054 | 32 BIT SWITCH |
| 0006A0 | 000058 | POINTER TO RUNCOM |
| 0006A4 | 00005C | POINTER TO COBVEC |
| 0006A8 | 000060 | POINTER TO PROGRAM DYNAMIC BLOCK TABLE |
| 0006AC | 000064 | NUMBER OF FCBs |
| 0006B0 | 000068 | WORKING STORAGE LENGTH |
| 0006B4 | 00006C | POINTER TO PREVIOUS TGT IN TGT CHAIN |
| 0006B8 | 000070 | ADDRESS OF IGZESMG WORK AREA |
| 0006BC | 000074 | ADDRESS OF 1ST GETMAIN BLOCK (SPACE MGR) |
| 0006C0 | 000078 | FULLWORD RETURN CODE |
| 0006C2 | 00007A | RETURN CODE SPECIAL REGISTER |
| 0006C4 | 00007C | SORT-RETURN SPECIAL REGISTER |
| 0006C6 | 00007E | MERGE FILE NUMBER |
| 0006C8 | 000080 | RESERVED — 4 HALFWORD FIELDS |
| 0006D0 | 000088 | PROGRAM MASK OF CALLER OF THIS PROGRAM |
| 0006D1 | 000089 | PROGRAM MASK USED BY THIS PROGRAM |
| 0006D2 | 00008A | RESERVED — 6 SINGLE BYTE FIELDS |
| 0006D8 | 000090 | LENGTH OF THE VN(VNI) VECTOR |
| 0006DC | 000094 | ADDRESS OF IGZEBST TERMINATION ROUTINE |
| 0006E0 | 000098 | DDNAME FOR DISPLAY OUTPUT |
| 0006E8 | 0000A0 | SORT-CONTROL SPECIAL REGISTER |
| 0006F0 | 0000A8 | POINTER TO COM-REG SPECIAL REGISTER |
| 0006F4 | 0000AC | CALC ROUTINE REGISTER SAVE AREA |
| 000728 | 0000E0 | ALTERNATE COLLATING SEQUENCE TABLE PTR. |
| 00072C | 0000E4 | ADDRESS OF SORT G.N. ADDRESS BLOCK |
| 000730 | 0000E8 | ADDRESS OF IGZCLNK DYNAMIC WORK AREA |
| 000734 | 0000EC | RESERVED |
| 000740 | 0000F8 | POINTER TO ABEND INFORMATION TABLE |
| 000744 | 0000FC | POINTER TO FDMP/TEST FIELDS IN THE TGT |
| 000748 | 000100 | ADDRESS OF START OF COBOL PROGRAM |
| 00074C | 000104 | POINTER TO VNs IN CGT |
| 000750 | 000108 | POINTER TO VNs IN TGT |
| 000754 | 00010C | POINTER TO FIRST PBL IN THE PGT |
| 000758 | 000110 | POINTER TO FIRST FCB CELL |
| 00075C | 000114 | WORKING STORAGE ADDRESS |
| | | *** VARIABLE PORTION OF TGT *** |
| 000760 | 000118 | BACKSTORE CELL FOR SYMBOLIC REGISTERS |
| 000768 | 000120 | BASE LOCATORS FOR SPECIAL REGISTERS |
| 000770 | 000128 | BASE LOCATORS FOR WORKING-STORAGE |
| 000774 | 00012C | BASE LOCATORS FOR LINKAGE-SECTION |
| 000784 | 00013C | BASE LOCATORS FOR FILES |
| 000794 | 00014C | VARIABLE NAME (VN) CELLS |
| 00079C | 000154 | INDEX CELLS |
| 0007A4 | 00015C | VARIABLE LENGTH CELLS |
| 0007AC | 000164 | ODO SAVE CELLS |
| 0007B4 | 00016C | FCB CELLS |
| 0007C8 | 000180 | TEMPORARY STORAGE-2 |

| | |
|---|---|
| TGT | LOCATED AT 000648 FOR 00000198 BYTES |
| FCB00001 | LOCATED AT 0007E0 FOR 00000100 BYTES |
| FCB00002 | LOCATED AT 0008E0 FOR 00000100 BYTES |
| DCB00003 | LOCATED AT 0009E0 FOR 00000060 BYTES |
| FCB00003 | LOCATED AT 000A40 FOR 00000100 BYTES |
| DCB00004 | LOCATED AT 000B40 FOR 00000060 BYTES |
| FCB00004 | LOCATED AT 000BA0 FOR 00000100 BYTES |
| DECLVECT | LOCATED AT 000CA0 FOR 00000050 BYTES |
| FWA00001 | LOCATED AT 000CF0 FOR 00000050 BYTES |
| FWA00002 | LOCATED AT 000D40 FOR 00000050 BYTES |
| WRK-STOR | LOCATED AT 000D90 FOR 000000B0 BYTES |
| SPEC-REG | LOCATED AT 000E40 FOR 00000031 BYTES |

pointing to an FIB or FCB. If the first byte pointed to by the address is an 'F,' it is pointing to an FCB; if it is an 'I,' it is an FIB.

[*In COBOL II FCBs and FIBs exist for both VSAM and non-VSAM files. In the listing of the fixed portion of the TGT you can find the NUMBER OF FCBs and the POINTER TO FIRST FCB CELL. The pointer indirectly points to the FCBs in use by the program; it points to a list of four-byte addresses, one address pointing to each FCB. Each FCB begins with the constant 'FCB' and is followed by two bytes (alphanumeric) indicating the number of the FCB.*

*Each FIB begins with the constant 'FIB.' At a displacement of six bytes it contains the DDNAME of the file to which it pertains. The address of the FIB is in its associated FCB at a hexadecimal displacement of 'A4.'*]

At hexadecimal displacements of '60' and '63' in the FCB is extremely valuable information concerning VSAM errors in a COBOL program. At X'60' is the file status code from the last I/O regardless of whether or not a program requested the file status with a FILE STATUS clause in the SELECT. [*Additional error information is now available in the extended VSAM feedback area in the file status. In addition to the two-byte file status that was previously used, a six-byte VSAM status is available. The information consists of three, two-byte binary fields and is only filled in when the file status is not zero. The first two bytes contain the VSAM return code (Register 15). When the return code is not zero, the next two fields are filled in. They contain the VSAM function code and the feedback code.*]

At X'63' is error information not normally available to a COBOL programmer, the error code from the VSAM Access Control Block (ACB) if an OPEN or CLOSE was just done or the feedback code from the VSAM Request Parameter List (RPL) on all other VSAM I/O. [*If all the detailed information of the VSAM RPL is needed, the FCB (hexadecimal displacement 'A8') points to a work area called GMAREA. GMAREA contains the address of two RPLs (displacements '1A' and '1E'). The first RPL is used for read/write access while the second RPL is used for read/only access.*]

## DCB Address Cells in the Program Global Table (PGT) [*DCB Locations listed after the TGT*]

While the Data Control Block (DCB) cannot be located through the TGT, this article would seem incomplete without showing how to find it. When information relating to a non-VSAM file is required, the DCB contains important information such as the record format, record length, blocksize, number of buffers and the record address of the current record.

The current record can also be found using the BL cells. [*The current record can also be found using the BLF cells. The address of the current record for either VSAM or non-VSAM files also can be found in the FCB at a hexadecimal displacement of 'AC'.*]

Each DCB ADDRESS CELL is a fullword. The DCB addresses appear in the

*See* TGT *page 94*

# Change Management In The DB2 Environment

By Stephanie A. Markham

Now that the dust has settled, it is clear that DB2 is the DBMS of the future, at least of IBM's foreseeable future. It is already a highly successful product installed in thousands of sites with the number of installed sites doubling in just the last year. IBM has placed a strategic emphasis on the relational DBMS approach by including DB2 and its Structured Query Language (SQL) as key components of the SAA direction. While the technology provides a great many advantages in the areas of data transparency, programmer productivity and general information distribution, the implementation of relational technology through DB2 and SQL have brought to the data processing world a number of added challenges, especially in the areas of managing and controlling change to DB2/SQL applications.

One of the primary strengths of the relational approach is the ability to structure a query for data without knowledge of or concern for the underlying physical structure within which the data resides. When correctly coded, a request should produce accurate information, regardless of the database search path used to obtain that information. And, for the most part, this is the case with SQL requests issued against DB2 databases. However, the actual access path used within the DB2 database does have serious ramifications with regard to the performance of the request.

When relational technology is being used for ad hoc query, requests can be dynamically interpreted at run time by the DBMS or the run-time query environment to determine the optimal path through the database. This way, the cost of up-front logic for figuring out an optimal path is traded against the performance nightmare of having to search an entire database for an accurate answer to a request.

DB2 contains some sophisticated logic for determining the optimal path through the database. During interactive requests against DB2 databases, the SQL is interpreted during request execution to determine the optimal path through the database. This interpretation process is time well spent, resulting in adequate performance and resource consumption during ad hoc query execution. However, when DB2 is used as a production, transaction-driven DBMS, some unexpected complications arise.

In general, performance and resource consumption issues become much more critical within production, transaction-driven applications regardless of the data structures accessed by those applications. To make DB2 perform as a production database, it makes little sense to interpret and optimize each SQL request as it is encountered. The procedure would simply result in too much overhead during execution to be even remotely acceptable in a production application. To resolve this issue, IBM took the approach of the *pre-execution bind,* essentially a process that optimizes the database access statements in advance, producing an *executable* form of the SQL. At run time, this executable SQL is used to perform the database access, providing improved production execution performance.

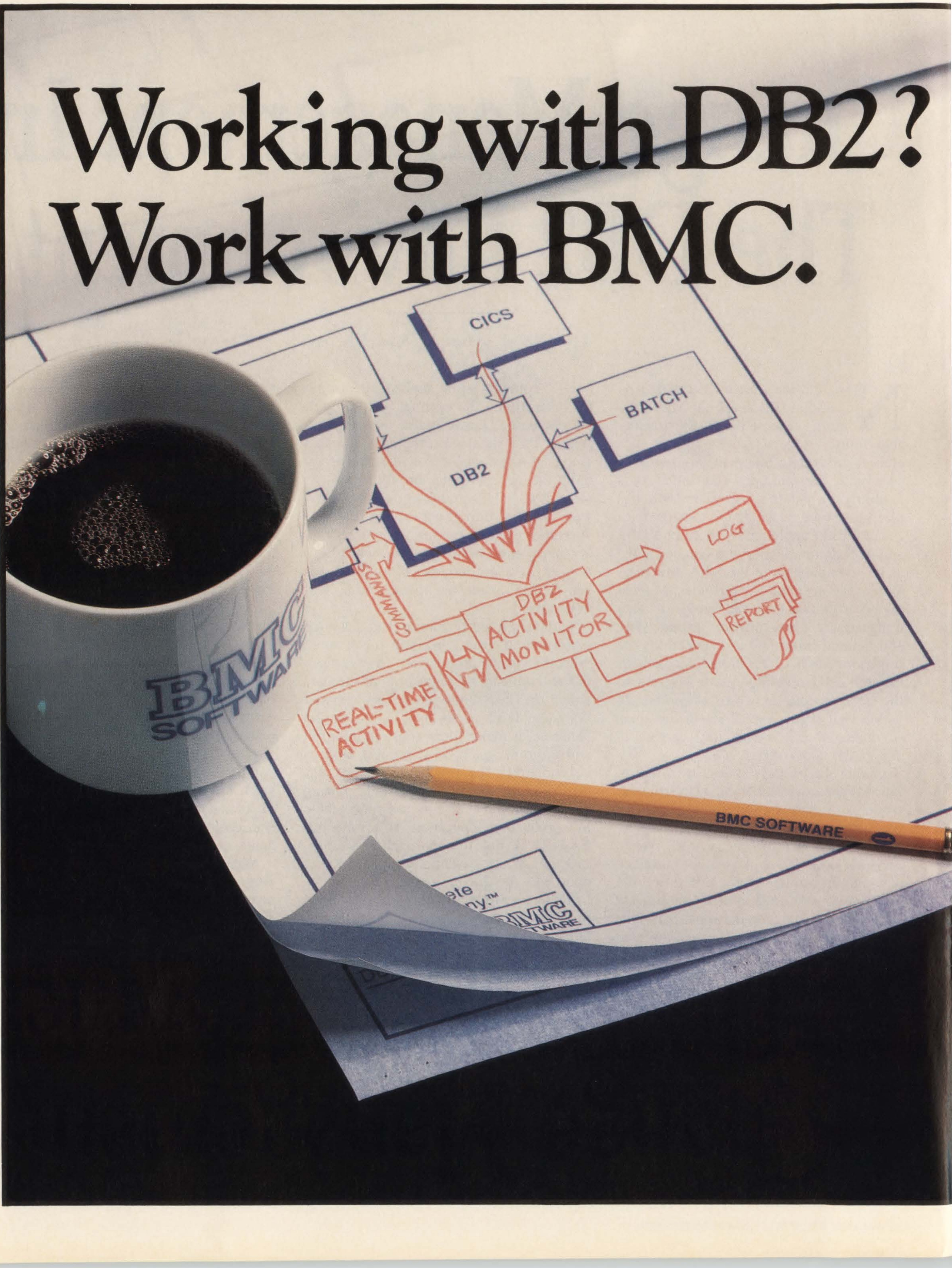
## What Are the DB2 Change Management Problems?

When a DB2 program is compiled, it first goes through a preprocessor. The preprocessor extracts the SQL statements, using them to build a Database Request Module (DBRM) which it then stores in a separate PDS. It also time stamps both the program and the DBRM. The program is then compiled and link-edited into a load module, carrying the time stamp through this process. Before the program can actually be executed, its DBRM must go through the *bind* process. The bind does a number of things, including validation of the SQL syntax. Ultimately it produces or causes the update of an application plan containing the optimized SQL execution code path and the resolution of the relationships between the SQL statements and the database. The plan is then stored in the DB2 catalog. The DBRM time stamp for the program within the plan is carried into the DB2 catalog as well. Figure 1 illustrates this process.

To some extent, the application plan shields the programs from certain database changes. For instance, if the database structure is altered in certain ways (such as the elimination of an index) a plan is dynamically rebound when a program executing with it runs. This lengthens the execution time for that one program but re-creates the executable form of the database access path for future executions. Program changes, on the other hand, are not so easily reconciled.

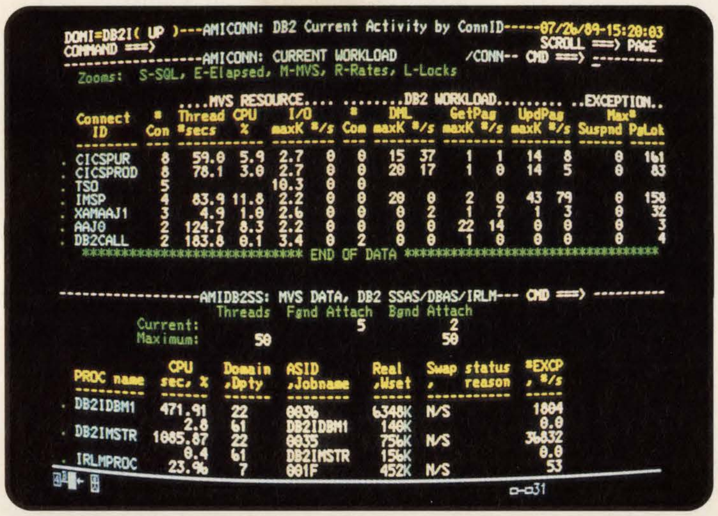
When an SQL program is executed, the corresponding application plan must be identified. DB2 ensures that the program and the associated plan are synchronized by comparing the time stamp for the program's DBRM in the application plan with the time stamp in the program. If the program time stamp is newer than the DBRM time stamp, the program will fail at run time. Consequently, if a program participating in a plan changes and needs to be recompiled, its DBRM must also be rebound.

To make matters more complicated, a program can participate in any number of application plans. Theoretically, a change to a single program could necessitate the rebinding of a number of plans. So, while the performance of production SQL ap-

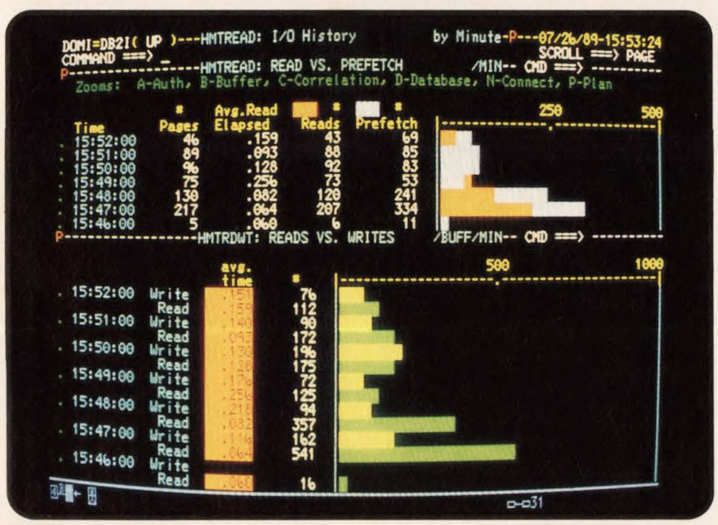

Working with DB2?
Work with BMC.
CICS
BATCH
DB2
LOG
COMMANDS
DB2
ACTIVITY
MONITOR
REPORT
REAL-TIME
ACTIVITY
BMC SOFTWARE

# DB2 ACTIVITY MONITOR Gives You Complete Performance Data

DB2 ACTIVITY MONITOR provides more functionality than any other DB2 monitor currently available. Because BMC has worked extensively with DB2, developing a complete line of DB2 products, we know what you need to keep your system running efficiently. We used that knowledge to develop DB2 ACTIVITY MONITOR, making it the most complete source of information available for people who work with DB2.




**CURRENT WORKLOAD screen.** Performance data summary for DB2 transactions, grouped by connection type. Transactions not performing at peak efficiency can be selected for more detail.




**READ VS. PREFETCH screen.** Recent historical breakdown of detailed I/O activity. Allows easy analysis of trends in DB2 workload.



**Features of DB2 ACTIVITY MONITOR include:**

- Displays and collects real-time and historical data from MVS, IMS, CICS and DB2.
- Generates batch reports from historical data and SMF.
- Supplies DB2 Console function.
- Conserves resources by providing control of Trace function.
- Notifies users of exceptions and sets limits dynamically.
- Provides Customization function for data and format.
- Displays increasingly detailed information with ZOOM function.

To begin a 30-Day-Plus Free Trial or for more information, fill out and mail the coupon below. Or for a faster response call **1-800-841-2031.**

BMC SOFTWARE

Australia (61-3) 819-6733
England (0276) 24622
France (16-1) 48 77 77 77
Italy (02) 48193845
Japan (03) 837-8651
West Germany (069) 66 4060

**The Complete DB2 Company.™**

IBM Business Partner

**BMC Software** • P.O. Box 2002 • Sugar Land, TX 77487-2002

☐ Please contact me about a 30 Day-Plus Free Trial of **DB2 ACTIVITY MONITOR.**

☐ Please contact me with more information about **DB2 ACTIVITY MONITOR.**

☐ Please contact me about BMC's complete line of DB2 products.

Name ______

Title ______

Company ______

Address ______

City ______ State/Prov. ______ Zip/P.C. ______

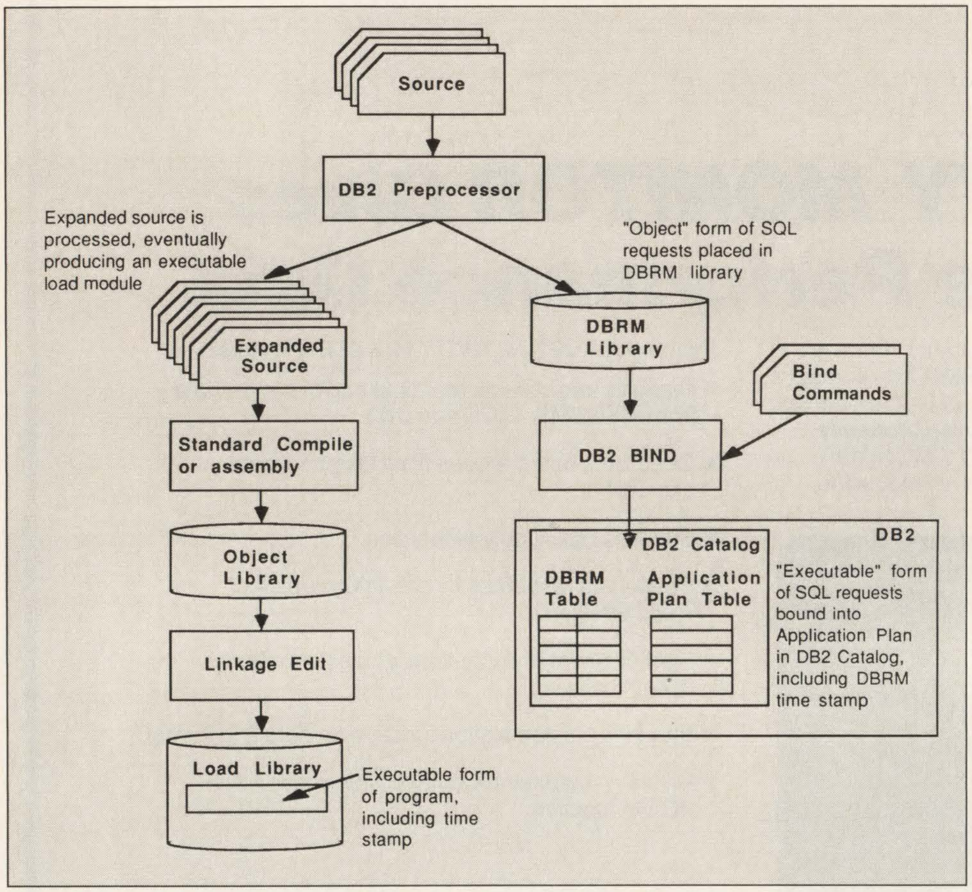
Phone ______

☐ Please contact me about BMC's no-cost Technical Reviews, Workshops and DB2 Round-ups in my area.

MJ

*CIRCLE #103 on Reader Service Card* ▲

FIGURE 1




plications is greatly enhanced by virtue of the executable access to the database structure in the form of the application plan and while database changes can often be made independently without significantly affecting program execution, the management of change in the DB2 production application environment is made significantly more complex.

## New Technology, Old Change Management Issues

The change management issues in the DB2 environment are really not very different from those encountered while trying to manage the change and distribution of any large, complicated application with independently constructed parts. Change to individual parts must be controlled and tracked to ensure accountability for and auditability of change. Transformation of source to its executable form(s) needs to be controlled to ensure that what is executing is definitely tied to the source from which it was created. And the impact of changes to related parts must be accurately identified, both to determine the entire effect of a seemingly innocuous change and to successfully implement, migrate, release and distribute production application changes.

The issues raised by the DB2 bind process are no more complex than those presented by the traditional data processing environment, an example being those caused by link-edited load modules. As with the bind process, the linkage-editing process takes a number of outputs from different transformation processes (compiles or assemblies) and combines them to form an executable entity. Since a single program object module can be linked into any number of load modules, a change to a component part (an object module) contained in multiple load modules generally requires the re-linkage of all of the affected load modules. Unfortunately, as with the DB2 bind process, there are no native tools which provide the information required to adequately control change to this relatively complex structure.

In effect, the DB2 pre-execution bind has simply caused us to re-visit a change management issue that has been with us for some time. And the construct bringing this issue to the forefront today has not replaced older constructs, but rather, is being used in conjunction with them, adding significantly to the complexity of managing and controlling application change to DB2 production applications. Additionally, many of the processes involved in changing DB2 applications can be performed on-line, have no retained source form when performed on-line and take effect immediately.

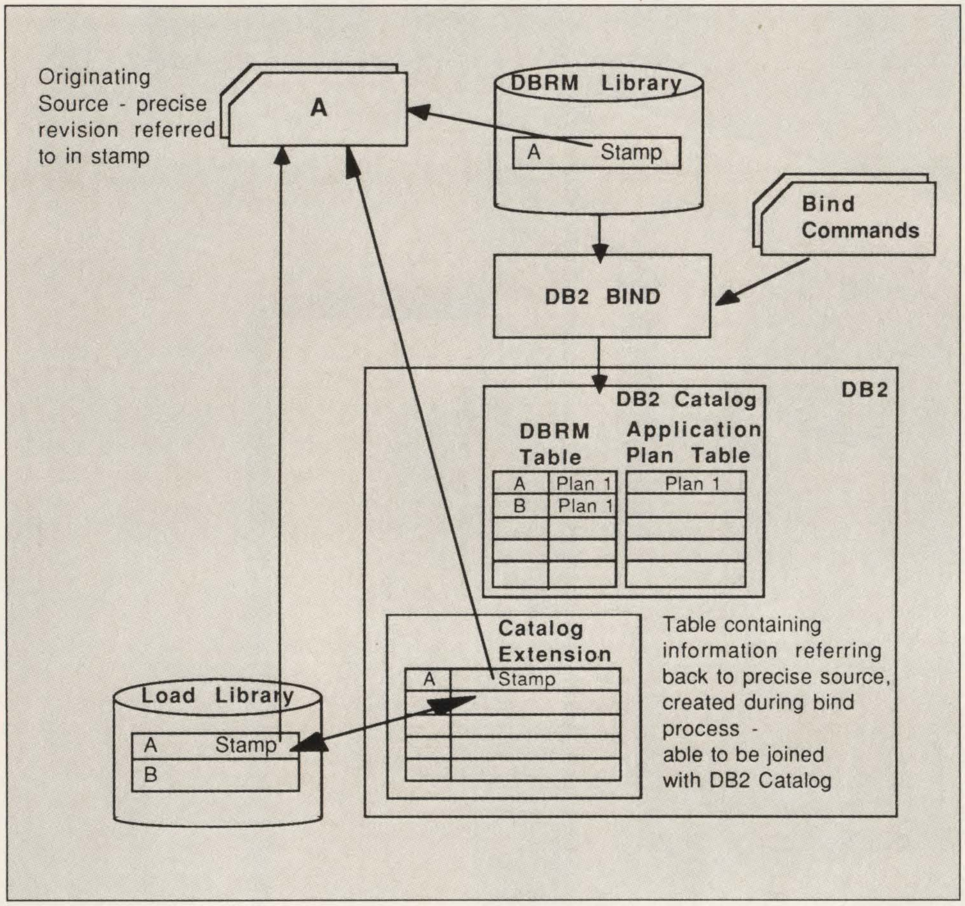
## What Are The Solutions?

There are several solutions to the change management problems inherent in the DB2 applications development environment. Many of these solutions are provided by currently available change management techniques and methodologies. By performing functions over which there is currently no control under the auspices of an automated change control and management system, regardless of whether they are unique DB2 functions or traditional functions, most of the problems we currently encounter can be resolved.

A complete solution to the management of link-edited load modules starts with control of the program source from which they are created. If a control methodology is used to manage all program source, control the creation of the outputs from that source and force those outputs to refer back to the exact source version(s) from which they were created, then it is possible to accurately track back from a composite load module to all of the source modules from which it was created down to the exact source versions. If that same change management methodology is capable of dynamically capturing *all* of the configuration information from the output creation process (including all subordinate structures complete with exact source version information for each subordinate structure) then it is possible to determine the precise makeup of a particular executable. The dynamic tracking of accurate component information, especially if it is kept and tracked over time, has the added benefit of providing a precise method of performing impact analysis on component changes. When a subordinate structure changes, the impact of that change is immediately obvious — and immediately addressable.

Now apply this methodology to the DB2 bind. The bind subcommand that controls the critical process of building and maintaining application plans can be issued through DB2 Interactive (DB2I) or from TSO through the DSN processor. If this process is performed on an ad hoc basis, as its execution methods encourage, the window of opportunity for the destruction of a production application, even with appropriate security in place, is large. If in-

FIGURE 2




stead, the process of creating and maintaining DB2 applications from source change through both the bind and linkage-edit processes is performed under the auspices of an automated change management system, the synchronization problems that can be caused by DB2's requirement for a pre-execution bind can be minimized or even eliminated.

To control this process, it is critical that the individual changes being made to the initial source be tracked and that these changes trigger the process of output creation. It is not sufficient that the process simply be triggered. The preprocessing of SQL programs must be done under the *control* of a change management system not just to ensure that the outputs are actually being created from the controlled source through the controlled process, but also to allow that process to stamp the outputs to refer back to the precise version of the source from which they were created. Additionally, both the linkage-edit process that creates the load module and the bind process that creates the executable form of the database access statements should be performed under the auspices of the change management system. Only when source for these processes is stored and maintained (bind subcommand statements and linkage-editor control cards), can the processes be accurately recreated. And when this controlled source is used to automatically perform the processes by which the executable forms are created, definitive ties back to the originating source can be carried into the ultimate executable forms (the load module and the DB2 application plan) regardless of their complexity.

Within the DB2 structures, tying the DB2 catalog information back to the program source is easier said than done. IBM does not, for very good reasons, allow user modifications to the application plan and DBRM table structures themselves. The only way to physically tie into DB2 catalog structures, such as the application plan and DBRM tables, is to create a table in the catalog extension containing the tie-back information along with a relational key into the DBRM and application plan tables as shown in Figure 2. This way the new table can be joined with the DBRM and application plan tables to provide all the necessary cross-referencing information.

Now consider the earlier problem stemming from the fact that a small program change may require the rebinding of the DBRM into any number of application plans — without any definitive way to know which ones require rebinding. When automatically controlled and maintained, both application plans and load modules contain information tying everything definitively back to the creating source modules — making the process accurate and reliable. Simple cross-referencing can determine which plans may require rebinding due to out-of-date DBRMs and also which load modules require re-linking due to out-of-date subordinate components. In the DB2 environment where an out-of-sync condition between a program and its database access information can have delayed execution implications, being able to use this information to control the rebinding process and being able to perform the re-bind from existing controlled source can significantly reduce the problems caused by the whole bind process. And because all of the information needed is available in a controlled environment, it can be used as the driving and controlling force behind DB2 application migrations.

## Bad News, Good News

As usual, the implementation of a new technology in an old data processing world has brought us both bad news and good news. The bad news is that some problems persist throughout the evolution of data processing and DBMS technology. The good news is that today there are technological solutions to these problems provided by systems that were not available a couple of years ago. And, fortunately, these systems can resolve both the old problems and the new wrinkles.


*ABOUT THE AUTHOR*

*Stephanie A. Markham has been involved with change control and configuration management since she joined Business Software Technology in 1987. During her time at BST, Markham has been involved in the support and implementation of advanced change control and configuration management technology in a variety of application development environments.*

# The Future Of Computer Center Automation

By Howard W. Miller

Information technology evolves in one direction — from the simple to the complex. Computers started out simple. They had no operating system; they processed a single program at a time and they required human intervention. However, the speed of the CPU increased, high-speed tape units were added, operating systems were introduced and direct access storage devices were added. Also, the operating systems became multi-tasking: on-line processing was introduced and the users increased from one to a thousand simultaneous users. Computing quickly became complex.

Although more complex, computer centers were based on a single large computer. Large computer centers might have had multiple computers that were independent or loosely coupled, but the concept remained the same. In the 1980s, this changed. The midrange, *departmental* computer became popular, the Personal Computer (PC) was introduced on a large scale and the multiple processor computer was introduced. The business case for choosing among these computers was different and it became economical to do some applications on one type of computer rather than on another.

Applications, such as spreadsheets and word processing, were easy to achieve on a PC and some applications, such as manufacturing systems, became easy to install on a midrange computer. More than ever, the selection of computers became a software decision rather than a hardware decision.

When viewed in the context of natural evolution, the evolution of information technology from simple to complex and from single processor to multiple processor is expected. In nature, organisms (both plant and animal) evolve from the simple to the complex and from single celled to multicelled. Computer evolution is paralleling this pattern of natural evolution.

*Computer center automation is achieved by systematically removing all manual computer center tasks and by replacing them with automation tools.*

As information technology becomes increasingly complex, there is a continuing need to look back and isolate obsolete technologies. There is a need to eliminate technologies that no longer fit into the environment and that make moving forward difficult. On one hand, information technology is becoming increasingly complex and, on the other, there is a need to look back and reduce this complexity by eliminating technologies that no longer fit into the environment.

Furthermore, as the technology moves forward, the opportunities for applying the technology increase. As a result, there are typically four or five opportunities to apply the technology for every one that an organization can economically afford to implement. Under these circumstances, the strategy that seems to work best is to choose the path of least resistance. Implement the technologies that the organization is receptive to or that are the easiest to implement and defer the others to a more opportune time.

A basic understanding of these two concepts, the evolution of information technology from the simple to the complex and the strategy of implementing technology using the path of least resistance, makes it significantly easier to understand the future direction of unattended computer center operation.

## Direction Of Computing

There is a parallel between natural evolution and the evolution of information technology. In natural evolution, life began as a single cell. The cells formed colonies. The cells in the colony eventually formed a symbiotic relationship and some cells began to perform specialized functions. The cells merged into colonies and the colonies evolved into a complex multicelled structure.

The evolution of computing is paralleling this process. Computers started out as small, single processing units and grew into large complex mainframe computers. In addition, midrange computers such as the DEC VAX, the IBM Systems 36 and 38 and the Hewlett Packard 3000 were introduced and these machines too grew in size and complexity. Finally, the PC and the workstation were introduced.

As midrange and PCs became more prevalent, two things began to happen: specialization and networking. Many were dedicated to specialized applications such as office automation, inventory management, manufacturing or scientific research. Furthermore, these computers are networked together into complex computing structures, similar to a network of cells.

To better understand the future direction of unattended computer center operation, it is beneficial to look at the classes of computers and the communication networks that tie them together.

### *Mainframe Computers*

In the 1960s, the first modern mainframe computers, the IBM 360s, were little more powerful than the PCs of today. They were relatively slow machines (less than one MIPS) and initially had little main memory (32K to 64K). It was not until the IBM 370 class computer that computers routinely had memories in the range of a million bytes or more.

Since the 1960s, mainframe computers have continuously grown in speed and capacity from one generation to the next at a rate of about 100 percent. Probably the most significant aspect of this evolution is the increasing complexity of the computers. A computer became a network of computers. Specialized computers were used as control units for tape drives, for disk drives, for communications and for I/O devices such as printers and terminals. All these devices are actually small processors in themselves. In fact, as the disk drive series evolved, the control units took on considerable specialization, including channel switching and cache memory. The computers not only became larger and faster, but also they became clusters of specialized computers.

Finally in the 1980s, IBM popularized the concept of a central computer that consisted of multiple processors. Computers had two, three or four processors within the machine. This is obviously the future direction of mainframe computers. Mainframe computers are going to continue to have more processors within a single box. The operating system makes these processors appear as a single image to the computer and to the application program.

The mainframe computer, therefore, started as a simple processing unit that has evolved into an extremely complex machine composed of multiple special purpose processors contained within a single device. This single image device is networked to other computers either directly attached to it or attached through a communications network.

### *Departmental Computers*

The history of small business machines (commonly referred to as minicomputers or, more recently, departmental computers) is almost as old as the mainframe computer. The difference between the two is that technology has fostered the development of most business systems using large mainframe computers. It is only recently that a wealth of software has become available for the departmental computer. Furthermore, the cost of small

# Only one automated operations product is a perfect fit.


Service level management began with OMEGAMON®. So we precision-crafted automated operations to fit perfectly with OMEGAMON. Because automation is just another part of performance management.

AF/OPERATOR™ learns about MVS, CICS, IMS, or DB2 problems from OMEGAMON. So as soon as an exception is triggered, AF/OPERATOR can issue commands to pinpoint the problem. Then speed a message directly to your best troubleshooter for fast resolution. It can even take action itself to improve service at machine speed.

Maybe the CICS claims application is way over its service level. AF/OPERATOR can issue OMEGAMON commands to determine which resource is delaying claims. Or if other jobs are causing the delay. It can then automatically track down the user to see if the conflicting job can be cancelled.

Of course, AF/OPERATOR lightens the load of your operations staff with console message management, automatic startup of time-dependent activities, reliable IPLs, and orderly shutdowns. And it simplifies your operations by becoming the single point of control for all your systems — MVS, IMS, and JES to name a few.

For over ten years, data center managers have trusted the performance of their systems to Candle. Now you can automate your data center with the same confidence. Expect AF/OPERATOR to fit your data center perfectly — now and in the future. For more information, call Terry Forbes at **(800) 843-3970**.

**!Candle®**



***CIRCLE #131 on Reader Service Card ▲***

departmental computers has now declined to the point where it is cost effective to use them for applications previously restricted to the mainframe computer.

The high capacity and relatively low cost of departmental computers seem to imply that this technology is evolving faster than the mainframe computer. In fact, most departmental computers, if not all, are designed exclusively for on-line processing while the common mainframe computers are essentially batch processors converted to on-line processing. As software systems become available for departmental computers, they are typically superior in both cost and performance to comparable software on the mainframe computer. As a result, the departmental computer has experienced a surge in popularity over the last decade.

However, most departmental computers are used essentially the same way as mainframe computers. A whole series of different applications are integrated together into a single general purpose departmental computer. Few organizations have attempted to dedicate departmental computers to single application (payroll, accounts payable, purchasing and general ledger, for example). In almost all cases, the departmental computer is used as a small general purpose computer.

The one notable exception is the manufacturing system. However, even here, a range of application systems runs within the manufacturing system shell on the same departmental computer (order entry, inventory, MRP, shop floor control and probably word processing and electronic mail), much the same as they do in mainframe-based manufacturing systems.

It is not common for departmental computers to be dedicated to a single application and for multiple departmental computers to be tied together in a network passing information from one system to the next or one computer to the next. Although networking computers could provide distributed or decentralized capabilities for specific departments, few organizations have attempted to do so. This is probably because of the pervasive mainframe attitude that exists within business computing and the difficulties in managing information that is distributed across multiple sites.

However, as more organizations come to recognize that there is a wide variety of software available and that departmental computers are cost effective, they are realizing that it is possible to have mainframe computing without mainframe computers. They are linking together many small departmental computers so that the organization has single image computing composed of many small computers. The intriguing aspect of this tactic is that it is possible to build this network without displacing the mainframe computer. The mainframe computer can be a permanent part of the network or, eventually, it can be phased out. But most likely, it will become a more or less equal partner in the network.

### The Personal Computer

In the late 1970s and the early 1980s, a whole new class of computers was introduced: the PC. It has the power and capacity of the early mainframe computer, but it is designed as a personal computing workstation. The most common applications for the PC are word processing, spreadsheets and small database applications. The PC is intended to satisfy the needs of individual workers.

The PC created a new market. Suddenly, people who had never thought of using computers were using the PC. Applications were designed that were not cost effective on mainframe or departmental computers. Furthermore, after the initial introduction, the power and the capacity of the PC quickly doubled or tripled and then leveled off at a price/performance ratio that organizations were prepared to pay.

The power and the capacity of the PC expanded. As it expanded, the cost increased until the cost of the computer was no longer in synchronization with what a person was prepared to pay for a single workstation on a desk. As a result, the capacity of the machine leveled off. Instead, a need developed for PCs to talk to each other, to talk to departmental computers and to talk to mainframe computers.

The evolution of the PC shifted from increasing in size to adding features onto an existing platform. Boards became available to allow the PC to talk to the mainframe; networks were introduced to allow the PCs to talk to each other. Large capacity direct access devices became available and designated computers became file servers for other machines. A wealth of applications software became available. The typical PC owner would spend from 20 to 50 percent of the initial purchase price every year in order to add software and hardware features to the computer.

Furthermore, PCs became front-end processors for the larger, more powerful departmental or mainframe computer. This caused organizations to end up with a hierarchy of computers: the PC replacing the "dumb" terminal on the desk and the PC talking to the departmental computer with pass-through capabilities into a mainframe computer.

### Workstations

The last class of computers is the workstation. The workstation is similar in many ways to the PC, except for its computing power. In some cases, workstations are in the one to three MIPS range — more powerful than some departmental computers and almost equivalent to the entry level mainframe computer.

The workstation is designed to provide a tremendous amount of computing power to scientific and engineering applications or any application requiring a tremendous amount of computing but not requiring large amounts of I/O activity. In terms of architecture, the workstation is similar to the PC. However, in terms of processing capacity, it is similar to the mainframe.

Workstations are used primarily in engineering and research where there is a need for large amounts of processing and little application software. They do not have complex, user-friendly operating systems and there is not a wealth of applications software available. What they provide is sheer raw computing power at a reasonable price. Workstations have windowing capabilities similar to the Apple Macintosh. However, because of the complex nature of the computer, they still are not as easy to use. Ease of use is usually a feature of a proprietary operating system and not part of the computer itself.

Workstations can either be stand-alone processors (diskless) or they can be file servers with large capacity disk storage units. In many cases, workstations are purchased without any disk capacity. They are networked together with one machine becoming a file server. The beauty of the workstation is the sheer computing power that is available at a relatively low cost.

### Communication Networks

Computers have evolved into four layers. Mainframe computers process work for hundreds to thousands of individual computer users and multi-purpose departmental computers service from 10 to a 100 computer users. PCs service a single person, but they are probably networked to a mainframe computer. Another layer is workstations that are high-powered computers servicing an individual and

networked to a file server.

Each has a different price range and each satisfies a different market. Each has a different set of available software and, in many cases, there is a lot of overlap in capabilities between one level of machine and another. Some of the functions that can be done on a PC can be done on a mainframe and vice versa. All justification aside, once a computer is installed, there is an almost universal desire to communicate with other computers.

Most organizations have a complex computing environment consisting of multiple computers and multiple computer suppliers. What started out as a single computer has evolved into a network of computers. This network is providing information technology with the ability of having single-image computing across multiple computers.

A computer user on the network, whether (s)he is using personal computers, workstations, departmental computers or mainframes, can access anything on the network. The objective of the network is to provide the computer user with access to an application system irrespective of the location of the hardware platform. What is developing is a community of computers similar to the complex collection of cells with the communications network tying them together in the same way as the nervous system.

## Computer Center Automation

Through the early 1980s, mainframe computing was the predominant form of computing. The common computer center had one or more large mainframe computers satisfying the requirements of hundreds or thousands of users. There was an economy of scale associated with this kind of processing. As the mainframe computers increased in speed and capacity in magnitudes of 100 percent at a time and as the storage capacity increased at a rate of 30 to 40 percent per year, it was not reasonable for the computer center to add staffing at the same rate. Furthermore, the computer center operating environment became more complex than it had ever been before.

In order to reduce cost and human intervention and to improve the quality of computer center service, computer centers began to automate. The early automation activities were limited to satisfying specific needs such as tape management. It was difficult to manage thousands of tapes without some sort of assistance. Computer job schedulers were added to control the increasing amount of work that was processed through the mainframe computer. A wealth of other utilities were added to the computer center to improve its operation. However, an overall theme for this automation was lacking. In the early to mid-1980s, that theme developed: the concept of unattended computer center operation.

Unattended computer center operation is achieved through the elimination of manual activities, implementation of a self-service approach to computing and by computer center automation. Computer center automation is based on installing three levels of software: 1) primary, 2) secondary and 3) support software systems.

### *The Primary Software Systems*

The primary computer center automation software packages are the automation tools that directly interface with the computer. They consist of three software packages: 1) the automated computer job scheduler, 2) the automated console response system and 3) the electronic report distribution system.

#### The Automated Computer Job Scheduler

The automated computer job scheduler manages the routine, daily batch computer processing schedule. The objective of the computer job scheduler is to provide computer center users with the ability to change the routine schedule or to process ad hoc work without computer center intervention. Business organizations purchasing a departmental or small business computer or a turnkey system need to make sure this is an integral function of the operating software.

#### The Automated Console Response System

The automated console response system eliminates all routine interaction between the computer operator and the computer. It is the tie that binds together most of the software packages used to achieve unattended computer operation. The automated console response system is the new quality assurance manager. It improves the quality of the service of the computer center by eliminating the human intervention between the applications software systems and the computer and between the computer operation software and the computer.

#### The Electronic Report Distribution System

Electronic report distribution software manages and distributes hard-copy reports electronically. This software directs reports to a storage device rather than to a printer. Once on the storage device, it can be retained for a predefined period, viewed and, if necessary, printed under the control of an end user. This is not a substitute for on-line query, but it is an outstanding intermediate step. It gets the computer user to begin to think in terms of the information (s)he requires rather than in terms of reports.

### *The Secondary Software Systems*

The automated computer job scheduler, the automated console response system and the electronic report distribution system are central to computer center automation. However, at the next level, there are four additional software systems that work closely with the primary software: security software, automated rerun/recovery systems, automated computer monitoring systems and automated problem notification.

#### Security Systems

Security software operates like an umbrella over all of the other computer automation tools. Security software is a prerequisite for a self-service computing environment. It enables users to access their information and software without interfering with the integrity of other users or without relying on central computer center administration. The most common security software is RACF from IBM and Top Secret from Computer Associates.

#### Automated Rerun/Recovery Systems

Rerun/recovery software enables the automatic restart of batch jobs without technical support, database or computer center personnel assistance. Automated batch job restart is crucial to unattended operation. It enables automatic housekeeping, so that the only intervention required is to correct the problem and resubmit the job.

Batch jobs are within the control of end users; reliance on computer center staff for cleanup and restart is not consistent with the direction of computer center management. Software is available to handle these conditions and new applications should be designed with this as a requirement.

#### Automated System Monitors

Interactive computer system monitors have been available for a long time. This software provides the technical services staff and computer operator with the ability to interactively monitor the perform-

*CIRCLE #65 on Reader Service Card* ▲

ance of the computer system. Within the software, thresholds are set for computer performance and when exceeded, corrective actions can be taken. This software permits the computer operator to manage the operation of the computer. However, when the automated system monitor interacts with an automated console response system, conditions that exceed threshold can be automatically corrected through an automated response. The combination of the automated console response system and the automated system monitor offer the opportunity to automatically correct performance imbalances before they impact the computer user.

**Automatic Problem Notification**

Security and environmental monitoring devices are available to monitor the vital aspects of the computer center in the absence of computer room staff. Such equipment can recognize failing equipment or intrusions and phone designated staff on an exception basis using voice synthesizers. Furthermore, the devices can be queried by cautious or inquisitive management.

A similar device is available for computer systems. Messages are passed to a microcomputer-based system where they are logged and filtered. Messages that require no action are ignored and those that are defined to the system initiate a phone call to on-call support personnel. In an unattended computer center, such a device is indispensable for the correction of software failures that are sure to occur in even the best run computer center.

### *Support Software Systems*

Achieving unattended operation requires that the power of the computer be used to manage itself. For a couple of decades or more, information technology experts have been installing automation tools in the computer center to solve independent problems. In addition to the primary and secondary software, there are many other areas where automation can be applied to the computer center. In some cases these tools will be installed, and in others they will need to be selected.

**On-line Data Entry Software**

Centralized data entry is an obsolete function. Studies indicate that only 48 percent of the computer centers surveyed had a centralized data entry function. In 1984, more than 90 percent of the computer centers had a data entry function, a decrease of almost half (42 percent). Online data entry software simulates the functionality of true on-line interactive software. It provides the same opportunity as on-line software to directly enter data. The edit and update facilities are not equal to those of custom on-line update systems, but the software usually provides some logical editing (numerics, date ranges and so on). Furthermore, when online data entry software is used in conjunction with a report management system, it eliminates the need for hard copy error exception reports.

**Automated Report Balancing And Control**

Automate the report balancing process. Frequently, 10 or more years of effort are expended on the efficient design of application systems and little or no effort is spent on automating the balancing process. An entire processing stream will be halted for hours or even days waiting for computer users to review and balance their systems. Report balancing software should provide the ability for computer users to define balancing rules and to change them as necessary. It should automatically check and balance reports where required.

**Library Management**

Library management should be performed by the programming and technical support staff under the protection of a security package. Programming and technical support staff should move program modules into production libraries. Remember that computer center librarians do not know what a new software module can do, only that the proper forms were provided and that they had proper authorization. Using the security system as the substitute for authorization, the computer center can allow programming and technical staffs to maintain their own libraries. The computer center can effectively take advantage of the security system in the same way as computer center users.

**Disk Space Management**

One of the alternatives that is becoming more cost effective as a trade-off for tape processing is to substitute disk or direct access storage devices for tape as permanent or temporary storage. As a result of these and other uses, the data stored on disk media is growing at a rate of 30 percent or more a year. This growth pattern has resulted in the increased use of disk management software to ensure that sufficient disk space is available, that it is used efficiently and that its use is not dependent on human intervention.

**Disk Space Abend Software**

Disk space abend software stops "space-not-available" DASD abends during step initialization. These conditions are associated with disk space availability and management and arise when the IBM operating system is not able to satisfy space allocations for a new dataset. These abends are found in the best run computer centers. Since they are not part of the programming staff's area of responsibility, recovery from these types of abend conditions can place a significant burden on computer center operation, technical service or database staff.

**JCL Scan Utility**

JCL continues to be a labor-intense and error-prone activity. JCL is the source of a significant portion of the computer center's interruptions and problems. Scanning JCL for syntax errors and/or conformance with computer center standards should be part of building the computer center batch operation schedule, library maintenance and production and test job submission. Syntax checking is part of the operating system; some automated job schedulers include it as an integral part of their software and stand-alone software is available for scanning JCL.

**Tape Management Systems**

If the computer center has a large tape inventory, it is almost mandatory that the computer center needs to have a tape management system. Tape management software helps to improve reliability and reduce the direct labor associated with the use of this media. Furthermore, tape management systems are valuable tools for isolating the use and reducing the inventory of tape volumes. They should reduce the labor associated with tape handling, improve the quality of the retention process and assist in identifying ways to reduce tape usage. Tape is a data storage medium that is likely to be with us for a long time. Start reducing the inventory of tapes by no longer designing software systems that require tape as a processing medium.

**Tape Dataset Stacking Utility**

Tape dataset stacking software should be used in concert with tape management software. Statistics indicate that 80 percent of all tapes use only a few inches or feet of the tape reel. Furthermore, many tapes are backups used only in exception processing and many of these backups use only a fraction of a tape volume. Install software to stack multiple datasets on a single reel. By stacking these kinds of tape datasets, the computer center can reduce

# The Secret of Our Success.

For over 15 years, Tower Systems has been quietly crafting powerful mainframe software. Software that helps thousands of customers be more efficient, productive and successful.

Software like EPIC/VSE, the only resource management system written specifically for DOS/VSE. Software like EPIC/CMS, which brings powerful resource management capabilities to the CMS user. And software like EPIC/MVS—now in development—which will bring new dimensions to tape resource management for OS/MVS.

**The Customer Comes First.** Compared to some of the "big guys" who market all kinds of software, we are a focused company—concentrating our energy on highly refined products. And on customer support and technical maintenance that are the envy of the industry.

The support comes from the same experts who designed and developed the product. And the upgrades and enhancements are available to every customer through a clear contract—with no surprises, no hidden purchase requirements and no arm-twisting.

The secret of our success is your success—with our products support. To learn more, call Tower Systems at **(714) 650-4900** or toll-free from outside California, **1-800-854-7551.**

*"Tower Systems software is powerful, functional and flexible. I know. I write it."*

Keith Johnson
Project Leader,
Software Development

# Quietly moving ahead.

**TOWER SYSTEMS**
INTERNATIONAL
2220 Fairview Road, Costa Mesa, CA 92627

the physical handling of tapes, reduce the volume of tape inventories, decrease off-site storage cost and improve cost containment. Stacking datasets increases the access time to retrieve them, but there is a good chance that the tape will never be used again or, at worst, that it will be used infrequently.

## Direction Of Unattended Computer Operation

Computer center automation is achieved by systematically removing all manual computer center tasks and by replacing them with automation tools. As a result of this direction, software suppliers are beginning to visualize automated computer center operation as a logical unit and they are attempting to draw these individual software products together into a tightly integrated group of packages. Although integrated, the packages will continue to be marketed separately.

Software vendors are beginning to recognize that a truly unattended computer must operate in a self-service mode. To achieve this, the functions of the automation software packages need to be externalized, allowing the computer center user to enter data, schedule jobs, write programs, electronically distribute reports, administer security and query data. Furthermore, all of these functions are accomplished under a security umbrella to ensure that staff members do not accidentally or intentionally interfere with the information of another computer user.

On one hand, there is a movement toward a single image computer center automation tool as a result of a tighter integration of the different software systems. On the other hand, central support organizations are disappearing and the functionality of the software systems is being extended to the end users of the computer center. However, despite all the progress in both of these areas, the main thrust of all computer center operations continues to be in managing functionality of the large mainframe computers. Departmental computers, workstations, PCs and networks are being completely ignored. It is assumed that there is no need for automation since there are so relatively few users of these computers.

The problem is different from what is perceived by computer center management or software vendors. For example, an East Coast university has two large mainframe computers and 125 to 150 departmental computers. The total computing capacity of these 125 to 150 computers exceeds the total capacity of the two large mainframe computers. Furthermore, the computing capacity of the PCs is equal to the total capacity of all the departmental computers and the large mainframe computers.

It is not uncommon for organizations to have far more computing capacity in their network of departmental computers, workstations and PCs than they do in the computer center. On one hand, there is a drive to automate the computer center while, on the other hand, the departmental computers, workstations and PCs are being completely ignored. Furthermore, PCs, workstations and departmental computers are being introduced with a tremendous amount of enthusiasm. The computer user sees a problem that can be solved. (S)he gets closely involved with the process and installs the computer with little or no additional staffing.

However, after the computer is operational, the user goes back to the primary job. As a result, the computer user justifies additional staff to operate the computer or oversee the network of PCs or workstations. The departmental computer is creating a small data processing department. It starts out with one person, then two persons and then a manager. In effect, 125 to 150 departmental computers or networks can lead to the staffing of 250 to 350 people. This is more staff than would be required to manage the same amount of mainframe computing. The answer is not to go back to mainframe computing. The answer is to automate departmental computers and PC networks.

What is the new direction for computer center automation? The direction is to externalize the software packages installed on the mainframe onto a stand-alone computer. If it makes good business sense to schedule a single machine, then it makes equally good sense to be able to schedule across multiple computers. If it makes sense to have an automated operator on a central computer center, then it makes equally good sense to have an automated operator on the network managing multiple computers. If it makes sense to have electronic report distribution for the central computer, then the same kind of report distribution should work across the network.

The direction is to externalize the mainframe based software to a dedicated computer on the network. This computer interacts with the other computers by responding to console messages, scheduling batch jobs, starting the computer in the morning, shutting the computer down at night and doing routine backups. The stand-alone computer eliminates the operator interaction at each of the individual computers and incorporates it into a single central machine tied into the network.

Achieving this objective will require a multi-phased approach. The first step is to externalize the functionality of the computer center operation software packages. Initially, the direction calls for a tighter integration of this software into fewer software packages and then to externalize it onto a single computer. In the first phase, the computer will operate as a peer on a network of like computers. If the network is a network of VAX computers, then the computer will handle all of the interaction with a VAX. If it is a network of Hewlett-Packard or UNIX based computers, the computer will handle all Hewlett-Packard or UNIX based computers.

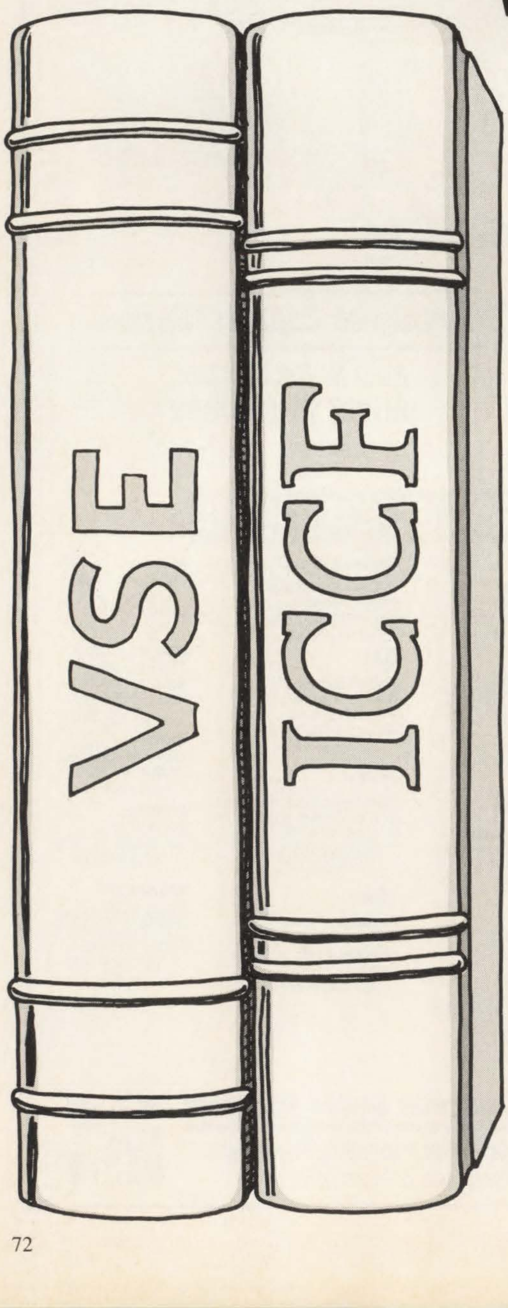
In the next phase, the computer is able to do precisely the same thing with unlike vendors. Obviously, the differences between dealing with a Unisys, a DEC, an IBM or a Hewlett-Packard computer and their proprietary operating systems is considerable. The level of complexity to manage computers across hardware vendors and software vendors is significantly more complex than dealing with a single vendor.

The future of unattended operations is to eliminate all the manual operation functions from the departmental and low-end computers in the same way that those manual functions are eliminated from the mainframe computing center. Further, the next stage is to tightly integrate all the packages into a single image automation tool that is externalized from the mainframe onto a stand-alone computer. In this way, the same functionality can be provided to all levels of computers: mainframe, departmental, workstation and PC. *The future of unattended operation is unattended network operation.*


*ABOUT THE AUTHOR*

*Howard W. Miller is responsible for administrative computing at Boston University. He has held senior-level positions in systems management for more than 20 years.*

# ICCF Library Management With VSE




By Sharon Hooper Martinez

Relying on user libraries within the VSE/ICCF library file, IBM's Interactive Computing and Control Facility (ICCF) offers an interactive work environment for information processing.

In order to ensure the viability of this system, IBM supplies a utility called DTSUTIL for library management. In this article, the main concepts of DTSUTIL will be examined with a brief look at a second utility, DTSANALS.

Through the use of specialized commands, these programs provide file maintenance capability off-line and interactively.

## The FORMAT Command

The FORMAT command initially establishes the maximum number of libraries and users for DTSFILE, which is the VSE/ICCF library file.

LIBRARIES and USERS, the two required parameters of the FORMAT command, should specify the maximum number of users and libraries — anticipating future needs. It is with the FORMAT command that a fixed area is defined that will contain a system record, user profile records and library header records. To redefine this area, the library file would have to be rebuilt. Either of the operands when specified in the RESTORE (SYSTEM) command would take precedence over the values established by the FORMAT command.

Although the user profile records and library header records are allocated by FORMAT, the records should be added (via ADD LIBRARY/USER) as the need arises and not at the time of allocation.

Another DTSUTIL command allows adding and modifying libraries, user profiles, broadcast records and library members.

## The ADD And ALTER Library Commands

If a new library is being added to the VSE/ICCF library file, the parameter for library number becomes available once the run is complete. Any number entered (it is an allowable parameter) is superfluous. A library deleted in a prior run of DTSUTIL could be picked up as the next available library number. Its characteristics would be determined by the parameters given in this ADD command.

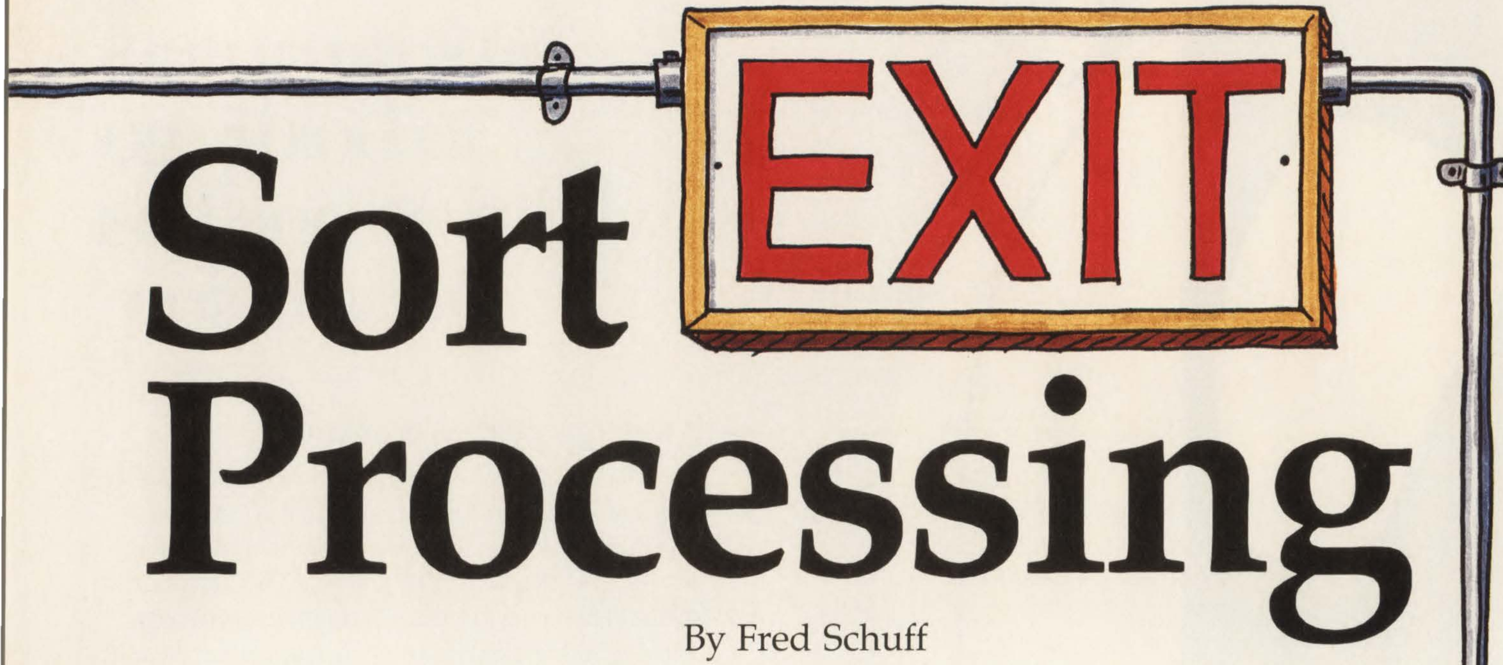
Options supplied will control the maximum number of directory entries available, the percent of freespace, whether the library will share data defined as *common data,* if the date in each directory entry for this library will be updated whenever a member is changed and if the library will be Public or Private.

If the maximum directory parameter is zero, the library will have an unlimited number of directory entries. This is not advisable. Searching a large library is highly inefficient. Several small libraries would be preferable.

The freespace parameter will provide for increasing the number of directory en-

*See* ICCF *page 88*


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.




# Sort EXIT Processing

By Fred Schuff

Sort exits exist to perform special user processing in conjunction with the standard process of sorting data into a desired sequence for processing. Customizing code to perform various functions (exits) can be written either by the user or come as part of third-party software packages.

The intent of this article is threefold: to discuss some basics of using the sort program with the available sort exits, to describe a special application for front-end supplied sort exits and the application of sort exits to implement an improved *sort and report* processing flow.

## Sort And Its Exits

There are a number of possible exits (that is, user written routines) which can be invoked at specific points during sort processing. These are referred to as *sort exits*. Sort exits are given control during the sort process by the sort program itself. A standard parameter list passed to each exit and a return code from the exit allows the user-coded exit routine to control the flow as well as the contents of data records. Exits can be invoked during both the input and output sort phases and can do such things as add, delete and modify records during sort.

Table 1 lists the exit names and points at which user coded sort exit routines are made available during sort. It is provided to give you a feel for the number and types of exits which are available.

## Invoking Sort

Sort exits are given control by the sort module itself. That is, they are LOADed and given control by the sort program. Control is passed to each of the specified exits at the time specified by the type of exit. As an example, the E15 exit (input exit) is given control just after a record is read from SORTIN by the sort program. The E15 exit can then change, delete or add records to the input stream. In fact, an E15 exit can create *all* of the input to sort without any external data file at all.

The two more common sort exits are the E15 (input record processing) and E35 (output record processing) exits. In fact, the COBOL internal sort (COBOL SORT verb) allows specification of COBOL code to be invoked at record input or record output processing time. The COBOL code is passed control as either the E15 or E35 exit (see Figure 1).

## Sort Invoked Via JCL

Sort can be initiated either from JCL as a step in a stream of processing or from an application program. Using JCL, the sort process is controlled with JCL statements, the PARM field on the EXEC JCL card and the sort control records from the SYSIN file (see Table 2).

## The PARM Field

The PARM field on the EXEC JCL card contains parameters that control the sort processing. These are overlooked by sort users too often. Some of these options are the following:

**SIZE** | **CORE** to specify the amount of storage to be used for sort or merge processing

**EQUALS** | **NOEQUALS** to retain or not retain the order or records with equal sort sequence

**FLAG(I** | **U)** | **NOFLAG** to control routing of messages

When using sort, it is recommended that *options* are used as appropriate to tailor sort to your requirements.

## The SYSIN Sort Control Cards

The *sort control cards* from the SYSIN file are the most commonly used and best understood method to communicate sort information. Below are some of the most common statements:

**SORT** statement: multiple functions —
- Identify the process as a sort
- Define the sort sequence that is desired
- Provide control information
- Provide parameters about the input data

**RECORD** statement: define the type and length of records

**SUM** statement: select records with equal control fields to generate totals of specified fields

**END** statement: to end sort or merge control statements

In addition, the control statement that is used to specify the sort exits which are to be invoked for this sort is the MODS statement.

**MODS** exit-name = (module-name,module-size,library-name,code), . . .
**exit-name** = E11, E14, E15 and so on
**module-name** = the name of your routine to be used for that sort exit

# ANNOUNCING

# A New More Productive Era in Systems Programming

**The Systems Programming Environment**
**Only in the SAS/C® Compiler**

Until now, higher-level languages just couldn't hack it in the systems programming world. Too many issues stood in the way—inefficiency, poor access to low-level system services, bulky and intrusive library requirements, and inflexibility in addressing the IBM 370 architecture.

But now you can write systems-level routines faster and maintain them better than you ever could with assembler—with the SAS/C compiler's exclusive Systems Programming Environment (SPE).

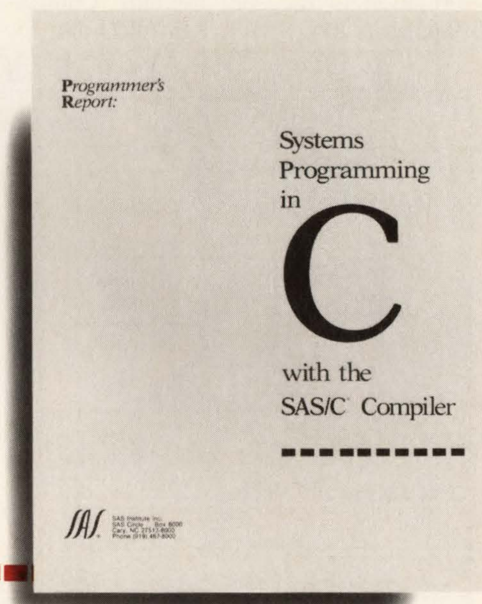
SPE is an extension to the C language that greatly simplifies the coding of user exits, tools, and utilities for JES2, VTAM, CICS, TSO, GCS, and other systems software. Included are support routines that allow you to write and execute C programs and a compact runtime library that features both general purpose and system specific functions for memory management, interrupt handling, low-level I/O, and more. There's also a utility that translates assembler DSECTs into C structure definitions—an enormous time-saver when you're writing programs that interface with assembler.

Together these tools provide a freestanding C environment designed to interact with the operating system the same way assembly language programs do. With SPE, your C programs can:

- call existing assembler routines
- generate any machine instruction in-line
- easily access system data and control blocks
- exploit BSAM or CMS file system I/O
- process asynchronous events and interrupts
- directly use SVCs and DIAGNOSEs

Then, at compile time, the SAS/C compiler's global optimizer will compress your code to produce routines that rival assembler for speed and efficiency.

With frequent updates and knowledgeable technical support—both provided free—the SAS/C compiler is the best investment you can make toward greater systems programming productivity.




**Learn More in a FREE Programmer's Report**

To find out more about systems programming with the SAS/C compiler, simply clip the coupon below and mail today. We'll send you our new **Programmer's Report: "Systems Programming in C".** Or call us today to find out how you can receive the SAS/C compiler for a free, 30-day evaluation.


SAS Institute Inc.
SAS Circle ☐ Box 8000
Cary, NC 27512-8000
Phone (919) 467-8000
In Canada, call (416) 443-9811


The SAS/C compiler runs under MVS (370, XA, and ESA) and VM/CMS on IBM 370/30XX/43XX/937X, and compatible machines.



____ **Yes, send me a FREE copy of "Systems Programming in C".**

____ **Contact me with details of a FREE 30-day trial of the SAS/C® compiler.**

**Mail to: SAS Institute Inc., Attn: ME, SAS Circle, Box 8000, Cary, NC, USA, 27512-8000. Telephone (919) 467-8000.**

MFJ989

**Please complete or attach your business card.**

**Name** ____________

**Title** ____________

**Company** ____________

**Address** ____________

**City** ________ **State** ____ **ZIP** ________

**Telephone** ____________

**Hardware System** ________ **Operating System** ________

**module-size** = the exact or estimated size of the routine for that sort exit
**library-name** = the DDNAME of the library if not in LINKLIB, STEPLIB or JOBLIB
**code** = N for pre-linkedited exit routines (recommended)
S for modules to be linkedited by sort
C for exit routines written in COBOL — they must be pre-linkedited as well

It is simple enough to linkedit your sort exit before sort processing. This also reduces the overhead of sort and the necessity for additional JCL to be coded for the sort itself.

## Sort Invoked From A User Program

When sort is invoked dynamically from a user program, there is a standard parameter list that is used to communicate information to the sort program. Control statements like SORT, RECORD and so on are passed via in-core buffer areas. This is handled internally by COBOL when using the SORT verb in the COBOL language. The basic form of this linkage is:

**Register-1:** A(POINTER)
**POINTER:** X'80' + AL3(SORT-PARAMETER-BLOCK)

(Note: this is identical to a parameter list with just one parameter, the SORT-Parameter-Block.)

### *SORT-Parameter-Block*

The first part of the parameter block (see Figure 2a) is fixed in location.

The second part (see Figure 2b) is optional, variable in length and contains optional parameters. The options are indicated by a one-byte code in the first byte of the field or by specific character literals for options. These start with the fullword at +34 in the SORT-Parameter-Block.

Then, as each exit point is reached and the appropriate exit is entered, data is passed to the exit routine from sort and returned from the exit to sort. The parameters are passed like a normal parameter list (for subprograms). The usual mechanism to control the flow of sort by the exits is values returned in the RETURN-CODE. A sample, the E35 parameter-list, is shown below:

**Register-1:** A(Parameter-list)
**Parameter-list:** A(address or record leaving Phase 3)
A(record in output area)
Switches

**RETURN-CODE**s from E35 to sort are:

**0** — ACCEPT this record
**4** — DELETE this record
**8** — DO NOT RETURN to this exit
**12** — INSERT a record (user R1 for the A(new record))
**16** — TERMINATE sort

Once these simple conventions are understood, it is fairly simple to write sort exits.

## Example: Sort Exit Front-end Code

There are situations in which it would

**TABLE 1**

**Sort Exits**

| Exit Name | Sort Phase | Description |
|---|---|---|
| E11 | 1 | Prepare for other exit routines |
| E14 | 1 | Delete, change, sum records |
| E15 | 1 | Create, add, delete, change, sum input records for sort |
| E16 | 1 | Act on insufficient storage |
| E17 | 1 | Close other exit datasets |
| E18 | 1 | Check labels, process read errors |
| E21 | 2 | Prepare for other exit routines |
| E25 | 2 | Delete, change, sum records |
| E26 | 2 | Close other exit datasets |
| E31 | 3 | Prepare for other exit routines |
| E32 | 3 | Create input records for merge |
| E35 | 3 | Add, delete, change, sum records |
| E37 | 3 | Close other exit datasets |
| E38 | 3 | Check labels, process read errors |
| E39 | 3 | Check labels, process write errors |
| E61 | 1,2,3 | Modify a collating sequence |

**FIGURE 1**

**Sort Exit Locations**

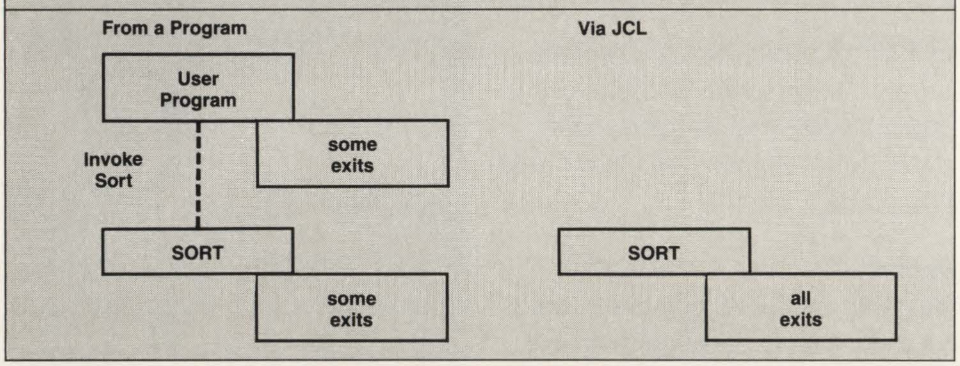



**TABLE 2**

**Common Sort JCL**

```
//stepname    EXEC PGM=SORT,PARM='option . . . option'
//SYSOUT       DD  SYSOUT=*
//SORTWK01     DD  DSN=&&SORTWK01,DISP=(,DELETE),
//                 UNIT=SYSDA,SPACE=(CYL, (ppp,sss))
//SORTWK02     DD  DSN=&&SORTWK02,DISP=(,DELETE),
//                 UNIT=SYSDA,SPACE=(CYL, (ppp,sss))
      .
      .            up to 32 SORTWKnn files and can
      .            be disk or tape
      .
//SORTWKnn     DD  DSN=&&SORTWKnn,DISP=(,DELETE),
//                 UNIT=SYSDA,SPACE=(CYL, (ppp,sss))
//SORTIN       DD  DSN=input-dataset-name,DISP=SHR
//SORTOUT      DD  DSN=sorted-dataset-name, . . .
//SYSIN        DD  *
sort control cards
/*
```

be helpful to be in between the sort program and existing sort exits. It may be because they are from a vendor and source code is not available or because rewriting an exit to add some small increment of processing is not practical.

*There are many products which employ sort exits. Exit routines, or hooks to user exit routines, are provided as part of a product. Most often, this code is a* black box *that cannot be changed or examined by the client site. The method described below is used for enveloping sort exits and is applicable to more than just the sort product.*

The following method was used to insert code between sort and two common exits, the E15 input and E35 output exits. The sort was dynamically invoked and the exit code was part of the vendor-supplied package. The original need was to trace records and return codes in and out of these exits to determine if a CPU looping problem was the fau't of a new version of the sort product or the vendor-supplied exit code.

IBM's DFS-SORT was the sort product installed where this method was developed. The assumptions in the coding refer to the linkage for this product version.

Method summary:

(1) Create a new module called *iceman* or *sort* that acts as a front end to sort and that is *non-reusable*

(2) When a sort is initiated, the pseudo sort module is invoked because it is STEPLIBed in your JCL

(3) That module, the pseudo sort, will then LOAD the real sort (or iceman) module into memory using a specific DCB on the LOAD Macro to force loading of the *real* sort module

(4) The pseudo sort will act as the intermediary between the exits and the real sort code to provide whatever *processing* is desired, like tracing data records and logging return codes which are set (see Figure 3).

When the *pseudo sort* receives control the first time, it replaces the addresses for the E15 and E35 exits (in the standard parameter list) with the addresses of the Pseudo E15 and E35 exit routine within the pseudo sort code. These pseudo exits will receive control (before the exit), handle your desired special processing (like tracing records) and then invoke the real E15 or E35 exit. They then get control again (after the exit) to perform user code (like return code tracing).

What makes this method work is the use of a module with the same name but which is marked *non-reusable*. Then, when this module loads the real code, another copy will be brought into memory. If the module was reusable, then the USECOUNT in the Contents Descriptor Entry (CDE) would be incremented on the existing *pseudo* module. That is not what is desired.

E15 and E35 exits are simple to intercept because their address is in the SORT-Parameter-Block. To front-end other exit routines, the process is similar but more complex. The MODS statement must be scanned to locate the other exit names. Then a pseudo exit name is substituted and the real exit is loaded by the front-end code or the pseudo exit.

The beauty of this method is that it is simple, requires little or no "hooks" into the vendor code and implements using minor JCL changes.

## Example: Sort And Report Processing

One of the most common uses of sort is the re-sequence records for reporting. This makes it simpler to generate a report when the data is in the desired sequence (like by account number). The normal process is to run a sort and then the report program. If you look at the required I/O processing against the data file, there are three full passes through the data — twice in sort and once more in the report program.

A clever method was devised to use sort exits to make this a single-pass process. In cases where there were large amounts of data, this represented significant savings in I/O and either storage space (if DASD files) or tape handling (if tape files).

**F I G U R E 2a**

**SORT-Parameter-Block**

Fullword Align — align on halfword boundary

| | | |
|---|---|---|
| +00 | | L'Following List |
| +02 | X'00' | A(start of SORT / MERGE statement) |
| +06 | X'00' | A(end of SORT / MERGE statement) |
| +10 | X'00' | A(start of RECORD statement) |
| +14 | X'00' | A(end of RECORD statement) |
| +18 | X'00' | A(start of MODS statement) |
| +22 | X'00' | A(end of MODS statement) |
| +26 | X'00' | A(E15 or E32 exit or zero if none) |
| +30 | X'00' | A(E35 exit or zero if none) |

**F I G U R E 2b**

**SORT-Parameter-Block**

| | |
|---|---|
| X'00' | Main Storage Value |
| X'01' | Reserved Main Storage Value |
| X'02' | A(start of MODS statement) |
| | A(end of MODS statement) |
| X'03' | A(start of DDNAME to replace SYSOUT) |
| X'04' | Number of input files (E32 Exit) |
| X'05' | A(start of DEBUG statement) |
| | A(end of DEBUG statement) |
| X'06' | A(start of ALTSEQ statement) |
| | A(end of ALTSEQ statement) |
| X'07' | A(start of SUM statement) |
| | A(end of SUM statement) |
| X'08' | A(start of INCLUDE/OMIT statement) |
| | A(end of INCLUDE/OMIT statement) |
| X'09' | A(start of OUTREC statement) |
| | A(end of OUTREC statement) |
| X'0A' | A(start of INREC statement) |
| | A(end of INREC statement) |
| X'F6' | A(ALTSEQ translate table) |
| X'FD' | IMS Flag for IBM Sort |
| X'FE' | A(STAE work area) |
| X'FF' | A(Message Option code) |

DIAG option "literal"

| | | |
|---|---|---|
| BALN | OSCL | POLY option for tape sort |
| CRCX | PEER | LIST options |

DDNAME prefix (4-char) to replace SORT in JCL

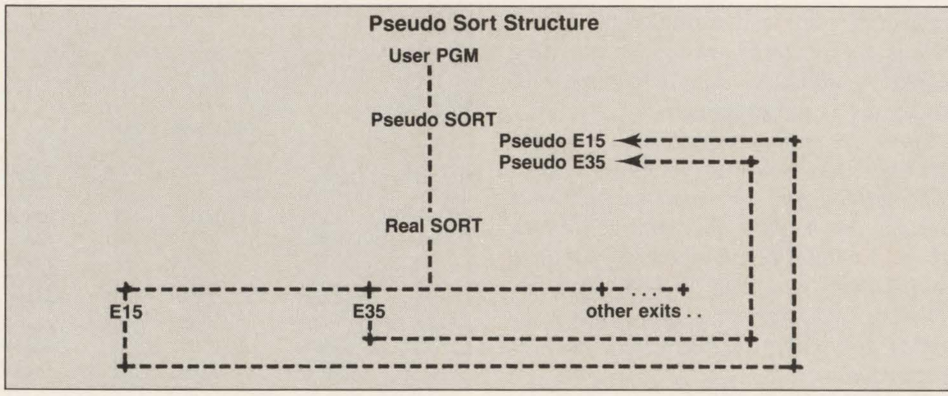
The concept is to make the report program the E35 sort exit by converting the READs in the program to E35 exit points in the sort process. What made this so useful was that the report program was not modified — it still executed the READ statements. The special driver code converts the READ statements to branches to the E35 entry point (see Figure 4).

In this process, there is one pass through the input file (normally SORTIN) rather than three. This is a significant reduction in the I/O processing and data handling that is required. In most cases, the sorted file is only needed for that single report so the absence of a SORTOUT file is really not any problem. The real beauty is that the user report program runs without changes (in most cases) with standard READ statements.

The sort and report manager program handles the interface. The general purpose method used was to filter the SVCs (using SVC screening) to trap the OPEN of the INPUT file for the user report program so that the READ could be converted into an E35 sort exit routine. At End-of-File (EOF), the E15 exit would signal sort that input was complete and then branch to the End-of-Data (EOD) routine within the user report program.

It would be possible to expand this interface to handle more than one user report program at a time. To do this, the E35 driver would just pass control to report program number one and then report program number two with the same record. It would affect run times because this would be a purely sequential operation, but it would still amount to the same single-pass through the input file for both reports.

FIGURE 3




To carry the method to the extreme, multiple reports using multiple sort sequences all handled in one pass through the input file is possible. This would be done by invoking "n" different sorts simultaneously. To do this, the "prefix" for sort ddnames would be required to create multiple sets of sort utility files. The E15 would handle reading the records for both report processes. Then, the "n" E35 exits, the report programs, would be given control to process the record just sorted. The major problem with this technique is the limitations of storage in the region to handle the "n" simultaneous sorts.

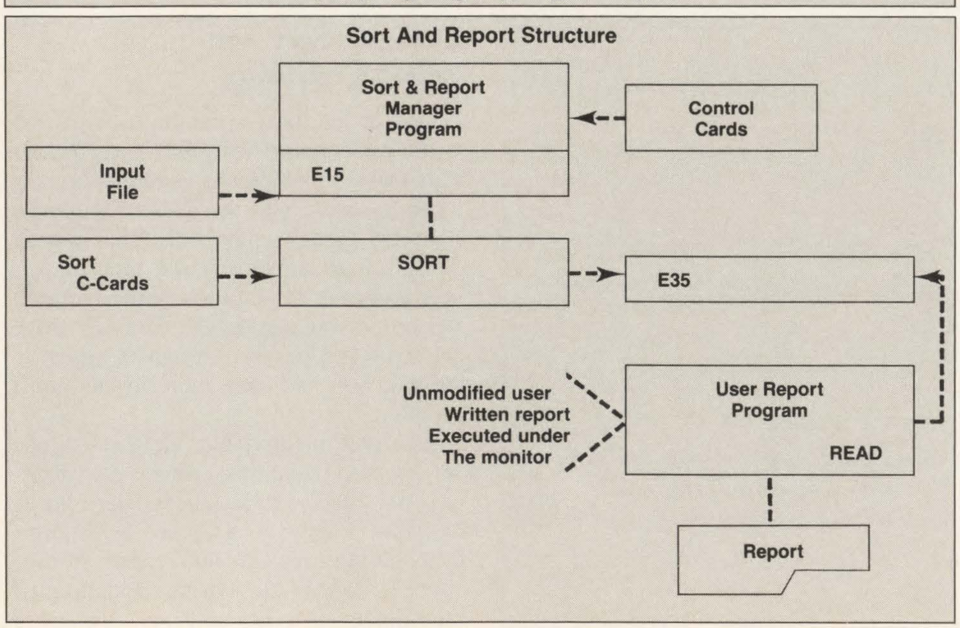
## In Conclusion

Sort is an example of standard processes which can be expanded to handle more tasks than one would expect. The key to doing more with sort or other resources at hand is to fully understand just how all the pieces work and fit together.

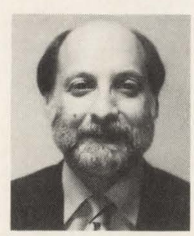
There are a number of uses for the technique described above. One such example was to *front-end* a DBMS interface module that passes user *calls* to the DB manager that services those requests. Many vendors like to have a *loadable* module with most of the interface code to avoid having to re-linkedit application modules when a change is applied to this interface code. The code that is linked with the user code is a small program that invokes the loadable interface. The result was a dynamically invoked DB-trace which was simple to access and did not require using the DB system resources or DB utilities and log tapes.

Putting the pieces together often requires first taking them apart. Preparation is critical to the success of such an undertaking. Also, having the right attitude and confidence to perform such tasks is the hardest obstacle to overcome. However, the results are certainly worth the effort.

FIGURE 4




*ABOUT THE AUTHOR*



*Fred Schuff is a senior consultant for Systems/Software Engineering, a systems and management consulting firm. He has worked for the past 20 years in a number of different industries in systems programming, database support and applications design and consulting. S/SE, 940 W. Valley Rd., Suite 1603, Wayne, PA 19087, (215) 341-9017.*

# Applications Tuning Tool Boosts Throughput for Bergen Brunswig

By David Shein

Frank Bolan had a problem. He is Manager of Technical Services for Bergen Brunswig Corporation, based in Orange, CA. Bergen Brunswig, with more than 40 divisions located throughout the United States, is a $2 billion per year distributor of pharmaceuticals and related products. The data center runs MVS/XA, supporting a mixed workload that includes batch, CICS and IDMS (Cullinet/Computer Associates).

The dilemma Bolan faced was that Bergen's data processing workload was about to overwhelm the data center's 3090/200 system and an immediate processor upgrade was financially unacceptable to corporate management. As he relates, ''In October of '87 we made a presentation to the steering committee saying, 'We've got to have an upgrade.' They weren't against the upgrade, but they were against the cost.'' That left Bolan between the proverbial rock and hard place. He needed something that would greatly increase the 200's throughput and he needed it quickly. In his own words, ''We were pretty well out of gas. I really needed a silver bullet — bad.''

## A Possible Solution

Shortly thereafter — and with his predicament still unresolved — Bolan attended the December 1987 Computer Measurement Group (CMG) conference in Orlando, FL. There he discovered STROBE, an application tuning tool marketed by Programart (Cambridge, MA). What initially captured Bolan's attention was Programart's claim that STROBE

> *"We were pretty well out of gas. I really needed a silver bullet — bad."*

could help him boost the performance of IDMS, the database management system on which many of Bergen's key applications depend. He quickly learned, however, that STROBE's usefulness reached far beyond IDMS and into just about every applications environment in the shop.

Based on information provided by Programart, Bergen decided to bring STROBE in for a two-day trial. Michael Cleary, Supervisor of Technical Support Services at Bergen, picks up the tale, ''We actually got some relief from STROBE before we got the product. The company held a workshop and asked us to identify some problem jobs we wanted to look at. So we had five or six of our worst resource ''hogs'' STROBEd the day they were here. We got a lot of good clues out of that.''

Once given the opportunity to work with the detailed diagnostic information produced by STROBE, the project managers participating in the trial were convinced that the tool could make a real difference. Bergen decided to purchase the product. STROBE was installed in the spring of 1988 and it did not disappoint anyone.

## Processor Upgrade Delayed

By this time the processor bottleneck was exhibiting early signs of a crisis in the making. As just one example, the new accounts receivable system's month-end job was running anywhere from eight to 14 hours and it was not even fully implemented yet. As Bolan tells it, ''They didn't have all the divisions fully implemented on that new A/R system and they were worried that they wouldn't be able to add all the divisions to it. There were directors getting pretty nervous." Gary Davis, software specialist, points out that although three-fourths of the divisions were on the system, "The last quarter was 50 percent of the work. So it was pretty significant . . . they could not implement that system as it was."

Using STROBE-develped information about the month-end program's internal processing, the developers were able to determine that the program was doing a great deal of unnecessary work. The redundant processing stemmed from changes made to the system's functional specifications after the month-end program had already been programmed. When the unnecessary processing was eliminated, the run time was slashed from eight hours to just 20 minutes — end of batch window problem.

STROBE proved equally effective at pinpointing hidden bottlenecks in many other areas. The net results were that Bergen was able to delay the anticipated processor upgrade. The upgrade (to a 300E) finally occurred in October 1988, a full year after the original steering committee presentation. Bolan gives full credit to STROBE for postponing the upgrade as long as possible, "I don't think we would have made it that entire year without STROBE helping us. It probably deferred a two-and-a-half million dollar processor upgrade for six to eight months."

## Applications Tuning And STROBE

So what is STROBE and what does it do?

First, a few words about what STROBE is *not*. It is not primarily a tool for system tuning (although it can help), nor is it a capacity management tool (although it can help there too). System tuning and capacity management aim to make currently installed resources operate as efficiently as possible and to plan and control the amount of computing resources required to service a given (often growing) workload. In other words, their job is to manage the *supply* side of the computing demand/supply equation.

The intent of applications tuning, by contrast, is to reduce the *demand* for computing resources by rendering applications more efficient. The idea is to reduce the applications' resource consumption so that the installed resources can effectively handle more work.

STROBE is an applications tuning tool. Its purpose is to identify and locate performance bottlenecks in application logic so that they can be reduced or eliminated.

## How STROBE Works

Using STROBE is a two-step process. Step one is to run the application with the real-time measurement part of STROBE turned on. Step two consists of generating reports which display the results in a usable form.

The measurement component of STROBE resides in the address space of the application(s) being measured. At specified intervals, STROBE "wakes up" and takes a comprehensive snapshot of program activity in the address space. The captured information is quite detailed and encompasses all relevant types of resource usage including CPU, main storage and I/O. The data thus collected is written to an ordinary OS dataset.

At some convenient later time, a series of report programs is run to digest the collected data and prepare a comprehensive profile of the internal performance of the STROBEd application(s). The reports cover all measured resource types. Resource consumption is depicted using a form of presentation specific to the environment being measured. For instance, processor usage reports can, if desired, display individual COBOL or PL/1 source statements showing the CPU consumption associated with each statement. In most cases, the amount of detail or "granularity" in STROBE's reports is controlled by the user. File I/O activity, for instance, can be displayed down to the cylinder level if desired.

Note that STROBE does not optimize or improve applications directly; such corrective surgery is the user's responsibility. What STROBE *does* do is provide critical information about what is happening inside an application — a kind of software x-ray. By basing optimization efforts on this kind of hard data, users can be sure that the efforts are being concentrated where they will do the most good. STROBE makes it possible to take applications tuning out of the realm of educated guesswork. As Davis says, it provides "a reality check." Cleary concurs, "Without STROBE you don't know what to tune for. You just throw resources at it and hope."

The fact that STROBE does not directly modify applications can take some explaining, as Bolan is well aware. "That made STROBE difficult to justify to management. Usually you buy a product and it does something for you; it produces some result . . . what do you get with STROBE? Information. We know where to look now to solve a problem."

It is also important to realize that STROBEing an application is an iterative process; when changes are made to the application, subsequent runs are needed to determine if the changes have the desired effect. Bolan points out, "It might take several runs . . . it's not like one listing is going to tell you everything. You STROBE it, make a change and STROBE it again." Too, eliminating one performance constraint often reveals another. Bolan again says, "You might move a problem to another area . . . you might fix a table search and then find that you've got a VSAM file that's way out of line."

STROBE is designed to be convenient enough to be used on production jobs. Its own resource consumption is modest: about 30K of virtual storage and a small amount of CPU and I/O in the address space being measured. This makes it entirely practical to use STROBE on actual production runs. Especially helpful is the fact that STROBE is completely transparent to the applications it measures, requiring no code or JCL changes. Asked to compare STROBE to other optimizing and measurement products he has used, Bolan says, "To me the significant difference is that you don't have to recompile the program to use it."

For specifying and controlling the measurement process, STROBE features a simple and straightforward ISPF dialog.

## STROBE At Bergen Brunswig

Today Bergen Brunswig STROBEs every type of program in the shop. STROBE has been instrumental in keeping batch processing windows under control, as illustrated by the A/R job mentioned earlier. It makes users' lives easier in other ways, too. Describing a semimonthly inventory job, Davis says, "They were getting billed $180,000 per year for this resource and we dropped it down to $10,000." Davis adds that STROBE in the batch environment yields information that is hard to obtain otherwise — things like I/O counts to files, CI splits and the like. And true to its promise at CMG, STROBE has been of tremendous value in milking the best possible performance out of IDMS. The eight hour A/R run mentioned earlier was an IDMS job and numerous others have felt the beneficial impact of STROBE as well.

CICS throughput has also benefited from STROBE. Cleary uses it to tune VSAM buffers, temporary storage and other key CICS performance regulators.

Bergen has used STROBE on third-party vendor software to uncover performance problems and even actual bugs. For instance, a CICS monitoring product came under suspicion when its installation caused a previously trouble-free application transaction to begin sporadically ABENDing. Using STROBE, Bolan was able to prove that the ABEND was in the vendor's code. "We STROBEd it and told them where they were having their ABEND . . . they were kind of surprised . . . we knew what we were talking about." Similarly, when problems occurred in a purchased inventory control application, "STROBE proved that it wasn't our code that was bad," Bolan adds.

Bergen also uses STROBE to *retrotune* older applications. This is important because an application may start out performing well but degrade over time due to unforeseeable changes in the mix of work. As Davis says, "Applications change without you really knowing. In this shop there's been a large development cycle rewriting some major applications. One of the benefits we found was STROBEing this old code and finding out why a job would run for three hours." In such cases, Davis points out, the problem code often turns out to be part of a routine which, when the program was designed, was not expected to be heavily used. "But," he explains, "the business cases had changed and the programmers were surprised."

Bergen finds STROBE highly useful in developing new applications. Bolan remembers the case of a new CICS transaction that was routinely delivering five-to-six second response times. As Bolan tells it, the project manager was upset about the "poor CICS performance" until STROBE revealed that each transaction was doing immense amounts of I/O to a particular dataset. Armed with this data, the development team made some design changes and the response time dropped to the subsecond range. According to Davis, "Now they STROBE new applications. QA requires it."

And the people at Bergen are still thinking up new ways to use STROBE. Currently in the planning stages is a mechanism to automatically invoke STROBE before honoring an operator-initiated CANCEL command. It is hoped that problem resolution can be expedited substantially if the programmer is provided with STROBE output instead of (or in addition to) an operating system dump.

## Intangible Benefits

Not every STROBE-related improvement is quantifiable. Where performance considerations are involved, Bergen has seen a distinct raising of consciousness on the part of management and technical staff alike. Davis reports that STROBE's output opens eyes especially in the cases with logical views of data with IDMS in which you have people developing code who are far away from the physical data itself. Bolan agrees, saying, "I was surprised by the reaction from the project managers and programmers. It opens a lot of eyes. Also, I could see that it was bringing two adverse groups together to solve a common problem . . . that's where I was shocked," he adds pointing out that data helps to mitigate finger-pointing behavior.

And what about management? Bolan reports, "Directors call me up and say, 'Could you STROBE this? We're having a real problem with the users complaining about how long this is taking.' " The vice president of MIS, while he may not be conversant with all the technical details, certainly knows how important STROBE is to his operation. Bolan adds, "When we have a performance problem, the first words out of his mouth are, 'Have you STROBEd it?' "

## Conclusion

One of the hoariest (albeit highly accurate) truisms in data processing is the 90/10 rule: 10 percent of the code is responsible for 90 percent of the execution time. STROBE lets Bergen Brunswig zero in on that critical 10 percent without having to waste time wading through the other 90. Not infrequently, the solution to a nagging performance problem turns out to be something as simple as changing buffer allocations in JCL, which does not even require touching program code.

Bolan, Davis and Cleary agree they would not want to operate the data center without STROBE. As Davis says, "We'd run it less efficiently. We'd use more resources doing the same work."

*ABOUT THE AUTHOR*

*David Shein is a systems programmer and free-lance writer. He has authored other articles in previous issues of* MAINFRAME JOURNAL.

# CA-OPTIMIZER Promotes More Maintainable Code

By John Kador

With maintenance chores consuming up to 80 percent of resources in some MIS shops, COBOL programmers can use all the help they can get. CA-OPTIMIZER, a popular COBOL productivity aid from Computer Associates International, is used by thousands of programmers to enhance their programming techniques and create more maintainable code. The most-often cited benefits of the product are structured, better-tested and more efficient programs.

The system has three main components. Through its optimization component, CA-OPTIMIZER reduces object code by eliminating redundant or unexecutable object code. Through the Analyzer component, programmers are taught more effective programming techniques. The Analyzer also guarantees that every line of code has been thoroughly tested before being released for production. With the Detector component, errors are displayed along with information on where and why they occurred. Multiple abends can be corrected in a single test session.

## Benefits

The general benefits of CA-OPTIMIZER can be categorized into four areas.

### *Benefits To Programmers*

Because CA-OPTIMIZER's automatic organization facility lets programmers concentrate solely on the *goal* of the program instead of the *how* of the program, it increases programmer productivity. The system works with the programmer to develop more reliable and efficient COBOL programs with less effort. In this way, a program requires fewer compiles and tests to maintain. The time required to effectively tune and improve programs is reduced sharply.

### *Benefits In Terms Of Program Reliability*

CA-OPTIMIZER provides the means to thoroughly test and tune COBOL programs. Thoroughly-tested programs, of course, are more reliable. Such programs exhibit a lower level of production errors, job failures and reruns. Sound testing procedures ensure the development of code that is logically solid and efficient. System time previously lost to repeated failures and reruns can now be put to more productive use.

CA-OPTIMIZER displays overall program flow, spotting hidden performance problems, untested logic and any unexecuted verb commands. It also provides reports that help programmers with program testing. These reports, which include an expanded program listing, contain comprehensive COBOL-oriented diagnostic information.

### *Benefits In Terms Of Program Performance*

A COBOL program is only as fast as the execution of its slowest statement. CA-OPTIMIZER scans every line and displays those statements that use the most CPU time and cause the most trouble. Thus, it identifies all COBOL statements that result in performance bottlenecks so that they can be rewritten. Using these facilities of the system, programmers can easily modify or remove the offending statements, producing a much leaner and faster executing program.

### *Benefits In Terms Of Sounder Programming Practices*

One of the indirect benefits of using CA-OPTIMIZER is that programmers become better programmers. CA-OPTIMIZER helps enforce better programming practices. First, by keeping all of the time-consuming, nonprogramming procedures to an absolute minimum, programmers will be able to concentrate much more on developing cohesive proficient coding. Second, because CA-OPTIMIZER functions as a shop standard, programmers will begin to work more as a team and not as a group of individuals.

In this way, techniques such as structured programming and team programming will become a larger part of the department effort. CA-OPTIMIZER applies techniques to automatically improve the efficiency of subscript calculations, data conversion, register usage, redundant instructions, verb usage and data and procedure addressing.

## Make Room For CA-OPTIMIZER

Danny Thomas, the entertainer, founded American Lebanese Syrian Associated Charities (ALSAC), the fund-raising arm of St. Jude's Children Hospital in Memphis, TN. The hospital was originally interested in conducting research on childhood leukemia, the survival rate for which was so dismal that it warranted the hospital being named for the patron saint of lost causes. It seems clear, however, that the hospital may have been misnamed because over the years medical research, much of it conducted at the hospital, has dramatically increased the chances for patients to survive a variety of cancers.

A large measure of credit for this achievement must go to ALSAC which annually raises $70 million for the hospital's research efforts. Heavy automation allows ALSAC to conduct a wide variety of mailings, telethons and other fund-raising activities with a modest staff, thus decreasing overhead. ALSAC operates an IBM 4381 P23 under VSE SP2.1.7. It uses Cullinet Software's (Westwood, MA) IDMS for data mangement as well as a large variety of software developed inhouse.

CA-OPTIMIZER's optimization feature is critical to the fund-raising operation, according to Gary Gaggiani, director of data processing. In the past, many of ALSAC's large jobs required 20 to 30 hours of wall clock time. "We desperately needed a way to cut those times down," Gaggiani says. CA-OPTIMIZER fit that bill by knocking 30 to 40 percent off the running of each job. Thirty percent of a 30-hour job is nine hours. In addition, these savings did not require any effort on the part of programmers. The inherent optimization works behind the scenes.

The optimizer feature of CA-OPTIMIZER works on the object code of a program. Its principal function is to eliminate redundant machine instructions without altering the actual processing of the programs. User-defined data areas remain untouched. An optimized program simply requires fewer instructions to do the same job.

Gaggiani believes that the bulk of the optimization caught I/O redundancies and inefficiencies. However, other times optimization can be effective are when it does the following.

- Reduces perform linkage to one instruction; the PERFORM verb can be made to provide a more efficient link.
- Simplify IF logic — in many cases when two tests are performed, the results of the second test may already be known from the results of the first. When this happens, CA-OPTIMIZER can eliminate the object code of the second test.
- Consolidate MOVES — when CA-OPTIMIZER detects a series of MOVEs of the same literal-to-contiguous storage locations, it will reduce them to just one move.
- Optimize register usage — through a detailed analysis of register usage, CA-OPTIMIZER can reduce the object code for data manipulation by sorting out and correcting inefficiencies caused by the compiler.
- Optimize addressing for data and procedure references — because it performs a global analysis of the program's data and procedure addressing schemes, CA-OPTIMIZER can minimize the loading and reloading of base registers.

ALSAC makes most use of the system's Detector Abend Report, a specially formatted report designed to eliminate all the searching and guesswork usually needed in debugging. But Gaggiani acknowledges that ALSAC could benefit from more extensive use of CA-OPTIMIZER for new systems development. "We know there are a lot of capabilities we are not using," he says. "We're so busy, we don't have time to sit back and thoroughly optimize our own operations."

## BancBoston Soups Up Application Package

BancBoston Mortgage Corp. in Jacksonville, FL originally acquired CA-OPTIMIZER to soup up some old, poorly performing DOS code before it converted to MVS/XA. It considered CA-OPTIMIZER along with others. The mortgage company concluded that if it ran CA-OPTIMIZER's Detector feature in production, it would get the benefits of several competing products as well as the dump formatting and optimized code features unique to CA-OPTIMIZER. BancBoston operates an IBM 3090 Model 120E under MVS/XA.

BancBoston even uses the system to optimize its largest piece of third-party application package which it prefers not to identify. Using CA-OPTIMIZER, BancBoston has optimized some of the largest COBOL modules, those in excess of 20,000 lines, and is seeing between a 40 to 50 percent reduction in utilization. As a result, it has reduced the size of load modules and instruction paths, reduced core fragmentation and decreased the I/O time required to load the modules. Obviously, the application software had been patched over the years until it became spaghetti code. Without any adverse affect on the functionality of the program, BancBoston reduced the run times significantly.

Systems and Programming Manager David Stern's main complaint is that CA-OPTIMIZER is unnecessarily complex to install. He would like Computer Associates to streamline the number of options required to install the product. There are a lot of hidden options and internal parameters that can be turned on and off. "I'd either like to know all about them or, better yet, eliminate them entirely. But don't feed me a little at a time. That's not very satisfactory," he says.

## Leggett & Platt Eliminates Loops

Leggett & Platt, a components manufacturer for home furnishings in Carthage, MO had an extremely slow application. According to Systems Development Manager David Earhart, this particular program used 100 percent of an IBM 3033 and required three to five hours to run. So he asked Computer Associates to demonstrate what CA-OPTIMIZER would do with this program. Somewhat surprisingly, after optimization, it did not run faster. But using the Analyzer to generate timing statistics on each statement in the program, it became obvious why the program was so CPU intensive. One statement was doing a test against an improperly coded table, resulting in a loop. When Earhart modified the IF test, the run time went to five minutes and the CPU utilization went to a reasonable 25 percent. Leggett & Platt justified CA-OPTIMIZER on those savings alone.

They operate an IBM 3083 J running VSE under VM.

## The Analyzer at Central Vermont Public Service

The Analyzer has come in handy at Central Vermont Public Service Corporation, an electric utility based in Rutland, VT. According to Sue Wilder, manager of application development, the Analyzer is immensely useful for testing programs faster and with greater assurance. For example, it was used when the utility developed an automated prebilling edit program for customer billing. This was a complex system that validated meter readings entered into the system through optical scanning. After entry, each reading must be compared to the master file for reliability and reasonability. "The Analyzer helped make this application as efficient as possible," she says. Central Vermont Public Service employs 10 application programmers and two database analysts. It operates an IBM 4381 and 4341 under VSE/SP.

By contrast, BancBoston does not allow the Analyzer to be run in production because of the overhead that it sets up. It is used in testing mode only. A rule of thumb is that if a COBOL program exceeds 2,000 lines, then Analyzer should be used to catch any inadvertent loops. With programs under 2,000 lines, it is hard to get into loops that do serious damage. Leggett & Platt also will not analyze every program. "If a program runs only five minutes, why spend a couple of hours analyzing it," says David Earhart.

## VS COBOL II

In 1984, IBM introduced its newest COBOL compiler, VS COBOL II, to conform to many of the 1974 ANSI standards. CA followed with CA-OPTIMIZER II. Although CA-OPTIMIZER II is a new product, it is backward compatible with the earlier version. Each of the three components — Optimizer, Detector and Analyzer — have been specifically redesigned to work with VS COBOL II. An on-line VS COBOL II HELP command is built into the product, reducing the learning curve for shops that are converting to the newer compiler.

All this is nice, but MIS executives are in no big hurry to convert billions of dollars worth of OS/VS COBOL code over to the new language. None of the users contacted for this article had any plans to convert to VS COBOL II. The new COBOL does offer certain benefits over its predecessor. Chief among these is the ability to make full use of 31-bit addressing and write truly structured programs.

CA-OPTIMIZER supports all VSE, MVS and VM operating systems. CA-OPTIMIZER II supports VS COBOL II running in an MVS environment. Neither requires changes to source programs or to the COBOL or COBOL II compiler and usually does not require changes to the operating system. License fees, based on CPU and operating system, range from $14,000 to $72,400.

For further information contact Computer Associates International, Inc., Route 206 and Orchard Road, CN-8, Princeton, NJ 08543-0008, (201) 874-9000

### *ABOUT THE AUTHOR*

*John Kador is a free-lance writer and a frequent contributor to* MAINFRAME JOURNAL.





Cycles *from page 44*

Similarly, efforts to optimize processing will normally show only minor improvements in efficiency.

It probably does not make that much difference whether programs are written in COBOL or Assembler. Such a small portion of processing is actually spent in application code, that the differences in resource utilization of Assembler and COBOL programs will usually not be significant compared to other processing.

Also, there is more potential for reducing CPU utilization by controlling which CICS services are used than by tuning application code. Efficiency in overall system design will normally have a greater impact than the efficiency of individual program statements.

The largest user of CPU in most CICS environments is its monitor facility (IBM or other vendors). In particular, one of the most expensive services is the logic to keep track of CPU usage. If accurate measures of CPU usage are not collected, CICS CPU usage can be tuned considerably.

In addition, it is possible to reduce CPU utilization by controlling which data is collected by CMP or other monitors. Also, the trace facility will add about 12 to 20 percent to CICS processing. Turning off trace is one of the easiest ways to conserve CPU cycles.

Be cautious with the use of exits. Highly used global exits for facilities such as file control and program control can be expensive.

Avoid over-modularization. While it is good to divide processing into logical units, excessively small modules can generate quite a bit of program-linkage overhead.

Furthermore, VSAM subtasking can reduce CPU demand in the main CICS task. It does this, though, at a cost of more total CPU utilization.

### *ABOUT THE AUTHOR*

*Ted C. Keller is the manager of a group responsible for CICS systems support, performance management and capacity planning at Yellow Freight System, Inc., 10990 Roe Ave., Overland Park, KS 66211, (913) 345-3274. He has worked in various data processing jobs for more than 21 years.*

# VM/SP HPO 5

# Spool File Constraint Relief

## *In eliminating the 9900 spool file limit, the spooling subsystem of VM/SP HPO 5 has become "virtual," but there still is a limit to the number of spool files.*

By Thomas E. Cooper

In addition to improving the performance of VM/SP for large processor installations, VM/SP HPO 5 introduced major changes and enhancements to the spooling subsystem of VM. HPO Release 5 provides spool file constraint relief by changing the 9900 system-wide file limit to 9900 per user. This article will present a brief introduction of the changes in the spooling system that HPO 5 introduced.

The three changes that are externally noticeable are the SYSSPOOL virtual machine and changes to the SPOOL and SPTAPE commands. HPO 5 requires the combination of userid and spool file identifier for spool file manipulation. Performance monitoring software started identifying a new virtual machine on the system called SYSSPOOL that had a virtual storage size of 16MB. All of these external changes are due to spool file constraint relief, but the changes internal to CP are not as simple or subtle.

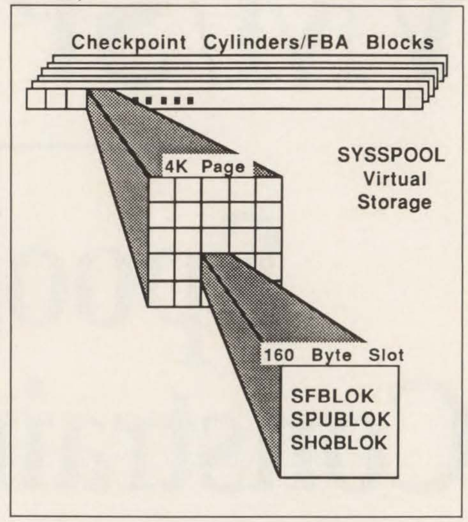
### "What is the maximum number of spool files for an HPO 5 system?"

*The VM/SP HPO Planning Guide and Reference (SC19-6223-7)* manual estimates the maximum number to be 100,000. The actual number is dependent on several factors: size of the checkpoint area on DASD, SYSSPOOL virtual storage, SYSSPL setting of the SYSRES MACRO in DMKSYS, the number of virtual machines with spool files, the number of real spooling devices and whether the HOLD command is used. There is a theoretical maximum; however, before touching on that there are dependent factors to address.

Because you can now create up to 9900 spool files per user, something needed to be done about the effect the spooling subsystem has on CP-free storage. Prior to HPO 5, each spool file has a corresponding control block (SFBLOK) in CP-free storage. If the number of files is increased in magnitude, CP-free storage is drastically impacted. This effect is eliminated by the use of the SYSSPOOL virtual machine. The SFBLOKs for reader files reside in the virtual storage of SYSSPOOL, not in CP-free storage. This increased the amount of storage available for spool file control blocks by 16MB.

The changes to the checkpoint area and the effect the SYSSPOOL virtual machine has on spool file constraint relief are both related to a new field in the SFBLOK control block. Because spool file identifiers are only user unique, another

**FIGURE 1**




*There is a one-to-one relationship between the virtual storage of SYSSPOOL and the checkpoint area on DASD.*

means is needed to uniquely identify spool files. This is accomplished by the SFBSYSID field. SFBSYSID is the virtual address of the SFBLOK in SYSSPOOL's virtual storage.

SYSSPOOL's virtual storage is mapped one-for-one onto the checkpoint pages defined by the SYSRES MACRO in DMKSYS. Every virtual page corresponds to a physical page on the checkpoint DASD volume. This is why the size of the checkpoint area is a limiting factor in determining the maximum number of spool files. There needs to be at least enough physical 4K pages in the checkpoint area to allow for the expected maximum of spool files (reader, punch and print) on the system. The greatest number of files is allowed when there are enough checkpoint pages to be mapped to a 16M SYSSPOOL virtual machine. Anything less reduces the number of spool files possible for the system. *The VM/SP HPO Planning Guide and Reference* manual gives suggested checkpoint area sizes for all the supported DASD types that allow for a 16MB SYSSPOOL virtual machine.

### "But I thought SYSSPOOL virtual storage is only used for reader files."

The other spool file types are uniquely identified via SFBSYSID, too. Each spool file in the system corresponds to a specific location in the checkpoint area and SYSSPOOL's storage. Although these SFBLOKs do not reside in SYSSPOOL's virtual storage (they reside in CP-free storage), the corresponding address in

**FIGURE 2**

| | | |
|---|---|---|
| 10 users * 9900 SFBLOKs/user = | 99,000 | SFBLOKs |
| 10 users * ((9900 + 1024 − 1 SFBLOKs)/ 1024 SFBLOKs/SPUBLOK = | 100 | SPUBLOKs |
| 1 user * 3296 SFBLOKs/user = | 3,296 | SFBLOKs |
| 1 user * ((3296 + 1024 − 1 SFBLOKs)/ 1024 SFBLOKs/SPUBLOK = | 4 | SPUBLOKs |
| | 102,400 slots | |
| 99,000 + 3,296 = 102,296 SFBLOKs | | |

*The theoretical maximum number of spool files on a VM/SP HPO 5 system.*

SYSSPOOL is unavailable for reader spool files. This means that non-reader files chew up SYSSPOOL storage.

Every spool file on the system corresponds to a unique location in the checkpoint area and in SYSSPOOL's virtual storage. SFBSYSID keeps everything straight.

The maximum number of spool files is a self-defining variable except when the SYSSPL parameter of the SYSRES MACRO in DMKSYS is used. The primary purpose of this parameter is for the migration to a pre-HPO 5 system from HPO 5. SYSSPL controls the maximum number of spool files that can be created. By setting SYSSPL to 9900, backward migration is permitted. The value of SYSSPL can range from 1000 to 100,000. However, except for the migration reason, it should not be specified so the system can use the checkpoint and SYSSPOOL virtual storage sizes to control the maximum number of files.

## "How does the number of spooling users, real spool devices and the HOLD command affect the maximum number of spool files?"

For CKPT IPL purposes, there are other control blocks that are written to the checkpoint area besides SFBLOKs. Because these blocks are written to the checkpoint area, they require slots in SYSSPOOL's storage.

User's unique spool file identifiers are controlled by bit maps called SPUBLOKs. SPUBLOKs are maintained in SYSSPOOL storage with reader SFBLOKS. In addition to containing their unique identifier (SPUSYSID; equivalent to SFBSYSID), they contain a 128-byte bit map that corresponds to the user unique spool file identifiers (SFBFILID). The first SPUBLOK for a user controls the assigning of files one through 1024 inclusively. SPUBLOKs are the same size as SFBLOKs.

For every real spool (unit record) device, a "dummy" SFBLOK is allocated to contain information needed for the automatic starting of these devices during CKPT IPL initialization. These SFBLOKs are allocated out of SYSSPOOL virtual storage and the checkpoint area just like real SFBLOKs as they are assigned a unique SFBSYSID value.

Each user on the system that has his/her virtual print or punch placed on hold via the HOLD command requires checkpoint and SYSSPOOL storage. The HOLD command causes the creation of an SHQBLOK in CP-free storage that is assigned a unique identifier (SHQCKPT; equivalent to SFBSYSID and SPUSYSID) for checkpoint purposes. Each SHQBLOK makes a 160-byte chunk of SYSSPOOL virtual storage unusable for spool files.

## "What is the theoretical maximum?"

Assuming that there is enough checkpoint area for a 16MB SYSSPOOL virtual machine, the maximum number of slots available for SFBLOKs is 102,400 (25 SFBLOKs/page * 16MB/4K/page). It is not possible for all of these slots to be SFBLOKs because user-unique bit maps reside in this area as well. With the 9900 limit per user, no unit record devices and no held users, the theoretical maximum of spool files is 102,296.

Now even the VM spooling subsystem is "virtual."

*ABOUT THE AUTHOR*

*Thomas E. Cooper is a senior consultant for VM/CMS Unlimited, Inc. From his home in Powell, OH, he supports the Single System Image (SSI) product by telecommuting to the San Diego data center. He is involved with various VM user groups and has given presentations on the CMS file system. VM/CMS Unlimited, Inc., 10060 Willow Creek Rd., San Diego, CA 92131, (800) 443-4317.*

ICCF *from page 72*

tries during a RESTORE. The payoff for using imbedded freespace comes for those libraries considered volatile with regard to adding and deleting members and in expected growth.

For obvious reasons, the ALTER library command requires that the library number be specified. Only those operands given on the ALTER command are changed. All other attributes remain intact as set up by the original ADD command.

**FIGURE 1**

```
// Job
// DLBL    DTSFILE,'ICCF.LIBRARY',99/365,DA
// EXTENT SYS011,SYSWK2,1,0,133148,75000
// EXTENT SYS011,SYSWK2,1,1,426500,204000
// ASSGN  SYS011,FBA,VOL=SYSWK2,SHR
// EXEC   DTSUTIL
FORMAT    LIBRARIES(38), USERS(32)
ADD       LIBRARY,MAX(150)
ADD       LIBRARY,MAX(150),FREE(20)
ADD       USER ID(MA01),LIB(1),PASS(D$ADMN),OPTA(01110111)
ADD       USER ID(MABA),LIB(3),P(MYINIT),MAXST(6000),SEC(18 19)
/*
/&
```

**FIGURE 2**

```
// Job
      .
      .
// EXEC   DTSUTIL
ADD       MEMBER (25,PR730C,SD27(password)     /*  see note below
include member
END OF MEMBER
/*
/&
Note:  25 = Library number   PR730C = member name   SD27 = Userid
```

**FIGURE 3**

```
// Job
// DLBL    DTSFILE,'ICCF.LIBRARY',99/365,DA
// EXTENT SYS011,SYSWK2,1,0,133148,75000
// EXTENT SYS011,SYSWK2,1,1,426500,204000
// ASSGN  SYS011,FBA,VOL=SYSWK2,SHR
// ASSGN  SYS005,TAPE
// EXEC   DTSUTIL
BACKUP
/*
// MTC    REW,SYS005
/&
// JOB
// PAUSE     *** CHECK BACKUP LISTING BEFORE CONTINUING ***
// ASSGN  SYS005,UA
// ASSGN  SYS004,TAPE
// TLBL   DTSRSTR,'DTSRSTR'
// EXEC   DTSUTIL
RESTORE
/&
/&
```

## The ADD And ALTER User Command

When adding a new user profile record, specify the user ID, Password and the Library. Nearly all of the numerous parameters available for this command have default values. Be sure you are aware of what these values entail and that they adhere to your installations guidelines (see Figure 1).

Once again, the ALTER side of this coin is used to modify existing parameters but leaves unchanged those not stated.

## Other Major Functions

DTSUTIL is used to backup and restore the ICCF/VSE library file. It may be used to pull individual members or entire libraries from the backup to be restored.

Under ordinary circumstances, DTSUTIL will be run while VSE/ICCF is inactive. Should the utility be run while VSE/ICCF is active, it is allowed read-only access to DTSFILE except for the adding or altering of Library and User options.

## The ADD Member Command

To add a member to an ICCF library, VSE/ICCF must not be active. Options required are: (1) the library number to which the member will be added (2) the name to be given the member (3) and a userid to be associated with the member (see Figure 2). Optionally, a password may be included. If the member is in compressed format, the CMPRSD option must be included.

## The BACKUP Command

The BACKUP command may be issued to backup the VSE/ICCF library file in an interactive partition but there must not be any other updating of the file during the backup.

This function will backup to either tape or disk. If the backup is to tape, then the assign for the backup file, DTSBKUP, should be made to SYS005. During backup the user profile records are written twice, once in their normal occurrence and once at the end of the backup file. At the final write of the user records, accounting information is updated. At the same time, the user profile records on DTSFILE are also updated. This accounting information provides data on space utilization.

If BACKUP is not run in an interactive partition, VSE/ICCF must be inactive (see Figure 3).

**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.

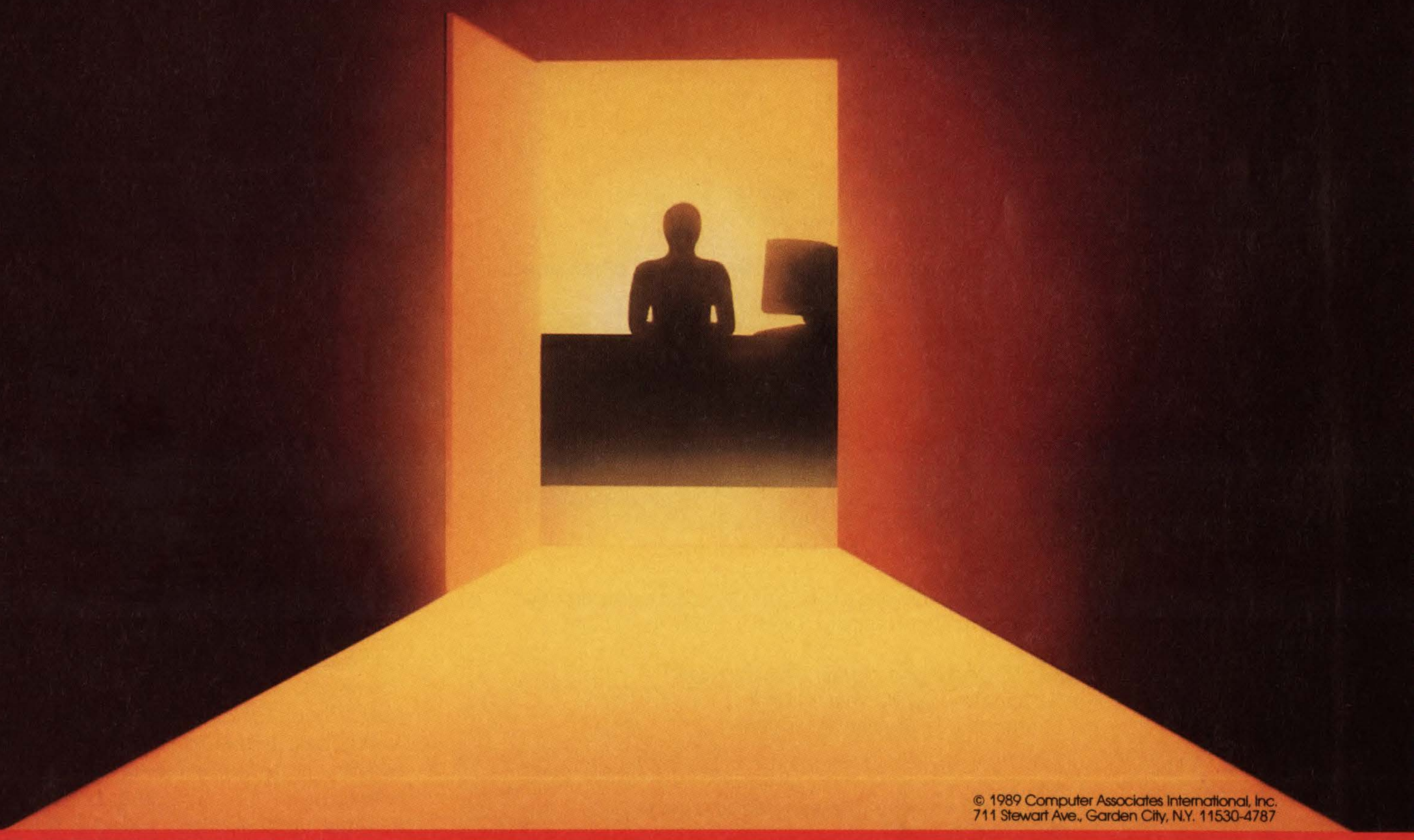
**CA-UNIPACK/DCA–DATA CENTER ADMINISTRATION**
Consisting of: CA-NETMAN®/FINANCIAL-INVENTORY, CA-NETMAN®/PROBLEM-CHANGE, CA-NETMAN®/OLCF and CA-NETMAN®/MRM.

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* ▲

When the backup is for a user library, a directory listing is output giving a record count by member name.

## The MERGE Command

The purpose of the MERGE command is to allow archiving of inactive members.

During execution of this function, the backup file (DTSMERGE) is input along with DTSFILE. The new backup file will be in normal backup format and contain everything on the current VSE/ICCF library file plus any members on DTSMERGE not found on DTSFILE.

When a restore is made from this backup, only the active on-line members will be restored. The archived members will not be restored, unless RESTORE ALL is requested. More later.

Both the input file and the output file must be located on the same device type. If the device type is tape, the input backup file must be assigned to SYS004 and the output backup file must be assigned to SYS005.

A directory listing is produced by MERGE just as in BACKUP, but there are some differences in the listings produced by these actions.

On the directory produced by MERGE, the first column gives a status code for each member. A blank entry indicates the member was copied from DTSFILE. An asterisk indicates the member was found on both input files and that the member found on the backup input file has an inactive status. In this situation the member currently on the VSE/ICCF library file will be carried over to the new backup. This could present a problem if a member had been archived and a totally new member was inadvertently given the same name.

If you are using MERGE for the archives feature, the asterisk status code should be checked carefully to be sure you will not be losing a good file.

Another cause of the asterisk status is that the member was restored since the last MERGE was run.

An "N" status code denotes a member, active at the time of the MERGE that created the input backup file, purged from DTSFILE. Now this member will be archived; that is, placed on the output backup with an inactive status.

The "C" status is the normal status for previously archived files which are simply transferred to the new backup without change.

## The RESTORE Command

When using any form of RESTORE, VSE/ICCF must not be active.

The system RESTORE will load a formatted DTSFILE from backup. It can also be used to change the maximum number of user profile records and the number of library records.

If there is a need to change the size of DTSFILE or other of its physical characteristics, a FORMAT command can be placed before the RESTORE.

When ALL is requested and the backup contains archived members, these will be restored also. Usually, ALL is not specified and only members active at the time of MERGE will be restored.

If the backup was not created by MERGE but by BACKUP, archival files will not be a problem.

Non-common members in libraries can be restored to each library directory in alphabetical order by choosing the SORTED option.

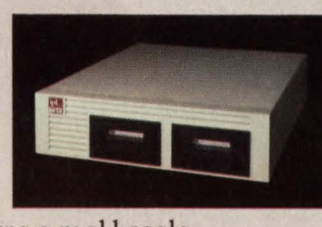
# Introducing a subsystem that cuts through the brown tape.

**IBM mainframe users can now compress 24 reels of data onto one 8mm cassette.**

It's called the 6800 Series Cartridge Tape Drive with the F1011 Data Compression Feature. Using IBM's SNA compression algorithm, the F1011 enables you to put a colossal 4.5 gigabytes onto one compact 8mm cartridge. Without operator intervention.

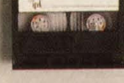
What's more, a fully configured 6860 tape subsystem features up to 7 cassette drives, which essentially replaces 196 reels. Goodbye stuffed storage rooms. Goodbye downloading monotony.

Of course, the IPL 6800 Series F1011 Feature uses no more of your system's resources than a normal tape drive operation. And as with all IPL products, the 6800 connects directly to your IBM system. There's no need for hardware or software alterations.



So if you're looking for a way to cut through the problems associated with reel-to-reel backup and storage, consider the IPL 6800 series. It's an innovation that turns a reel hassle into a real pleasure.

Call IPL today at 1-800-338-8ipl, in MA (617) 890-6620. Or contact us at 360 Second Avenue, Waltham, MA 02154.



ipl systems, inc.

## The RESTORE Member Command

In the member RESTORE the library number given as a parameter of MEMBER will determine which library on the backup file will be searched. If this number is not given in the RESTORE, DTSUTIL will search for the given number from the beginning of the file and accept as *found* the first match. The library where this match is made will determine the library to which the member is added if the TOLIBRARY option is omitted.

The TOLIBRARY option must be used if the library where the member was stored no longer exists on DTSFILE.

Before the member is added to a library, DTSUTIL checks for duplicate members and deletes the duplicate before restoring from backup.

The member cannot be deleted if it is a common member. In order to restore such a member, the common member must be removed from DTSFILE using PURGE prior to the RESTORE MEMBER command.

## The General Maintenance Commands

### *DELETE*

The DELETE command is used to reset the library header record or user profile record specified in the command.

A library cannot be deleted unless either the associated user profile record is also cleared or the profile record is altered to change the primary/alternate library to another library.

The space used by members and the directory attached to a library which has been deleted will be released, becoming part of VSE/ICCF freespace after the next reorganization of the library file.

At this point, I would like to mention that there are several methods of reorganizing DTSFILE. Running BACKUP and RESTORE is perhaps the most popular and efficient method. But, DTSANALS, the utility program mentioned earlier, offers a REORG command which only runs after the ANALYZE command has executed and has the same reorganizing effect.

### *PURGE*

The PURGE command is used to delete common members, broadcast records, all members within a library except common members, members within a library that meet specified criteria and message members.

### *SHARE*

This command is used to change a member or members from Private or Public to Common.

In order for this to be accomplished, the member name(s) must not be duplicated in any other library.

## The DISPLAY Command

Through the use of DISPLAY, the ICCF administrator has access to accounting information which could point out areas of library and system usage inefficiencies.

All options of the DISPLAY command may be run while VSE/ICCF is running with the exception of the ACCOUNT option.

Some of the accounting information available follows: (1) userid (2) library assigned to this user (3) number of logons to ICCF (4) number of ENTER requests (5) number of execution requests (6) total execution units used (7) date of last logon (8) total space usage (9) total logon time (10) total time in the interactive partition.

This information continues to be updated until a request is made to CLEAR the accounting information via the DISPLAY Account Clear command.

## Accessing Members From Backup

The PRINT, PUNCH or PRTPCH commands are used to reproduce a member, message or library from either DTSFILE or a backup file.

To give an example, an archived file may be restored by first issuing the INPUT BKUP command to direct the following PUNCH command to access a backup file as opposed to its normal access of DTSFILE. Follow this command with the actual PUNCH command to move the member to the punch or punch queue. Finally, restore the member to DTSFILE through the ADD member command.

The INPUT BKUP command is valid only prior to issuing a DSERV (directory listing), PRINT, PRTPCH or PUNCH command.

## DTSANALS

The primary usage for DTSANALS' ANALYZE function is for recovering from a system failure. When the primary objective is either file reorganization for efficiency or recovery from an I/O error, the BACKUP and RESTORE functions for DTSUTIL are recommended. But when recovering from a system failure where VSE/ICCF was terminated without the normal EOJ routine or the ABEND STXIT routine was incomplete, the RECOVER function should be run.

## The ANALYZE Command

The first phase of the ANALYZE execution sets a bit map with the location of all recognized chains of records. The second phase then processes any records found to be unchained. These are placed at the first of the free chain area and will be printed. It is then the responsibility of the user to determine if these records need to be rechained.

After ANALYZE has completed, the next step is execution of the RECOVER command. If any updating commands are executed between the ANALYZE and RECOVER commands, ANALYZE will automatically run again before RECOVER is done.

Though ANALYZE may be used followed by REORG to reorganize the VSE/ICCF library file for improved performance, as stated earlier it is better to use BACKUP in conjunction with RESTORE. One reason for this recommendation is that with BACKUP/RESTORE, location of records used for spool allocation is optimized. Also records captured for input and any additions to library members are physically closer on the backup file.

There is much to be said about the functions of DTSANALS. So much, in fact, that I will leave that for another day. But from the information here, you should begin to have some idea of the scope of this utility.

## IN SUMMARY

DTSUTIL is the prime VSE/ICCF source of library maintenance in the VSE environment, but your forethought and planning can help make file management an integral part of the daily processing routine.

*ABOUT THE AUTHOR*

*Sharon Hooper Martinez is an independent consultant under contract with Champion International Corporation (Lufkin, TX). Her IBM and DP experience spans more than 11 years including her position as a programmer/analyst with both Oshman's Sporting Goods (Houston, TX) and Temple Associates, a subsidiary of Temple-Inland (Diboll, TX). (409) 829-5648.*

ISPF *from page 24*

supply ASA carriage controls (for example, "0" for double spacing) instead of a blank line: the zero in column one is a change in indentation from the normal blank (ASA single spacing carriage control) in column one.

Getting TF to work in paragraphs formatted with hanging indents (after the first line all lines are indented, such as would be used in numbered points or field descriptions) requires the RIGHT Primary Command. TF only re-formats text in the columns being displayed on your screen. If you are using a hanging indent of 20 columns, you would enter the RIGHT 19 Primary Command to display columns 20 on. After the display has been moved to the right, you would enter TF64 on the first line of the paragraph and the text would be distributed between columns 20 and 64 of each line of the paragraph.

One situation may leave you scratching your head, however. If you have column bounds set for the member (the BNDS Line Command will display the current column bounds), only the text within the column bounds will be formatted by TF. This can make a real mess of your text, because text outside of the column bounds is left in place, potentially splitting a word and leaving the two halves several lines and many columns apart.

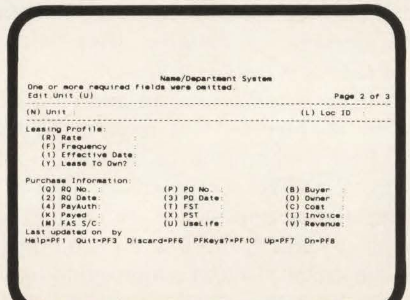
### *Next Time*

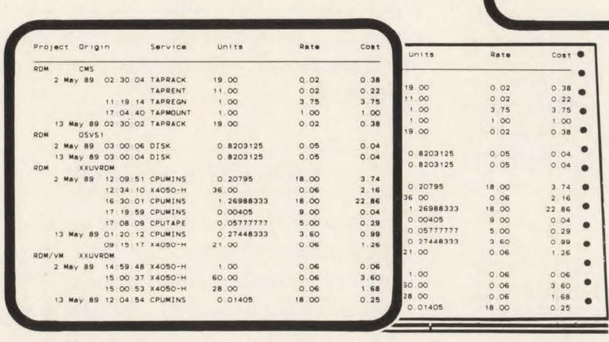
Are there other ways to work with text using ISPF? Certainly. After all, text formatting is a popular topic. If you have text-related suggestions or any other ideas or questions about ISPF, contact *MAINFRAME JOURNAL*.

*ABOUT THE AUTHOR*

*Jon E. Pearkins is President of Certified Software Specialists Ltd., a Canadian company with a worldwide customer base for its line of IBM mainframe systems software. Certified Software, 54015 Range Road 212, Ardrossan, Alberta, Canada, T0B 0E0. Phone (403) 998-0607.*











TGT *from page 52*

same order as the non-VSAM files are defined in the FILE SECTION of the program. The DCB addresses are located in the PGT which is listed directly after the TGT and the literal pool.

[*Directly after the TGT is the storage map of I/O control blocks and of WORKING-STORAGE. It lists displacements from the program beginning and lengths of various areas including all DCBs, FCBs and WORKING-STORAGE. (The addresses of the areas are not listed if the program is compiled with the RENT option. In this case the DCB (for non-VSAM files) or the ACB (for VSAM files) can be found by using its address located at hexadecimal displacement '78' in the FCB.)*]

Think of the processing needs of a major bank. If its overnight processing ABENDs and fails to transfer funds to the government, more than a million dollars in interest is lost. The bank does not want to play debugging roulette when one of its programs is not working at 3 a.m.

Even problems of a much smaller magnitude are critical to companies. If a problem occurs during a production job that must run, you often do not have enough time in the day, nor computer processing time, to guess at solutions and hope they work. Remember that a dump may occur after many hours of CPU processing or in a program that runs just before daybreak. Knowledge and understanding of COBOL, including use of the TGT, is sometimes the only chance to resolve the problem in the necessary time frame.

The TGT is an important source of information for a COBOL programmer. Although the TGT is not an area used by a programmer on a daily basis, it often contains the key, the vital information, in debugging. Its proper use can make the difference between solving and not solving a problem.

*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.*

NCP *from page 32*

If you have EP lines in your communications controller, you will have a separate operating system channel address for each EP line. For a PEP, you will have both EP channel addresses and the Native Subchannel Address.

## Intermediate Network Node

The NCP provides the capability to receive data that is bound for other NCPs or for a VTAM system. The Intermediate Network Node (INN) function looks at the destination address to see where this particular data is going, then looks at the PATH tables you provided for routing to decide how to forward it to the next point along the path.

All data passing through the NCP will be viewed by the Intermediate Network Node modules.

## Boundary Function

The boundary function is done for devices which are using SDLC for their line protocol and is sometimes called the Boundary Network Node (BNN) function.

When the NCP receives data in which the address indicates that it should be delivered to a cluster controller which is attached to this communications controller, the boundary function is performed. The SNA headers for the data are changed to eliminate unnecessary information; for example, origin and destination addresses are changed from network addresses to local addresses. In addition, if any data is too large to be delivered to the cluster controller, it will be segmented as part of the boundary function. For a 3274, which can receive a maximum of 265 bytes of data at once, the NCP will break down anything larger into multiple segments with no more than 265 bytes in each.

## Link Scheduler

There are modules which are concerned with sending and receiving the control information for those SDLC devices which are attached. As bits come in across the line, scanners in the communications controller pass them to the link scheduler programs. The bits are assembled into bytes and headers and trailers are read and understood. If errors are detected, retransmission is requested. Otherwise, the data is passed to the INN function to be forwarded on the path to its destination.

The link scheduler is also responsible for sending data out to devices which use SDLC protocol. Then the headers and trailers for the data must be created and the data is made available to the scanners to be sent. Of course, the data must be kept in storage until the receiver confirms delivery.

In cooperation with the scanners, the link scheduler polls the clusters on the SDLC lines. Polling allows the NCP to solicit input from devices on a regular basis in the order you specified in the NCP gen.

The link scheduler handles all SDLC lines; these include lines to other NCPs as well as lines to cluster controllers.

## BSC Scheduler

When SNA was first announced, most installed devices used BSC line protocol. Large IMS and CICS systems had BSC interactive devices and printers which were controlled by BTAM. There were no SDLC devices until VTAM and NCP were installed.

In order to help users migrate to an SNA environment, IBM provided the BSC Scheduler modules as part of the NCP. VTAM was able to communicate with certain 3270-type BSC devices, allowing them to logon to various VTAM applications. If you have devices defined in your NCP gen with TYPE=NCP and LNCTL=BSC on the GROUP macro, you have lines which are using the BSC Scheduler.

This function takes the SNA data format and translates the information into a format that can be sent to the device using BSC line protocol. The only BSC devices that are supported by VTAM and the NCP are 3270-type controllers.

## SNI Modules

SNA Network Interconnection (SNI) allows you to connect your network with its naming conventions and addressing schemes to other networks which may have conflicting conventions. To do this, you define a VTAM system and its System Services Control Point (SSCP) as the Gateway SSCP and your NCP as a Gateway NCP. The Gateway NCP will perform certain SNI functions, one of which is to maintain an alias table to translate addresses which come in from the *foreign* network to addresses set aside in your own NCP. Therefore, applications which are communicating with devices from the foreign network think that those devices are attached directly to the Gateway NCP.

SNI has allowed many companies to exchange information with other firms which provide services to them or which are suppliers or customers for goods they produce.

## Physical Services

The NCP has certain modules that communicate with VTAM to help control the resources defined and attached to the NCP. These modules are called Physical Services modules or sometimes the Physical Unit (PU) for the communications controller. If you issue a command to VTAM to activate a line defined in your NCP gen,

```
V NET,ACT,ID=LINE015
```

an SNA ACTIVATE LINK (ACTLINK) command will be sent from VTAM to the NCP. NCP will then perform the necessary function and return a response to VTAM.

Every time your operators issue commands to start or stop NCP-defined remote devices, VTAM will be sending corresponding commands out to the Physical Services modules of the NCP.

## An Important Part Of The Network

The NCP constantly assists the VTAM system on your host by picking up information that is arriving asynchronously from devices attached by remote lines and by delivering data to those devices. The communications controller hardware is specifically designed to handle these activities and has a scheme that allows the highest priority task to be running at any point in time. Consequently, unless there is some unusual problem, there is little delay in getting data back and forth from the host and the remote device.

The NCP has proven to be a reliable software product allowing you to offer your users resource sharing, improved performance, better error recovery and, in some cases, reduced costs.

*ABOUT THE AUTHOR*

*Beverly J. Weaver is an independent consultant living in Springfield, IL. She is currently leading seminars for Verhoef Information Packages, Inc., a firm that specializes in training and education with emphasis on MVS, CICS and data communications. Her primary areas of focus are data communications and IMS. Verhoef Information Packages, Inc., 799 Bloomfield Ave., Suite 310, Verona, NJ 07044, (201) 239-1005.*

# Innovation Data Processing

## *System Managed Software Tour Packs Them In*

Why have more than 2,000 storage administrators and systems programmers from approximately 1,000 of the nation's largest and most prestigious companies been meeting in major cities across the country to hear one software company speak on system managed storage? Why have managers in more than a dozen cities sent their best technical representatives (and it is not IBM making the presentations)? The answer is Innovation Data Processing, one of the most influential vendors of DASD storage management software for large scale IBM systems in the world today.

Actually it should be no surprise that the company's extensive install base would provide such large, attentive and supportive audiences. When Innovation introduced what is now its most well-known product, FDR, there was no independent software market offering DASD management products for IBM systems — that was in 1972. Today, with more than 6,000 FDR licenses sold, the company is the leader in the field and unquestionably commands a distinguished audience.

IBM caused quite a stir in February 1988 with dual announcements of Enterprise Systems Architecture (ESA) and the Data Facility Storage Management Subsystem (DFSMS). Bold headlines in the trade papers questioned if the independent vendors of storage management software would be able to compete. It has been 18 months and not only is Innovation still around, but also it is packing them in across the country. Additionally, the message being sent seems to be just what customers want to hear. For the company and users of the FDR family of DASD storage management products, it will be "business as usual."

The style of the company's presentation reflects the style of the company. It is fast-paced, reliable, based on an abundance of technical detail and simply presented. More than anything else, however, it is the level of technical detail that elicits a special respect from audiences. A comment heard most often is, "Really enjoyed listening to a presentation that gave a better understanding of system managed storage and how to deal with it, instead of the usual sales hype."

*Anthony Mazzone, President, Innovation Data Processing.*

The message that it will be business as usual, in the face of what might seem to be a formidable endeavor, should not be unexpected. When FDR was introduced in 1972, it was competing against IBM utilities that were provided free with the operating system. In 1980, the first year ABR was sent into the storage management fray, it secured two percent of the market. Today, better than one of every three sites using a DASD storage manager use ABR. The company had a stand-alone restore for the XA architecture before IBM did. It delivered support for the device within three weeks of its availability when IBM delivered the triple density 3380-K. That is quite an accomplishment. However, more importantly, it is a tribute to the talent and quality of the staff.

By maximizing and leveraging the company's talents, it was one of the first software firms to have its own dedicated development machines, having had its own OS development machines since 1975. Today, there are machines running SP, XA and, since January 1989, ESA on a 4381-91 E that was installed specifically for ESA/SMS development.

The company's objective is to be recognized by management as the best in the business of DASD management. President Anthony "Tony" Mazzone urges, "Look at our products — FDR, CPK, ABR and IAM. DASD management is our business. We want to provide products that meet your needs. We would not be in business now if we hadn't met those needs in the past and we know you are going to be tough judges of how well we do in the future."

Based on the response to its system managed storage perspectives seminars, Innovation is not likely to be judged poorly. Managers and technicians alike seem to appreciate not only the company's products, but also its straight-forward candor and strength on technical detail. These two attributes combine to characterize the company's approach to its products as well as its way of doing business.

***Vendor Profile is a regular forum whereby a vendor is given the opportunity to introduce the company and its products to MAINFRAME JOURNAL readers.***

# Tuning VSAM Index Control Intervals

## *Examining Index Records*

By Michael D. Sachais

There is a misconception about tuning index Control Intervals (CIs) in today's DP environment. Most literature you read about VSAM implies it will use the proper index CI size for your KSDS clusters if you: 1) do not code the CI size parameter for the index component of a cluster or 2) code an index CI size which VSAM calculates to be insufficient for the cluster you are defining. These implications could not be farther from the truth.

If VSAM calculates that you have coded an insufficient index CI size that is too small to address all of the CIs within a Control Area (CA), it will attempt to adjust the index CI size to what it determines to be a proper size. The problem, however, is that VSAM cannot possibly do this effectively because the proper size of an index CI depends on the key compression occurring in the index component of the cluster. At the time a cluster is being defined, VSAM has no idea what the data stored in a cluster will look like. It is therefore impossible for VSAM to calculate an accurate key compression for the key entries to be stored in the index CIs.

This is the first in a series of articles discussing VSAM index CIs and how to tune them. The series will be broken down into three parts, each covering a different aspect of index CIs and how to tune them. In this article you will learn about the format of VSAM index records and how they are used to store the keys of a VSAM KSDS cluster.

The following two articles will: teach you how to print and read index records (similar to reading a dump) to determine VSAM's key compression rate; and how to test and choose the most effective index CI size for a given cluster. After reading all three articles you will be able to accurately tune index CI size which in turn will minimize DASD space utilization by a KSDS cluster and minimize BATCH and CICS processing times when processing the cluster.

### How VSAM Adjusts Index CI Sizes

When you define a KSDS cluster, VSAM analyzes the index CI size you code (if you code one) to determine if it is large enough to store all the key entries needed to address the CIs in a CA. If VSAM determines the index CI size you coded is not large enough, it will begin adjusting some of the parameters you coded on the DEFINE CLUSTER command to try to achieve a proper index-to-data size.

Figure 1 illustrates a partial LISTCAT for the KSDS cluster named MY.KSDS.FILE. In this figure you can see the cluster has a data CI size of 4096 (CISIZE in DATA ATTRIBUTES subsection) and an index CI size of 1536 (CISIZE in DATA ATTRIBUTES subsection). However, when I DEFINED this cluster I specified a data CI size of 4096 and an index CI size of 512. As a result, VSAM adjusted the index CI size to 1536. How did VSAM determine that 1536 was the proper index CI size to use?

First, VSAM calculates the number of data CIs that will fit into one CA. In the LISTCAT in Figure 1 you can see that

**F I G U R E 1**

**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
```

there are 150 CIs per CA in the data component of the cluster. Next VSAM calculates whether the index CI size that has been coded is large enough to store the key entries for all the CIs in the CA. VSAM assumes that the average key entry in an index record will be eight bytes long. The eight bytes are composed of a five-byte compressed key and three bytes of control information.

VSAM's assumption of eight-byte key entries holds true in a DFP 2.2 environment. What VSAM determines the average key entry length to be differs between VSAM releases. However, regardless of the release of VSAM you have, VSAM cannot correctly assume the size to which your keys will compress and, therefore, its assumption of the average key entry length will be a generalized estimate.

If VSAM determines that the index CI size is not large enough to store all the key entries, VSAM increases the index CI size (if possible) to a size that it calculates to be sufficient. VSAM calculated the proper index CI size for the cluster MY.KSDS.FILE as:

8 bytes per entry X 150 data CI/CA = 1200, rounded up to the nearest valid control interval size which is 1536.

Notice that VSAM rounded the answer 1200 up to 1536. This is because 1200 is not a valid index CI size to VSAM. VSAM therefore rounded the number 1200 up to the next valid index CI size (1536). If VSAM cannot increase the index CIs size to a valid CI size, it will reduce the CA size to reduce the number of data CIs that will fit in one CA.

The problem with VSAM's method of calculating the proper index CI size is that VSAM makes a broad generalization. VSAM assumes that the average size of a key entry is eight bytes. What happens if the keys in your cluster compress poorly and as a result the key entries are larger than eight bytes? VSAM's attempt to correct the index CI size for you is still wrong.

As you continue through the series of articles you will see that the cluster MY.KSDS.FILE has this exact problem. The average compressed key length in the cluster is greater than eight bytes. Therefore, VSAM calculated an insufficient index CI size for the cluster.

## Format Of An Index Record

Index records are stored in CIs in the same way that data records are, except that there is only one index record per index CI and there is no FREESPACE left in the CI. Therefore each index record is stored in its own index CI.

The size of each index record will be the index CI size minus seven bytes. The seven bytes are used for the CI information. Because each CI will store only one index record, the CI information in every index CI will consist of: one three-byte RDF that contains the length of the index record and one four-byte CIDF that also contains the length of the index record along with the amount of FREESPACE (0) in the index record.

Index CIs are not grouped into CAs like data CIs. Instead, they are stored independently of one another. When a new index record is created, it is stored in a new index CI at the end of the index component. Therefore, index set and Sequence Set Index (SSI) records will be intermingled in the index component unless the IMBED parameter is used. When the IMBED parameter is used, the SSI records will be stored with the data

CIs instead of with the index set index records.

A SSI record can be broken into four parts: a header, a list of the free CI pointers, unused space and the key entries. Each part will be analyzed separately.

## The Index Header

The first 24 bytes of each index record store the index header. The index header contains information describing the information contained within the index record such as the length of the index record and the start of the unused space in the index record. This information will be used to calculate the average key compression that is occurring to the keys stored in the index record.

## The Free CI Pointer List

The free CI pointer list begins immediately after the index header. It contains pointers to all of the unused (free) CIs in the CA governed by the sequence set index record. The pointers are stored in descending order and are used from right to left.

When a data CI becomes used, its pointer is converted to binary zeros and becomes part of the unused space in the index record. A key entry is also created for the newly used CI and stored in the key entries portion of the index record.

If all of the data records are deleted from a CI causing the CI to become empty, the pointer to the CI is re-added to the end of the pointer list. Therefore, the CI will be the next CI within the CA to be utilized.

## The Unused Space

The unused space in the index record is made up of binary zeros. It is used to hold additional key entries which need to be stored in the index record. Key entries are stored in the unused space from right to left. The lowest key in the index record will be located in the rightmost key entry and the highest key in the index record will be located in the leftmost key entry. Look at an example of how key entries are stored in the index record.

Figure 2 illustrates a SSI record that governs a CA containing five CIs. Only two of the five CIs are used. Looking at the SSI record you can see that the free CI pointer list contains the three unused CIs in descending order (five, four and three). You can also see that the key entries in the index record contain the highest key in each used CI and a pointer to the CI containing the key. To make the example as simple as possible, it is assumed that each key entry in the index record will be eight bytes long (a five-byte key plus three bytes of control information) and no key compression is occurring.

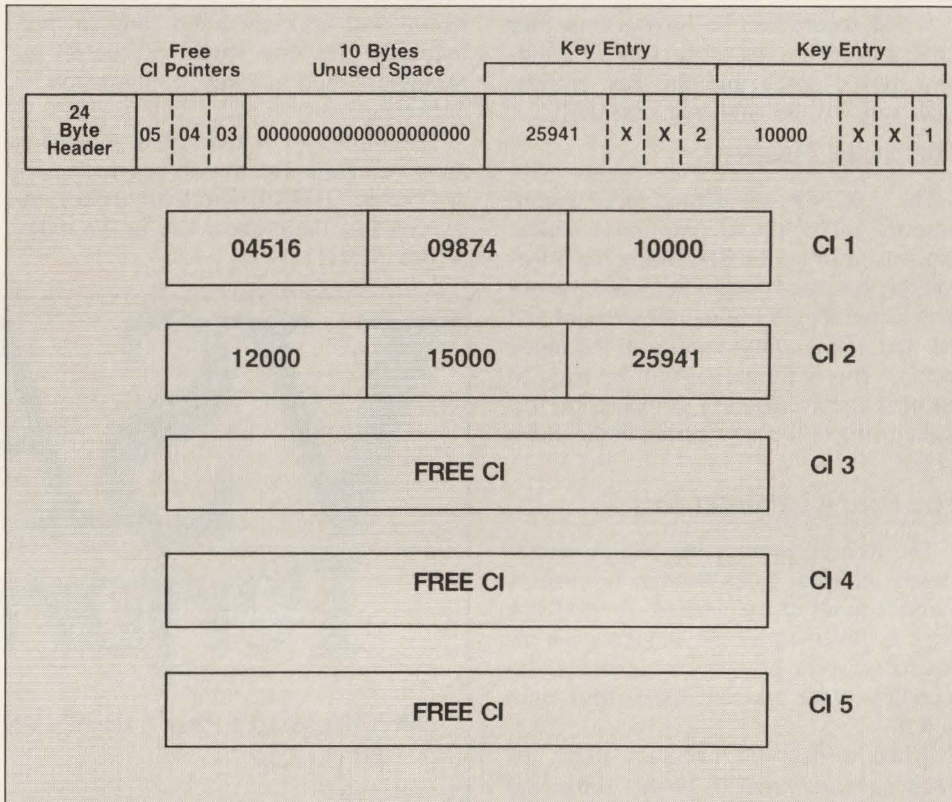
The rightmost key entry in the index record contains the lowest key in the index record (10000). The leftmost key entry contains the highest key in the index record (25941).

Figure 3 illustrates the SSI record and data CIs after record 30000 has been added to the CA. VSAM determined that record 30000 will reside in data CI three. Notice that the key entry for the new CI was stored in the eight rightmost bytes of the unused space. In addition, because CI three is no longer unused, the free CI pointer for CI three was converted to binary zeros and became part of the unused space.



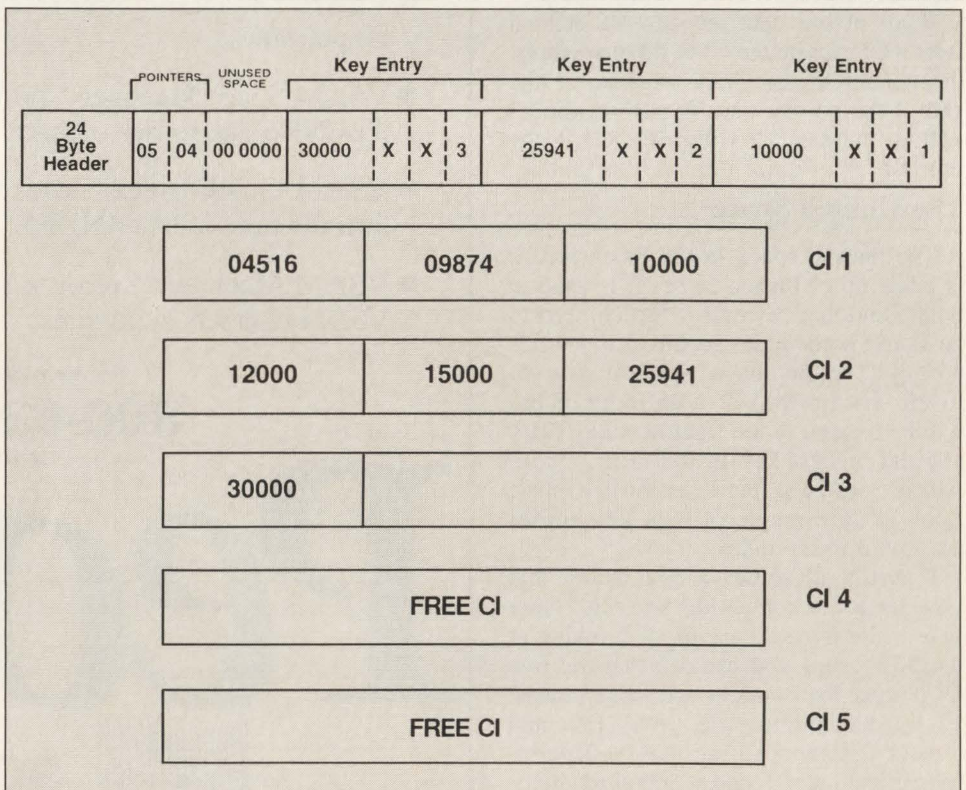
 

**FIGURE 2**




*An SSI record containing two key entries; the SSI record governs a CA containing five data CIs.*

**FIGURE 3**




*The key entry for data CI three is added to the SSI record.*

An index record will continue to store key entries for data CIs as long as there is enough unused space to store the key entry. When the amount of unused space in an index record is insufficient to store an additional key entry, the index record is considered to be full. If a sequence set record becomes full before the unused CIs in the CA are used, the unused CIs in the CA cannot be utilized. This could result in excessive amounts of wasted DASD space.

The amount of unused space left in the index record in Figure 3 is three bytes. Because each key entry requires eight bytes, there is insufficient room in the SSI record to store another key entry. Therefore, data CIs four and five in the CA cannot be addressed by the SSI record. This will result in 40 percent (two out of five CIs) of the CA in the cluster being wasted. If two out of every five CIs in each CA are wasted, DASD space requirements for this cluster will increase by 40 percent.

From the above example you can see why it is crucial to choose an index CI size that can store the key entries for all of the CIs in the CA governed by the index record. Knowing the average key entry length can help you calculate the proper index CI size to use. This value can be calculated by using the information stored in the index records of a VSAM cluster.

In the next article you will learn how to calculate an index CI size that can store the key entries for all of the CIs in the CA governed by the index record. This will minimize the amount of DASD space wasted by a cluster. This in turn can reduce both CICS and batch processing times.


*ABOUT THE AUTHOR*

*Michael D. Sachais is a technical consultant for Management Science America (MSA) where one of his responsibilities is tuning VSAM/CICS systems. 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. MSA, 3445 Peachtree Rd., NE, Atlanta, GA 30326, (404) 239-2648.*

Age Of A Page *from page 22*

second are lost. The same method is used for race horses, of course, and their feelings are not hurt either.

### *LRU*

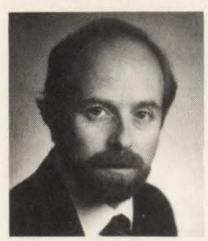
The UIC update routine is used to implement a modified LRU page replacement policy. (UIC update is more generally known as the clock algorithm.) Page UIC values are used to order all the page frames in the system from most recently referenced to least recently used. Technically, UIC is a partial order, while pure LRU is a total order.

An LRU total ordering of the age of all pages in the system, normally implemented as a stack, is considered prohibitively expensive for large memories. A variation of LRU using associative sets is used in CPU caches where performance is also critical and straight LRU is used to manage cached DASD controllers.

LRU may be more expensive to maintain, but it is much faster than the UIC update or clock algorithm for page replacement. For instance, SRM knows that there is at least one page in the system as old as the System High UIC value, but it does not know how many pages exist that are that old. More importantly, it does not know where to find the oldest page without searching all of main memory sequentially. On the other hand, in the usual stack implementation of LRU, pages are explicitly chained together by age. See Figure 1.

UIC update can be a severe bottleneck in a large, multiprocessor with a lot of memory to manage and significant paging activity to a high-speed paging device. For instance, almost any 3090-400 with 256MB of central storage, plus expanded storage for paging may be subject to this large system effect.

### *ABOUT THE AUTHOR*



*Mark Friedman is a Director in the Technical Division of Landmark Systems. He is a frequent speaker on MVS performance topics at SHARE and CMG. Friedman has 10 years experience as an MVS performance analyst and a developer of performance analysis software tools. Landmark Systems Corp., 8000 Towers Crescent Dr., Vienna, VA 22180, (703) 893-9139.*

# Implementing A HyperText Prototype On An IBM Mainframe

By Patricia C. McGrew and William D. McDaniel

Discussions of HyperText have typically assumed that the software system was running on a PC, a stand-alone workstation, some type of Local Area Network (LAN) or minicomputer. Few researchers or writers have addressed the issues involved in bringing the concepts behind HyperText or HyperMedia to the world of the large commercial IBM mainframe user. This article addresses that issue and describes how a prototype for a functional HyperText system which runs in a VM environment on an IBM mainframe was created.

Using what was learned from the process of creating this prototype will make it possible to create a viable production HyperText system for IBM mainframe environments.

## HyperText, HyperDocument and HyperMedia

There are a number of definitions for HyperText and HyperMedia. The following will be used in this article: *HyperText* is the concept of accessing and perusing text in a non-linear manner through the use of software links between document elements, as well as through sequential and associative methods.

It embodies the ability to peruse text using links between related portions of one or more documents. The term *text* applies not only to words, but also to a limited subset of graphic representation such as simple flowcharts and other types of line art which may be displayed on mainframe-attached graphics terminals.

*HyperDocument* describes the universe containing a variety of discrete documents which share links. A HyperDocument can contain only one document that uses only internal links. At the other end of the spectrum are HyperDocuments that contain hundreds of discrete documents which share thousands of common links.

*HyperMedia* extends the concept of HyperText beyond pure text and limited graphics to include on-line, real-time access to all manner of graphics representations, animation, audio and video.

## Why Implement HyperText On A Mainframe?

While many academic and research institutions believe in the use of PCs, LANs and minicomputers, corporate America is the realm of the IBM mainframe. It has a huge investment in IBM hardware, IBM look-alike hardware, IBM software and software which runs only in the IBM mainframe environment. Corporations are also great producers of paper. Every piece of hardware and software has some type of user and reference manual. Internal policies and procedures are also put on paper and distributed.

It begins to make sense that many of those environments must consider providing on-line access to those documents. It makes even more sense to provide that access using the same methods and techniques which are heralded in trade journals. That means using HyperText and related concepts to develop and implement on-line text access and text management systems. This type of system can provide more up-to-date information to more people within an organization in a more timely manner than is possible when dealing with documents on paper.

While not proposing that on-line manuals using HyperText links will *replace* paper documents, we, the authors, believe that once the methods for creating HyperDocuments in a mainframe environment are documented and disseminated, there will be a demand from the mainframe user community to make manuals available in HyperText (non-linear) format.

## Our Development Environment

Our primary environment at Image Sciences, Inc. (Dallas, TX) is VM/CMS, using XEDIT as the text editor. We use IBM's Document Composition Facility (DCF) as our primary host-based text composition system; in fact, we can run DCF Releases 3.0, 3.1 and 3.2. Our language of choice for prototyping is REXX, which we used extensively in the development of this prototype. Our developmental language of choice is C.

## Text Access Styles In A Mainframe Environment

We are primarily concerned with providing on-line access to documents in three situations:

- From within an application program to provide access to syntax and execution information
- In a stand-alone mode for the development, review and approval of on-line documents
- In a stand-alone mode for reference documents not associated with an application program.

There are three styles of on-line text access which could be appropriate for a mainframe environment: associative navigation, HyperText and HyperMedia.

### *Associative Navigation*

Associative navigation encompasses a style of on-line text perusal in which the user is responsible for developing the search arguments and performing the search. It does not require any type of high-tech user interface, although we believe that the best system permits user entry into the on-line document and navigation within the document using function keys or dialog boxes.

When an associative navigation method of on-line text access is provided, only one document is accessed at a time. The user has a great deal of freedom to browse the document looking for information, usually with the aid of a table of contents or an index to help guide a search. Graphics may be included in the document, but the ability to display graphics will depend on the terminal available. We looked at this type of on-line access as an entry point from which we might be able to develop a prototype.

### *HyperText*

The buzzword of the eighties in document processing is HyperText. We defined HyperText earlier, so here we want to further refine the definition based on our findings as we pursued the prototyping process.

First, a HyperText-based on-line text access system no longer uses words as keys in search arguments. It uses something closer to user-controlled Key Word In Context (KWIC) searching with the aid of predefined links to similar or related information items. Access may be to only one document or to the larger group of related documents we call a HyperDocument which is structured for nonlinear access.

As we further refined the concepts we wanted to explore in our prototype, we found it necessary to further divide HyperText into several tiers:

Tier 1: The document author defines the keywords (links) which will be available during the search process

Tier 2: The document reader also has the ability to do associative searches

Tier 3: The document reader can define links during the search process, permitting a customized reconstruction of the HyperDocument.

The creation of HyperText modules is object-oriented, analogous to object-oriented programming. The search process is built on the use of links and nodes to associate concepts across document boundaries. With so much freedom to navigate a HyperDocument, the user interface becomes important. We believe that at this point the most logical presentation method is through the use of windows which is not common in mainframe applications today but will become more so. We will discuss this more as we look at the development of our prototypes in a later section.

### *HyperMedia*

The logical extension of HyperText is HyperMedia. A HyperMedia is not confined to words as the basis of information dissemination. It permits access to animated pictures, videos, voice data and CAD-style three-dimensional graphics.

*There are three styles of on-line text access which could be appropriate for a mainframe environment: associative navigation, HyperText and HyperMedia.*

It became immediately obvious to us that, at least in the short term, we could not create a HyperMedia prototype for the mainframe. Most mainframe-attached terminals do not have the resolution to support the types of graphics required for such a system. The support hardware and software to integrate voice, animation and video display, with links into all of the media for a single HyperDocument, simply are not available to us.

## Project Design

After we reviewed the various concepts in HyperText systems and felt more secure in our definitions, we tried to establish goals. These goals came with some built-in bias toward what is and is not currently possible in a standard IBM mainframe environment. First, our goals.

### *Goals*

The high goal of developing a prototype HyperText system for the mainframe was to demonstrate the feasibility of such a system. We wanted to prove that it is possible to use source code which already exists in the rather extensive text libraries in our environment and recompose them for screen display. The problem we had to solve was how to maintain the information provided by font changes (such as highlighting and heading level differences) in the document composed for the screen that is there when you compose for hard copy output.

When the prototype was complete, we wanted to have a system that would permit us to distribute technical reference or procedural documents in an on-line environment.

### *Constraints*

Implementation of a HyperText system on an IBM mainframe is subject to several constraints which guided the design. These constraints arose from the interaction of a number of factors but mostly from the IBM 3270 terminal hardware. IBM 3270 terminals are in wide use throughout the mainframe user community and come in a variety of models, each having a different set of capabilities. The configuration that forms the lowest common denominator is a 24-line by 80-column 3278 with a single font and two levels of brightness. The upper end of the 3270 line generally available in our offices is a 3179-G that displays 32 lines of 80 columns, has seven colors, four extended highlights such as blinking or reverse video and can display graphics.

We wanted our prototype to support all types of 3270 terminals, so the display of high-resolution graphics was ruled out although one of the early design goals was to simulate font changes via highlighting and color as much as possible on each device.

IBM 3270 screens are also pseudo-interactive. That is, no actual interaction with the CPU occurs until the ENTER key, a Function key or a Program Attention key, is pressed. This limits the amount of direct interactivity the user can have with any program. When one of these keys is pressed, the program can detect the cursor position. Consequently, our system is designed around cursor positions and the 24 Function keys available. Programmed attention keys are not assigned specific actions in our prototype. VM assigns default uses to these keys that we felt were appropriate.

The multi-user, shared resource aspect of an IBM VM system means that response time would always be a factor and that sharing access to a HyperText and its associated link file would be problematic. VM Release 4 does not provide protected multi-user write access to a file. Typically such protection is offered by funneling the I/O requests through a VM service machine that has sole access to the protected file.

To address the response time issue we decided not to recompose text during display. Rather, all HyperDocument text was precomposed and merely displayed. This, coupled with a desire to avoid left-right scrolling, meant that the window size had to be static in width and the composed line length had to be relatively small.

Muli-user write protection is not an issue in the current prototype because it is a perusal system only. Annotations, when they are incorporated, will be personal and will not be shared until a later phase in the project.

Ultimately our HyperText system will provide collective authoring and review facilities utilizing the concept of serializing I/O and shared access in the traditional VM manner of a disconnected service machine. These constraints by no means pose insurmountable problems. The existing prototype is already quite usable for perusing reference documentation. Producing a viable and pleasing HyperText perusal system while working within the constraints posed by IBM mainframe architecture has proven challenging.

### *Advantages*

There were several advantages to using the IBM mainframe facility for developing the HyperText perusal system. Since IBM's VM operating system supports virtual storage concepts, there was never any storage constraint. This meant the system could be prototyped without concern for optimizing storage usage. If the prototype began to run out of storage, a user could always define up to 16MB of virtual storage and continue. This also meant that we did not have to be overly concerned with optimizing data structures. There was no need to conserve storage for links or text files, consequently there was no time spent determining how to do this.

The VM facilities for prototyping are quite sophisticated. The REXX language we used to write our system is interpretive, allowing for rapid changes. In addition, it provides a vast array of functions particularly suited for text manipulation. The XEDIT facility used to provide full-screen support has a simple and efficient interface to the REXX language. As well as using XEDIT for the full-screen driver of the windowing interface, we also used it for management and search of the HyperText link database. XEDIT provides both structured and unstructured associative search capabilities.

A final advantage to developing such a system on an IBM mainframe was the issue of connectivity. All users in the VM system can share access to the HyperDocuments we build and can add personal annotations which are not viewable by others. As we enhance the system to allow group annotations and dynamic link definition by readers, we already have the mechanisms in place to extend support to an environment where multiple users can add and delete links, annotations and bookmarks.

## Project Definition

With the knowledge of what we could and could not do, we defined our project in terms of the minimum features we could implement and then built on that base in a series of layers. Before we describe the base system and the changes that brought us through the various layers, look at how text might exist in any environment implementing an on-line text access system, including ours.

### *Text Databases*

Many mainframe users have already integrated a host-based composition system into their environment. The most common is IBM's Document Composition Facility (DCF). Over time most users build an extensive text database, often storing files so that information which must be used in more than one book is stored as a separate file and imbedded as required. This allows great flexibility in the creation of new books and the maintenance of old books. Libraries tend to consist of files coded using DCF Starter Set Generalized Markup Language (GML), custom-written GML and SCRIPT control words. Documents may be composed using the standard profile (style file) or a custom profile.

We knew it would be necessary to create an alternate macro library to recode some of the macros behind the GML tags for display to the screen and to use an alternate font profile. We found this to be a reasonable trade-off for the flexibility of on-line display.

We recognize that while many sites have an extensive investment in host-based text, other environments which might be interested in moving to the mainframe may have text stored in other formats, such as diskettes used with PC-based software, workstation-based publishing systems or typesetting systems. For these environments there are a variety of translation programs already commercially available to bring text up to the mainframe environment in some type of format. Some conversions strip all of the previous controls out of the file leaving pure text which must have the new coding integrated; other systems can provide a fairly clean conversion which includes the correct coding on the host side. However, these are not matters to cover in depth here.

Finally, we also recognize that some environments have no access to a machine readable version of their documents at all. For these environments there are two options: scan or re-key. If you scan the documents you must be sure to use an Optical Character Recognition (OCR) scanner so that your result is text characters in a file and not the bit-pattern of the page image. You must have the text characters so that you can add the coding required for your composition language. If you re-key, which is often more cost effective than it might initially appear, you can have your typists enter most of the composition markup during the keying process.

### *The Base System*

We began by identifying a base from which we could work. This was fairly easy since we knew that the starting point was to use DCF to compose source files which already existed for the terminal. This provided a file which could be browsed using the facilities already available in XEDIT, but the user interface was minimal at best.

The act of composing for the terminal did not change the defined line or page lengths, so browsing through the document meant looking at a lot of inappropriate page breaks, often in the middle of the screen.

The table of contents was still as accurate as the hard copy version of the document, as was the index. Using them, however, only provided a place from which to begin the search for the topic in question.

Regardless of the terminal, DCF could

not take advantage of color or other types of highlighting. Instead, all of the highlighting and other font changes were pared down to changes to upper case, upper case and underscored or underscored only. This makes it difficult to use font changes as a clue to what element in a document you are reading.

Despite the primitive nature of this method of providing the material back to the screen, at least two people we tested this approach on preferred having manuals on-line in even this crude fashion to paging through mounds of paper.

### *Layer 1: Reformatting For The Screen*

The next logical step was to use the facilities provided within DCF to format the text a bit more logically for the screen without going into the documents to alter the coding. The major tasks were to alter the line length to 72 characters and change the page depth to 25 lines. We also redefined the fonts to eliminate any underscoring, since underscores caused formatting to the terminal which we found hard to look at. The underscore had to be on a line by itself under the text.

We also took advantage of the large number of symbols set in the DCF profile to cut down the number of blank lines between document elements, in the running headers and footers and between headings.

The only problem we could not conquer without touching the source files at this stage was to control places where formatting had been turned off, causing lines to exceed the 72-character limit. Even at this point in the cycle we had a viable prototype from which to continue working.

### *Layer 2: Windows And Font Management*

The next phase of our project included several steps:

- Incorporate a window interface into the system
- Create a DCF profile to format off-the-shelf DCF documents for display by the HyperText system
- Develop code in REXX to transform the DCF output into text that could display on a 3270 device with user defined highlighting substituting for multiple fonts.

Several months before this project began, we had created a REXX/XEDIT windowing system that provided basic windowing facilities on 3270 terminals under VM. This system provided subroutines to:

- Open and close windows of varying size anywhere on the screen
- Overlap windows and establish a hierarchy of window interactions
- Generate and maintain attribute characters allowing for the display of multiple colors or highlighting on 3270 terminals
- Perform input and output in windowed areas with automatic clipping.

Windowing interfaces on 3270 terminals are uncommon; having one ready to use made it possible to provide a dramatically improved look to the system from the beginning. Without it, HyperText nodes would have to be displayed over the entire screen area, overlaying any previous nodes.

Since one of our prime goals was to simulate font changes in printed text while remaining constrained to a single-font terminal, it became necessary to have DCF mark font changes in such a way that a program could operate on the composed text from DCF and transform DCF's marking into 3270 attribute characters. A method was found to allow DCF to define up to 256 different fonts when comprising for a fixed-pitch IBM line printer. The technique involved DCF's ability to define logical fonts as the underscoring or overstriking of text.

DCF uses a profile of control statements to symbolically define fonts, spacing, page and line lengths and other aspects of a text composition. By creating another new profile which reset line and page length, redefined highlight, heading and other fonts and redefined spacing constants for heading, paragraph and other transitions, we were able to leave the DCF source code for our documents untouched while formatting them into completely different looking documents for inclusion into the HyperText system.

A post-processing program was written to convert the DCF output that was formatted for an IBM line printer into text suitable for display on 3270 devices. A profile was used to indicate what screen attributes were to be substituted for each DCF generated font.

Due to limitations of XEDIT it was ultimately feasible to define only 60 fonts to DCF instead of its maximum of 256. This was quite sufficient, however, since the screen had only 28 possible combinations of display attributes, many of which were unacceptable — blinking and reverse video text, for instance.

Because the profile developed removed running headers and running footers, the HyperDocument elements have no intrinsic page boundaries. Consequently, we dynamically define window depth as a function of screen depth and construct window heading and footing lines dynamically during the display.

When this phase of the project was complete we had several of the final components of our prototype:

- A DCF profile that could compose DCF source text in such a way that multiple fonts were programatically distinguishable even though the designated output device was an IBM single font line printer
- A post-processing program that could transform the DCF composed text into a HyperText document element suitable for display on IBM 3270 type terminals
- A windowed browser for the post-processed text which could substitute color and/or highlighting for font changes.

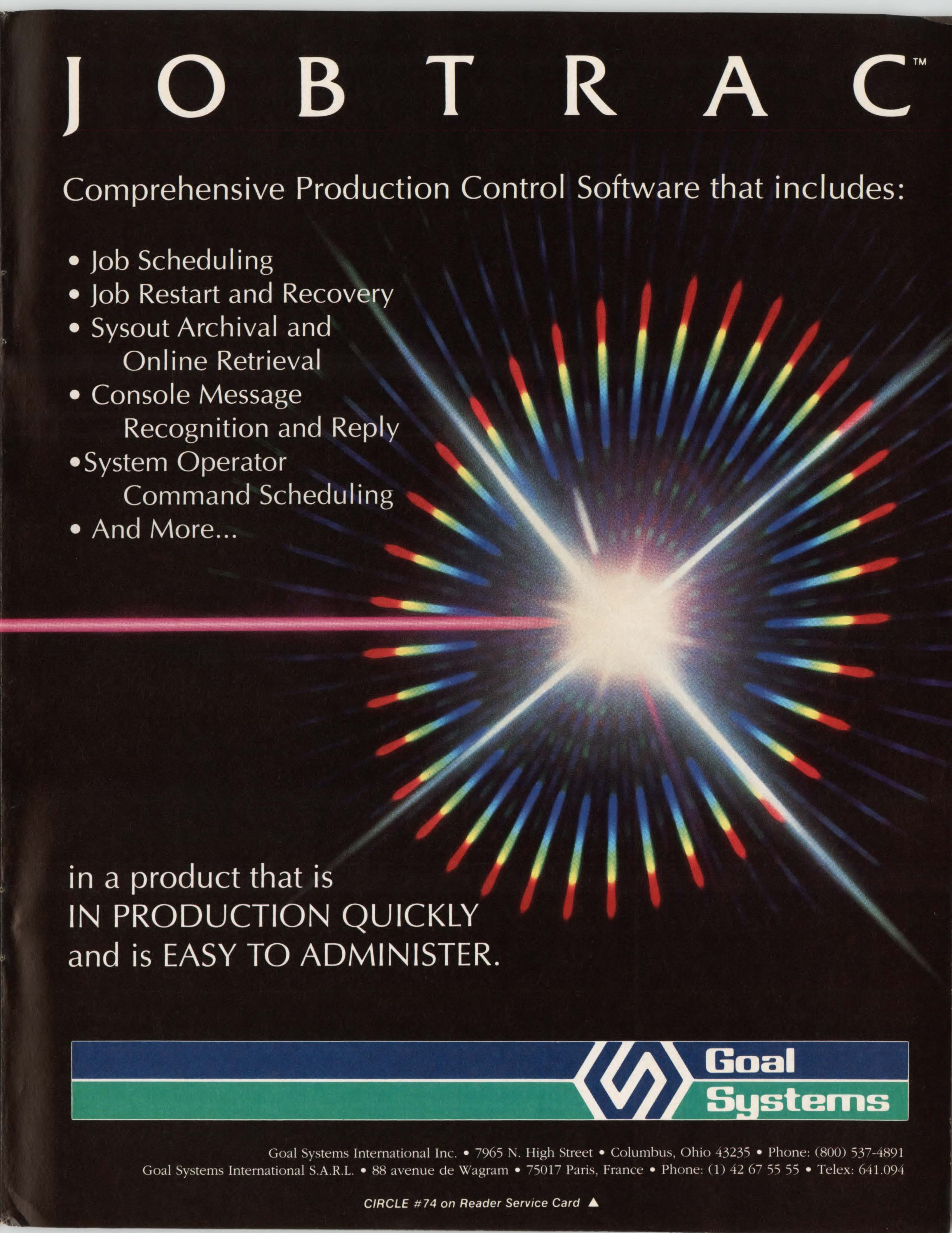
### *Layer 3: Intra-Document Links*

The next step was to turn the windowed browser into a basic HyperText system by adding links between HyperDocument nodes. The goal during this phase was to define the link file and its processing, but to add only intra-document links.

The link model we settled on has the following characteristics:

- Source nodes are defined as strings of text at specific places in a document
- Target nodes are defined as a specific line of text in a document
- Links are defined as pointers from a source node to a target node
- There are no intrinsic semantic assignments to the links; that is, there are no link *types*
- Links may have variable length *names*
- Links are one way only, pointing from a source node to a target node; source nodes may be target nodes and vice versa
- Source nodes may have up to 12 links emanating from them
- Target nodes may have any number of links pointing to them
- It is possible to generate both *I reference it* and *It references me* lists of links and nodes.

Source nodes are highlighted by a unique combination of attributes when displayed. When the cursor is inside a source node, pressing Function key produces a list of the links emanating from this source node. Each defined link is dynamically assigned to a Function key so that when the key associated with a defined link is pressed, the HyperText system locates and displays a window


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 ▲

containing the target node and text following it.

We explored the idea of giving each HyperDocument its own link file but decided this was unnecessary. XEDIT makes it possible to partition the link file by source node HyperDocument element name and exmaine only the links whose start and end addresses surround the cursor position. From this list, a menu of up to 12 link names can be displayed.

At this phase of the project all links had to be manually defined by editing the link file. While we were already designing the automated link definition system consisting of GML tags and a link resolver routine, we felt that a rapid prototype of the HyperText perusal system would serve our goals better. Consequently, our link file was designed to be edited by hand easily and to work well with the REXX/XEDIT environment of the prototype.

At the end of our fourth project phase we had more major components of our eventual system:

- A HyperText model which was simple yet versatile
- A few mono-element HyperDocuments consisting of a single element and a set of intra-document links
- A HyperText link data structure that operates efficiently within the REXX/XEDIT environment of our prototype and which allows for multi-element HyperDocuments without change.

### *Layer 4: Inter-document Links Plus User Navigation*

This phase of the project had two items as its goals:

- The expansion of the existing HyperText system to include multi-element HyperDocuments
- The addition of an associative search facility to aid sequential browsing of any specific HyperDocument element.

The first of these two goals was simple. Since the link file contained the CMS filename of the HyperDocument element that contained the target node, as soon as other CMS files were referenced inter-document links were supported. Due to the flexibility of the REXX/XEDIT prototyping environment we were using, no other changes had to be made.

Providing associative searching capabilities involved a bit more effort but was accomplished by opening a search argument window and again using the facilities of the prototyping environment to perform a search. The result is displayed either in the current window or in an error window if the search fails.

While searches are only supported in this minimal fashion now, we do intend to enhance the capabilities to allow more complex search arguments. One of the major enhancements will be to offer search that is case *insensitive* which is impossible with the current system.

The system has not been changed since this fifth phase was completed. We now have a functional, if minimal, HyperText system on an IBM mainframe under VM. It has also been ported to C in the PC environment. The largest obstacle currently is the definition of links for the HyperDocuments. Beyond that, there are many functional enhancements planned which will improve the usability of the system.

### *Layer 5: Better Perusal And Autolinks*

At this time the development directions for our HyperText project will diverge into two parallel paths. One of these paths will concern itself with automating the link definition process. The other development path will concentrate on the addition of functionality to the perusal system. This will come in two major phases.

#### Autolinks

Automating the link definition path will make it possible to write HyperDocuments from scratch, as well as altering existing documents so that they automatically generate their linkage information. The automation process will have two sides:

- DCF GML tags used in the DCF source to define source and target nodes
- Post-processing software to process and resolve the links.

DCF source code for our documents is marked up in GML which consists of tags to define document elements such as paragraphs and headings. We intend to create a new document element, the *HyperText node*.

Since target nodes are simply a starting point in the text and not a text string, they will be marked by a tag that assigns them a unique reference ID. Text strings which are to be source links will be delimited by start and end tags and will note the link name and the target's reference ID for each link emanating from them. In addition, the source node string will be placed in a special *link-font* which will be used solely for the purpose of indicating HyperText source nodes.

DCF allows logical devices with different default characteristics such as page length and physical output device to be defined. We will define a HyperText device that will physically be an IBM line printer. The tags will examine the logical device and if it is not a *HyperText device*, they will perform no action. In this manner it will be possible to leave the HyperText node defining tags in the DCF source files of our documents when they are composed for hard copy output on laser printers or other such devices. The tags will only become activated when a particular profile and logical device combination is used.

It will be possible to specify up to 12 target nodes in each source node definition. Since target nodes will each have a unique reference ID, it will be possible for any number of source nodes to point to a single target node.

Once composition is complete, the post-processor will begin the task of converting the IBM line printer output into HyperDocument files while resolving link references to build the final HyperText link file. The first instantiation of this process will likely be a multi-pass program that will not be terribly efficient. Later versions will incorporate more efficient routines.

When documents are changed and recomposed, it will be necessary to determine which document's link definition file needs to be integrated into the existing link file. We are currently examining strategies for performing this *link integration* function.

#### Enhancing Text Perusal

While development is underway to automate the link definition process, the parallel phase of the project, enhancing functionality, will also be proceeding. Functional improvements currently planned include:

- Introduction of the ability for readers to add annotations
- Improving associative search capabilities
- Enhancing the help processor
- Providing user control over windows.

Adding annotations appears, at first glance, to be simple. A window is opened and text is entered. A link is then built that allows the user to find the annotation later. There are problems with this approach, however.

- How is the user made aware that annotations exist and what pieces of text they are linked to?
- How is the annotation file maintained across revisions to the source text? Is the file maintained at all?

- Are annotations always personal and private or are they sometimes to be shared? If so, how is this accomplished?

Improving the associative search capabilities will not prove too much trouble. XEDIT has sophisticated search capabilities which can be employed. Barring that, a specific search program can be implemented without too much difficulty.

The help processor, at the moment, merely provides a list of Function key definitions. Planned enhancements to it include:

- The ability to activate a function from the menu
- Context sensitive help based on desired function
- HyperText links into a full reference and tutorial.

Since the text is now composed without page breaks, it is possible to display the text in a window of any depth. Window width needs to be static because the composed text cannot have its width easily changed. There is no reason, however, to not allow the user to resize windows vertically in order to see more text in any given window. In addition, users will be given the ability to move windows about on the screen to better utilize display real estate.

These functional improvements will add a lot to the usability of the system as a whole. The next and final phase of the project will add the last few missing pieces to provide a fully implemented HyperText system on an IBM mainframe.

### *Layer 6: Graphics, Bookmarking And User-Defined Links*

The final phase of the project will be geared toward adding the most sophisticated features of the system. The goals of this phase are quite challenging given the environment we are working in and will require significant innovation to achieve.

- Add a HyperDocument structure browser that can graphically display the links between nodes
- Add bookmarking facilities as a reference tool available to users
- Provide real-time group perusal and review facilities
- Allow users to define their own intra- and inter-document links.

While a structure browser is a logical and obvious tool for use with any HyperText system, it is one of the most difficult to provide to IBM-based users. Recall that the terminals we are dealing with may have no graphic capabilities, not even box characters. In addition, the screen is so limited in terms of width and depth that it is next to impossible to display any graph theoretical structures such as a HyperText network. We continue to explore options and alternatives, most of which center around text and indention as the structure indicator.

Bookmarks are the concept of *paper-clipping* pages for later referral. In a HyperText environment it is useful to apply such marks in a hierarchical fashion, so that the user may not only return to a point left earlier, but also determine the path through the document which led there. It is this last, historical, aspect of bookmarking which makes it such a challenge.

The most technologically challenging goal we face with respect to our HyperText prototype is the implementation of real-time group authoring and review facilities. Our plans in this area include providing the ability for a collection of users to simultaneously review, edit, annotate and discuss a common HyperDocument on-line, at remote distances.

IBM mainframe architecture in general and the VM operating system in particular do not lend themselves to such facilities simply, although there have been several announcements from IBM of facilities in VM which will make this much more feasible. Once again, the problem of limits on screen real estate and functionality pose the greatest challenges. Another major challenge, however, will be the coordination and update of group activities and annotations.

We see this process evolving into a full computer conferencing system with a HyperText interface as the front end. It is certainly possible and sensible to think of even real-time commentary and discussion over a computer network as a kind of living HyperDocument, constantly being rewritten and relinked.

As currently instantiated, our HyperText system provides readers with a non-linear but static document. The links are preconceived by the author and are cast in stone. While we provide a rudimentary associative search and will be providing better facilities in this area, there is still something missing.

HyperText systems become truly unique in documenting technology when they allow users to define their own links. This, effectively, allows a user to *rewrite* the document on the fly with a personal flair. It also adds a danger to the process of research.

One presumes that HyperDocuments will be written to be such. The links will be significant and intentional on the part of the author. A question must be raised as to the validity of allowing readers to *rewrite* such a document to fit their own tastes. The entire meaning of a HyperDocument could be changed by a reader given such a facility. On the other hand, a reader may divine insights and significant associations specifically by building personal links.

The entire question may be academic in that such personally defined HyperDocuments can be made strictly personal either by:

- Allowing users to track through only their own personal link files if they have defined such
- Clearly delineating the author's HyperDocument structures from the readers' structures when such exist.

In either case, however, personal HyperDocument definition facilities pose philosophical as well as technological problems and will be all the more interesting to solve because of that fact.


*ABOUT THE AUTHORS*

*Patricia C. McGrew is a senior advisor to the vice president of product development at Image Sciences, Inc. She is the in-house consultant on text composition and document management and developed and taught the initial class offering in IBM's Document Composition Facility offered by Image Sciences. McGrew is involved in both the SHARE Document Composition and Electronic Publishing Project and Technical Communication Project as a speaker, planner and deputy project manager.*

*William D. McDaniel is director of research and design at Image Sciences, Inc. Previously he was manager of product development. He is involved in the SHARE Document Composition and Electronic Publishing Project and Technical Communication Project as a speaker and is a member of the SHARE Strategic Issues Task Force. Image Sciences, Inc., 5910 N. Central Expressway, Dallas, TX 75206, (214) 891-6500.*

*McGrew and McDaniel have coauthored a book,* On-line Text Management: Hypertext and Other Techniques, *(J. Ranade Series/McGraw-Hill), ISBN 0-07-046263-1, 1-800-2MCGRAW.*

**Q Why do I need to journal VSAM updates created by batch programs in an MVS environment when I already have CICS doing the journaling?**

**A** CICS journaling will journal only VSAM records that have been updated by CICS transactions. If VSAM updates occur in a batch environment, you need to either backup the entire dataset prior to running the batch job or create a journal of records that are being updated by the batch job. This assumes that you desire to recover the VSAM dataset if an error occurs during processing.

If you elect the backup method, you may be expending resources (CPU, DASD I/O, TAPE I/O) over and above what is necessary to maintain an adequate recovery process. By using an MVS journaling utility, you only journal those records that are being affected by the batch job.

Many recovery packages on the market today allow users to combine both batch and CICS journals to effect an entire dataset recovery. This can simplify the recovery process when the datasets are updated in both environments.

(Answer provided by the Davis, Thomas & Associates Hotline staff in Minneapolis, MN)

**Q Why do I need to archive my journal backups?**

**A** You do not need to, but it is highly recommended. Depending on your shop, on a daily basis you may be creating one or more journal files for each CICS that is running. Or, in the case of journaling batch updates, you may also be creating several of those journals on a daily basis. When a journal fills, it can require manual operator intervention to back up the journal file. Archiving is a method of automatically backing up journals and maintaining a record of what is contained on each journal backup. This can speed the recovery process when such a need arises.

(Answer provided by the Davis, Thomas & Associates Hotline staff in Minneapolis, MN)

**Q Can I code my own journaling routines for journaling updates that occur in batch?**

**A** Yes, of course you can. However, it can be difficult to determine just when VSAM will commit the write to the file. Depending on the dataset definition, your application may say, "Do it now," and VSAM may do it later. Also, you need to code journaling routines into every batch program that runs and also provides a process that will allow you to recover the dataset from the journal. Then you need to standardize the recovery process so that operations can recover datasets without programmer intervention (we have enough to do as it is now).

In summary, it can be done.

(Answer provided by the Davis, Thomas & Associates Hotline staff in Minneapolis, MN)

**Q We are a VM/VSE/CICS shop and are trying to use a VM-owned line on a remote controller through a 3705 which would allow us to have CICS and VM on the same controller. We understand that the terminals need to be defined to CICS as local and the line is taken away from VSE. But, after we added two printers on the line, the terminals stop until the printing is done. What is causing this?**

**A** We have experienced this in one of our sites already. What seems to be going on is that the SDSCI definition in CICS is treated as a Physical Unit (PU) to VTAM. When the printer is printing it ties up the ENTIRE PU, that is, anything under TYPE=SDSCI. The solution is to put an SDSCI in front of every printer you have so it does not tie up the entire SDSCI block. We also put a TYPE=SDSCI in front of the terminals and this seems to help the response.

(Answer provided by Dale Holtberg and Paul Alden of Davis, Thomas & Associates, Minneapolis, MN)

**Q We are a VTAM VSE/CICS environment. We have currently added more terminals to the TCT and we are running out of non-pageable buffer pool memory. We have been told to use the ITLIM parameter in VTAM to take care of the memory allocation at open time. When we change this parameter to a low number, we seem to run out sooner. What are we missing?**

**A** Only a few parts are missing. First of all, we are assuming that you are using connect=auto in the TCT. Next, your information about the ITLIM parameter is correct; however, one more piece of information needs to be addressed. When CICS is running as a VTAM application, CICS will use OPNDST ACQ and SIMLOGON instead of a plain OPNDST. The problem is that ITLIM is controlling the plain OPNDST. When CICS issues the SIMLOGON or OPNDST ACQ for a connect=auto terminal, it goes to VTAM and an OPNDST is performed. However, CICS is still firing the other connect=auto terminals SIMLOGONS or OPNDST ACQ. Storage is excessively used and the VPBUF area is flooded. When the ITLIM is lowered (under the above circumstances), the SIMLOGON and OPNDST ACQ are being fired off to VTAM much faster than the OPNDSTs are performed so the buffer fills up sooner. A solution is to remove the connect=auto statement and use the logmode= parameter in VTAM instead. Then the ITLIM OPNDST controlling function will perform as stated and the storage pools will not become excessively used at startup.

(Answer provided by Dale Holtberg and Paul Alden of Davis, Thomas & Associates, Minneapolis, MN).

**Q Can I attach a digital modem to any telephone line?**

**A** The straight forward answer to your question is that there is not such a device as a digital modem. Let me explain the difference between analog and digital circuits and this will more fully answer your question.

*Analog circuits* use modems to convert the computer's digital (1s and 0s) signal to analog electrical signals. These electrical signals are then sent over the same types of lines that you use for telephone calls.

The clocking (speed at which data is transmitted) is determined by the modem in the analog circuit.

*Digital circuits* use a different type of facility than voice lines, so the digital signal does not have to be converted to analog electrical signals. Because digital circuits require special telephone company switching equipment, this equipment is not available everywhere. With digital circuits, error rates are dramatically lower (better) than for analog circuits.

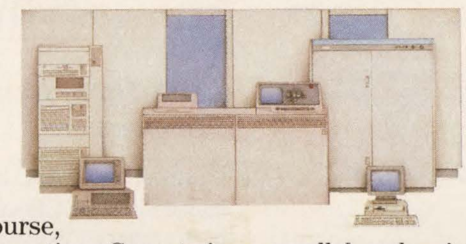
Clocking for the entire U.S. digital network is done by one clock and you order a certain speed circuit — not modems of a certain speed. There are no digital modems, but there is a piece of equipment called a Digital Service Unit (DSU) that sits where the modem would go. Data communications people generally refer to DSUs as modems, thus further confusing the issue.

(Answer provided by Steve Steinmetz, Davis, Thomas & Associates, Minneapolis, MN.)

# 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

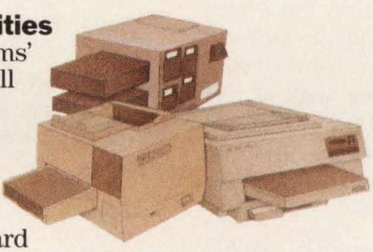


Printer Systems laser printers can be shared by several different kinds of computers. Like IBM. And DEC. And, of course, PCs. All at the same time. Coax, twinax, parallel, and serial connector ports are provided on all our laser printers.

### 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

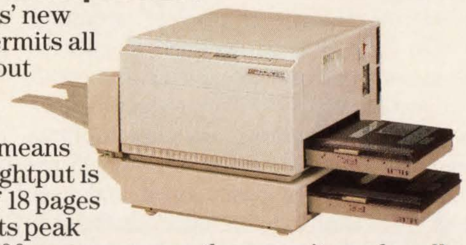


With Printer Systems' new laser printers, full emulation of the IBM 3812, the DEC LN03 Plus, and the HP LaserJet II is a given. In fact, our printers not only come standard with all the same fonts as IBM's, DEC's, and HP's laser printers, but our font matrices are precisely identical to theirs. So ours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

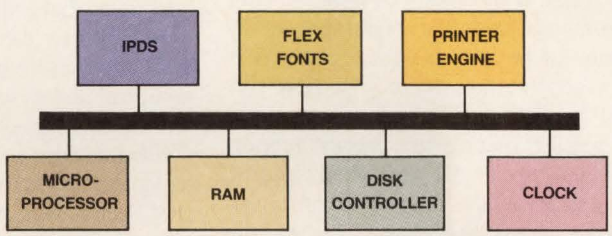


Printer Systems' new Intelliprint 218 permits all this sharing without giving up performance. Its 32-bit RISC technology means that its real throughtput is the rated speed of 18 pages per minute. And its peak duty cycle of 50,000 pages per month means it can handle however much you demand of it.

### 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* ▲

## *SQL/PM, Performance Monitor For SQL/DS Announced*

VM Systems Group recently announced, SQL/PM, a new real-time performance monitor for the SQL/DS relational environment for the VM operating system. SQL/PM is designed to address the serious performance issues that come with use of SQL/DS. The product focuses on providing organizations with comprehensive, real-time capture of SQL/DS resources at the database, agent and user level. Provisions are included for the collection of historical data that can be used in trend analysis or capacity planning.

For more information contact Bruce Mancinelli at VM Systems Group, Inc., (800) 233-6686 or:

**Circle #200 on the Reader Service Card**

## *AF/REMOTE Adds Beeper To Speed Problem Resolution*

Candle Corporation announced that its most recent OMEGACENTER product release, AF/REMOTE Version 150, enables users from remote locations to diagnose and solve crucial system problems day or night at the speed of a phone call. The product is said to alleviate time-delay situations of having to wait for the "expert" to rescue the system. The new version of AF/REMOTE includes an automatic paging feature that beeps support technicians on- or off-site when specific events occur on the host. It permits operators to access hardware consoles, the operating system console, VTAM applications and even non-VTAM applications from a single host-connected PC in the data center, providing the ability to completely IPL, control and monitor a system. Using modem access, remote PCs or ASCII terminals can connect to all these functions.

For more information contact Jan Gruenwald at Candle Corporation, (800) 843-3970 or:

**Circle #201 on the Reader Service Card**

## *VITAL SIGNS Provides VM Users 22 Enhancements*

VITAL SIGNS Version 4.0, BlueLine Software's VM performance monitor, includes 22 major enhancements. The new version includes RECALL, which provides real-time access to historical data previously collected by VITAL SIGNS which is readily available on DASD. It also includes monitoring of (CMS) response time, scheduling of time and event commands, reporting by performance groups, minidisk activity seek and national language support. Increased security, extension of report writer capability and user ability to customize VITAL SIGNS are other features of Version 4.0.

For more information contact Denny Yost at BlueLine Software, Inc., (800) 826-0313 or:

**Circle #202 on the Reader Service Card**

## *POINTER CHECKER PLUS Provides High-Speed Pointer Verification*

BMC Corporation's POINTER CHECKER PLUS is a new product that provides high-speed pointer verification for full-function databases during the IMS/VS image copy process. With the product, users can check pointers with each image copy while increasing elapsed time by less than five percent. This allows pointer verification on a more timely and frequent basis, greatly increasing data integrity. POINTER CHECKER PLUS also provides stand-alone verification to track down pointer errors after they have been detected, allowing for quick problem analysis and resolution because of faster verification and detailed error analysis. An ISPF interface is provided for on-line access to reports, option specification and JCL generation and it allocates most required datasets.

For more information contact Sandy Richardson at BMC Corporation, (800) 841-2031 or:

**Circle #203 on the Reader Service Card**

## *VSE/XRM Extends Real Memory Addressing Capacity To 64MB*

Software Pursuits has announced the availability of VSE/Extended Real Memory (XRM), an extension to IBM's VSE operating system. VSE/XRM allows the VSE/SP operating system to address up to 64MB of real memory. VSE/XRM takes advantage of a hardware feature known as the Extended Real Addressing Facility (ERAF) which is available on all mainframes that support more than 16MB of memory. The product is transparent to the VSE operating system, requires no source code modifications or changes to existing software. It is compatible with VSE/SP Release 3.1 and higher and runs in a native VSE environment under VM/HPO using Preferred Machine Assist (PMA), under VM/XA and under PR/SM. There are no known incompatibilities with any IBM system software, third-party system software or any application systems.

For more information contact Ed LeClair at Software Pursuits, Inc., (415) 769-4900 or:

**Circle #204 on the Reader Service Card**

## *CICS Shops Conquer Compressions*

Axios Products has significantly enhanced its FETCH and FETCH/XA CICS performance software products. The FETCH products, which reportedly improve CICS performance by providing more efficient program loading, now provide significant CICS program compression relief. By monitoring the amount of DSA available for program storage, the FETCH products can release low-use programs before DSA goes "short on storage" and is forced into a compression. The result is said to be further improved CICS performance because compression is infrequent or never. The user controls the activation of this feature depending on whether or not CICS compression is a performance factor in the shop.

For more information contact Linda Curto at Axios Products, Inc., (516) 348-1900 or:

**Circle #205 on the Reader Service Card**

## *DR/VFI Facilitates Application Recovery In Disaster Planning*

Systems/Software Engineering has announced the availability of DR/VFI which provides a software solution that identifies vital files needed to effect recovery of application software. S/SE's vital file identification technique ensures that a systematic approach is being taken in the identification and storage of critical data which is vital to the business resumption process. The product eliminates the need to save entire DASD farms for recovery purposes and greatly reduces the manual efforts associated with keeping recovery plans current.

For more information contact Frank DeBella, Systems/Software Engineering, (215) 341-9017 or:

**Circle #206 on the Reader Service Card**

## Exceptional Opportunity

. . . to join the recognized leader in the software industry. BMC, ranked first by Business Week magazine in per-employee investment in research and development, is looking for qualified software personnel. BMC Software employees design, develop, market and support systems software products to enhance IBM mainframe technology. Opportunities are available for:

### Product Developer/Programmer

Requirements Include:

- Previous product design or development in one or more of the following IBM systems:
  - MVS/XA
  - MVS/ESA
  - VTAM
  - CICS
  - IMS
  - DB2
  - Strong 370 Assembler Progamming Skills

### Product Support Representatives/ Systems Programmers

Requirements Include:

- Excellent testing, quality assurance and diagnostic skills
- Excellent oral and written communication skills
- 4-5 years experience in one of the following technical positions.
  - Installation, support and maintenance of VTAM, CICS and RACF
  - Installation, support and maintenance of IMS, DB/DC, DB2
  - Installation, support and maintenance of VM/XA and MVS/XA
- Strong 370 Assembler programming skills

### Technical Writers

Requirements Include:

- Experience in writing technical documentation for systems software
- Familiarity with TSO/ISPF and SCRIPT is highly desirable
- Experience with at least one of the following IBM mainframe systems: MVS, VM, IMS, CICS, and DB2

At BMC, serious technical professionals will discover an atmosphere uniquely conducive to both professional and personal growth. Be a part of the continuing growth where talent, dedication and an innovative spirit have made BMC Software an industry leader in software development.

If you are a non-smoker and meet the above requirements, send your resume in strictest confidence to:

BMC SOFTWARE
Attn: Personnel
P.O. Box 2002
Sugar Land, Texas 77487-2002

*Equal Opportunity Employer M/F/H/V*

## LEADING EDGE TECHNOLOGICAL ENVIRONMENT

### San Antonio, Texas

Work for the company that...

- Keys 5,200,000 full function IMS transactions daily (7 million projected by 1990)
- Is known for using technology as a strategic weapon
- Assisted in developing an "Image of the Future" for IBM*
- Has a 4 day work week
- Provides a flexible benefit package
- Works with tomorrow's technology...today

**ISD is comprised of seven departments:**

Planning and Administration; Property & Casualty Systems; Financial Services Systems; Financial & Corporate Systems; Claims Systems; Computer Center; Business Systems Communication.

**PRODUCTS & SERVICES:**

• Property & Casualty Insurance • Investment Services • Discount Brokerage Services • Banking Services • Satellite Communications Company • Travel Services • Buying Services • Life and Health Insurance Annuities

We are currently seeking **Systems Programmers** who have strengths in the following areas:

**DATA BASE MANAGEMENT**

Requires 5 years experience in IMS application programming, including:

- Proven experience in design, performance and tuning of IMS databases
- IMS application dump debugging skills
- Experience with DB2 table maintenance a plus

**IMS PERFORMANCE AND TUNING**

Requires 2-3 years IMS systems programming experience with:

- IMS application programming skills
- Knowledge of MVS/ESA concepts, IMS internals and basic performance management tools

**CICS SYSTEMS PROGRAMMER**

Requires 1-2 years experience as a CICS Systems Programmer, including:

- Experience in CICS programming, dump debugging, SMPE maintenance and system administration
- Knowledge of MVS/ESA concepts and CICS internals (1.6 or higher)
- Experience with MRO and ISC a plus

**NETWORK MANAGEMENT SUPPORT**

Requires a minimum of 8 years experience in IBM's Systems Network Architecture with:

- Proven understanding of SNA Control and Flow, Problem Determination, and transmission protocols
- Proven experience in VTAM, NCP, Netview and Communication Controllers
- Experience in Network Capacity Planning, Modeling, Tuning, Automating and Monitoring

**STRATEGIC SYSTEMS DEVELOPMENT**

Requires 3-5 years experience in systems programming on large IBM mainframe with: • Emphasis on CICS

San Antonio, the 9th largest city in the U.S., has many amenities to offer...a scenic Riverwalk, the symphony, live theater, fine dining, night life, professional sports, cultural events, as well as 5 major institutions of higher education.

Qualified candidates please send resume to:

**USAA**
**USAA Building**
**San Antonio, Texas 78288**
**Attn: Employment & Placement/TLL MJ**

USAA

*As appeared in IBM's full page ad in "The Wall Street Journal", September 23, 1988.
An Equal Opportunity Employer, M/F. Principals only, please.

## Looking For IBM Mainframe Expertise?

Place your recruitment ad in MAINFRAME JOURNAL and you'll reach more DP Managers, Systems/Applications Programmers, and Operators with IBM mainframe expertise than any other publication.

MAINFRAME JOURNAL is still the only publication specifically targeted to DP professionals in organizations using IBM mainframe systems.

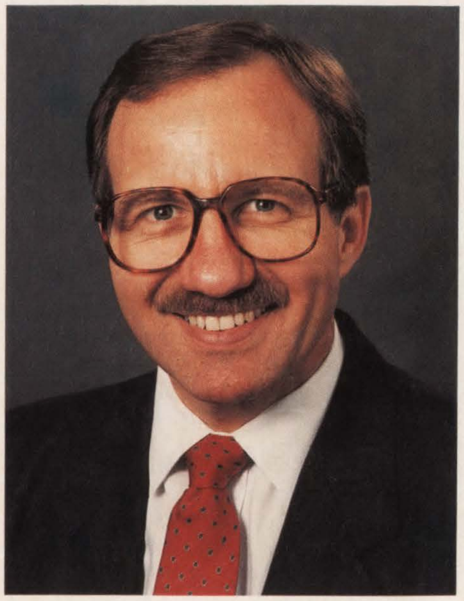
Why pay almost twice as much for a recruitment ad in a generic DP publication that reaches fewer of the people you are looking for?

**For more information, contact Diane Dishman, Marketplace Ad Sales, at (214) 343-3717.**

# Mergers And Acquisitions

## *Steroids Or Hemorrhoids?*

By Willard J. Cecchi



*Willard J. Cecchi, President, BlueLine Software, Inc.*

Computer Associates buys Cullinet Software, Duquesne and Morino merge then acquire BST; IBM invests in Computer Task Group Policy Management Systems, MSA and others for a total of 12 deals so far in 1989. VM Software buys System Center and then changes names.

The list of merger and acquisition deals seems endless. Each year the pace increases. A fact borne out by the new Broadview/ASAPSO report, showing acquisitions of information services/software firms was up 21 percent in 1988 counting the number of transactions. The value of these deals leaped 70 percent at a time when the total M&A activity across all industries has been flat for the last 10 years. The pace seems to have slackened so far in 1989 with the number of software related deals down 12 percent. However, our interest in this situation is heightened when companies we know which played a founding part in the mainframe software business are acquired.

### What Are The Benefits?

Everyone has an opinion on this frenzy of mergers, acquisitions, LBOs, strategic partnerships, hostile takeovers and related activities happening in most business sectors including mainframe software. Is this the law of the jungle, survival of the fittest, big fish eating little fish, terror on Wall Street or is there some actual benefit?

If you've listened to the horror stories which seem to follow each Computer Associates acquisition, you're probably thinking there are benefits, unfortunately, none of them good, at least for users. If you own a software company and need capital to expand or are a customer who wants the comfort of dealing with a larger company or a shareholder in a software company, you might feel more positive.

I've been lucky (?) enough to experience all sides of the M&A event. I've been part of a company that was acquired, I've used products that were discontinued after an acquisition and I've participated in several acquisitions. To date, for example, BlueLine has done six company/product acquisitions in building our current product line. As a result of my experience, I believe, all three parties — buyer, seller and user — can benefit if the deal is done right.

The beneficial aspects to the buyer and seller are financial. Sellers get capital for expansion and development, a return on their hard work, liquidity and access to new markets. Buyers can fill in the gaps in their product line, enter new markets, such as DEC, and expand their customer base by acquisition with less risk than starting from scratch. The software business is a financially attractive one for companies seeking to diversify into new and growing areas strategic to their future, so companies with no previous interest in marketing computer software are buying into our industry (that is, GM's EDS deal).

### What About The User?

That's great you say, but how does the user get anything out of these deals?

First, assuming the deal isn't being done on the bridge of the Titanic, the *merged* company is stronger and better able to provide support. Historically, mainframe software companies have been thinly capitalized. Many began in the proverbial garage and were boot strapped by the technical founders. With a growing customer base, they may not be able to continue developing and supporting the product in keeping with user demands. Now, the merged company, with increased resources and a larger customer base, should improve support. Of course, the company may choose not to, hence the horror stories.

Secondly, users frequently get a broader choice of products. Many small companies or individuals have demonstrated great creativity in developing innovative products but don't have the necessary marketing capabilities to reach prospective customers. A company that acquires these products and brings them to the marketplace gives users a better choice of solutions. Users also deal with a familiar supplier. The R&D funding available after an acquisition means improvements to the acquired product such as additional features, support of additional operating systems or new hardware platforms.

Thirdly, M&A can also result in reduced costs to users as the product is offered to broader markets, new financing options are made available and competition gets stronger.

### Acquisitions And Steroids

Acquisitions, like steroids, can deliver faster growth and increased strength. Also, like steroids they can have undesirable side effects, dropped products, poor support and confused sales coverage which brings to mind the other word used in the title.

My view of acquisitions is positive: there is no shortage of great product ideas waiting to be developed. Knowing that there are companies interested in acquiring products provides fuel for the individual software entrepreneur.

# Why ADABAS 5 is thousands of dollars faster.


The investments your organization makes in its data base technology will either cost it a fortune—or save it one. That's why you need a DBMS that assures optimum performance in a high production environment: ADABAS 5.

In a recent series of standard, fully scaled benchmarks, audited by Coopers & Lybrand, ADABAS 5 proved again that it is thousands of dollars faster. Each benchmark was conducted on a National Advanced Systems AS/EX™100 (equivalent in power to an IBM 3090 500S).

In the **standardized TP1 debit-credit benchmark,** ADABAS 5 worked with a data base containing **1 million accounts.** The results: a **record-breaking 388 transactions per second** (tps)—99.3% serviced in under one second!

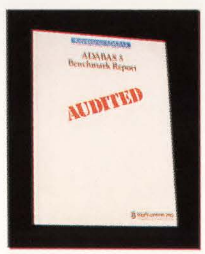
For the first time, an authentic **ET1 debit-credit benchmark** was conducted with ADABAS 5 managing **10 million accounts.** The results: **167 tps** (from terminal in/to terminal out)—99.5% serviced in under one second!

These figures represent major savings in time and money. But they're not surprising—at least not to the thousands of organizations which already use ADABAS 5.

They expect more performance. Plus the benefits that come from 18 years' experience in DBMS technology:—relational functionality which allows adaptable data structures and fast information retrieval based on multikey criteria—document management and free text retrieval—24 hour processing in a fully integrated DB/DC environment—portable applications across various hardware and operating systems—distributed processing—entity relationship data bases with recursive retrieval functions for knowledge-based systems.

Discover how much more profitable your DP organization can be. **Conduct your own ADABAS 5 benchmark,** using your own data and application profile in your own production environment. The facts will speak for themselves.

™ AS/EX is a trademark of National Advanced Systems Corporation.




**Demand the performance and functionality only ADABAS 5 can offer.**

**To order your free copy of the ADABAS 5 Benchmark Report, call toll-free:**

**1-800-843-9534 today**

(In Virginia or Canada, call 703-860-5050.)

**SOFTWARE AG**

*PROGRAMMING BUSINESS SUCCESS*

*CIRCLE #75 on Reader Service Card* ▲